Planning for Avaya Aura® Experience Portal

April 2012

Notice

While reasonable efforts have been made to ensure that the information in this document is complete and accurate at the time of printing, Avaya assumes no liability for any errors. Avaya reserves the right to make changes and corrections to the information in this document without the obligation to notify any person or organization of such changes.

Documentation disclaimer

“Documentation” means information published by Avaya in varying mediums which may include product information, operating instructions and performance specifications that Avaya generally makes available to users of its products. Documentation does not include marketing materials. Avaya shall not be responsible for any modifications, additions, or deletions to the original published version of documentation unless such modifications, additions, or deletions were performed by Avaya. End User agrees 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 End User.

Link disclaimer

Avaya is not responsible for the contents or reliability of any linked Web sites referenced within this site or documentation provided by Avaya. Avaya is not responsible for the accuracy of any information, statement or content provided on these sites and does not necessarily endorse the products, services, or information described or offered within them. Avaya does not guarantee that these links will work all the time and has no control over the availability of the linked pages.

Warranty

Avaya provides a limited warranty on its Hardware and Software (“Product(s)”). 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 to Avaya customers and other parties through the Avaya Support Web site: http://support.avaya.com. Please note that if you acquired the Product(s) from an authorized Avaya reseller outside of the United States and Canada, the warranty is provided to you by said Avaya reseller and not by Avaya.

Licenses

THE SOFTWARE LICENSE TERMS AVAILABLE ON THE AVAYA WEBSITE, HTTP://SUPPORT.AVAYA.COM/LICENSEINFO/ ARE APPLICABLE TO ANYONE WHO DOWNLOADS, USES AND/OR INSTALLS AVAYA SOFTWARE, PURCHASED FROM AVAYA INC., ANY AVAYA AFFILIATE, OR AN AUTHORIZED AVAYA RESELLER (AS APPLICABLE) UNDER A COMMERCIAL AGREEMENT WITH AVAYA OR AN AUTHORIZED AVAYA RESELLER. UNLESS OTHERWISE AGREED TO BY AVAYA IN WRITING, AVAYA DOES NOT EXTEND THIS LICENSE IF THE SOFTWARE WAS OBTAINED FROM ANYONE OTHER THAN AVAYA, AN AVAYA AFFILIATE OR AN AVAYA AUTHORIZED RESELLER; AVAYA RESERVES THE RIGHT TO TAKE LEGAL ACTION AGAINST YOU AND ANYONE ELSE USING OR SELLING THE SOFTWARE WITHOUT A LICENSE. BY INSTALLING, DOWNLOADING OR USING THE SOFTWARE, OR AUTHORIZING OTHERS TO DO SO, YOU, ON BEHALF OF YOURSELF AND THE ENTITY FOR WHOM YOU ARE INSTALLING, DOWNLOADING OR USING THE SOFTWARE (HEREINAFTER REFERRED TO INTERCHANGEABLY AS “YOU” AND “END USER”), AGREE TO THESE TERMS AND CONDITIONS AND CREATE A BINDING CONTRACT BETWEEN YOU AND AVAYA INC. OR THE APPLICABLE AVAYA AFFILIATE (“AVAYA”).

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 originally sold by Avaya and ultimately utilized by End User.

License types

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 Software that permits one user to interface with the Software. Units may be linked to a specific, identified Server.

Shrinkwrap License (SR). Customer may install and use the Software in accordance with the terms and conditions of the applicable license agreements, such as “shrinkwrap” or “clickthrough” license accompanying or applicable to the Software (“Shrinkwrap License”). (see “Third-party Components” for more information).

Copyright

Except where expressly stated otherwise, no use should be made of materials on this site, the Documentation, Software, or Hardware provided by Avaya. All content on this site, the documentation and the Product provided by Avaya including the selection, arrangement and design of the content is owned either by Avaya or its licensors and is protected by copyright and other intellectual property laws including the sui generis rights relating to the protection of databases. You may not modify, copy, reproduce, republish, upload, post, transmit or distribute in any way any content, in whole or in part, including any code and software unless expressly authorized by Avaya. Unauthorized reproduction, transmission, dissemination, storage, and or use without the express written consent of Avaya 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 regarding distributed Linux OS source code (for those Products that have distributed the Linux OS source code), and identifying the copyright holders of the Third Party Components and the Third Party Terms that apply to them is available on the Avaya Support Web site: http://support.avaya.com/Copyright.

Preventing Toll Fraud

“Toll fraud” is the unauthorized use of your telecommunications system by an unauthorized party (for example, a person who is not a corporate employee, agent, subcontractor, or is not working on your company's behalf). Be aware that there can be a risk of Toll Fraud associated with your system and that, if Toll Fraud occurs, it can result in substantial additional charges for your telecommunications services.

Avaya Toll Fraud Intervention

If you suspect that you are being victimized by Toll Fraud and you need technical assistance or support, call Technical Service Center Toll Fraud Intervention Hotline at +1-800-643-2353 for the United States and Canada. For additional support telephone numbers, see the Avaya Support Web site: http://support.avaya.com. Suspected security vulnerabilities with Avaya products should be reported to Avaya by sending mail to: securityalerts@avaya.com.

Trademarks

Avaya, the Avaya logo, Avaya Aura® Experience Portal, AvayaAura® Communication Manager, and Avaya Aura® Orchestration Designer are either registered trademarks or trademarks of Avaya Inc. in the United States of America and/or other jurisdictions.

All non-Avaya trademarks are the property of their respective owners, and “Linux” is a registered trademark of Linus Torvalds.

Downloading Documentation

For the most current versions of Documentation, see the Avaya Support Web site: http://support.avaya.com.

Contact 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 Web site: http://support.avaya.com.

Contents

Chapter 1: Overview of the Avaya Aura® Experience Portal offers

Avaya Aura® Experience Portal 6.0 consists of three components:

- Media server
- Experience Portal Manager (EPM)
- Application Execution Environment

The architecture and configuration options differ as per the media server that is used to run the software.

In this release MPP runs on Avaya Enterprise Linux Release 6.0.32 bit or later or Release 6.0 32 bit or later.

The EPM web interface that provides a centralized administration and configuration tool.

Important:

Avaya Aura® Experience Portal 6.0 release does not support Avaya Media Server (AMS). However, the Avaya Aura® Experience Portal 6.0 documentation library includes information for AMS. Please ignore any references to AMS.

Chapter 2: System Description of Avaya Aura® Experience Portal with MPP

EPM server overview (MPP)

An Experience Portal Manager (EPM) is a server that runs the Avaya Aura® Experience Portal software. All Experience Portal systems with Media Processing Platform (MPP) must have a primary EPM server. In addition, if your system is configured to use dedicated server machines for the EPM and MPP software, the system can also have auxiliary EPM servers that handle outgoing calls when the primary EPM server is unavailable.

Primary EPM server

The EPM software on the primary EPM server:

- Includes the EPM Web interface that provides a centralized administration and configuration tool. When a user logs into the EPM Web interface, the user role associated with the user name dictates which pages the user can see and what actions the user can perform.
- Sends relevant configuration information to each MPP server.
- Routes outgoing calls made with the Application Interface web service to an available MPP server.
- Collects the operational status from each MPP server and displays it on the EPM Web interface.
- Monitors the heartbeat of the MPP servers and redistributes telephony ports when an MPP fails.
- Receives event and alarm messages from all MPP servers.
- Downloads report data from all MPP servers and stores it in the Experience Portal database so that users can create reports that contain information from all MPP servers in the system.
- Interacts with the Avaya WebLM license server to distribute and manage Automatic Speech Recognition (ASR), Text-to-Speech (TTS), and Telephony ports across all MPP servers.

- Provides an optional Simple Network Management Protocol (SNMP) interface to monitor Experience Portal alerts.
- Handles Application Logging web service requests.

Auxiliary EPM server

The EPM software on the auxiliary EPM server:

- Assigns outgoing calls made with the Application Interface web service to an available MPP server. However, Experience Portal does not provide load balancing or failover. You must use a third-party product for these purposes.
- Shares Application Logging web service requests when the primary EPM server is in service and handles all the application logging requests when the primary EPM is not functional.

Note:

When using the Application Logging web service, Orchestration Designer 6.0 provides failover and load balancing between the primary and auxiliary EPM servers. Applications written with other tools must provide their own load balancing and failover mechanisms for this web service.

- Does *not* include the EPM Web interface, therefore the Auxiliary EPM server cannot be used to administer the system or monitor the status of the MPP servers.

Experience Portal network architecture

The Experience Portal network consists of the Experience Portal system and other external systems.

