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

Abstract

These Application Notes describe the steps required to package and deploy Avaya Communications Process Manager (CPM) sample SDK web application on a SAP NetWeaver Application Server. The SAP NetWeaver Application Server 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 SAP NetWeaver Application Server via the SAP NetWeaver Developer Studio. This client application was used to verify the Avaya CPM Simple Object Access Protocol (SOAP) web services interaction between the Avaya CPM and SAP NetWeaver Application Server.
1. Introduction

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

The SAP NetWeaver Application Server is an open integration and application platform. It supports Java technology and provides Java Enterprise Edition web services. The SAP NetWeaver Developer Studio is based on the Eclipse platform and supports the development of standard Java Enterprise Edition (Java EE) components. For these Application Notes, a sample Avaya CPM web client application available in the CPM SDK was deployed to the SAP NetWeaver Application Server via the SAP NetWeaver Developer Studio. This client application was used to verify the Avaya CPM Simple Object Access Protocol (SOAP) web services interaction between the Avaya CPM and SAP NetWeaver Application Server.

The Avaya CPM SDK sample web client application supports the following Avaya Communication Enabled Business Process (CEBP) 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.
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, SAP NetWeaver Developer Studio, and the SAP NWAS. The Avaya 9600 Series H.323 and SIP Telephones are registered with Avaya Communication Manager (running on the Avaya S8710 Server) 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
### 3. Equipment and Software Validated

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

<table>
<thead>
<tr>
<th>Equipment &amp; Software</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avaya S8710 Server</td>
<td>Avaya Communication Manager 4.0.1 (R014x.00.1.731.2)</td>
</tr>
<tr>
<td>Avaya G650 Media Gateway C-LAN MEDPRO</td>
<td>HW20 FW116</td>
</tr>
<tr>
<td></td>
<td>HW01 FW024</td>
</tr>
<tr>
<td>Avaya 9600 Series IP (SIP) Telephones</td>
<td>1.0.2.2</td>
</tr>
<tr>
<td>Avaya 9600 Series H323 Phones</td>
<td>1.5</td>
</tr>
<tr>
<td>Avaya Communications Process Manager</td>
<td>2.1.52</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>• Voice Portal Management System</td>
<td>4.0.0.0.2901</td>
</tr>
<tr>
<td>• Media Processing Platform</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 CPM SDK</td>
<td>Release 2.1.52</td>
</tr>
<tr>
<td>SAP NetWeaver Application Server</td>
<td>Release 7.1 SP 3</td>
</tr>
<tr>
<td>• Microsoft Windows XP Professional</td>
<td>2002, Service Pack 2</td>
</tr>
<tr>
<td>SAP NetWeaver Developer Studio</td>
<td></td>
</tr>
<tr>
<td>• Microsoft Windows XP Professional</td>
<td>Release 7.1 SP 3</td>
</tr>
<tr>
<td></td>
<td>2002, Service Pack 2</td>
</tr>
<tr>
<td>Sun Microsystems Java JDK</td>
<td>1.5_0_13</td>
</tr>
<tr>
<td>Microsoft Active Directory and DNS Server</td>
<td></td>
</tr>
<tr>
<td>Microsoft Windows Server 2003</td>
<td>Service Pack 2</td>
</tr>
<tr>
<td>Apache Software Foundation</td>
<td>1.4</td>
</tr>
<tr>
<td>Axis</td>
<td></td>
</tr>
<tr>
<td>Apache Software Foundation</td>
<td>1.7.0</td>
</tr>
<tr>
<td>Ant</td>
<td></td>
</tr>
</tbody>
</table>

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

In these Application Notes, it is assumed that the Avaya CPM software and the license file have already been previously installed. These Application Notes further assume that Avaya Communication Manager, Avaya SES, Avaya Voice Portal, and Avaya Meeting Exchange Express (as shown in Fig. 1) have already been configured and are operational with Avaya CPM. For additional information on these installation tasks, refer to [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.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Launch a web browser and enter the URL [http://&lt;Name or IP address of CPM server&gt;/VIA](http://&lt;Name or IP address of CPM server&gt;/VIA). When prompted for a user name and password, enter the credentials of the sever administrator account.</td>
</tr>
<tr>
<td>2.</td>
<td>The Communication Process Manager administration portal page appears. To add a new user, click <strong>Account</strong>.</td>
</tr>
</tbody>
</table>

