one-X™ R5.2 Web Application SDK

Workstation

16-603494
Release 5.2
November 2009
Issue 1

Notice

While reasonable efforts were made to ensure that the information in this document was complete and accurate at the time of printing, Avaya Inc. can assume no liability for any errors. Changes and corrections to the information in this document might be incorporated in future releases.

Documentation disclaimer

Avaya Inc. is not responsible for any modifications, additions, or deletions to the original published version of this documentation unless such modifications, additions, or deletions were performed by Avaya. Customer and/or End User agree to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation to the extent made by the Customer or End User.

Link disclaimer

Avaya Inc. is not responsible for the contents or reliability of any linked Web sites referenced elsewhere within this documentation, and Avaya does not necessarily endorse the products, services, or information described or offered within them. We cannot guarantee that these links will work all the time and we have no control over the availability of the linked pages.

Warranty

Avaya Inc. provides a limited warranty on this product. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya’s standard warranty language, as well as information regarding support for this product, while under warranty, is available through the Avaya Support Web site:

http://www.avaya.com/support

License

USE OR INSTALLATION OF THE PRODUCT INDICATES THE END USER'S ACCEPTANCE OF THE TERMS SET FORTH HEREIN AND THE GENERAL LICENSE TERMS AVAILABLE ON THE AVAYA WEB SITE http://support.avaya.com/LicenseInfo/ ("GENERAL LICENSE TERMS"). IF YOU DO NOT WISH TO BE BOUND BY THESE TERMS, YOU MUST RETURN THE PRODUCT(S) TO THE POINT OF PURCHASE WITHIN TEN (10) DAYS OF DELIVERY FOR A REFUND OR CREDIT. Avaya grants End User a license within the scope of the license types described below. The applicable number of licenses and units of capacity for which the license is granted will be one (1), unless a different number of licenses or units of capacity is specified in the Documentation or other materials available to End User. "Designated Processor" means a single stand-alone computing device. "Server" means a Designated Processor that hosts a software application to be accessed by multiple users. "Software" means the computer programs in object code, originally licensed by Avaya and ultimately utilized by End User, whether as stand-alone Products or pre-installed on Hardware. "Hardware" means the standard hardware Products, originally sold by Avaya and ultimately utilized by End User.

License type(s)

Designated System(s) License (DS). End User may install and use each copy of the Software on only one Designated Processor, unless a different number of Designated Processors is indicated in the Documentation or other materials available to End User. Avaya may require the Designated Processor(s) to be identified by type, serial number, feature key, location or other specific designation, or to be provided by End User to Avaya through electronic means established by Avaya specifically for this purpose.

Concurrent User License (CU). End User may install and use the Software on multiple Designated Processors or one or more Servers, so long as only the licensed number of Units are accessing and using the Software at any given time. A “Unit” means the unit on which Avaya, at its sole discretion, bases the pricing of its licenses and can be, without limitation, an agent, port or user, an e-mail or voice mail account in the name of a person or corporate function (e.g., webmaster or helpdesk), or a directory entry in the administrative database utilized by the Product that permits one user to interface with the Software. Units may be linked to a specific, identified Server.

Database License (DL). Customer may install and use each copy of the Software on one Server or on multiple Servers provided that each of the Servers on which the Software is installed communicate with no more than a single instance of the same database.

CPU License (CP). End User may install and use each copy of the Software on a number of Servers up to the number indicated by Avaya provided that the performance capacity of the Server(s) does not exceed the performance capacity specified for the Software. End User may not re-install or operate the Software on Server(s) with a larger performance capacity without Avaya's prior consent and payment of an upgrade fee.

Copyright

Except where expressly stated otherwise, the Product is protected by copyright and other laws respecting proprietary rights. Unauthorized reproduction, transfer, and or use can be a criminal, as well as a civil, offense under the applicable law.

Third-party components

Certain software programs or portions thereof included in the Product may contain software distributed under third party agreements ("Third Party Components"), which may contain terms that expand or limit rights to use certain portions of the Product ("Third Party Terms"). Information identifying Third Party Components and the Third Party Terms that apply to them is available on the Avaya Support Web site:

http://support.avaya.com/ThirdPartyLicense/

Trademarks

Avaya, the Avaya logo, and MultiVantage are either registered trademarks or trademarks of Avaya Inc. in the United States of America and/or other jurisdictions.
Microsoft is a registered trademark of Microsoft Corporation. All other trademarks are the property of their respective owners.

Downloading documents

For the most current versions of documentation, see the Avaya Support Web site:

http://www.avaya.com/support

Avaya support

Avaya provides a telephone number for you to use to report problems or to ask questions about your product. The support telephone number is 1-800-242-2121 in the United States. For additional support telephone numbers, see the Avaya Support Web site:

http://www.avaya.com/support

Content:

1 Introduction

The **one-X™ Web Application SDK** is a product created by AVAYA's Market Development Organization (MDO). It is intended for use by AVAYA customers and partners to efficiently develop web applications that consume unified communication services hosted by Avaya platforms.
This SDK specifically targets development of event-driven web applications which use web browsers as their user interface.

The SDK is based on **one-X™ Server** (1XS) **Release 5.2** as the application runtime framework. 1XS is a set of applications hosted on an IBM WebSphere JEE Application Server. As shown in Figure 1 below, the 1XS applications represent interfaces for client communication with various AVAYA server products (Communication Manager, Application Enablement Services, Meeting Exchange, Modular Messaging, Intelligent Presence Server) which serve to control the services they offer from a central point of access.

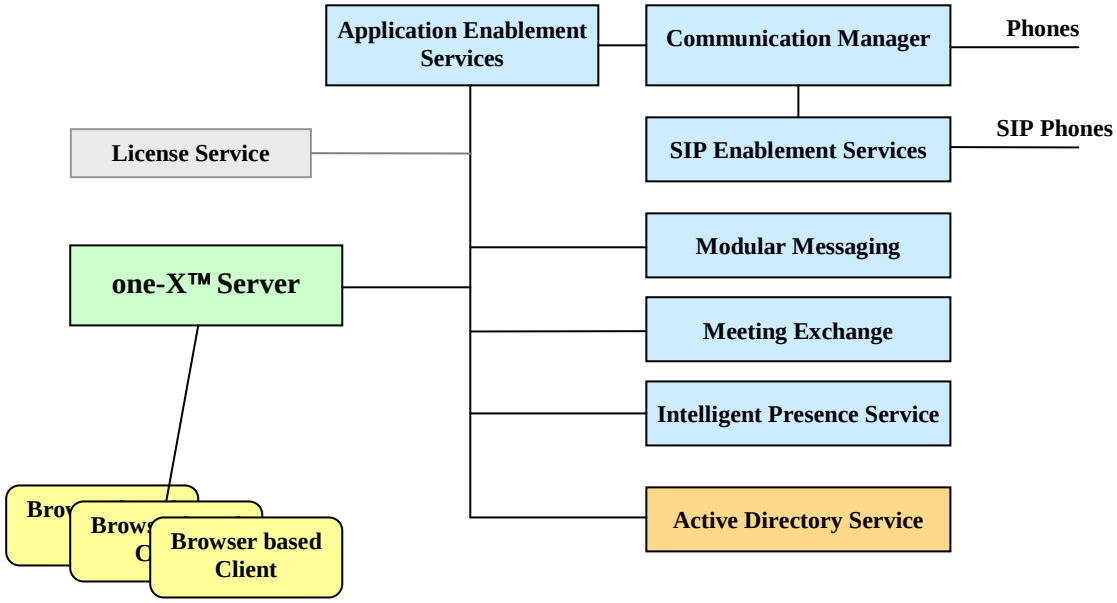

FIGURE 1: AVAYA Unified Communications environment with one-X Server (1XS) serving as a gateway between web clients and native AVAYA services

Components of the SDK include:

- Software, libraries, documentation, and demonstration applications
- Workstation tools. A workstation (MS Windows™ based) serves as the frontend for developing 1XS-hosted applications. In order to support this capability, the workstation must be enabled with a set of tools delivered with the SDK.

For deployment and testing of applications, a development laboratory infrastructure is needed. Different lab structures and configurations are imaginable:

- A single server which hosts all necessary AVAYA and non-AVAYA products in virtual machines. This fully virtualized, turnkey development lab environment may be provided by AVAYA upon request. It is delivered with a predefined configuration and works out-of-the-box. This configuration includes developer editions of some AVAYA- products. For example, AVAYA Communications Manager (CM) is delivered in its IPCoDE configuration, a software-only version of CM which has fully functional signaling and switching but with limited performance and advanced media processing capabilities.

- A multiple server setup. Production (vs. developer) editions used, with some of the server products co-resident on a single hardware server in virtual machines. This configuration also may be provided by AVAYA.
- A one-X™ Server-based Unified Communications infrastructure – full production version -- available at the customer's or partner's site.

1.1 Purpose

This document targets the workstation used in the development lab environment.
It s purpose is to document

- how the workstation tools are installed and configured;
- how the AVAYA server applications are accessed by the tools;
- how tools can be used effectively to develop web applications consuming AVAYA unified communications (UC)...

1.2 Prerequisites

It is assumed, that a user of this document is basically familiar with all mentioned tools. Additional information may be obtained from the web sites of the tool providers.

The user should at least have basic knowledge of the AVAYA Unified Communications infrastructure and of one-X™ Server, i.e., they should understand Figure 1 above.

Reconfiguration of AVAYA server applications requires in-depth knowledge and is out of scope of the SDK.

In order to understand the demo applications and to develop 1XS-based applications, the SDK user should be familiar with the following technologies:

- Java
- JSP (Java Server Pages)
- HTML, CSS, JavaScript
- JEE Applications structures, especially for Web Applications
- Eclipse IDE concepts

Some basic knowledge of the IBM WebSphere Application Server 6.1 is also helpful.