The Experience Portal system consists of two major subsystems:

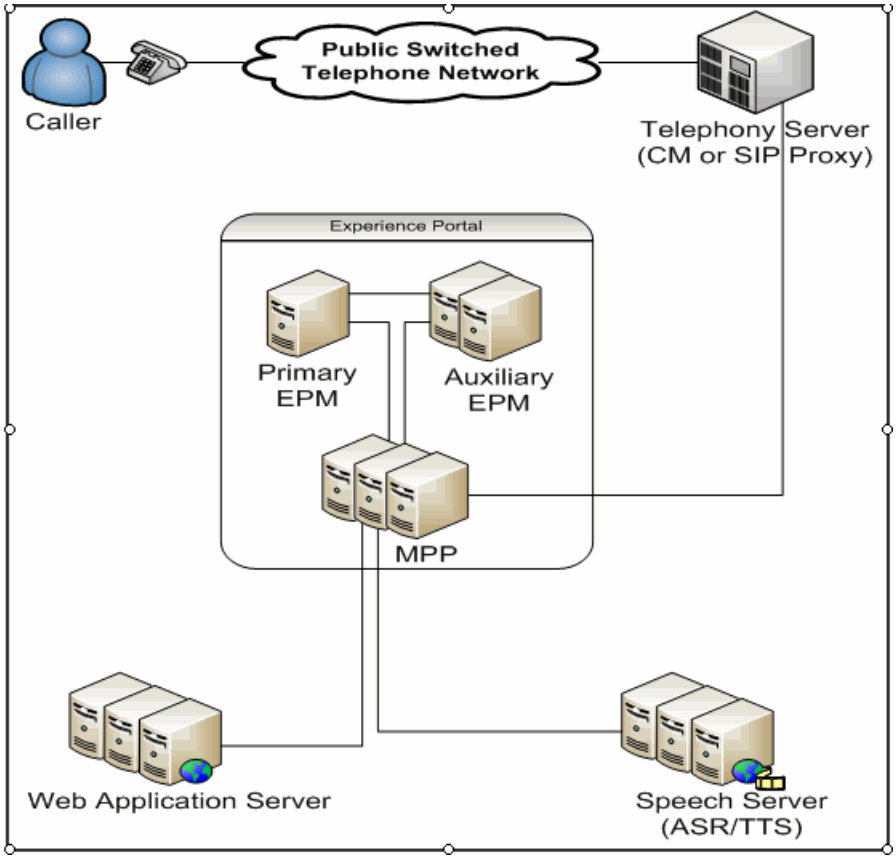
- The EPM, which controls the Experience Portal system
- One or more MPP servers, which process all incoming and outgoing calls

In the Experience Portal network, external systems include the following:

- Private Branch Exchange (PBX) servers
- ASR and (TTS) speech servers
- Application servers

Experience Portal network diagram

The following figure shows the Experience Portal network architecture and the connections between the components.



Component	Description
Public Switched Telephone Network (PSTN)	—
Private Branch Exchange (PBX)	Experience Portal supports one or more PBXs.
Avaya Aura® Session Manager	—
Primary Experience Portal Manager (EPM)	Experience Portal requires one primary EPM server.
Auxiliary Experience Portal Manager (EPM)	Experience Portal supports one or more auxiliary EPM servers.
Media Processing Platform (MPP)	Experience Portal requires at least one MPP server. Experience Portal supports up to 30 MPPs running on dedicated servers, or one MPP running on the same server as the Primary EPM.
(ASR) and (TTS) servers	Experience Portal supports one or more ASR and TTS servers.
Application server	Experience Portal supports one or more application servers.

Overview of the Experience Portal with MPP

Experience Portal provides two offers depending on whether you want to buy the server hardware and operating system software along with the Experience Portal software.

With either offer, the Experience Portal platform must be installed on a LAN and must have connectivity to a Private Branch Exchange (PBX). LAN connectivity also provides connections to optional speech servers and other external systems.

Avaya-provided or bundled, server offer

This offer includes the following items from Avaya:

- Hardware requirements for the number of Experience Portal servers you purchased.
- The Enterprise Linux Installer, which installs the Avaya Enterprise Linux operating system.
- The Experience Portal software that runs on each EPM and MPP server in the system.
- The Orchestration Designer software. Orchestration Designer is an Eclipse plug-in that provides an integrated GUI for application design and implementation. Orchestration Designer creates speech applications that automatically conform to the Experience Portal requirements and recommendations.

In addition, when you select this offer an Avaya representative visits your site to install and configure the Experience Portal servers.

Customer-provided server offer

This offer includes only the Experience Portal and Orchestration Designer software from Avaya.

Customers are required to:

- Obtain the hardware for all planned Experience Portal servers. The hardware must meet the requirements described in Minimum (Linux) server machine hardware requirements on page 28.
- Either install the physical server machines or arrange for their installation by an Avaya technical support representative or third-party service provider.
- Install Release 6.0 32 bit or later in 32-bit mode on all planned Experience Portal servers.

Overview of the Experience Portal server configuration options

When you install the Experience Portal software, you can use a single server or multiple servers, depending on the number of telephony ports required.

Single server configuration

This configuration includes a single server running both the Experience Portal Manager (EPM) and Media Processing Platform (MPP) software.

The advantages are:

- Only a single server is required.
- There are no network problems between the EPM and the MPP.
- There is no time synchronization problems between the EPM and the MPP.
- You can also install a Tomcat application server on the Experience Portal server.

The limitations are:

- The system is limited to 60 telephony ports. If you need additional ports, you must use the dedicated server configuration.
- There is no failover mechanism for Application Interface web service and Application Logging web service requests if the EPM server is unavailable.

Dedicated server configuration

This configuration includes two or more servers, one dedicated to running the primary EPM software and at least one dedicated to the MPP software. In addition, you can have an auxiliary EPM server that handles failover for Application Interface web service requests.

The advantages are:

- You can configure up to 30 dedicated MPP servers for the Experience Portal system, up to a maximum of 5,000 telephony ports. You can also link multiple systems through an external database.
- You can configure an auxiliary EPM server that can handle Application Interface web service and Application Logging web service requests if the primary EPM server is unavailable.
- If one MPP server is unavailable, Experience Portal can redistribute its ports to the other MPP servers as long as the MPP servers are not already running at full capacity. For more information, see The MPP server capacity on page 22.

The limitations are:

- The application server must reside on its own dedicated server machine.
- The EPM and MPP servers require LAN in order to communicate. Network issues can disrupt this communication.

Tip:

To determine exactly what your installation requires, consult your Avaya Services representative or Avaya Business Partner.

EPM components

Installed on the Linux operating system, the EPM software consists of the following components:

- Experience Portal Manager web application
- Experience Portal web services
- Application log manager
- Alarm manager
- Network log manager
- Avaya License Manager
- Experience Portal database

Additionally, the EPM relies on several third-party components, which are installed automatically as part of the EPM installation, including:

- Java, Standard Edition Software Development Kit: Java run-time environment
- Apache Tomcat: web servlet container
- Apache Axis: web services container
- Apache Axis2: web services container
- PostgreSQL: SQL database server

Experience Portal Manager Web application

The Experience Portal Web application serves several purposes, including:

- Provides graphical Web pages for configuring and administering the Experience Portal system.
- Sends relevant configuration information to each media server
- Collects operational status from each media server
- Collects report data from each media server
- Collects license information from the Avaya License Manager

Application log manager

The application log manager receives log entries generated by applications developed by using Orchestration Designer and writes those entries to the Experience Portal database.

Alarm manager

The alarm manager monitors the entries logged by the network log manager. When appropriate, the alarm manager generates an alarm.

Network log manager

The network log manager receives log entries from several Experience Portal components and writes those entries to the Experience Portal database.

Avaya License Manager

Several Avaya products share the Avaya License Manager (WebLM) component. When you purchase Experience Portal, you receive a license file from Avaya that specifies the number of Telephony ports, Automatic Speech Recognition (ASR), and Text-to-Speech (TTS) resources that you have purchased. Experience Portal must be able to communicate with the WebLM server in order to process any incoming or outgoing calls.

The WebLM server software is automatically installed with the Experience Portal primary EPM software, but you can also connect your Experience Portal to a dedicated WebLM server machine which is shared among all Avaya products.

Experience Portal database

The Experience Portal database stores important Experience Portal data for both the EPM and the media servers.

Because the database is located on the EPM server, the MPP servers do not need to be backed up.

All important data from the Avaya Media Server database is backed up in the Experience Portal database from the System Backup feature in EPM.

Note:

You should not modify the Experience Portal internal database. For assistance to modify the database, contact your Avaya technical support representative.

Related topics:

Directory details of the EPM system components on page 15

Directory details of the EPM system components

The EPM system components are located in different directories on the Linux operating system. The following table provides the location where most of the files for each EPM component are installed:

Note:

In addition to the directories listed below, some of the EPM components modify or update several other directories and files on the Linux operating system.

Component	Directory
Experience Portal Manager web application	/opt/Tomcat/tomcat/webapps/VoicePortal
Avaya Aura® Experience Portal Management web services	/opt/Tomcat/tomcat/webapps/axis2
Application log manager	/opt/Tomcat/tomcat/webapps/axis
Alarm manager	/opt/Tomcat/tomcat/webapps/axis
Network log manager	/opt/Tomcat/tomcat/webapps/axis
Avaya License Manager	The co-located WebLM is installed in the /opt/Tomcat/tomcat/webapps/WebLM directory. **Note:** If you use an external WebLM, the license manager may be installed in a different directory on the external system.
Experience Portal database	The Postgres files are installed in the /var/lib/pgsql directory. **Note:** Most of the database data is in the /var/lib/pgsql/data directory.
Java, Standard Edition Software Development Kit: Java run-time environment	/usr/java
Apache Tomcat: web servlet container	/opt/Tomcat
Apache Axis: web services container	/opt/Tomcat/tomcat/webapps/axis
Apache Axis2: Web services container	/opt/Tomcat/tomcat/webapps/axis2
PostgreSQL: SQL database server	/var/lib/pgsql

Media Processing Platforms

MPP server overview

A Media Processing Platform (MPP) server runs the Experience Portal MPP software.