---

![Communications Process Manager](image_url)
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.</td>
<td>The <strong>Account Home</strong> page appears. To create a new user account, for CPM Notifications and Response system, click <strong>Create Account</strong>.</td>
</tr>
</tbody>
</table>

### Account Home

**Account Home**  
**Help**

- The services on the left allow you to:
  - Specify personal information including your time zone
  - Change your PIN
  - Specify your preferences for being contacted

- **Points Of Contact**
  - If you are a new user, please review your points of contact, and make any necessary changes. If you do not establish points of contacts and notification profiles, the system default to notify you through email and your Inbox will be used. You may also specify time profiles if you wish to control when devices are used to contact you or specify timers that count business hours or special time intervals. You may filter your notifications by application, requester and subject to select the right notification profile for each kind of message you receive.

- **User Search**
  - You can search for users by entering letters or digits in the search box in the sidebar. If you then click on **Search**, or press **Enter**, the system will then return those users whose handle (user id) or common name matches the sequence of letters, or whose phone number contain the given digits. (The match is actually more liberal, if the handle or common name contains the digits, they will match as well.) You may also enter a **LDAP Search Filter** - All **inetOrgPerson** attributes are supported.
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 4.   | The **Create Account** screen appears. To add a new user to CPM LDAP directory, click **Create Account**. 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 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**.  
  
  The **Create Account** screen is shown in **Steps 5 and 6**. |
### Step 5

<table>
<thead>
<tr>
<th>Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Administrator:</strong></td>
</tr>
<tr>
<td><strong>CPM User:</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Handle:</strong></td>
</tr>
<tr>
<td><strong>ID Number:</strong></td>
</tr>
<tr>
<td><strong>Display Name:</strong></td>
</tr>
<tr>
<td><strong>First Name:</strong></td>
</tr>
<tr>
<td><strong>Last Name:</strong></td>
</tr>
<tr>
<td><strong>Common Name(s):</strong></td>
</tr>
<tr>
<td><strong>Phone Number:</strong></td>
</tr>
<tr>
<td><strong>Mobile Phone Number:</strong></td>
</tr>
<tr>
<td><strong>Fax Number:</strong></td>
</tr>
<tr>
<td><strong>Pager Number:</strong></td>
</tr>
<tr>
<td><strong>Electronic Mail Address:</strong></td>
</tr>
<tr>
<td><strong>Honorific:</strong></td>
</tr>
<tr>
<td><strong>Title:</strong></td>
</tr>
<tr>
<td><strong>Affiliation:</strong></td>
</tr>
<tr>
<td><strong>Manager:</strong></td>
</tr>
</tbody>
</table>

---
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.</td>
<td><strong>Department:</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Organization:</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Room:</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Street:</strong></td>
</tr>
<tr>
<td></td>
<td><strong>City:</strong></td>
</tr>
<tr>
<td></td>
<td><strong>State:</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Postal Code:</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Language:</strong> US English</td>
</tr>
<tr>
<td></td>
<td><strong>Time Zone:</strong> America/New_York</td>
</tr>
</tbody>
</table>

### Applications

- **Advisory Service:** Yes |
- **Click To Find Service:** Yes |
- **Notification And Response Service:** Yes |
- **Notify And Conference Service:** Yes

7. Repeat **Step 4** to create additional CPM users as necessary.
5. Import and Configure Sample Avaya CPM Web Client Application Project

This sample configuration uses SAP NetWeaver Developer Studio (NWDS) to deploy the CPMClient-2.1.war file to the SAP NetWeaver Application Server (NWAS). These Application Notes assume that the SAP NWAS and SAP NWDS are already installed and operational.