1.3 Terminology

Term	Meaning
1XP	AVAYA one-X Portal Server (**same as 1XS**)
1XS	AVAYA one-X Server **(1XS)**
ADS	Microsoft Active Directory Service
AES	AVAYA Application Enablement Services
AJAX	Asynchronous JavaScript and XML
A/S	Application Server (JEE)
CM	Avaya Communication Manager
DADS	Definity ACD Simulator (CM simulator)
DTMF	Dual Tone Multi-frequency
EAR	Enterprise Archive
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IPS	AVAYA Intelligent Presence Service
JAR	Java Archive
JEE	Java Enterprise Edition
JRE	Java Runtime Environment
JSP	Java Server Pages
LDAP	Lightweight Directory Access Protocol
MM	AVAYA Modular Messaging
MS	Microsoft
MX	AVAYA Meeting Exchange
OS	Operating System
SDK	Software Development Kit
SFTP	Secure File Transport Protocol
UI	User Interface
URL	Uniform Resource Locator
VM	Virtual Machine
VOIP	Voice over IP
VPN	Virtual Private Network
WAR	Web Archive
WAS	(IBM) WebSphere Application Server
WTP	Web Tools Platform
XML	Extensible Markup Language

2 Lab Overview

As noted in the introduction, depending on a user's requirements or already available parts, the lab infrastructure may differ.

This document is written with a fully virtualized "single server" lab environment perspective in mind. Any references to IP-addresses or user account and data configuration is valid for this infrastructure only.

There is no functional difference between the different setups.
Differences are in configuration and in the methods for accessing server components.

- In the fully virtualized environment, all server components have fixed, well-known IP addresses, they are configured for collaboration, and they have predefined user data prepared.
- All other possible lab infrastructures have their individual IP addresses for all server modules.
- In a customer-provided infrastructure, the customer is responsible for proper configuration.

2.1 Single Server Lab

The following diagram provides an overview of the fully virtualized SDK lab environment.

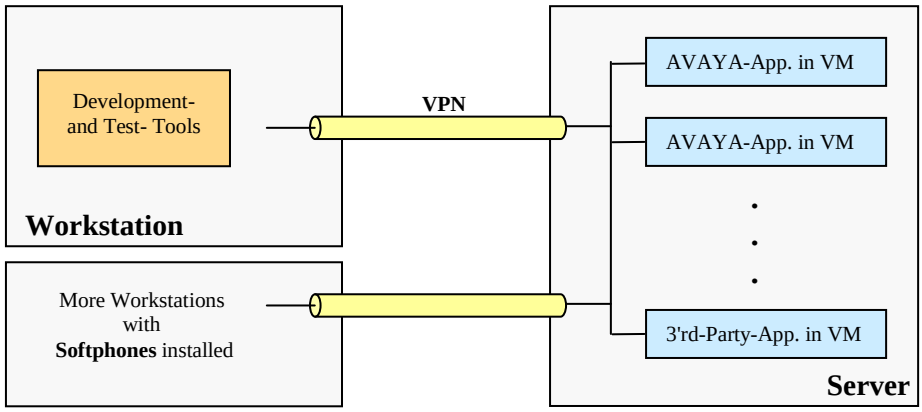

FIGURE 2: Fully virtualized SDK development laboratory environment

Several AVAYA- and 3'rd-party- server applications are running on virtual machines (VMWare) installed on a single server.

The server provides external access via a VPN (OpenVPN). Each of the virtual machines has a well defined IP-address.

Workstations run a VPN client which maps the fixed VM addresses in their address spaces.
To integrate this 'Virtual Lab' into any network, only the main server's IP address has to be adjusted.

In addition to the development workstation, some more workstations should be added to provide testability using AVAYA softphones.

Development workstations are running on MS Windows™ (XP, Vista), the main server on RedHat Linux, and the various virtual machines on the main server either on RedHat Linux or Windows 2003 Server.

2.1.1 Virtual Machines in the Single Server Lab

The following virtual machines are hosted by the server:

VM Name	OS	Server-Functionality
ADEXCH	Windows	MS Active Directory Service (LDAP server) for user authentication
CMx	Linux	AVAYA Communication Manager (Simulator-Version)
AESx	Linux	AVAYA Application Enablement Services (Simulator-Version)
SESx	Linux	AVAYA SIP Enablement Services (Simulator-Version)
1XPx	Linux	AVAYA 1X-Portal Server
MASx	Windows	AVAYA Modular Messaging
MSSx	Linux	AVAYA Modular Messaging Storage Server
MXx	Linux	AVAYA Meeting Exchange Express (Simulator-Version)
IPS	Linux	AVAYA Intelligent Presence Service

All server modules run on VMWare Server (www.vmware.com).

ADEXCH is equipped with a desktop-sharing **VNC**-server. The VNC-password is: **interop**.

2.1.2 VM IP-Addresses, Administration-URLs, User-Names and Passwords

Login parameters for the base server which hosts the VMs are:

User= root, Password=interop

The lab-server's IP-address has to be adjusted according to local network requirements.

For some of the server virtual machines, direct access to the OS might be needed:

VM Name	IP	User	Password
ADEXCH	192.168.17.126	Administrator	interop
CMx	192.168.17.129	root	interop
1XPx	192.168.17.134	root	interop
MASx	192.168.17.135	mmcust	Interop123

For administration of some of the server applications, Web-interfaces are available:

Application	URL	User	Password
1XP-Admin-Client	http://192.168.17.134:9080/admin	websphere	Interop123
1XP-WebSphere Console	http://192.168.17.134:9043/ibm/console/login.jsp	websphere	Interop123
MAS-Administration	http://192.168.17.136	craft	interop

2.1.3 Predefined 1XS Users and their Communication Infrastructure

For a list of configurations see the Appendix 6.1 "**SDK-Lab: Phones, Accounts, Passwords …**".

2.1.4 1XP Client

One-X™ Portal Client is AVAYA's full featured 1XS client.
Its start URL is http://192.168.17.134:9080.

See Appendix 6.1 for a list of 1XS users enabled to login.

2.1.5 Limitations of the Single Server Lab

The fully virtualized lab built with simulator versions of AVAYA server applications in virtual machines has some limitations:

- The performance of software running on this lab is low but acceptable as development environment for a single user

Any operation based on DTMF signals, like placing a call to a user's voicemail-box or sending host/participant codes upon entering a bridge conference, are basically unsupported.
DTMF requires a Media-Gateway added to the lab

- Voice channels in conferences on CM do not work (but conferencing works).
  Voice channel support in CM conferences requires a Media-Gateway added to the lab.
- The lab does not provide web-based user interfaces for AVAYA Meeting Exchange and Modular Messaging ("Web Subscriber Options").

3 Workstation Tools

3.1 Overview

The SDK Development workstation tools may be classed into several groups:

- Networking and management tools for access of the fully virtualized Single Server infrastructure.
  These tools are not needed in a non-virtualized lab.
- Tools for file management on remote servers and appropriate scripts.
- Softphones or phone simulators (the latter are used in the Single Server Lab only, where a simulator version of CM is installed)
- Software web development tools in the form of an Integrated Development Environment (IDE)
- Client test tools

The tools delivered with the SDK should be understood as a recommended selection.
Other tools with equal or similar functionality are available for free or as commercial ones.

Feel free to work with your favorite tools if they match the SDK needs.

AVAYA provides support for the tools and configurations included in the SDK but may not support other tools you may choose to use.

3.2 How to set up the Workstation

There are no specific requirements for workstation hardware.
All modern hardware should have sufficient performance to run a Java-based development environment.

For preparation of the workstation, the following steps have to be performed:

- For the Single Server Lab infrastructure:
  Install OpenVPN-Client which maps the server-hosted virtual machines into the accessible IP address space of your workstation
  Install VMWare-Console which enables you to start/stop server components or to login to them if you want to modify configurations.
- Install Putty and WinSCP.
  Putty is a command-line- and WinSCP a graphical SFTP-client. They facilitate file and folder operations on a remote server (copy to/from, delete, modify).
  The tools are used for deployment and modification of software on the 1X-Server.
- If ADS runs as a virtual machine and is equipped with a VNC server (that's the case for the Single Server Lab), you may want to install VNC client (for Windows XP only).
  VNC is a "remote desktop" product. It offers higher performance than access of the ADS operating system via the VMWare-Console.
  If you change user configurations in ADS from time to time, use of this tool may be convenient.
- If a Java Runtime Environment isn't yet installed, then install the one delivered with the SDK.
  You can take any JRE version 1.5.x ("Java 5.0") and higher.
  1.5 is recommended because it is the Java version used to run IBM WebSphere 6.1.
  You need a JRE to run the IDE.
- Install the Eclipse IDE delivered with the SDK.
  The IDE is the "Galileo" version (3.5) of Eclipse bundled with a suite of plugins called the Web Tools Platform (WTP). WTP adds JEE support to Eclipse.

- Install some softphones, one on your main development workstation and some others on additional workstations.
  For the virtualized Single Server Lab environment, the OpenVPN client has to be installed on additional workstations too.
  For this type of lab, you have the possibility to run simulated phones on your main workstation. If you want this, then install the DADS Console.

  Your test environment should at least comprise 3 softphones and 1-2 additional phones (e.g. DADS simulated phones)!
  If desired, you are able to connect SIP-based hardphones to the Single Server Lab (not preconfigured).

- Decide which client test tools you want to use.
  It is mandatory to have a powerful JavaScript debugger. See Chapter 3.5.2 for recommendations.

3.3 Tool Installation and Usage

All mentioned tools are available in the subfolder **/Tools** of the SDK distribution.

3.3.1 OpenVPN

You need this tool in the fully virtualized Single Server Lab only!

OpenVPN is a free VPN tool (http://openvpn.net) for creation of server-to-multiclient encrypted tunnel connections.
The virtualized Single Server Lab runs an OpenVPN server; any workstation which wants to access the server has to install an OpenVPN client.

Perform the following steps:

- On your workstation: Ensure that you are able to install drivers which are not certified (see *YourWorkstation/Properties/Hardware/Driver Signing*):
- Install OpenVPN:
  Deselect installation of the OpenVPN service during the installation process.
- Go to the installation folder of OpenVPN (e.g. */Program Files/OpenVPN*) and add subfolders "config", "keys" and "log".
- In SDK distribution, the OpenVPN folder contains subfolders "config" and "keys". Copy folder content from there in the added folders on your workstation.
- Open the file *vsil.ovpn* in subfolder */config* of your OpenVPN installation with a text editor.
  Set the lab-server's IP-address in the attribute *remote*.
  Set one of the available certificate- and key- files (see folder */keys*) in attributes *cert* and *key*.
  Note: Each workstation which accesses the lab server must have its own unique certificate/key pair. Otherwise, the connections interfere with each other.
- On your workstation: Examine *Settings/Network Connections* and ensure that a driver "TAP-Win32 Adapter V9" is installed.
  If the TAP-driver is missing: Install it by starting *../Program Files/OpenVPN/bin/addtap.bat*
- You are done.
- Start OpenVPN via *../Program Files/OpenVPN/bin/openvpn-gui-xxx.exe*, right-click on the taskbar icon and select *Connect*.