The MPP software:

- Runs on Avaya Enterprise Linux or Red Hat Enterprise Linux 6.0
- Uses Voice over IP (VoIP) protocols to communicate with the telephone network
- Uses the Media Resource Control Protocol (MRCP) protocol to communicate with the speech servers
- Runs Voice eXtensible Markup Language (VoiceXML) speech applications deployed on the application server
- Runs Call Control eXtensible Markup Language (CCXML) applications

Note:

Experience Portal uses the Oktopous™ ccXML Interpreter. The **CCXML URL** field is not applicable for AMS.

Multiple MPP servers

When you configure a system with multiple MPP servers:

- An individual MPP server is not aware of any other MPP servers in the system, nor can it communicate directly with them.
- Using the EPM Web interface, administrators can control any MPP server in the system.

Data storage

The Experience Portal system is designed so that all persistent data is stored on the primary EPM server. For example, all configuration information is stored on the primary EPM server and downloaded to the MPP server when required.

Any persistent data created on the MPP server is uploaded to the EPM either on-demand or through scheduled jobs. For example:

- The EPM regularly polls the MPP server status.
- Event and alarm data is delivered to the EPM on demand.
- Report data, including Call Detail Records (CDRs) and Session Detail Records (SDRs), are delivered to the EPM according to a schedule that you administer.

The MPP has additional data that can be used for debugging, but the additional data is not required to be persistent. For example:

- Trace data and MPP-specific log files
- Session transcriptions and utterances

MPP server components

The MPP server consists of the following components:

- System Manager
- Web services
- Session Manager
- Avaya Voice Browser
- CCXML Browser
- Speech proxies
- Telephony
- Event Manager

Related topics:

[System Manager component](#) on page 18
[The Web services component](#) on page 19
[The Session Manager component](#) on page 20
[The Avaya Voice Browser component](#) on page 20
[The CCXML Browser component](#) on page 21
[Speech proxy component](#) on page 21
[The Telephony component](#) on page 22
[The MPP server capacity](#) on page 22

System Manager component

The System Manager component works in conjunction with the EPM to keep the MPP functioning in an optimal state. In addition, System Manager provides the following functions:

Function	Description
State management	Starts and stops all processes in response to start or stop commands from the EPM. Monitors the health of the processes and attempts to restart any processes that exit prematurely, appear deadlocked, have stopped responding.
Configuration management	The EPM downloads configuration information to the MPP during startup. Configuration updates can also be downloaded to the MPP while it is running. The System Manager transfers the information to the other MPP components of the change, if needed.

Function	Description
License management	The EPM manages port licensing for each MPP and passes that information during MPP startup and later if licenses need to be redistributed. The EPM downloads all licensing changes to the MPP.
Resources monitor	The EPM monitors CPU usage, memory usage, and disk usage for each MPP. The EPM checks the state of these resources at predetermined intervals during EPM polling operations. If at any time the use of these resources crosses thresholds set on the EPM, Resource monitor issues an alert. The System Manager also monitors for network errors between the MPP and the EPM.

The Web services component

The EPM accesses the web services of the MPP to monitor and control the MPP. The Apache Web server implements the web services and ensures that communication between the EPM and the web services is secure. The MPP web services are:

Service name	Description
Call Data Handler (CDH) service	The EPM uses the CDH service to transfer Application Detail Records (ADRs), Call Detail Records (CDRs), and Session Detail Records (SDRs) from the MPP. The EPM stores the record data in the Experience Portal database and uses this information to generate the call and session reports.
MPP Management Service (MMS)	The EPM uses the MMS to send heartbeat requests, configuration changes, and commands. The MMS then forwards these requests to the System Manager for execution.
Application Interface web service	Also known as the “Outcall web service”, using this Web services the developers can: • Start a CCXML or VoiceXML application that has been added to Experience Portal. • Send an event to a specific application session running on an MPP. • Query the system for the total number of: - Used and unused outbound resources available - Unused SIP outbound resources - Unused H.323 outbound resources
TransService	This process uploads any transcription data to the Experience Portal database.

The Session Manager component

A *session* covers the time between the start of the inbound or outbound call and the completion of that call.

When the MPP initiates a call or is assigned a call, the Session Manager:

1. Starts a new session.
2. Assigns the session a unique ID.
3. Associates the call with the appropriate Call Control eXtensible Markup Language (CCXML) or Voice eXtensible Markup Language (VoiceXML) application.
4. Depending on the MPP settings, the administrator selects for the MPP, records all or some of the following data during the session:
    - Call Detail Records (CDRs)
    - Application Detail Records (ADRs)
    - Session transcriptions
    - Performance trace information

The MPP Session Manager also coordinates all interactions between the MPP and:

- Any Automatic Speech Recognition (ASR) servers
- Any Text-to-Speech (TTS) servers
- Any telephony components
- The Avaya Voice Browser.
- The CCXML Browser

The Avaya Voice Browser component

The Avaya Voice Browser is a Voice eXtensible Markup Language (VoiceXML) interpreter that communicates with the application servers to interpret the VoiceXML documents of a speech application.

For each incoming call:

1. Session Manager starts a new Avaya Voice Browser session and passes the Universal Resource Indicator (URI) of the VoiceXML application to the new session.
2. The Avaya Voice Browser contacts the application server and waits for the VoiceXML page to be returned.
3. After the application starts, the Avaya Voice Browser is responsible for:
   - Interpreting the VoiceXML page returned by the application server.
   - Managing the user interaction including playing prompts and interpreting input from the caller through Dual-tone multi-frequency (DTMF) or Automatic Speech Recognition (ASR).

The CCXML Browser component

The CCXML Browser component is responsible for providing low level call control support including the setup, monitoring, and tear-down of telephone calls.

For VoiceXML applications, Experience Portal includes a default CCXML application that provides the basic call control functionality. If you want to use advanced features such as call merging and all conferencing, you need to create a custom CCXML application.

Note:

Experience Portal uses the Oktopous™ ccXML Interpreter. The **CCXML URL** field is not applicable for AMS.

Speech proxy component

The MPP speech proxy component integrates third-party media resources, such as Automatic Speech Recognition (ASR) and Text-to-Speech (TTS) speech servers, into the Experience Portal system by employing Media Resource Control Protocol (MRCP).

When a speech application requests ASR or TTS resources, the speech proxy component communicates with the speech servers and selects the appropriate server to provide those resources. The MRCP proxy reports the state of the speech servers to the MPP System Manager.

If directed by the EPM, the speech proxy component can also add or remove communication ports between an MPP and any speech server in the system.

The Telephony component

The MPP Telephony component provides all telephony services required by the Experience Portal system, including call control and media processing.

The telephony subsystem can be connected to:

- Communication Manager, a VoIP-based PBX gateway, using the International Standard for Multimedia Communication Over Packet-switched Networks (H.323) and Real-time Transport Protocol (RTP) to transport the actual audio data stream in a connection
- Communication Manager using Session Initiation Protocol (SIP) and RTP
- The Avaya G860 Media Gateway using SIP
- Third-party SIP gateways

The MPP server capacity

The number of telephony ports and the maximum number of simultaneous calls that the MPP server can handle depend on many factors, including the hardware characteristics of the MPP server and the complexity of the applications that the Experience Portal system is running. For assistance in sizing your MPP server capacity and setting the correct value for the **Maximum Simultaneous Calls** parameter for each MPP server, contact your Avaya Services representative or Avaya Business Partner.

When configuring your Experience Portal system, make sure that you have enough MPP servers to handle the telephony ports that you purchase. Ideally, you should have enough reserve capacity so that when one MPP server goes out of service, all of your telephony ports can be handled by the remaining MPP servers. You must have enough MPP servers so that the sum of the maximum simultaneous calls is larger than the number of configured ports.

For example, if your Experience Portal system needs to handle 400 simultaneous calls, you must purchase 400 telephony port licenses and configure a sufficient number of MPP servers to run 400 simultaneous calls.

If your Avaya Services representative or Avaya Business Partner determines that each MPP server can handle a maximum of 100 simultaneous calls, you can configure:

- Four MPP servers, each with the **Maximum Simultaneous Calls** parameter set to 100. During initialization, the Experience Portal distributes the 400 available telephony ports across the four servers so that each server runs at a maximum capacity of 100 calls each, and the entire system can process 400 simultaneous calls. However, there is no failover capability in this configuration. If one MPP server goes out of service, Experience Portal cannot reassign the ports because the other three servers are already running at full capacity.
- Five MPP servers, each with the **Maximum Simultaneous Calls** parameter set to 100. During initialization, the Experience Portal distributes the 400 available telephony ports

across the five servers so that each server is assigned 80 telephony ports and the entire system can process 400 simultaneous calls. In this configuration, if one MPP server goes out of service, Experience Portal can reassign the 80 ports to the other four servers, bringing up the servers to maximum capacity.

If desired, you can add up to 30 MPP servers to a single Experience Portal system, and that system can handle up to 5,000 telephony ports. You can also link several Experience Portal systems together through an external database.