This section describes how to import and configure the Avaya CPM SDK web client application to the SAP NWDS to illustrate importing and configuring an application in NWDS environment. 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. For detailed SAP NWDS information, refer to [9] in the Additional References section.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>

1 The “war” file shipped with the Avaya CPM SDK can be deployed out of the Box on the SAP NetWeaver Application Server. See Section 6 for how to deploy the war file to the SAP NWAS.
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td>To import the CPM clientsdk-webui project, navigate to <strong>File ➔ Import ➔ Existing Projects into Workspace</strong>. The <strong>Import page</strong> appears. Click <strong>Existing Projects into Workspace</strong>. Click <strong>Next</strong>.</td>
</tr>
</tbody>
</table>

![Import dialog box](image-url)
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.</td>
<td>In the <strong>Select root directory</strong> field, enter the root directory of the CPM clientsdk-webui project (i.e., “C:\CPMSDK2.1\javasdk\clientsdk-webui”). The <strong>clientsdk-webui</strong> appears in the <strong>Projects</strong> pane. Click <strong>Finish</strong>.</td>
</tr>
</tbody>
</table>

![Import Projects](image)
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 4.   | Verify that the **clientsdk-webui** appears in the **Project Explorer** pane.  

**Note:** The red check next to the clientsdk-webui indicates the clientsdk-webui project has not been successfully built. Proceed Proceeding with Steps 5 – 7 will clear the error indication. |
5. To specify the Project Facets, right click on **clientsdk-webui** and then click **Properties**. The **Properties for clientsdk-webui** page appears. Click on **Modify Project**. The **Modify Faceted Project** page appears. Select **Dynamic Web Module** and **Java** with version **2.4** and **5.0** respectively. The result is shown in the screen below. Click **Finish**.
6. To modify the validation preferences, right click on `clientsdk-webui` and then click `Properties`. The `Properties for clientsdk-webui` page appears. Click on `Validation`. Uncheck `HTML Syntax Validator` and `JSP Syntax Validator`. Click `OK`.

**Note:** If validation is not required, then check **Disable all** to disable all validations.
7. To add CPM_LIB as a class path variable, navigate to `clientsdk-webui` → `Build Path` → `Configure Build Path`. **Classpath Variables** appears in the right pane. Click **New** to add the “CPM_LIB” variable pointing to the “C:/CPMSDK-2.1/javasdk/lib” folder.

Click **OK**.
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.</td>
<td>Verify that the error indication showed on Step 4 is cleared.</td>
</tr>
</tbody>
</table>
5. Create Sample Avaya CPM Web Client Application WAR file

There are two ways to create a sample Avaya SDK web application WAR file. Section 5.1 uses Ant to create the WAR file, while Section 5.2 uses Eclipse Export tool. Either method can be used.

5.1. Create Sample Avaya CPM Web Client Application WAR file Using Ant

The build.xml file is an Ant script that is provided by the CPM SDK to compile and create a deployable war file. When using Ant to create the Avaya CPM SDK web application, Ant will construct the deploy file structure (i.e., WebContent). This file structure is specified in the build.xml file and is used for creating the war file. Refer to Appendix A for more detailed information on build.xml.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>To create a CPM web client war file, click <strong>run</strong> → <strong>External Tools</strong> → <strong>clientsdk-webui build.xml</strong>.</td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>2.</td>
<td>When the build is completed successfully, the <strong>CPMClient-2.1.war</strong> file is created under the <strong>dist</strong> folder. The result of the Ant build is shown in the screen below.</td>
</tr>
</tbody>
</table>

![Ant build result screenshot](image.png)
5.2. Create Sample Avaya CPM Web Client Application WAR file Using Export WAR

Alternatively, this section describes the steps that can be used to generate the CPM web client war file (i.e., CPMClient-2.1.war) using Eclipse Export tool.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>To associate additional JAR files, right click on clientsdk-webui and then click Properties. The Properties for clientsdk-webui page appears. Click on J2EE Module Dependencies. Click Add External JARs.</td>
</tr>
</tbody>
</table>

![Image of Eclipse J2EE Module Dependencies]
2. The WAR file deployment may fail, when the JAR files are stored under the "generated" subfolder. Steps 2 and 3 move the JAR files from the "CPM_LIB\generated\" folder to the "CPM_LIB" folder. In addition, Step 3 also removes additional JAR files that should not deploy to the WAR file.