3.3.2 VMWare Console

You need this tool in a fully or partially virtualized lab only!

Install the software, there are no options.

If you start the VMWare console, you have to provide the lab-server's IP address, a valid login name and a password.
When logged in, you can see the running virtual machines and you are able to control them.

A click on one of the VMs opens the virtual machine's window.

Click into this window in order to enable it for reception of mouse and keyboard events.

For MS Windows™ based VMs it might be necessary to send "Ctrl-Alt-Del" to the window. This can be done via the "VM" menu of the VMWare console.

If keyboard/mouse control is set to a VM window, it can be switched back to the VMWare console using "Ctrl-Alt".

3.3.3 Putty

The command-line oriented SFTP client PUTTY (www.chiark.greenend.org.uk/~sgtatham/putty) will be needed to

- copy files to/from 1XS, especially to the WebSphere A/S
- call scripts (application installation/deinstallation) on 1XS
- login to CM for configuration purposes

Install PUTTY using all default settings.

Usage example (command-line access to CM):

- In the fully virtualized Single Server Lab, enable the OpenVPN connection first
- Start PUTTY
- Select the category "Session"
- Enter the CM server's IP address (see Chapter 2.1.2), Port=22 and Connection-type=SSH.
- Click "Open"
- Enter login-name and password (see Chapter 2.1.2)

Notes:

- Session settings can be stored for reuse
- PUTTY allows to set colors (foreground/background) per session

3.3.4 WinSCP

The graphical (Explorer-like) SFTP client WinSCP (http://winscp.net) will be needed to

- explore folders/files on the 1XS, especially the WebSphere A/S
- copy files to 1XS, change file access permissions, etc.

Install WinSCP using all default settings.

Usage example (explore files on 1XS):

- In the fully virtualized Single Server Lab, enable the OpenVPN connection first
- Start WinSCP
- Select "Session"
- Enter the 1XS (1XP) IP address, login-name and password (see Chapter 2.1.2), Port=22 and Protocol=SSH.
- Click "Login"
- In the left window of WinSCP click to open folder /opt/IBM/WebSphere/AppServer61/profiles/default/installedApps/onexpNode01Cell to see the enterprise applications installed on the WebSphere A/S.
- Select "Disconnect" in menu "Session" to disconnect from 1XS.

Notes:

- Session settings can be stored for reuse

3.3.5 VNC Client

You may want to reconfigure settings in the ADS database and therefore want to login to the ADS server.
In case of the Single Server Lab, login can be done using the VMWare console, but there is a solution (optional) which provides better performance.
Because the ADS server is equipped with VNC server desktop sharing software (www.realvnc.com), you are able to view the server's desktop via VNC Viewer.

Note: The free version 4.1 of VNC inluded in the SDK is limited to Windows versions XP and Server 2003.

Install VNC and deselect VNC Server in installation options.

Use VNC by starting the viewer (see Chapter 2.1.2 for ADS IP-address, VNC password and authentication information).

3.3.6 Phones

At least 4 phones should be available in the lab infrastructure, which can be

- Softphones installed on the main development workstation and additional workstations
- Hardphones (SIP only, not preconfigured)
- Simulated phones in case of the Single Server Lab which comes with a CM simulator.

For the Single Server Lab, it is recommended to have 2 additional workstations and install a softphone on your development workstation and each of the additional workstations.
In the Single Server Lab, more phones are available by using simulated phones.

Limitations of the phone setup:

- **Sometimes unpredictable behaviour of simulated phones (e.g. unexpected connection end).**
- **Missing voice connectivity of simulated phones.**
- **Simulated phones cannot be used in bridge conferences (Meeting Exchange).**

- **One-X™ Portal Client and a softphone cannot simultaneously run on the same workstation, but the "Dashboard" demo application and a softphone can run on the same workstation.**

3.3.6.1 Softphones

SDK tools contain the **AVAYA IP Softphone**.

In the virtualized Single Server Lab infrastructure, each workstation running a softphone has to be equipped with an OpenVPN connection. Therefore OpenVPN has to be installed first.

Installation and configuration of the AVAYA IP Softphone R6:

- Start the installation wizard
- On the Feature Selection page select "IP Softphone"
- On the Sub-Feature Selection page: deselect all check boxes
- Finish installation
- Open "Telephone and Modem Options" in the System Configuration of your operating system
- On the tab for dialing rules: Add a location (arbitrary)
- On the extended tab: Add the "Avaya IP Service Provider"
- Start the softphone, the configuration wizard will start
- Login configuration: "Login to AVAYA call server"
- Set a station call number and the password (e.g. call-nr=32135/ password=1234, see Appendix 6.1, you can use any of the phones of type **4620**)
- Set primary server address to the IP address of CM (no secondary server)
- Select the dialing location set up before
- Agree to the emergency call handling feature (emergency calls sent to same station)
- Set "Road Warrior" mode and LAN bandwidth
- Follow the audio tuning wizard
- You are done, login to the call server is possible now

3.3.6.2 DADS Console

You may need this tool in the fully virtualized Single Server Lab only!

The DADS console (Definity ACD Simulator) is a tool for control of the CM simulator. Its main purpose is generation of traffic load, a feature which isn't needed for the SDK.

The console offers the ability to open multiple UI windows for simulated phones.

The DADS console is a Java application, you need to have a Java Runtime Environment installed. There is a JRE version 1.5 in the SDK distribution, install it first.

The DADS console consists of a JAR library *vfmgui.jar* and a Batch file for start, put them in any directory of your choice.
Open the Batch file with an editor and ensure that the path to the JRE is set correctly.

Start the console by running the Batch file. A windows console window and the DADS console will be opened.

! Attention!:
Never close the DADS console via button "Stop DADS" or "Stop DADS" in menu "Control" - this would stop the simulator on the server!
Always end the console by clicking the "Close" button in the upper right corner of the DADS console window.
If you inadvertently have stopped the simulator, you have to login to the CM server (using PUTTY) and restart the simulator there by calling "startdads".

3.3.6.3 Using DADS simulated phones

To start the UI of a simulated phone, select *8434* in menu *Phones* of the DADS console and enter an extension number (PN number remains empty). Use any of the call numbers **400xx** (see Appendix 6.1).
You can start multiple simulated phones.

The UI of a simulated phone looks like this (main parts only, don't care for the buttons on the right side):

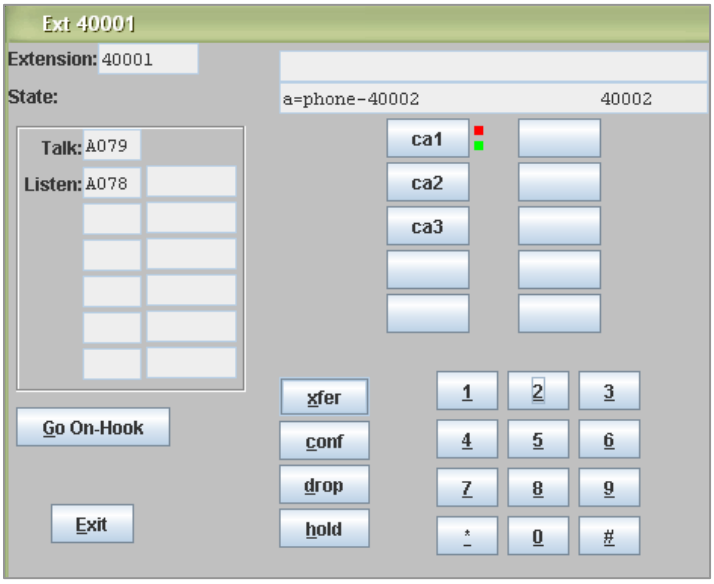

FIGURE 3: DADS simulated phone

How to use it:

- **Make a call:**
  Click on one of the call appearances **cax**. Click dialpad buttons to enter a call number.
- **Receive a call:**
  A green flashing signal next to a call appearance button signals an incoming call.
  Click the **cax** button to answer the call. Timeslot information in the **Listen** field signals that a voice connection is set up.
- **End a call:**
  Click the button **Go On-Hook**.
- **Transfer a call:**
  Make a call via, let's say, **ca1** and wait for answer.
  Click **xfer** and another call appearance becomes active (e.g. **ca2**): Dial the call number of the extension to be transferred to and wait for answer.
  The green signal on **ca1** blinks fast.
  Click **xfer** again to complete the call transfer.
- **Conference calls:**
  Make a call via, let's say, **ca1** and wait for answer.

Click **conf** and another call appearance becomes active (e.g. **ca2**): Dial the call number of the 3*rd extension for a conference and wait for answer.
The green signal on **ca1** blinks fast.
Click **conf** again to set up the conference.

Notes:

- **If button text on a simulated phone UI changes to something like "l…", the phone is no longer synchronized with the server. It should be closed (button "Exit") and opened again.**
- **Simulated phones cannot be used in bridge conferences (Meeting Exchange).**
- **Simulated phones cannot be used if a Media-Gateway is connected to the fully virtualized Single Server Lab. In this case simulated phones cannot call real softphones and vice versa.**
  For CM experts: The Media-Gateway can be set out of order by applying the command "release board 1a04" in CM's SAT window (System Administration Terminal). This may be reverted by "busyout board 1a04".

3.3.7 Voice Recording Software

For voice messaging an audio recording tool (recording into *.wav- files) might be needed.
You may use your favorite recording tool or one included in the SDK.

Any type of voice file format can be send to another user's voice mailbox as message attachment.
But, in order to be able to playback voice messages via phone, the file format has to be:

Wav-file, sample rate 8 KHz, 8-Bit samples, CCITT µ-Law encoded

3.3.7.1 AVAYA Voice Player

If you want to record µ-Law encoded wav-files using the Voice Player, perform the following steps:

- Select *Options/Advanced* in the menu of the Voice Player and check *Force Recording to Modular Messaging MULAW Format* as well as *8 kHz Play/Record*.
- Click *Record* in the *Audio* menu.
- After having stopped the recording, select the *File/Save As...* menu and store the audio in a file with the extension **wav**.