MPP processes

The following table provides an overview of the processes that run on the MPP. For information about the log files created by these processes, see the MPP server logs topic in the *Troubleshooting Avaya Aura® Experience Portal* guide.

Process Name	Descriptive Name	Notes
`ccxml`	CCXML Interpreter	Controls all call handling behavior for each VoiceXML application that runs on the MPP. CCXML Interpreter also controls each request to obtain or release a telephony resource for a given VoiceXML application. **Note:** Experience Portal uses the Oktopous™ ccXML Interpreter. The **CCXML URL** field is not applicable for AMS.
`CdhService`	Call Data Handler (CDH)	A web service that runs when the EPM is downloading Call Detail Records (CDRs) and Session Detail Records (SDRs).
`EventMgr`	Event Manager	Collects events from other MPP processes and sends them to the network log web service on the EPM.
`httpd`	Apache Web Server	Enables the other web services running on the MPP. The first Apache Web Server process started by the daemon runs as root. The root process starts nine other processes that run as the `avayavp` user in the avayavpgroup group.
`MmsServer`	MPP Management Service (MMS)	With a Web service interface, the EPM server sends commands to the MPP server. MMS runs only when the EPM is polling or sending commands to the MPP.
`mppmaint`	MPP Maintenance Utility	The `cron` process runs the MPP Maintenance Utility daily at 4 am to purge CDRs, SDRs, and transcriptions data based on the retention period specified in the EPM.

Process Name	Descriptive Name	Notes
`mppmon`	MPP Monitor	Runs as root and monitors the `httpd` service, restarting them if necessary.
`mppsysmgr`	System Manager	Handles the majority of tasks required to manage the MPP. For example, this process monitors system resources such as CPU usage, memory usage, and disk usage. If any of these values exceed the baseline set in the EPM, the System Manager issues an alarm message. When instructed by the EPM, the System Manager starts or stops all MPP processes and distributes EPM configuration updates to all MPP processes as updates occur.
`SessionManager`	Session Manager	Runs as root and integrates and controls the interaction between the MPP and media resources, as well as between the speech application and the ASR, TTS, and telephony components.
`TransService`		Uploads any transcription data to the Experience Portal database.
`vxmlmgr`	VoiceXML Manager	Works with the Session Manager to run multiple VoiceXML dialog sessions. VoiceXML Manager also interfaces with the CCXML, telephony, ASR, and TTS subsystems. The VoiceXML Manager and the Session Manager communicates through messages. The Session Manager is responsible for interpreting these messages and routing the calls to the appropriate platform subsystems on behalf of the VoiceXML Manager.

Chapter 3: External system requirements

External requirements worksheet

Use this worksheet to make sure that all external requirements have been met before you begin installing the Avaya Aura® Experience Portal hardware or software.

The site at which the Experience Portal servers are located, the server hardware, and the network connecting the servers must meet certain requirements.

In addition, there are external systems, such as Communication Manager and third-party speech servers, that support Experience Portal operation. External systems are optional while others are an integral part of an Experience Portal system. You must purchase these external systems separately and you are responsible for installing, administering, and maintaining them.

✔	External requirement
	The physical site at which you intend to install Experience Portal must have sufficient space and the proper network connections, as described in Site requirements on page 26.
	Experience Portal requires: • A 100/1000 Base-T LAN full duplex network switch connection so that all Experience Portal servers can communicate with the application server, the speech servers, and the Private Branch Exchange (PBX). • Each Experience Portal server has a static IP address and host name. For details, see LAN requirements on page 27.
	The PBX must be running the appropriate version of Communication Manager as described in PBX requirements on page 27.
	If the customer is supplying the Experience Portal servers, each planned Experience Portal server must meet the minimum hardware requirements described in Minimum (Linux) server machine hardware requirements on page 28.
	To access the EPM Web interface, Experience Portal requires Microsoft Internet Explorer 6 (IE6) SP2 or later configured to use TLS security. For details, see Configuring browsers to use TLS security on page 40.
	Experience Portal requires a third-party application server to deploy speech applications. For details, see Application server requirements for Avaya Aura Experience Portal with MPP offer on page 28.

✔	External requirement
	Voice over IP (VoIP) requirements: • To use H.323 connections, make sure that the version of Communication Manager meets the requirements described in H.323 requirements on page 29. • To use SIP connections, make sure that the versions of Communication Manager and Avaya SIP Enablement Services meet the requirements described in SIP requirements on page 30.
	If your speech applications require Automatic Speech Recognition (ASR) or Text-to-Speech (TTS) resources, you must purchase and install one or more third-party speech servers. For a list of supported servers, see Speech server requirements for Experience Portal on page 33.
	Before you can configure Experience Portal, you need the site-specific licensing information from Avaya, as described in License requirements on page 37.
	During Experience Portal installation, you will be prompted for several passwords. These passwords must meet the minimal requirements described in Password requirements on page 38.

Related topics:

Site requirements on page 26
LAN requirements on page 27
PBX requirements on page 27
Minimum (Linux) server machine hardware requirements on page 28
Application server requirements for Avaya Aura Experience Portal with MPP offer on page 28
H.323 requirements on page 29
SIP requirements on page 30
Comparison of features supported on H.323 and SIP on page 30
Speech server requirements for Experience Portal on page 33
Speech application requirements and recommendations on page 36
License requirements on page 37
Password requirements on page 38

Site requirements

Verify that the site where you are installing the Experience Portal hardware platform is equipped with the following:

- Rack space for the servers that host Experience Portal.
- At least one network connection for each Experience Portal server. Depending on your network topology, two network connections might be required for each media server.

- Power supply.
- (Optional) Analog telephone line provisioned for Avaya Secure Access Link (SAL) or the Avaya Access Security Gateway (ASG) solution.

LAN requirements

Connectivity requirements

Experience Portal requires a 100/1000 Base-T LAN full duplex network switch connection so that Experience Portal servers can communicate with each other, with any other speech servers, any application servers, and any Private Branch Exchange (PBX) servers.

Each server in your Experience Portal system must be able to connect to all the other servers in the system using the host names of the other servers. You must use a Domain Name Server (DNS) for this purpose.

Server name requirements

Each Experience Portal server must have a static IP address and a host name. Each host name must be unique and cannot contain a . (period) or a (space) character.

PBX requirements

A Private Branch Exchange (PBX) functions as a gateway between the public and corporate telephony networks and the Experience Portal system as follows:

1. A caller connected to the Public Switched Telephone Network (PSTN) dials a telephone number that is associated with a speech application within the Experience Portal system.
2. The PSTN routes the call to the PBX associated with the number.
3. The PBX routes the call to an available media server.

You are responsible for managing and maintaining the PBX. The PBX must be accessible to the Experience Portal servers through a LAN, and the PBX must be running the appropriate version of Communication Manager. The required Communication Manager version is based on whether you want to use H.323 connections, SIP connections, or both.

To use...	Required...
H.323 connections	Communication Manager version 3.1 or later
H.323 with supervised transfer or the Application Interface web service for outbound calls	Communication Manager 3.1 build 369 or later with the Avaya Special Application SA8874 feature

To use...	Required...
SIP	Avaya SIP Enablement Services version 4.0 or later with either Communication Manager version 3.0 or later or a third-party SIP Gateway or SIP Trunk
SIP with SRTP	Communication Manager version 4.0 build 730.3 or later with Avaya SIP Enablement Services (SES) version 4.0 or later

Minimum (Linux) server machine hardware requirements

You must have the following minimum specifications for Experience Portal with Media Processing Platform (MPP) customer-supplied server machine:

- Compatibility with Release 6.0 32 bit or later running in 32-bit mode. For information about hardware compatibility, go to the *Certified Hardware* section of the Red Hat website, http://www.redhat.com.
- Dual Quad Core 1.6 GHz Pentium 4 or equivalent processors.
- 4 GB of RAM.
- 120 GB Disk, 7200 RPM.
- One 100/1000 Base-T Ethernet controller that is full duplex (onboard Network Interface Cards (NICs)).
- DVD drive.
- Keyboard.
- Monitor.
- Mouse.
- Avaya Secure Access Link (SAL) or Avaya ASG solution. If you purchase a maintenance agreement with Avaya Services, the Experience Portal system requires SAL or Avaya ASG solution so that Avaya Services can remotely access the system for maintenance purposes. Contact Avaya Support to determine the version of SAL and Avaya ASG supported.

Application server requirements for Avaya Aura® Experience Portal with MPP offer

In an Avaya Aura® Experience Portal with MPP network, the application server is a Web server that hosts your Call Control eXtensible Markup Language (CCXML) and Voice eXtensible Markup Language (VoiceXML) speech applications.

Avaya Orchestration Designer

For detailed Orchestration Designer requirements, see the documentation on the Orchestration Designer installation CD.

Dedicated server requirements

If you are installing the Experience Portal Manager (EPM) and the Media Processing Platform (MPP) software on separate servers, you must also install the application server on a separate server.

Single server requirements

If you are installing the Avaya Aura® Experience Portal EPM software on the same server machine as the MPP software, you have the option of installing a Tomcat application server on that machine as well. However, you must install the application server on a dedicated server machine even in a single server configuration.

Note:

Avaya Aura® Experience Portal includes an installation script for the Tomcat 6.0.32 application server. If you select any other version of Tomcat, you must install the Application server manually.