**Jar Selection** page appears (Not shown). Navigate to the target folder (i.e., "C:\CPMSDK2.1\javasdk\lib\generated\"). Select all JAR files and click Open.
3. Check all JAR files under CPM_LIB/. Uncheck the following JAR files:

- Catalina-ant.jar
- junit-3.8.jar
- serverlet-api.jar
- All JAR files under the CPM_LIB/Generated folder

Click OK.
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.</td>
<td>To create the war file, navigate to <strong>File → Export → WAR file.</strong></td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>5.</td>
<td>The <strong>Export</strong> page appears. Click the <strong>Browse</strong> button to select the directory and then enter the file name (i.e., “C:\CPMSDK2.1\javasdk\clientsdk-webui\dist\CPMClient-2.1.war”). Click <strong>Finish</strong>.</td>
</tr>
<tr>
<td>6.</td>
<td>The result is shown in the screen below.</td>
</tr>
</tbody>
</table>
6. Deploy a Sample Avaya CPM Client Application to SAP NetWeaver Application Server

This section illustrates how to deploy the CPM SDK client application to the SAP NWAS. These Application Notes assume that the SAP NWAS and SAP NetWeaver Developer Studio (NWDS) are already installed and operational. Refer to [9] in the Additional References section for the SAP NWAS installation. These Application Notes further assume that a proper version of the Sun JDK is downloaded and configured on SAP NWAS.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Verify that the SAP NetWeaver application server is running. Launch the SAP NetWeaver MMC console. Verify that SAP NWAS is up and running.</td>
</tr>
</tbody>
</table>
Step 2. To connect to the SAP NWAS from the SAP NWDS, navigate to **Window → Preferences → SAP AS Java**. Enter the following values:

- **Instance host**: Enter the SAP NWAS host IP address (i.e., “192.8.139.91”).
- **Instance number**: Enter the instance number “0”.

Click **Register SAP Instance**. The result is shown in the screen below.
3. To deploy the war file created in Section 5.2, Step 6, navigate to **Window → Show View → Other**.
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.</td>
<td>The Show View page appears. Click <strong>Deploy View → Deploy View</strong>.</td>
</tr>
</tbody>
</table>

**Show View**

- CMS
- Composite Application Framework
- Connectivity
- CVS
- Debug
- Deploy View
- Config View
- Deploy View
- Deploy View Console
- Repository View
- Undeploy View
- Design Time Repository
- Development Infrastructure
- Dictionary
- Help
- Java

**OK**  **Cancel**

| 5.   | In the Deploy View, click on deploy icon |
6. The **Export** screen appears. Enter the following values:

- Web module field: Enter CPM web client project name (i.e., “clientsdk-webui”).
- Destination: Enter the path and file name (i.e., “C:\CPMSDK2.1\javasdk\clientSDK-webui\dist\CPMClient-2.1.war”).

![Export Screen](image.png)
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.</td>
<td>The CPMClient-2.1.war appears in the <strong>Deploy View</strong>. Click the deploy icon to deploy the war file.</td>
</tr>
</tbody>
</table>
8. The result is shown in the screen below. In the Deploy View pane, a green check is shown next to the CPMClient-2.1.war file.
7. Verification Steps

The following steps may be used to verify that Avaya CPM and SAP NWAS are configured properly:

- Verify that the sample Avaya CPM Web Client application is deployed to the SAP NWAS.
- 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 that the Avaya CPM and SAP NetWeaver Application Server 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 CPMClient is deployed to the SAP NWAS. From the SAP NWAS Admin Console, launch the CPMClient web application (i.e., CPMClient-2.1).</td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>2.</td>
<td>Verify that the CPM Web Client application can be launched.</td>
</tr>
</tbody>
</table>

- Launch a web browser and enter the URL `http://<Name or IP address of SAP NWAS>:50000/CPMClient-2.1`.  
- Verify that the **Communications Process Manager Client** screen appears.