3.3.7.2 Audacity

Audacity (http://audacity.sourceforge.net) is a freeware for audio recording and playback in a variety of formats.

If you want to record µ-Law encoded wav-files using Audacity, perform the following steps:

- Select *Edit/Preferences* in the menu of Audacity. On the *Quality* tab of the preferences page set the *Default Sample Rate* to *8000 Hz* and the *Default Sample Format* to *16-bit*.
  On the *File Formats* tab set *Uncompressed Export Format* to *Other../WAV(Microsoft),U-Law*.
  Set a folder for recorded files on the *Directories* tab.
- On Audacity's main page select "Microphone" as recording source and click the *Record* button to start a recording. Click the *Stop* button to end a recording.
- Select *File/Export As WAV* to store the recorded audio in the correct format.

3.3.8 Eclipse IDE and the WebTools Platform Plugins

For development of web applications, the SDK delivery contains the Java-based Eclipse IDE (**Eclipse "Galileo" version 3.5**, www.eclipse.org) bundled with a set of plugins for JEE development, called the Web Tools Platform (**WTP version 3.1**, www.eclipse.org/webtools).

In order to run Eclipse, you need to have a Java Runtime Environment installed.
You find a JRE version 1.5 in the SDK distribution, install it first.
Although any JRE 1.5 and higher can be used, the SDK comes with a 1.5 version because IBM WebSphere 6.1 runs on Java 1.5 ("Java 5.0").

To install and run Eclipse:

- Take the "Eclipse_Plus_WTP" ZIP file and expand it in any folder of your choice, recommended location is *C:\Program Files\Eclipse350*.
- Set up a link (e.g. on the desktop) with the following target
  "C:\Program Files\Eclipse350\eclipse\eclipse.exe" -vm "C:\Program Files\Java\jre1.5.0._17\bin\javaw.exe"
  The additional parameter points to the default JRE used to run Eclipse.
- Start Eclipse via the added link.

3.4 1XS Filecopy and Installation Scripts

The Eclipse IDE is not able to manage (deploy, remove) applications on a remote WebSphere 6.1 A/S. Commercial IBM tools are necessary to do this. For applications targeted by the SDK, the IBM tools do not offer a real advantage over the Eclipse environment.

In folder */IDE/Scripts* of the SDK distribution you find some scripts for various application handling tasks.
Each of the scripts uses PUTTY or PSFTP (of the PUTTY suite) to copy files or to perform some operation on the 1XS server. The operation(s) are defined in form of a text file in folder */IDE/Scripts/commands*.
Some of the commands start scripts on the server. You find these server scripts in */IDE/Scripts/Server-Scripts*.

All scripts are prepared for management of the SDK demo application, the "Dashboard".
The server-side scripts are already available in folder */opt/avaya/1xp/dashboard* on 1XS.

Scripts are:

- **install-on-server.bat, install-ear-command.txt**
  Install the Dashboard application on the remote WebSphere A/S by calling the remote shell script **run_was_operation.sh** which starts the remote script **install_1XDashboard.py**.
  For installation the Dashboard enterprise application **1X_SDK_Dashboard.ear** has to be available in the remote script folder.
  On 1XS, enterprise applications are installed in */opt/IBM/Websphere/AppServer61/profiles/default/installedApps/onexpNode01Cell*.
- **uninstall-on-server.bat, uninstall-ear-command.txt**
  Uninstall the Dashboard application on the remote WebSphere A/S by calling the remote shell script **run_was_operation.sh** which starts the remote script **uninstall_1XDashboard.py**.
- **restart-on-server.bat, restart-app-command.txt**
  Restarts the Dashboard application on the remote WebSphere A/S by calling the remote shell script **run_was_operation.sh** which starts the remote script **restart_1XDashboard.py**.

- **copy-ear.bat, copy-ear-command.txt**
  Copies the Dashboard enterprise application **1X_SDK_Dashboard.ear** to the remote script folder. The install script takes it from this location.
- **copy-files.bat, copy-files-command.txt**
  This script can be applied to a Dashboard application already installed on 1XS.
  It copies most of the files (web application files like *.jsp, *.js, *.css only) in WebSphere's Dashboard application folders. These files can be modified during runtime, the A/S has to deliver updated file content upon the next browser request.

Each of the Batch scripts starts either PUTTY or PSFTP and instructs them to perform a script contained in a *command.txt file. The command textfiles contain PSFTP commands or a reference to a server shell script.

Some useful tips:

- If, during application development, single jsp, js, or css files have to be updated, the most convenient way is to directly copy them to the remote WebSphere folder using WinSCP (drag and drop) or to use WinSCP's local/remote folder synchronization facility.
  Updates should be available after re-login to the client application.
- If updates are not recognized in the client browser, in most cases you have to clear the browser's cache. The most convenient way is to let the browser clear its cache automatically upon every restart (can be configured for e.g. Firefox).
- JAR libraries, compiled Java class files, or configuration files (deployment descriptors, properties files) cannot be changed during runtime. Update of Java libraries/classes require at least a restart of the enterprise application. For update of the deployment descriptors the application has to be removed and installed again.
- If, after having made manual updates in a running application (especially if folders were added), deinstallation of the enterprise application may partially fail (the application is deinstalled for WebSphere, but some of the folders/files are still not deleted).
  Remove these artifacts from WebSphere's enterprise application folder using WinSCP.
  If problems insist, look into
  /opt/IBM/Websphere/AppServer61/profiles/default/temp/onexp.vsil.localNode01/server1
  and delete remaining temporary application files there too.

How to adapt the scripts for your own application:

The workstation BATCH script files refer to the location of the application under development, which usually is an Eclipse "workspace", or to a location of an EAR file.
These source references have to be adjusted.

The workstation script command *.txt files either contain SFTP commands for file transfer "workstation ->1XS" or commands which start *.py scripts on the server via a remote shell-script *run_was_operation.sh*.
Look into the server shell script to see which parameters have to be set upon start of the script and adjust the workstation command *.txt files according to a modified application name, folder from where to install or to call a script on the server.
The *.py scripts should be renamed and modified. They contain mostly WebSphere standard scripting, only some lines at the end of each script file have to be adjusted.

Notes:

In order to be executable, the scripts on 1XS must have file permission code "0775" set.

The scripts contain fully qualified references to the local filesystem and therefore, without modification, work in their original position only.

3.5 Client Test Tools

3.5.1 Browser

Any modern web browser can serve as web application test client.
The SDK's "Dashboard" demo was tested using Internet Explorer versions 6 and 7 and Firefox versions 2 and 3.

The following **browser settings** are mandatory: **JavaScript enabled, Cookies enabled, no Popup Blocker**.

The Dashboard application tries to block unwanted key (e.g. F5 = reload) and context menu operations. On Firefox, the context menu can be disabled only, if this is explicitly allowed in extended JavaScript settings.

Some important notes:

- **If your workstations are configured to use a proxy: For the virtualized Single Server Lab, don't forget to disable proxy-usage for IP addresses 192.168.17.***
- **Due to the different session handling of IE and FF, a 1X web application can only be started once on a workstation using Firefox but multiple times using Internet Explorer.**

3.5.2 JavaScript Debugger

A 1X web application consists of client-side JavaScript to a fairly wide extent.

You definitely need a powerful JavaScript debugging tool for development.

A very good one is the **Firebug** plugin for Firefox.

The Firebug plugin can be obtained from http://getfirebug.com.

4 Server Administration Tools

Each of the server modules in the 1XS Unified Communications infrastructure comes with its own configuration tools.

It is beyond the scope of the SDK documentation to introduce all tools in depth.

In the AVAYA-provided lab, server interoperability data and basic user data are preconfigured.

For preparation of some specific tests, it may be useful to modify the basic user configuration.

This introduction to the most important server configuration tools intends to provide the basic knowledge needed. Additional information is available in the online documentation of the tools.

4.1 AD Database Administration

1XS uses an ADS LDAP database for storage of enterprise users and their authentication info.

To review or change the configuration, you have to login to the ADS server via VMWare console or VNC viewer (see Chapters 2.1.1, 2.1.2 for login information).

The ADS database can be entered via
Programs/Administrative Tools/Active Directory Users and Computers.

In the Single Server Lab configuration, you will find users **onexpuser1** … **onexpuser8**, they all belong to the Security Group "**onexuser**" (see Appendix 6.1).

Notes:

- The security group "**onexuser**" was preset upon installation of the 1XS system, therefore this user group is mandatory for all 1XS deployed users.
  Modification of the user group name in ADS would break the configuration!
  On a standard 1XS system the user security group is "1XP Users".
- Changes in the ADS database (except changes of the authentication information) will not be recognized by 1XS without a synchronization. Repeated synchronization is done by 1XS on a regular basis. The sequence can be adjusted by the 1XS Admin Client.
  This client allows triggering of an immediate synchronization.

4.2 WebSphere Console

The WebSphere Console is the Administration Web Application of the IBM WebSphere A/S, see Chapter 2.1.2 how to login.

The most interesting section in the WebSphere Console is application management.
Select *Applications/Enterprise Applications* in the left side task list to enter application management.
Here you can see all deployed enterprise applications and you are able stop, start or configure them individually.

The Console offers the ability to install an enterprise application.

Although the SDK recommends using scripts for application installation (especially if installations are performed frequently), here is a brief introduction to how installations may be done by the WebSphere Console (for the Dashboard demo application as example):

- On the *Applications* page, click *Install* and wait for the next page

- Select *Local file system* and search the EAR file to install in the local workstation file system (e.g. 1X_SDK_Dashboard.ear)
- Click *Next* and wait for EAR upload to 1XS
- Accept default settings on the *Installation Options* page and click *Next*
- On the *Map Modules To Servers* page, select the single web application "1X-Dashboard-war" contained in the EAR and click *Next*
- Accept the *Summary* page and click *Finish*
- Save changes to the WebSphere master configuration on the next page
- Back on the *Applications* page, select the added "1X_SDK_Dashboard" application and click *Start*
- The installation is complete

4.3 MM Admin Client

The MM Admin Client is a web-based administration tool of AVAYA'S Modular Messaging service. See Chapter 2.1.2 for login information.