Additional information

For more information about:

- Java: go to http://java.sun.com.
- WebSphere Express: go to http://www.ibm.com/software/webservers/appserv/express/.
- Tomcat: go to http://tomcat.apache.org/.
- Orchestration Designer, see the documentation delivered with that product.

H.323 requirements

For H.323 connections, you must have Communication Manager version 3.1 or later.

You must use Communication Manager 3.1 build 369 or later with the Avaya Special Application SA8874 feature. This combination provides:

- VoiceXML supervised transfers. Without the SA8874 feature, supervised transfers have no access to call progress information and behave like a blind transfer.
- The Application Interface web service for outbound calling. Without the SA8874 feature, the web service has no access to call progress information and may start a VoiceXML application even when the connection attempt receives a busy signal.

Note:

The SA8874 feature comes with Communication Manager 3.1 or later but requires a separate license before you can enable the feature.

For information on how to configure Communication Manager to work with Avaya Aura® Experience Portal H.323 connections, see *Avaya Configuration Note 3910* on the Avaya online support Web site, http://support.avaya.com.

SIP requirements

For SIP connections, Experience Portal requires Avaya SIP Enablement Services version 4.0 or later with either Communication Manager version 3.0 or later or a third-party SIP Gateway or SIP Trunk..

If you want to use Secure Real-time Transport Protocol (SRTP), you must use Communication Manager version 4.0 build 730.3 or later with Avaya SIP Enablement Services (SES) version 4.0 or later.

Note:

For information on how to integrate SIP with Experience Portal, see *Avaya Configuration Note 3911* on the Avaya online support Web site, http://support.avaya.com.

Comparison of features supported on H.323 and SIP

This table compares:

- Standard H.323
- H.323 with the Avaya Special Application SA8874 feature enabled in Communication Manager
- SIP

Feature	H.323	H.323 with SA8874 feature	SIP
Outbound calling using the Application Interface web service	Partially supported. No call progress information is available, so an application may start before a call is answered.	Supported	Supported
Call conferencing	Supported	Supported	Supported
Call classification	Supported	Supported	Supported
Blind transfer	Supported	Supported	Supported

Feature	H.323	H.323 with SA8874 feature	SIP
Supervised transfer (also called consultative transfer) **Note:** If a connection cannot be established, use the Consultative Transfer feature in Experience Portal to allow the application to regain control of the call.	Operates like a blind transfer. **Note:** The only supported VoiceXML event for this transfer is `error.connection.noroute`.	Supported	Supported
Bridge transfer. See also Bridge transfers in a mixed SIP or H.323 environment on page 33	Partially supported. No call status information, such as "line is busy", is available.	Supported	Supported except for the VoiceXML `<transfer>` tag's `connecttimeout` parameter, which is not supported
DTMF detection **Note:** Experience Portal supports only out-band DTMF detection.	Supported	Supported	Supported **Note:** In case of SIP VoIP connection, the signaling group doesn't support the out- band option. It supports the in-band and RTP-payload DTMF options.
Playing prompt files	Supported	Supported	Supported
Recording	Supported	Supported	Supported
Converse-on vectoring	Supported	Supported	Not supported

Feature	H.323	H.323 with SA8874 feature	SIP
Encryption options	• Disabled • Advanced Encryption Standard (AES) • Avaya™ Encryption Algorithm (AEA)	• Disabled • AES • AEA	• Disabled • TLS • SRTP
Quality of Service	Supported	Supported	Supported
User to User Information (UUI)	Not supported	Not supported	For an incoming call, UUI values are populated in the VoiceXML session variables for both UUI and Application to Application Information (AAI). For more information, see User-to-User Interface (UUI) data passed in SIP headers on page 72.
Universal Call Identifier (UCID)	Supports the capability to receive UCID over H323 from Communication Manager. **Note:** This capability is available in Communication Manager 5.2. To enable this feature, you need to administer ucid-info on button 10 on the 7434ND stations used by Experience Portal. For more information, see Universal Call Identifier (UCID) values included in UUI data on page 73.	Supports the capability to receive UCID over H323 from Communication Manager. **Note:** This capability is available in Communication Manager 5.2. To enable this feature, you need to administer ucid-info on button 10 on the 7434ND stations used by Experience Portal. For more information, see Universal Call Identifier (UCID) values included in UUI data on page 73.	Supports the capability to both send and receive UCID. For more information, see Universal Call Identifier (UCID) values included in UUI data on page 73. **Note:** Also supports the GSLID used by AACC

Feature	H.323	H.323 with SA8874 feature	SIP
Switch failover	An alternate gatekeeper address can be specified in the EPM. Communication Manager can supply an alternate gatekeeper address list.	An alternate gatekeeper address can be specified in the EPM. Communication Manager can supply an alternate gatekeeper address list.	Experience Portal does not supply additional support, but the Avaya SIP Enablement Services (SES) hardware has failover support and MPPs can be configured as members of an adjunct in the SES.
Merge (Refer with replaces)	Not supported	Not supported	Supported

Bridge transfers in a mixed SIP or H.323 environment

If you have both SIP and H.323 connections defined in your Experience Portal system, Experience Portal handles bridge transfers in the following manner. For an outbound call with:

- `SIP` or `SIPS` in the `ToURI` field, a SIP outbound channel must be available.
- `TEL` in the `ToURI` field, Experience Portal tries to get an outbound port from the same H.323 port group. If none are available, Experience Portal tries any H.323 port.

  If no H.323 ports are available, Experience Portal converts `TEL` into `SIP` in the `ToURI` field and tries and get a SIP outbound channel.

Speech server requirements for Experience Portal

If your speech applications require Automatic Speech Recognition (ASR) or Text-to-Speech (TTS) resources, you must purchase and install one or more of the following third-party speech servers. The Experience Portal product does not include any speech server types.

Note:

You must purchase the recommended versions of ASR and TTS from the vendors, and use the matrix mentioned in this section to install the correlated components.

Supported ASR speech servers

Required versions

Speech server	Minimum version required	Also required
Nuance Recognizer (using Real Speak)	Recognizer 9.0.7 RealSpeak 4.5 w/patch 2	Nuance Speech Server (NSS) version 5.0.5

Speech server	Minimum version required	Also required
		Note: You must use NSS version 5.0.5 or later.
Nuance Recognizer (using Vocalizer)	Recognizer 9.0.9 Vocalizer 5.0.3	Nuance Speech Server (NSS) 5.1.1 **Note:** You must use NSS version 5.1.2 or later.
Loquendo ASR	LASR 7.8.1 w/Patch 13 LTTS 7.8.4 (Engine Full)	Loquendo Speech Suite (LSS) LSS 7.0.8 w/Patch 3 – for Linux LSS 7.0.13 – for Windows

MRCP support

Speech Server	MRCP V1 Support	MRCP V2 Support
Nuance Recognizer **Note:** Supports both MRCP V1 and V2 simultaneously.	MRCP V1	MRCP V2/TCP and MRCP V2/TLS
Loquendo **Note:** Supports both MRCP V1 and V2 but only one at a time.	MRCP V1	MRCP V2/TCP **Note:** • MRCP V2/TLS is not supported. • AMS platform does not support LSS in this release.

SRGS support

Speech Server	SRGS support	SRGS format support with SISR tag
Nuance Recognizer	Yes	Yes
Loquendo	Yes	Yes

NLSML and EMMA recognition result support

Speech Server	NLSML recognition result support	EMMA recognition result support
Nuance Recognizer	Yes	Yes
Loquendo	Yes	Partially supported

Note:
AMS does not support EMMA recognition result.

Supported TTS speech servers

Speech Server	Minimum Version Required	Also Required	MRCP V1 and MRCP V2 Support
Nuance RealSpeak	4.5 w/patch2	Nuance Speech Server version 5.0.5	MRCP V1, MRCP V2/TCP, and MRCP V2/TLS
Nuance Vocalizer	5.0.3	NSS 5.1.1	MRCP V1, MRCP V2/TCP, and MRCP V2/TLS
Loquendo	7.8.4	Loquendo Speech Suite (LSS) 7.0.8 – for Linux 7.0.13 – for Windows	MRCP V1 and MRCP V2/ TCP **Note:** MRCP V2/TLS is not supported.

Note:
Only MPP supports both MRCP V1 and MRCP V2 (TCP & TLS).

AMS does not support MRCP V2 or LSS in this release.

Recommended releases for the speech servers

Speech Server	MRCP V1	MRCP V2
Nuance Speech Server (NSS) 5.0.5 or later	ASR: Recognizer 9.0.7 or later TTS: RealSpeak 4.5 with SP2 or later	ASR: Recognizer 9.0.7 or later TTS: RealSpeak 4.5 with SP2 or later
Nuance Speech Server (NSS) 5.1.3 **Note:** NSS 5.1.x is not downward compatible to NSS 5.0.x.	ASR: Recognizer 9.0.9 or later TTS: Vocalizer 5.0.3	ASR: Recognizer 9.0.9 or later TTS: Vocalizer 5.0.3

Speech Server	MRCP V1	MRCP V2
Loquendo Speech Server (LSS)	TTS: Engine 7.8.4 ASR: Engine 7.8.1 w/Patch 13 LSS: 7.0.13 for Windows, 7.0.8 for RH Linux	TTS: Engine 7.8.4 ASR: Engine 7.8.1 w/Patch 13 LSS: 7.0.13 for Windows, 7.0.8 for RH Linux

Recommended Nuance version for using SRTP

The following matrix shows the compatibility of Nuance versions for using SRTP when Nuance is configured to use MRCP V2 with TLS.