![Avaya Communications Process Manager Client](image)
### Step 3
Launch CPM Advisory service. Verify that the recipients acknowledge receipt of the notification.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.</td>
<td>Launch CPM Advisory service. Verify that the recipients acknowledge receipt of the notification.</td>
</tr>
</tbody>
</table>

### Advisory Service

- **CPM Host/IP**: 192.168.1.12
- **CPM User Name**: spAdmin
- **CPM Password**: ********
- **Originator**: spAdmin
- **Recipients**: Jenny
- **Subject**: CPM Integration test
- **Message**: CPM and SAP Integration Test

### Step 4
Launch CPM Notify and Conference service. Verify that the Notify and Conference service is successfully completed.

### Step 5
Launch CPM Notify and Respond service. Verify that the Notify and Respond service is successfully completed.

### Step 6
Launch CPM Find and Call service. Verify that the Find and Call service is successfully completed.
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.</td>
<td>Launch a web browser and enter the URL <code>http://&lt;Name or IP address of CPM server&gt;/VIA</code> 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 <em>Communications Process Manager Inbox</em> screen appears. Click <em>Outbox</em>. Select the following fields to view the completed notifications.</td>
</tr>
</tbody>
</table>

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

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

![Communications Process Manager](image)

<table>
<thead>
<tr>
<th>Application</th>
<th>Session Id</th>
<th>Start Date</th>
<th>End Date</th>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click To Find Service</td>
<td>AAAAAXv5Sw6=hIq4Mw==</td>
<td>Tuesday 15 January 2008, 12:44:58 pm</td>
<td>Tuesday 15 January 2008, 12:45:47 pm</td>
<td>Invitation to conference call.</td>
</tr>
<tr>
<td>Notification And Response Service</td>
<td>AAAAAXv5Sw4=hidQ3tW==</td>
<td>Tuesday 15 January 2008, 12:43:36 pm</td>
<td>Tuesday 15 January 2008, 12:44:11 pm</td>
<td>Notify and Respond</td>
</tr>
</tbody>
</table>

[Delete Selected Request(s)]
### Step 8

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`

### Step 9

To view the SAP NWAS log entries, from the SAP NetWeaver Application Server, click **SAP NetWeaver Administrator → Problem Management → logs and Traces → Log Viewer**. Select topics (for example, **SAP Logs (Java)**) to view the log files.
8. Conclusion
As illustrated in these Application Notes, a SAP NetWeaver Application Server can be used to successfully interoperate with Avaya Communications Process Manager via web services.

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

The SAP NetWeaver online help documents may be obtained from http://help.sap.com/
[9] “SAP NetWeaver Process Integration 7.1 Knowledge Center”

Additional documents may be downloaded:
[12] Axis 1.4 may be downloaded from http://tomcat.apache.org
10. 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.war file.

```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 applicaiton.

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

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

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

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

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

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

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

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

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

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

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

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

```xml
<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?

```xml
<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.

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

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

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

Normally, this task is executed indirectly when needed.

```xml
<target name="prepare">
  <!-- Create build directories as needed -->
  <mkdir dir="${build.home}"/>
  <mkdir dir="${build.home}/WEB-INF"/>
</target>
```
<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>
    </fileset>
  </copy>

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

<!-- ==================== Clean Target ==================================== -->
<!-- The "clean" target deletes any previous "build" and "dist" directory, so that you can be ensured the application can be built from scratch. -->

<target name="clean"
description="Delete old build and dist directories">
  <delete dir="${build.home}"/>
  <delete dir="${dist.home}"/>
</target>

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

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

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

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

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

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

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

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

<!-- The "dist" target creates a binary distribution of your application
in a directory structure ready to be archived in a tar.gz or zip file.
Note that this target depends on two others:

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

--> 

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

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

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

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

</project>