The areas of most interest (see section **Messaging Administration** of the menu on the left side) are **Subscriber management** and **Classes-of-Service**.

If you select *Subscriber management* and then click *Manage* for *Local Subscribers*, you get the current list of subscribers and their mailboxes.
Select one of them and click *Edit the Selected Subscriber* to see the subscriber's main configuration data:

- Name
- Password
- Mailbox Number
- Associated Extension
- The Class of Service
- The Email Handle (the MSS mail address)
- Mailbox locking flag

The Class of Service defines the mailbox behaviour (e.g. the storage limit for voice mails in recorded minutes or the duration how long messages are retained on the server).
It can be reviewed via menu *Class-of-Service*.

Note:

- Changes in the MM will not be recognized by 1XS without a synchronization. Repeated synchronization is done by 1XS on a regular basis. The sequence can be adjusted by the 1XS Admin Client.
  This client allows triggering an immediate synchronization.

4.4 1XS Admin Client

The most important tool for 1XS administration is the web based Admin Client. See Chapter 2.1.2 for login information.

The open Administration Client shows the following tabs:

- **Users**: The 1XS user configuration
- **Servers**: Configuration of the communication between 1XS and AVAYA services like AES, CM, MM, MX and IPS
- **Scheduler**: Definition of cyclic synchronization and backup tasks
- **System**: Configuration of the communication between 1XS and
  - ADS (Enterprise Directory)
  - License service
  - Database
- **Monitors**: Monitors the status of communication between 1XS and AVAYA services

From a user configuration point of view, the "Users" and "Scheduler" tabs are of interest.

Synchronization settings:

Click the *Scheduler* tab and select *Enterprise Directory Synchronization* to see how often and when 1XS synchronizes with the ADS server.

Click on *Modular Messaging Synchronization* to review MM synchronization settings.

Both ADS and MM can be synchronized immediately upon request.

Each time modifications are made in ADS or MM, you should synchronize 1XS to make the changes available there!

User configuration:

Click the *Users* tab, select *Portal Users* and then click *Search* to see all users provisioned in 1XS. By clicking on one of the users, he can be configured individually.

To be available on 1XS, each user first must be provisioned by performing the following steps:

- Add a user in the ADS database, he has to be member of a user group with same name as the preset (upon 1XS installation) 1XS security group (which is "**onexuser**" in case of the predefined Single Server Lab).
- Synchronize 1XS to ADS using the 1XS Admin Client.
- The new user now appears in the list of *Unprovisioned Users* in the Admin Clients user configuration and can be provisioned there.
- Provision him, that is: Set a group profile
- Configure the user

For each user a *Group Profile* has to be set. Group Profiles can be freely defined (don't mix up with the ADS group).

Configuration settings of a user are defined in a *System Profile*. Some of the System Profile settings can be overridden by a Group Profile and some of the resulting settings can be individually overridden for a single user.

1XS client applications have to pay attention to some of the user configuration settings!
The Dashboard application documentation of the SDK will address these issues.

If you select one of the provisioned users, his communication resources can be configured.
Resources are:

- A phone extension (CM)
- Voice Messaging accounts (AVAYA Modular Messaging)
- A Conference Bridge (AVAYA Meeting Exchange)
- Presence configuration (IPS)

1XS client applications have to pay attention to non- or improper configured communication resources and prohibit using them!

Typical client software testcases include checking the reaction if connectivity to one of the AVAYA services is lost. You may use the "Monitors" tab to simulate the cases.

Suspend/Resume service connections:

Suspending the service means that the connection between 1XS and the service is shut down (not the service itself).

5 Working with the Eclipse IDE

Eclipse is a powerful development platform. Lots of free or commercial plugins for various tasks are available.

For the SDK, Eclipse was bundled with a suite of JEE development plugins called the Web Tools Platform (WTP).

For deeper Eclipse knowledge, please refer to online documentation, tutorials and existing literature. This document mainly focuses on Eclipse functionality relevant for 1XS-based web applications.

5.1 Eclipse IDE Basics

5.1.1 Workspace preparation

At the first start of the Eclipse IDE a welcome screen appears. Please close it to see the Eclipse Workbench.
If Eclipse prompts to select a workspace folder, then read the following first.

Eclipse handles application projects in the form of Workspaces. A **Workspace** is a collection of Java packages or projects stored in a folder of the file system. This folder contains additional Eclipse-specific configuration information.

The SDK comes with a 1X client demonstration application called "**Dashboard**". In order to explore the source files of the application, you need to make them available in the IDE by import the source files into an existing workspace. Thus, you have to set up a workspace first.

Although being free to select any workspace folder, there are reasons to use a recommended folder structure:

- Scripts for Dashboard filecopy, installation and deinstallation on 1XS do not work at an arbitrary position in the filesystem without modification.
- This documentation refers to subfolder names.

The recommended SDK work folder is: ***C:\1X-WebApp-SDK***.

Prepare the folder as follows:

- Add a subfolder ***IDE***. It will be the Eclipse workspace folder.
- Add a subfolder ***IDE-Export***. This will be the target location for export of EAR and WAR files from Eclipse.
- Copy the subfolder ***IDE/lib*** and its content from the SDK distribution the local SDK work folder.
- Copy the subfolder ***Scripts*** and its content from the SDK distribution to the local SDK work folder.

Set the workspace in Eclipse:

Select menu *File/Switch Workspace/Other..*, browse for folder *C:\1X-WebApp-SDK\IDE* and accept.

5.1.2 Perspectives

Eclipse provides the feature to display project content in different ways, called Perspectives.
To closer match a development task, a given perspective may present different tools than other perspectives.

As a Java IDE, the default perspective is the **Java Perspective**.
The Eclipse bundle delivered with the SDK is enriched with the WTP JEE plugins, therefore another perspective, the **JEE Perspective** is available.
Ultimately, it is up to the user to select his favorite perspective.

Upon import, it might be that Eclipse selects a perspective on its own. This can be misleading if import results are interpreted as different due to the selected perspective.

In order to avoid ambiguities, this documentation refers to the JEE Perspective only.

You can select a perspective at any time in the upper right corner of the Eclipse workbench.

FIGURE 4: Eclipse Perspective switching

If the JEE ("Java EE") perspective switch still isn't available there, then select it via menu *Window/Open Perspective/Other...*

5.1.3 IDE Preferences

Before working with the IDE, some basic preferences should be set. **Preferences are specific to a workspace!**

You can open the preferences dialog via menu *Window/Preferences*.

Preferences are mostly are matter of personal taste, but it can be annoying to look into sources which were created having e.g. different editor preferences set.

Therefore a list of editor preference settings used during the Dashboard application development is included here.
In the preferences dialog, select General/Editors/Text Editors. Dashboard settings were:

- Displayed tab width = 2
- Insert spaces for tabs = yes

Select General/Editors/Text Editors/Spelling. Dashboard settings were:

- Enable spell checking = false

Dependent on the Eclipse version and the type of a file, editor tab settings may not properly work, a fixed policy is used by the IDE instead.
Especially for Java source files, there is another concurrent adjustment. Select *Java/Code Style/Formatter* to examine the current formatter profile. A new one may be added to introduce a new tab policy.

Many of the workspace-specific preferences can be overridden for a single project in the project's properties.

5.2 Application Import

There are several, more or less simple, ways to do this.

- If available, the simplest one is to import existing Eclipse projects.
- Another way is the import of an existing application EAR file. This type of import has to rely on runtime data only, project specific information known during development is unavailable and must be restored manually.

The import process will be demonstrated for the **Dashboard** demo application packaged with the SDK.

Although it is recommended to import the Dashboard by using the first method, a description of the second method is provided also.

5.2.1 Import of existing Projects ("Dashboard")

Locate two (2) projects in folder ***IDE*** of the SDK distribution:

- **1X-SDK-Dashboard**
  This is an Enterprise Application EAR project, which represents the envelope for a single web application.
- **1X-SDK-Dashboard_war**
  This is a Web Application WAR project. The WAR file built from this project is the web application contained in the EAR project.

Copy both distribution folders and their complete content into the folder ***IDE*** of your local workspace.

Import the projects:

- Select menu *File/Import*.
- In the following dialog, select *General/Existing Projects into Workspace* and click *Next*.
- In the following dialog, check *Select root directory* and browse for the workspace folder which is *C:\1X-WebApp-SDK\IDE*.
- Both projects copied to this folder will now be visible in the dialog.
  Uncheck *Copy projects into workspace*.
- Check **both** projects for selection, then click *Finish*
- The projects will be imported and validated.
- If the perspective selected by the IDE is not **Java EE**, then set this perspective (see instructions above).

The Project Explorer window on the left side of the workbench should look like this now.

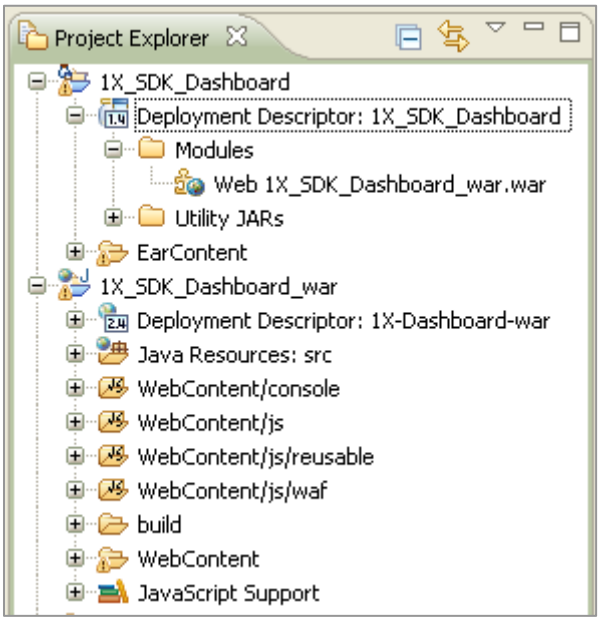

FIGURE 5: "Dashboard" project layout

Please ignore red, quadrantal, white-crossed error icons or warning icons (exclamation mark) which may decorate some of the project folders. They signal validation errors which are caused by imperfect JavaScript validation. A later Chapter will discuss this issue.