Nuance Speech Server (NSS)	Nuance Recognizer (NRec)	TTS	Function
5.0.4	9.0.4	4.5.1	Supports remote DTMF and TLS with SRTP enabled
5.0.5	9.0.7	4.5.2	Contains valid certificates
5.1.3	9.0.9	Vocalizer 5.0.3	

Note:

RS-4.5 patch 2 is required for installing Nuance TTS 4.5.1 on Linux. For more information, see *Nuance release note of RS-4.5 patch 2*.

Additional information

If you need more information about:

- Nuance servers: go to http://www.nuance.com
- Loquendo servers: go to http://www.loquendo.com

Speech application requirements and recommendations

The following technologies are required for Experience Portal speech applications:

CCXML	Experience Portal supports Call Control eXtensible Markup Language (CCXML) applications that comply with most of the standards defined in Call Control eXtensible Markup Language (CCXML). Of these standards, Experience Portal does *not* support: • The <createccxml> tag. • The <move> tag. • The <join> tag for dialogs. Dialogs can attach to a call or conference using the <dialogprepare> or <dialogstart> tags.

	• The <unjoin> tag for dialogs. Dialogs remain attached to a call or conference session for the entire duration of the dialog or the session, whichever ends first. • The Basic HTTP Event I/O Processor described in Appendix K of the W3C Working Draft. For more information, see the W3C CCXML Version 1.0 Web site. **Note:** CCXML is not applicable for Experience Portal with the AMS offer.
VoiceXML	Voice eXtensible Markup Language (VoiceXML) applications are required to comply with the W3C VoiceXML Version 2.1 Recommendation. For more information, see the Voice Extensible Markup Language (VoiceXML) Version 2.1, W3C Recommendation Web site.
ASR	If you plan to use Automatic Speech Recognition (ASR) technology in your speech application, you must adhere to the Automatic Speech Recognition (ASR). For more information, see the Speech Recognition Grammar Specification Version 1.0, W3C Recommendation Web site.
TTS	If you plan to use Text-to-Speech (TTS) technology in your speech application, you must adhere to the Text-to-Speech (TTS). For more information, see the Speech Synthesis Markup Language (SSML) Version 1.0, W3C Recommendation Web site.

Note:

Speech applications designed and created with the Orchestration Designer tool meet these requirements and recommendations.

License requirements

Before you configure Experience Portal, ensure that Avaya provides the following site-specific items:

- Product ID: The unique product ID for your site. This is a numeric identifier that must be provided when the EPM software is installed.
- The Experience Portal license file: Determines the maximum number of telephony ports available to the Experience Portal system, and whether the speech applications in the system can use ASR or TTS resources. The license file must be installed on the Avaya WebLM server.

Note:

Before upgrading the Experience Portal system to a newer version, you must upgrade the license to a newer version. If the Experience Portal system is upgraded to a newer version, for example, from 3.0 to 4.0 or from 4.0 to 5.0) and the license is not upgraded, the system provides a grace period of 30 days. During this grace period, you must

upgrade the license as per the following compatibility matrix otherwise the license grace period will end and the system will no longer be functional.

The compatible versions of Experience Portal and WebLM licenses are:

Voice Portal/Avaya Aura® Experience Portal Version	License Version
Voice Portal 3.x	All versions of licenses.
Voice Portal 4.x	4.0 or later versions.
Voice Portal 5.x	5.0 or later versions.
Avaya Aura® Experience Portal 6.0	6.0 or later versions.

- If Avaya Services maintains the Experience Portal system, then the Avaya Services representative should get:
  - The Avaya Service Account authentication file used to create Avaya Service accounts after the Experience Portal software is installed.
  - The Listed Directory Number (LDN) in the Avaya Services database for each EPM and MPP/AMS server, and each associated speech server.

Password requirements

During the installation, the system prompts you for several passwords. The passwords must:

- Be at least eight characters in length.
- Contain at least one alphabetic character and one digit.
- Not be the same as the associated user name.

Tip:

Passwords are case-sensitive, and you must use a combination of upper and lower case characters in your passwords.

External system requirement worksheets

External systems configuration worksheet

In order to work with Experience Portal, you need to set configuration options in the 3rd party products.

✔	Description
	You need at least one Windows system with a Microsoft Internet Explorer 6 (IE6) SP2 or later browser that is configured to use TLS security as described in Configuring browsers to use TLS security on page 40.
	For all speech servers running IBM WebSphere, Nuance OSR, and Nuance RealSpeak 4.0.12 speech servers with Red Hat Enterprise Linux, you need to set the LD_ASSUME_KERNEL environment variable to handle a multi-threaded environment as described in Configuring Red Hat Enterprise Linux Server 6.0 environment variables for speech servers on page 40.
	If you are running Orchestration Designer applications with a WebSphere Application Server (WAS) and Nuance speech servers, you need to configure the MIME type declarations as described in Configuring a WebSphere Application Server to work with Nuance speech servers on page 41.
	To use A-Law encoding with a Nuance server that supports Automatic Speech Recognition (ASR), you need to configure Nuance server as described in Configuring A-Law encoding for Nuance ASR servers on page 41.
	If you want to use Nuance SWI_rawScore, you need to configure additional parameters on the Nuance speech server as described in Adding support for Nuance SWI_rawScore.

Related topics:

Configuring browsers to use TLS security on page 40

Configuring Red Hat Enterprise Linux Server 6.0 environment variables for speech servers on page 40

Configuring a WebSphere Application Server to work with Nuance speech servers on page 41

Configuring A-Law encoding for Nuance ASR servers on page 41

Configuring parameters for getting recognition results from Nuance server on page 42

Configuring browsers to use TLS security

A web interface to the EPM for administering Experience Portal is included with the EPM software. To access the EPM Web interface, you must use a Microsoft Internet Explorer 6 (IE6) SP2 or later browser that is configured to use TLS security.

Procedure

1. In an IE browser window, select **Tools** > **Internet Options**.
2. Go to the **Advanced** tab.
3. In the **Security** section, ensure that the **Use TLS 1.0** check box is selected. If not, select the check box.
4. Click **OK**.

Configuring Red Hat Enterprise Linux Server 6.0 environment variables for speech servers

For all IBM WebSphere, Nuance OSR, and Nuance RealSpeak 4.0.12 speech servers running Red Hat Enterprise Linux Server 6.0 in the Experience Portal system, you need to set the LD_ASSUME_KERNEL environment variable to handle a multithreaded environment.

Procedure

1. If you are running IBM servers or Nuance servers started from the command line:
   a) On each speech server in the Experience Portal system, open the `/etc/profile` file in an ASCII editor.
   b) Add the line `Export LD_ASSUME_KERNEL=2.4.19v` to the file.
   c) Save and close the file.
2. If you are running Nuance servers as a Linux service:
   a) On each Nuance speech server in the Experience Portal system, open the `/etc/init.d/OSSservice` file in an ASCII editor.
   b) Add the line `LD_ASSUME_KERNEL=2.4.19v; export LD_ASSUME_KERNEL` to the file.
   c) Save and close the file.

Configuring a WebSphere Application Server to work with Nuance speech servers

If you are running Orchestration Designer applications with a WebSphere Application Server (WAS) and Nuance speech servers, you need to manually declare the grammars that Orchestration Designer uses on the WAS.

Procedure

1. Open a Web browser and go to `http://<WAS_ipaddress>:9090/admin`, where *<WAS_ipaddress>* is the IP address of your WAS server.
2. Log in as `AnyOne`.
3. Expand **Environment** in the left-hand pane.
4. Click **Virtual Hosts** in the expanded list.
5. In the right-hand pane, select the virtual host that manages your speech applications or, if you have not created a separate virtual host, select **default host**.
6. Click **MIME Types**.
7. Look for the `application/srgs+xml` MIME type. If it does not exist, click **New** and add it. If it does exist, select it and click **Edit**.
8. Add `grxml grammar` to the `application/srgs+xml` MIME type extensions.
9. Stop and then restart the WAS server.

Configuring A-Law encoding for Nuance ASR servers

If you want to use A-Law encoding with a Nuance server that supports Automatic Speech Recognition (ASR), you need to configure the additional parameters.

Procedure

1. On each Nuance server machine, log in to the operating system and navigate to the directory in which the Nuance `Baseline.xml` file is stored.
2. Open the `Baseline.xml` file in an ASCII editor.
3. Add the following additional value to *both* the `swirec_audio_media_type` and `swiep_audio_media_type` parameters:
   `<value>audio/x-alaw-basic;rate=8000</value>`
4. Save and close the file.

5. Restart the Nuance server.
6. Repeat this procedure for any other Nuance ASR servers in the Experience Portal system.

Configuring parameters for getting recognition results from Nuance server

You must configure parameters in the NSSserver.cfg and Baseline.xml files of the Nuance speech server to get the recognition results of *no match* from the Nuance server.

Before you begin

About this task

Make sure that you have installed the following applications:

- NSS - 5.0.7 or higher
- NRec - 9.0.11 or higher

Procedure

1. On each Nuance server machine, log in to the operating system.
2. Navigate to the `usr/local/Nuance/SpeechServer/server/config` directory in which the Nuance `NSSserver.cfg` file is stored.
3. Open the `NSSserver.cfg` file in an ASCII editor.
4. Define the values as given below:
   ```
   server.mrcp2.osrspeechrecog.mrcpdefaults.VSP.server.osrspeechrecog.result.sendnomatch VXIString true
   server.mrcp1.osrspeechrecog.result.sendnomatch VXIString true
   ```
5. Save and close the file.
6. Open the `Baseline.xml` file in an ASCII editor.
7. Define the value as given below:
   ```
   <param name=”swisr_result_enable_speech_mode"> <value> 1 </value> </param>
   ```
8. Restart the NSSservice.

Chapter 4: System Security

Security overview

The design of a self-service solution must include security considerations that are appropriate for your environment, to ensure:

- Sensitive customer data is not logged in plain text files
- Data is protected from unauthorized access and modification
- Applications do not inadvertently expose customer data
- Applications do not allow attackers access to the Private Branch Exchange (PBX)
- Machine operational status is not compromised through denial of service attacks

You can use the capabilities of the operating system or other custom-developed solutions to implement the required application-level security. Avaya realizes that many companies employ the use of third-party software to enhance system security. Any additional software that is installed on the system must be installed under a policy of permissive use. Avaya cannot ensure that such software does not affect the operation or performance capabilities of the Avaya Aura® Experience Portal system.

If you choose to install additional software, you must accept the responsibility of ensuring that it does not degrade system performance to an unacceptable level. Although you can choose to trade some system performance for the use of third-party applications, Avaya does not warrant that full system capacity be maintained. Furthermore, Avaya does not verify or ascertain the validity of third-party software unless prior business arrangements are made through Avaya. If you install additional software that causes problems on the system, Avaya might charge for any assistance required in troubleshooting the problem. Avaya might require that the software be removed before Avaya starts the troubleshooting process.

No telecommunications system can be entirely free from the risk of unauthorized use. You have the ultimate control over the configuration and use of the product and are solely responsible for ensuring system security. You can administer and tailor the system to meet your unique needs, and you are in the best position to ensure that the system is secure. You are responsible for keeping informed of the latest information, such as:

- Security patches
- Hot fixes
- Anti-virus updates

System managers and administrators are also responsible for reading all product recommendations, installation instructions, and system administration documents to

understand the risks and to identify any preventative measures that they should take in order to keep their systems secure.

Avaya does not guarantee that this product is immune from or prevents unauthorized use of telecommunications services accessed through or connected to this product. Avaya is not responsible for any damages or charges that result from unauthorized use of this product. Avaya also is not responsible for incorrect installations of the security patches that are made available. To aid in combating unauthorized use, Avaya maintains strong relationships with its customers and supports law enforcement officials in apprehending and successfully prosecuting those responsible.

Report suspected security vulnerabilities with Avaya products to Avaya by sending email to securityalerts@avaya.com. Reported vulnerabilities are prioritized and investigated. Any corrective actions resulting from the vulnerability investigation are posted at the Avaya online security Web site, http://support.avaya.com/security.

Whether or not immediate support is required, report all toll fraud incidents perpetrated on Avaya services to Avaya Corporate Security to securityalerts@avaya.com. In addition, for information concerning secure configuration of equipment and mitigation of toll fraud threats, see the *Avaya Toll Fraud and Security Handbook* at http://support.avaya.com/css/P8/documents/100073832 .

The Avaya Enterprise Security Practice, part of Avaya Network Consulting Services, can provide the following services to help protect against unanticipated threats and security hazards:

- Application assessment
- PBX assessment
- Network assessment
- Auditing
- Hardening services

For more information, or to contact the Avaya Enterprise Security Practice , call 1-866-832-0925.

If you want to perform the hardening steps, follow the steps described by the operating system manufacturer and security best practices. Security best practices are detailed in the National Security Agency Guides, http://www.nsa.gov/snac/.

In addition, to find related security advisories, report product vulnerabilities, and locate the latest software patches and upgrades, go to the Avaya online support Web site, http://support.avaya.com.

Secure system access

One key step in ensuring the security of a system is to the limit ways by which people can use the system. The following topics detail some of the ways you can limit access to Experience Portal:

- Physical system security
- Isolated LANs
- Firewalls

Physical system security

The Experience Portal system must be placed in a physically secure environment so that only a limited number of trusted people can use the system. Putting the system in a location that allows free access by anyone creates a risk that Experience Portal operation can be disrupted, whether unintentionally or maliciously. Isolate the Experience Portal system from everyone except trusted individuals.

Isolated LANs

Any server that is connected to the Internet is potentially subject to unauthorized use and malicious attacks. Experience Portal systems can be protected by configuring them on a LAN that has no physical connection to the Internet or to any internal unsecured networks. Sometimes referred to as an "island LAN," this type of network environment has its own LAN switch and contains only those network elements that the Experience Portal system needs to interface with. These elements include:

- Application servers
- Text-to-Speech (TTS) (TTS) and Automated Speech Recognition (ASR) servers
- Database servers, if used by the application
- PBX
- Backup server

If a LAN has no physical connection to the Internet, no risk of unauthorized access from external sources exist. As such, a firewall is not needed to protect the system from unauthorized use.

Physically isolating the LAN provides strong protection against fraudulent access. However, isolating the LAN can restrict the ability to remotely administer and maintain the Experience Portal system. Before deciding whether to place the Experience Portal system on an island LAN, you must consider the requirements of the operating environment.

Firewalls

If the LAN cannot be isolated, you can use firewall product to protect the LAN, and any Experience Portal servers connected to the LAN, from unauthorized access. The firewall should be installed on a machine that sits between the Internet and Experience Portal, so that all communication that comes into Experience Portal must first pass through the firewall.

A firewall also controls access of designated ports that use particular protocols or applications. They are commonly used to prevent the following:

- Denial of service attacks to application servers
- Snooping of sensitive data
- “Hijacking” access sessions that take control of a user session

  Session hijacking is the act of taking control of a user session after successfully obtaining or generating an authentication session ID. Session hijacking involves an attacker using captured, brute forced or reverse-engineered session IDs to seize control of a legitimate user's web application session while that session is still in progress.

Most firewalls can be configured to allow specified remote IP addresses to connect to designated ports by using specified protocols.

Even if a firewall protects the internal LAN, the Experience Portal system might still be accessible to unauthorized people who have access to the internal network. Therefore, you must still restrict access to the Experience Portal system in this environment to decrease the risk of fraudulent use by an insider. For more information about restricting access, see Account management on page 47.

Antivirus software

You can install antivirus software on the Experience Portal servers. The type of antivirus software used and the method of installation depends on the requirements of your company.

Make sure you use on-demand scanning, where scans are run at scheduled intervals. Do not use a message-scanning method, such as on-access scanning as that can impact the performance of Experience Portal. If your antivirus software runs whenever a file is changed, it can have a negative impact on Experience Portal performance.

In addition, some virus scan applications automatically start scanning at system startup by default. Disable this feature because it interferes with the time that it takes for an Experience Portal system to come back online after a reboot.

You must administer the antivirus software as follows:

- Scan the hard disk daily during off-peak hours, or at least once per week. Scans can be run on all Experience Portal servers simultaneously. Do not schedule the antivirus scan at the same time as a backup.
- Schedule antivirus definition updates to occur automatically at least once per week. The updates must occur before the next scheduled scan time to ensure that the latest data files are used during the scan. Do not schedule updates to occur during a virus scan.
- If the antivirus software detects a virus, it must attempt to clean the file. If the attempt fails, the software must move the infected file to a different directory on the server.

Administering accounts and passwords

Account management

You must follow the same practices for Experience Portal administrative accounts as you do for any proprietary enterprise system. These practices must be implemented as part of the operational procedures and must include the following management strategies:

- Minimize the number of accounts, especially privileged accounts.
- Strictly limit privileged accounts, such as `root`, Administration, and User Manager to those people who have a business need for access.
- Do not set up user accounts with a user ID of 0. User ID 0 designates the root login account.
- Use unique user IDs for each user account.
- Make sure that the passwords associate with each account are secure, as described in Password administration on page 47.
- Delete logins if they are not used for a specified number of days or if the user leaves the company.
- Review account information, such as permissions, ownership, and unexpected changes, on a regular basis.
- Review the Audit Log report for unusual activity such as:
    - Login failures
    - Unexpected user logins
    - Unexpected log-in times
    - System processes that should not be running

Password administration

Passwords are keys to an Experience Portal system. They must be protected and *strong*. A strong password is one that is not easily guessed and is not listed in any dictionary. Protected and strong passwords are especially important for root and administrative-level passwords

since they have no access restrictions. Passwords created during Experience Portal installation are checked for minimal characteristics as follows:

- Passwords must contain at least one alphabetic character and one digit.
- Passwords are case-sensitive and should contain a combination of upper and lower case letters.
- Passwords cannot include any special or accented characters.
- A password cannot be the same as its associated username.
- Although you can determine the minimum password length, you should not use any fewer than eight characters.

After installation, when you use the EPM to create additional user accounts, the minimal characteristics for passwords are enforced. However, administrators can customize the minimum password length. You must set this value to at least eight characters.