A couple of things might have failed during import and need correction:

- Your workspace folder does not match the workspace folder of the imported WAR project. The reference to additional libraries used at development time only is incorrect.
  In the Project Explorer: Open the *Java Resources/Libraries* node of the WAR project and check the path to the library *websphere_apis.jar*.
  If it is incorrect, right-click the entry, select *Build Path/Configure Build Path* in the context menu to open the projects properties dialog. Correct the library path using the *Edit* function.
  Note:
  The *websphere_apis.jar* is a Java library which represents the WebSphere 6.1 system API. The library is added in 1X client projects to enable Eclipse to resolve references to the Java **Servlet API** in JSP web pages. This JAR file will not be added to the final application WAR file (the Servlet API is basically available on a JEE A/S).

FIGURE 6: Java Built Path libraries

- The Java Runtime Environment referred in the imported project does not have a matching version or installation folder.
  In the Project Explorer: Open the *Java Resources/Libraries/JRE System Library* node of the WAR project and check the path to the various JRE libraries.
  If they are incorrect, right-click on *JRE System Library*, select *Build Path/Configure Build Path* in the context menu to open the projects properties dialog. Correct the JRE path using the *Edit* function.

- It may happen (rare) that Eclipse removes or modifies the EAR projects's reference to the WAR project in the EAR's deployment descriptor file **application.xml**.
  In the Project Explorer: Open the node *Deployment Descriptor:.../Modules* of the EAR project. A reference to the WAR project should be visible there.
  This reference is caused by some content in the EAR deployment decscriptor file. You may double-click the deployment descriptor node in the Project Explorer to open the file for editing.

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://java.sun.com/xml/ns/j2ee"
  <display-name>1X_SDK_Dashboard</display-name>
  <module id="WebModule_1XSDashboard">
    <web>
      <web-uri>1X_SDK_Dashboard_war.war</web-uri>
      <context-root>/dashboard</context-root>
    </web>
  </module>
  <security-role id="SecurityRole_1XSDashboard">
    <description>Security Role for Users of 1XS</desc
    <role-name>1XP_User_Role</role-name>
  </security-role>
</application>
```

FIGURE 7: WAR project reference of the EAR project

If the *<module>* node in the file is missing or if it was modified during import, the file should be corrected.

- It may happen (rare) that Eclipse did not resolve the references between EAR and WAR projects correctly.
  In the Project Explorer: Open the *Java Resources/Libraries/EAR Libraries* node of the WAR project. You should see a lot JAR files in this folder which can also be seen in the node *EAR Content* node of the EAR project.

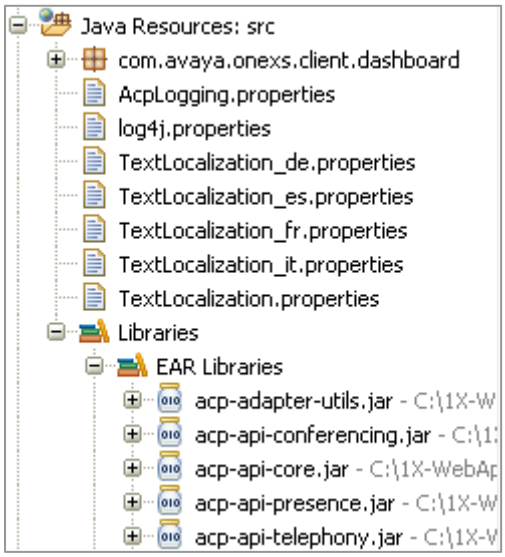

FIGURE 8: EAR libraries referenced by the WAR project

If not:

Select the EAR project's root node in the Project Explorer, right-click it and select *Properties* in the context menu to open the project's property dialog. Select *Project References* and ensure that the reference to the WAR project is selected.
Do the same for the WAR project to ensure that the reference to the EAR project is set.
Close both projects: Right-click their root node and select *Close Project* in the context menu.
Close Eclipse and start it again.
Right-Click the EAR project's root node and select *Open Project* in the context menu. Accept to have referenced projects opened too.
Wait until the project validation is complete and check the EAR library references in the WAR project again.

If the references still are incorrect, remove both projects from the workspace (in the Project Explorer: select each of the projects and choose *Delete* in the context menu) without deleting the physical files and repeat the import.

If the references still remain incorrect, check the JEE Module Dependencies of the EAR Project.
In the Project Explorer: Right-Click the EAR project's root node and select *Properties* in the context menu. Select *Java EE Module Dependencies* in the Properties dialog and make sure that in the list of JEE modules the module **1XS_SDK_Dashboard_war.war** is selected.

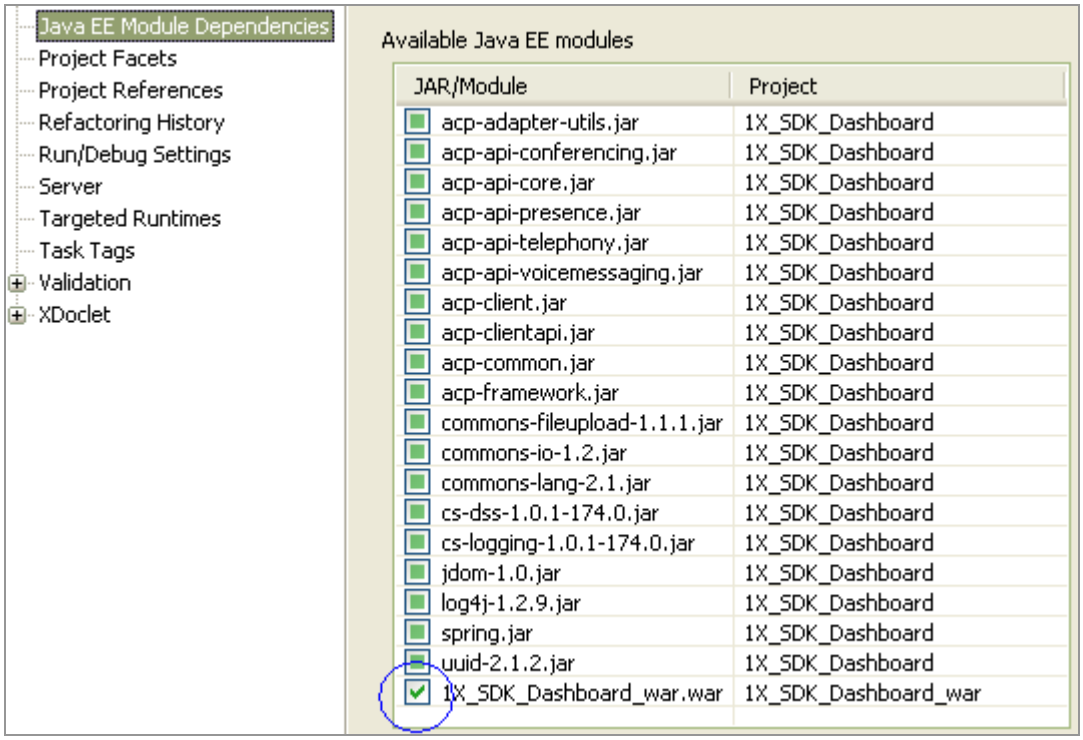

FIGURE 9: WAR module reference of the EAR project

Note:
The WAR project accesses some of the JAR libraries which are part of the EAR project. This has to be declared in order to enable the IDE to include the libraries upon generation of the final EAR archive.
The WAR project's EAR library references are defined in file *WebContent/META-INF/MANIFEST.MF* of the WAR project.

For the JEE perspective, in the right bottom area of the workbench, a tabbed window (Problems, Tasks, Properties, Servers) shows up.
This window ("Problems" tab excepted) is not of interest for 1XS client application development and can be minimized (use the "Minimize" icon in the window's upper right corner).
If done, some icons appear in the lower right workbench corner. They allow open/close operations on each of the mentioned tab windows separately.

5.2.2 Import of an Enterprise Application EAR ("Dashboard")

Another way to get an existing enterprise application into the IDE is to import the application EAR file. As already mentioned, this type of import has to rely on runtime data only. Project specific information known during development is unavailable and has to be restored manually.

Import:

- Start the IDE and ensure that the correct workspace is selected (if multiple workspaces are defined).
- Select menu *File/Import*, open the *Java EE* node in the following dialog and select *EAR file*.
- In the following dialog, browse for the Dashboard's EAR file **1X_SDK_Dashboard.ear**. It is stored in folder *IDE-Export* of the SDK distribution.
- There is no need to enter the next dialogs using the *Next* button. They show that the IDE automatically selects the EAR internal WAR file **1X_SDK_Dashboard_war.war** for import

as a separate project. The JAR libraries inside of the EAR file will not be imported as separate projects. This is the correct setting.

- Click *Finish* to import.
- After import and validation, you should see two (2) projects in the Project Explorer (JEE Perspective) which look quite similar to those created by the project import described before.

The application is completely available now in the IDE. You should be able to recreate the EAR file from the project's content.

But there are some differences compared to the results of a projects import:

- The Dashboard WAR project contains a single Java class **DashboardVersion.java** in the Java classpath *com.avaya.onexs.client.dashboard*. It simply contains the Dashboard application's version info.
  For the imported projects, the source code of this class is available in the node *Java Resources: src* of the WAR project (Project Explorer). The IDE compiles it during its build process and the final WAR file (inside of the EAR file) contains the compilation product **DashboardVersion.class**.
  Import of the EAR file can restore the "class" file only. In the Project Explorer, the file can be seen in node *Java Resources/Libraries/Web App Libraries/Imported Classes*.

  If you want to add the missing source again (there is no real need to do so):
  Right-click the node *Imported classes* and delete it via the context menu.
  Take the Windows Explorer and open the WAR project folder in the workspace directory.
  You will find a folder *src\com\avaya\onexs\client\dashboard* there, which is empty.
  Take the file **DashboardVersion.java** from the SDK distribution (WAR project, same folder) and copy it to the local workspace folder mentioned above.
  Go back to Eclipse, select the WAR project in the Project Explorer and hit *F5* for refresh.
  The missing source file will be available now.

  To get the new source compiled immediately:
  Open menu *Project*, unset *Build Automatically*, then click *Build Project*.

- If you open the "Problems" page (if minimized, click the icon in the lower right workbench corner), you will see problems like this.

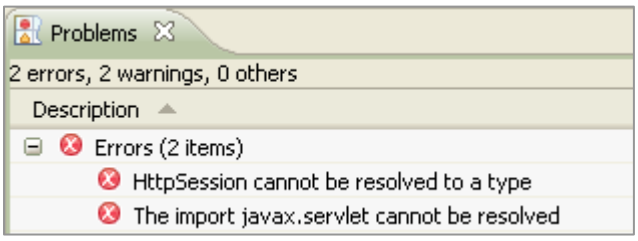

FIGURE 10: Unresolved Servlet API problem

They are caused by unresolved Servlet API references.

If you want to resolve these missing references:
Recall that your workspace was already prepared with an additional folder *IDE/lib* which contains the file *websphere_apis.jar* (see Chapter 5.1.1).
Right-click the WAR project in the Project Explorer and select *Properties* in the context menu.
Select *Java Build Path* in the left window of the dialog and select the *Libraries* tab in the right window.
Click *Add External JARs* and browse for the file *websphere_apis.jar*.
Close the dialog via *OK* button.

The validation can be repeated manually:
Right-click one of the projects in the Project Explorer and select *Validate* in the context menu.

Note: The file websphere_apis.jar should not be exported upon WAR file generation.
Therefore, on tab *Order and Export* of the *Java Build Path* dialog, it must not be checked.

- There is a small amount of validation problems only ("Problems" page), but, if you look in one of the WAR projects JavaScript source files (double-click e.g. *WebContent/console/console.js* to open it in the editor), a lot of "Unresolved" errors show up.
  This is caused by imperfect JavaScript validation capabilities of the IDE (discussed in Chapter 5.2.6). There are ways to improve things, which automatically get configured using the project import method described before.

5.2.3 Renaming of a 1XS Application

If you want to take the Dashboard demo application as a starting point for building a new application, then you should:

- Rename it
- Modify it to be an application which can be installed and run on 1XS in parallel to the Dashboard application

Let's say, the target is to create an application **1X_DemoApp** which can be accessed via URL **http://<1xs-ip-address>:<1xs-port>/demoapp**.

To achieve this, perform the following steps:

- Import both Dashboard projects into your workspace as already described before.
  The next steps will modify the projects. That doesn't matter, after having renamed the projects, the original Dashboard projects can be imported in the same workspace again (e.g. for having an unchanged/baseline reference).
  Another solution would be to establish a separate workspace.
  Upon import: Wait until import and validation is complete (see progress signal in the lower status-bar of the workbench).
- Ensure you have the **Java EE** perspective of the IDE selected.
- Right-click the WAR project (**this one first!**) in the Project Explorer and select *Refactor/Rename* in the context menu. Set the name to **1X_DemoApp_war** and click *OK*.
  Wait until import and validation is complete (see progress signal in the lower status-bar of the workbench).
  Again: Wait until validation is completed.
- Right-click the EAR project in the Project Explorer and select *Rename*. Set the name to **1X_DemoApp** and click *OK*.
- Right-click the renamed WAR project and select *Properties*. In the properties dialog: Select *Project References* in the left window. In the right window: Uncheck *1X_SDK_Dashboard* and check *1X_DemoApp*. Then click *OK*.

The basic steps are done. Now deployment descriptors of both projects must be adjusted.

For a more detailed description of the deployment descriptor files, refer to the SDK's "Dashboard" documentation.

The JEE standard deployment descriptor XML file of an EAR application is **application.xml**, it is located in subfolder *META-INF* of the enterprise application (*EarContent/META-INF* in the IDE).
IBM-specific extensions to the standard descriptor file are:

- **ibm-application-bnd.xmi**
- **deployment.xml** in subfolder
  ibmconfig/cells/defaultCell/applications/defaultApp/deployments/defaultApp

  - In the IDE's Project Explorer: Double-click **application.xml** to open it.
    The values used in the following modifications should be understood as example, you are free to use others.
  - Set the id of the *<application>* (second line, scroll right) to **id="Application_1X_DemoApp"**.
    Set the application's *<display-name>* to **1X_DemoApp**.
    Set the module's *<web-uri>* to **1X_DemoApp_war.war**.
    Set the module's *<context-root>* to **/demoapp** (the IDE might have set the name of the WAR project here).
    Set the id of the *<security-role>* to **id="SecurityRole_1X_DemoApp"**.
    Store the file.
  - Open **ibm-application-bnd.xmi** (don't care for the IDE's "Problems" message, the IDE can provide a standard text editor for xmi-files only).
    Adjust the authorization *<role>* to refer to the security role defined in application.xml:
    href="META-INF/application.xml#SecurityRole_1X_DemoApp".
    Adjust the *<application>* to refer to the application-id defined in application.xml:
    href="META-INF/application.xml#Application_1X_DemoApp".
    Store the file.
  - Open **deployment.xml** (only standard text editor provided by the IDE).
    Adjust the deployed object's module's URI to refer to the WAR module defined in application.xml:
    uri="1X_DemoApp_war.war".
    Store the file.

The JEE standard deployment descriptor XML file of a WAR application is **web.xml**, it is located in subfolder *WEB-INF* of the web application (*WebContent/WEB-INF* in the IDE).
IBM-specific extensions to the standard descriptor file are:

- **ibm-web-bnd.xmi**
- **ibm-web-ext.xmi**

  - In the IDE's Project Explorer: Double-click **web.xml** to open it.
    The values used in the following modifications should be understood as an example, you are free to use others.
  - Set the id of the *<web-app>* (second line) to **id="WebApp_1X_DemoApp"**.
    Set the web-application's *<display-name>* to **1X_DemoApp**.
    Store the file.
  - Open **ibm-web-bnd.xmi** (only standard text editor provided by the IDE).
    Adjust the *<webapp>* to refer to the application defined in web.xml:
    href="WEB-INF/web.xml#WebApp_1X_DemoApp ".
    Store the file.
  - Open **ibm-web-ext.xmi** (only standard text editor provided by the IDE).
    Adjust the *<webapp>* to refer to the application defined in web.xml:
    href="WEB-INF/web.xml#WebApp_1X_DemoApp "
    Store the file.

The modifications are already sufficient now. Nevertheless, one additional step should be done to remove "Dashboard" terms from the deployment descriptors completely.

- Select *File* in the IDE's *Search* menu.
- Ensure that the *File Search* tab is selected in the search dialog and search for **xmi:id** in all (name pattern = *) files in the workspace.
  You will find matches in files **ibm-application-bnd.xmi** and **deployment.xml** of the EAR project.
- Replace all the xmi:id- values so that they point to your application, e.g. RoleAssignment_1X_DemoApp, Deployment_1X_DemoApp, ApplicationBinding_1X_DemoApp, ….

The xmi:id- values have formalistic meaning only, they should be unique throughout an application.

The project modifications are complete.

You now have application projects with content identical to the Dashboard. If you generate WAR and EAR files from the projects, then the final EAR application **1X_DemoApp** may be installed on 1XS and co-resident with the Dashboard application.

5.2.4 Building Application WAR and EAR Files

How to create the final EAR application file for installation on the WebSphere A/S:

- Right-click the WAR project in the Project Explorer of the Eclipse workbench and select *Export/WAR file* in the context menu.
- Browse for the destination folder, e.g. *IDE-Export* in your SDK work folder and hit *Finish* to create the WAR file.
- Use Windows Explorer to copy the created WAR file into the folder *EarContent* of the EAR project.
- Again in the Project Explorer of Eclipse: Right-click the EAR project and select *Export/EAR file* in the context menu.
- Browse for the destination folder, e.g. *IDE-Export* in your SDK work folder and hit *Finish* to create the EAR file.

Now the EAR file is ready for installation on 1XS.

5.2.5 Installation of a 1XS Application on the 1X Server

As already mentioned, there are two (2) ways to install an application on 1XS:

- Use the WebSphere Console
- Use scripts provided with the SDK

For a quick first shot, the first method is the best one, see Chapter 4.2 for usage instructions.

During development, an application may be installed/uninstalled frequently. Therefore it is useful to automate these processes using the provided scripts (see Chapter 3.4).
In their original form, the scripts work for the Dashboard only and they rely on fixed file positions on development workstation and 1X server.
Some work has to be invested for adaptation of the scripts. Information needed is provided inside of the scripts.

Note:
Starting of scripts may be integrated in the Eclipse workbench.
Via menu *Run/External Tools/External Tools Configurations*… a dialog for external tool settings may be opened.

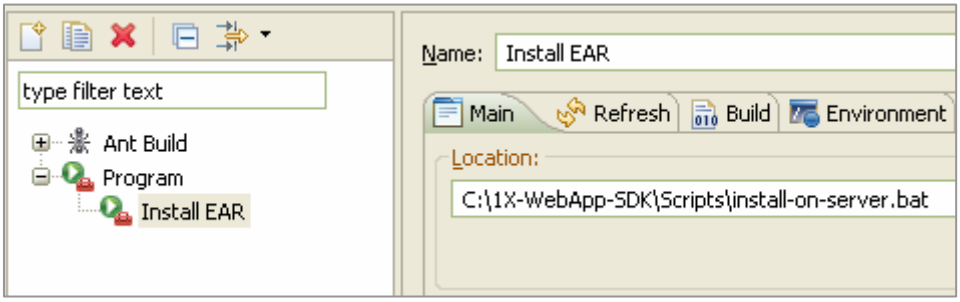

FIGURE 11: Referencing external tools

You are able to add calls to any external tool there for later start via the workbench's  toolbar button.

5.2.5.1 The 1XS User Security Group

Upon installation of a 1XS system, a user security group name is selected.
All users deployed on 1XS must belong to a user group of equal name in the ADS database.

Currently there is no easy way to modify the security group for an existing system, simply changing the group name in ADS is not a solution.

An EAR application for WebSphere contains the user group name in the deployment descriptor file **ibm-application-bnd.xmi** (*<authorization><groups… name=…>*).
WebSphere takes the group reference from the descriptor upon installation.

The group name is "**onexuser**" for the virtualized Single Server Lab, standard 1XS installations mostly have "**1XP Users**" set.

If the Dashboard application has to be installed on a 1XS system prepared for a user group name other than **onexuser**, the application's group name has to be adjusted (otherwise users would be unable to login).
Assuming that the Dashboard EAR application file content remains unchanged, there are two (2) ways to do the adjustment:

- Install the application using the provided installation scripts, they allow to set the matching user group upon installation (see parameters of the installation command at the end of the server script file **install_1XDashboard.py**).
- Use the WebSphere console for adjustment after having installed the application:
  Select *Applications/Enterprise Applications* from the menu and then click the installed application.
  In the following dialog click *Security role to user/group mapping*.
  Check the single role definition appearing in the next dialog and click *Look up groups*.
  Adjust the search string in the next dialog to match the user group you look for in ADS (use a search pattern like ***onex**** to find a group name like **onexuser**) and click *Search*.
  Select the correct group, remove the link to the old group, store data and update WebSphere's master configuration.

5.2.6 Handling JavaScript

Due to use of AJAX communication, 1XS client web applications are strongly JavaScript-oriented. Logical decisions and dynamic rendering of updated data in the browser-based UI have to be coded in this language.

Because of loose typing and the mixing of JavaScript with HTML (JSP-page script directives define available scripts at runtime), editor tools lack precise code validation and reference checking.

Object-oriented JavaScript coding (such as for 1XS clients) even increase requirements on tools (e.g., the Eclipse-based AST – Application Server Toolkit -- provided by IBM for the WebSphere 6.1 A/S does not provide any effective support for this activity).

The Eclipse IDE included in the SDK is bundled with the Web Tools Platform (WTP) plugins. JavaScript development tools of WTP are part of WTP's **Source Editing** subproject (www.eclipse.org/webtools/sse), which comprises the **JavaScript Development Tools** – **JSDT** (http://wiki.eclipse.org/index.php/ATF/JSDT).

Although JSDT is a very powerful plugin, it cannot overcome all the issues arising from projects with a large amount of JavaScript code organized in various files.
But, compared with a simple text editor, it is of outstanding usefulness.

If you look into the WAR project's properties and select the *Project Facets* dialog, an item *JavaScript Toolkit 1.0* appears.

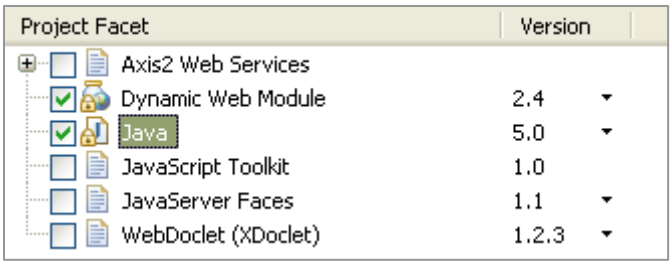

FIGURE 12: Project Facets

You will find that this facet is not enabled, but, in fact, it is (otherwise the section "JavaScript" in properties would not be available).

Attention: Don't enable this facet and disable it afterwards!
Some basic JSDT libraries may be removed from the project (not reversible).

5.2.6.1 Validation Limitations

Working with JavaScript in the IDE, problems may be reported upon validation and many errors/warnings may be signaled in the editor.
The causes of most of them are:

- Missing type recognition
- Unknown predefined objects or object attributes
- Inability to find referenced data, methods and functions in foreign files
- Suspicious, but legal, code constructs

Note:

Issues shown in the validation "Problems" window do not vanish upon code changes and new validation. It may be necessary to select/delete content in this window (via context menu).

It is up to the developer to decide which of the criticized code fragments are real errors and which are not.

There are several means of optimization:

- The JavaScript validator may be partially configured.
  In the WAR project's properties, a dialog *JavaScript/JavaScript Validator/Errors/Warnings* dialog offers several options.
- To enable the JavaScript validation process and the editor to resolve references, JavaScript libraries and source files can be declared for global lookup.
  This can be configured in the *JavaScript/JavaScript Libraries* dialog of the WAR project's properties.
  The following setup was selected for the Dashboard development:

  Folders with reusable libraries were declared as "Libraries".

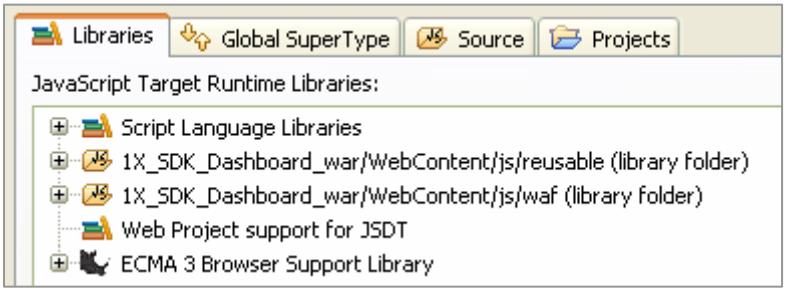

FIGURE 13: JavaScript Runtime Libraries

Other folders with JavaScript content referenced by many scripts were declared as "Source" (Note that the already declared library folders must be excluded if they are subfolders of declared source folders).

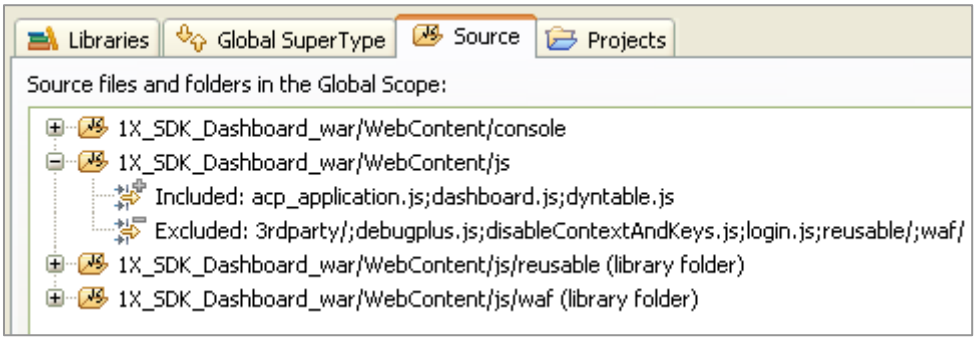

FIGURE 14: JavaScript Source References

6 Appendix

6.1 SDK-Lab: Phones, Accounts, Passwords ...

The following configuration can be found on the fully virtualized Single Server Lab only.

User-Group in ADS (= Security Group in WebSphere): **onexuser**

Call-Nr. of CM: **+1 73285**

Phones: (CM system name: **cmhandle**)

Call-Nr	Type	VOIP	CM-Name	CM-Passw.	CP	ADS/1XS-Name	ADS/1XS-Passw.	1XS-Group	ADS/1XS-Display-Name
32135	4620	y	onexpuser1	1234	1	onexpuser1	Interop123	lincroft	Harold Norton
32136	4620	y	onexpuser2	1234	1	onexpuser2	Interop123	lincroft	Jill McNeill
32138	4620	y	onexpuser3	1234	1	onexpuser3	Interop123	lincroft	Rudy Valentino
40010	8434D	y	onexpuser4	1234	-	onexpuser4	Interop123	lincroft	Doris Day
40011	8434D	y	onexpuser5	1234	-	onexpuser5	Interop123	lincroft	Steve Miller
40012	8434D	y	onexpuser6	1234	-	onexpuser6	Interop123	lincroft	Dorothy Young
32139	4620	y	onexpuser7	1234	1	onexpuser7	Interop123	lincroft	Eileen Rudden
32141 *1	4620	y	onexpuser8	1234	1	onexpuser8	Interop123	lincroft	Karen Lovett
32142 *2	4620	y	onexpuser1-2	1234	1	--	--	--	
32137	9630	y	Avaya 9630	1234	1	--	--	--	
40001	603E1	n	phone-40001	--	-	--	--	--	
40002	603E1	n	phone-40002	--	-	--	--	--	
40003	603E1	n	phone-40002	--	-	--	--	--	

CP = Coverage path
Coverage path **1**: 5 rings until coverage, 5 more rings until transfer to MM-mailbox via hunting-group **h4**

*1 32141 has an additional call-appearance which is bridged to the 1. call appearance of 32139
*2 32142 is set up for access of the second voicemail-account of onexpuser1 in Modular Messaging

Notes:

- 1XS can control VOIP stations only. Therefore the 8434D stations are configured as VOIP (although not being VOIP phones).
  1XS-Client can logon to all phones except 4000x.
- Call-Nr's 400XX are simulated phones, they work without having a physical phone or softphone connected to CM.
- Call-Nr's 40010 - 40012 have a special meaning in DADS. Therefore they sometimes show strange behavior like unwanted 'hangup'.
  This is not a real problem for 1XS test but should be known.
- Put call-numbers in E164-format (e.g. +73285<call-nr>) in ADS-database to get display-names for calls in 1XS
- **Attention:** Make sure that no two (2) users in ADS have the same phone-number set.
  Otherwise names in 1X- call-logs cannot be resolved!

Messaging Accounts: (MM system name: **mmhandle**)

Mailbox-Nr	Extension	Display-Name	MM-Passw.	COS	AD/1XS-User-Name	Messaging-Address
32135	32135	Harold Norton	2121	1	onexpuser1	hnorton@vmmss.vsil.local
32136	32136	Jill McNeill	2121	1	onexpuser2	jmcneill@vmmss.vsil.local
32138	32138	Rudy Valentino	2121	1	onexpuser3	rvalentino@ vmmss.vsil.local
32142	32142	Harold Norton	2121	1	onexpuser1	hnorton2@vmmss.vsil.local

Notes:

- Phone number of the mailbox is **32140** (for each user).
- Mailbox access via Hard- or Softphone (control by DTMF tone sending) requires a Media-Gateway (e.g. AVAYA G350)!
- **Attention:** Ensure that "Email-Handle" in MM-administration and "Email-Address" in ADS are not equal.

Conference-Bridges: (MX system name: **mxhandle**)

Bridge-Nr	Host-Code	Participant-Code	PIN	Moderator
8555	5678	1234	--	onexpuser1, onexpuser2

Note: Use **8502** to call into the bridge via phone (DTMF sending of the participant code requires a Media-Gateway)

Presence:

Server	SES ID	Password
presence	presence	presence