To ensure that strong passwords are created, you must use a nonsensical combination of letters and digits when creating passwords.

User authentication

A user must be authenticated before gaining access to the Experience Portal system. The combination of a username and password confirms or authenticates the user. Authentication is required before accessing the EPM, Experience Portal database, and the MPP Service Menu/ AMS.

Experience Portal administrators can limit failed log in attempts to prevent unauthorized users from guessing passwords to gain access to the EPM.

To prevent unauthorized users from using Experience Portal, you can specify the following:

- The number of successive failed log in attempts before the system locks the account.
- The amount of time to lock out users who do not successfully log in within the number of defined log in attempts.
- The number of successive failed log in attempts before the system triggers an alarm.

Role-based authorization for system administration

The Experience Portal system provides role-based authorization for controlling access to the EPM for system administration. Role-based authorization controls which users are allowed to administer the Experience Portal system. These roles can be administered on the EPM.

User roles define access to Experience Portal web pages and the ability to make changes to the system based on the role assigned to the user account. For a complete description of the

roles in Experience Portal, see the *User Roles* topic in the *Administering Avaya Aura® Experience Portal* guide.

Root access security

Root and administrator logins have the highest level of authority (or privilege) on the Experience Portal system. Root and administrator access can modify any capabilities and features on the system. Therefore, you must control access to these logins. You must provide root and administrator login access information only to a limited number of trusted people.

In addition, the Experience Portal system must be administered so that direct root logins are restricted to the system console only. This is the default configuration on all Experience Portal systems.

Restricting direct root access to the console requires users to have physical access to the system. Remote users must log in as another user and then use the **su** command to log in as root. Restricting root access provides an extra measure of security, since remote users must authenticate themselves twice. Remote users must enter their normal user login and then a second password for root access. In addition, all use of the **su** command is logged for accountability.

Network services

Network services are subject to security vulnerabilities which unfortunately allow unauthorized users to gain access to the system. The Experience Portal system uses relatively few network services, and several unneeded services and ports are disabled during the installation of Avaya Enterprise Linux as part of the bundled server offer.

The network services that are enabled during Avaya Enterprise Linux and Experience Portal installation are:

- Secure Shell (SSH) (server-side), which runs on all Experience Portal servers.
- Apache Tomcat, which runs on the EPM server. Tomcat is a J2EE compliant servlet container and is the default application server for the EPM.
- Network Time Protocol (NTP), which runs on all Experience Portal servers.
- PostgreSQL (SQL server), which runs on the EPM server. Postgres is an SQL compliant, open source, object-relational database management system for the Experience Portal database.
- Apache HTTPD, which runs on the MPP servers. The MPP servers use the Apache Web Server to implement web services for EPM monitoring and control and the MPP Service Menu.

For more information about how Experience Portal protects sensitive data, see the Avaya Aura® Experience Portal 6.0 Security White Paper in the Print guides section of the Avaya

Aura® Experience Portal Documentation Library. For more information about how Experience Portal protects sensitive data, see the Avaya Aura® Experience Portal 6.0 Security White Paper.

Related topics:

Secure Shell on page 50
Network Time Protocol on page 50

Secure Shell

Secure Shell (SSH) is a program that includes capabilities for doing the following:

- Logging in to another computer over a network
- Executing commands on a remote computer
- Moving files from one system to another

Secure Shell provides strong authentication and secure communications over untrusted networks. Secure Shell provides a more secure way to connect to remote systems than protocols such as telnet and FTP. Unlike telnet and FTP, users can connect to remote hosts over an encrypted link with SSH. Encryption protects against interception of clear text logins and passwords.

Network Time Protocol

If your Experience Portal system is configured to use a dedicated EPM server and one or more dedicated MPP servers, Experience Portal uses Network Time Protocol (NTP) to synchronize the time between the EPM server and all other Experience Portal servers.

In order to do so, the Experience Portal software installer changes the `ntp.conf` file on each server on which the software is installed. When you install the:

- Primary EPM software, the `ntp.conf` file on that server is set to point to the local clock.
- MPP software or the auxiliary EPM software, the `ntp.conf` file on that server is set to point to the primary EPM server as the reference clock.

Linux hardening efforts

The general distribution of Red Hat Enterprise Linux includes the Red Hat Package Management (RPM) modules for most, if not all, possible Linux configurations. These

distributions include a complete development suite, complete graphics support for the X Windows System, numerous development debugging tools and a variety of network administrative tools. For Experience Portal, only a small portion of the distributed RPMs is needed. When distributions of Red Hat Enterprise Linux grow to include more RPM modules, the relative percentage of RPMs needed by Avaya applications will be even smaller.

Experience Portal does not require most packages provided in the general distribution, and these unused RPMs are removed from the Avaya Enterprise Linux.

Aside from making the software product file images smaller and more manageable, the removal of unneeded RPM modules makes Linux more secure.

To make Linux even more secure, you must configure Linux to log security-related events, if possible. You must log the following events:

- Account privilege changes
- Logins and logouts
- System configuration changes
- Additions, modifications, or deletions of installed packages
- Activities of root or administrative logins

SNMP Agents and Traps

The Avaya Aura® Experience Portal Simple Network Management Protocol (SNMP) network includes agents, traps, and managers.

SNMP agents

You can configure Experience Portal to act as an *SNMP agent* so that a third party network management software can retrieve the Experience Portal system status.

An SNMP agent is a software module that resides on a device, or node, in an SNMP-managed network. The SNMP agent collects and stores management information and makes this information available to *SNMP managers*. SNMP agent communication can be:

- Solicited by an SNMP manager.
- Initiated by the SNMP agent if a significant event occurs. This type of communication is called an *SNMP trap*.

The commands and queries that the SNMP agent can use, along with information about the target objects that the SNMP agent can interact with using these commands and queries, is stored in a Management Information Base (MIB) that resides on the managed device.

SNMP traps

An SNMP trap is an unsolicited notification of a significant event from an SNMP agent to an SNMP manager. When an internal problem is detected, the SNMP agent immediately sends one of the traps defined in the MIB.

Important:

If you configure Experience Portal to send SNMP traps, you must configure the appropriate SNMP managers to receive those traps.

SNMP managers

SNMP managers collect information from SNMP agents. SNMP managers are usually used to display status information in a type of graphical user interface (GUI).

For Experience Portal, the SNMP manager can be an Avaya Services Security Gateway (SSG) or a Network Management System (NMS) station such as HP *OpenView* or IBM *Tivoli*. SNMP traps sent to the Avaya SSG contain specific information that generates Initialization and Administration System (INADS) notifications, which in turn generate customer trouble tickets.

Note:

You can only configure the Experience Portal SNMP agent and SNMP trap destinations if you are an administrator.

Secure Sockets Layer

Secure Sockets Layer (SSL) is a protocol that provides a mechanism for securely transmitting documents over the Internet. The protocol allows client/server applications to use encrypted transmissions to perform client/server authentication. SSL:

- Moves the traffic from port 80, which is used for http, to port 443 for https.
- Exchanges a set of keys for encryption and authentication and encrypts all traffic on that port until the session completes.
- Helps prevent eavesdropping, tampering with transmissions, and message forgery

Experience Portal provides SSL support for the following:

- EPM administration traffic runs over an SSL/HTTPS connection. Using an SSL/HTTPS connection ensures that no web administration data is transmitted in clear text. Encrypted data includes logins and passwords, configuration changes, and views of the Experience Portal system configuration.
- The EPM software must authenticate itself with the MPP before the MPP accepts any requests. Similarly, the MPP must authenticate itself with the EPM. All communication between the EPM and an MPP uses SSL/HTTPS.
- You can configure the Avaya Voice Browser to use SSL to access an application on the web server. In this case, VoiceXML data is transmitted in an encrypted format instead of clear text.

Note:

Although Experience Portal provides the framework for using SSL for VoiceXML, you must install an SSL certificate for each web server domain referenced by an application to fully implement client authentication using SSL for VoiceXML.

Data transmission

When sending sensitive data from one place to another, use care because transmissions can be intercepted. Risks arise when transmitting data in clear text. Whenever you have the option, consider encrypting the data you are transmitting.

Data encryption

By design, communication between the EPM server and the MPP server is always encrypted. However, you have the option of enabling or disabling encryption for other types of data transmissions. Encrypting communication is more secure for your system, but keep in mind that encryption can slow system response times.

To encrypt the H.323/RTP media streams between an MPP and the PBX, use the encryption standard supported by the switch or gateway. In an Experience Portal system, Communication Manager supports the 128-bit Advanced Encryption Standard. After enabling encryption on the switch, you use the web interface to the EPM to enable encryption on Experience Portal.

Nonsecure data transport

Certain Avaya and third-party products with which Experience Portal interacts do not support secure data transport. Since these remote systems do not yet support this capability, there is not a secure communication link between the following:

- IC connector in a Orchestration Designer application and the Avaya Interaction Center.
- Computer Telephony Integration (CTI) connector in a Orchestration Designer application and the Avaya Computer Telephony (CT) telephony server.
- MPP and the TTS and ASR speech servers.
- Orchestration Designer application reporting and the EPM.

Note:

If this data needs to be secure, a private LAN card can be used to isolate the data from other Experience Portal traffic.

Avaya Secure Access Link (SAL) and Access Security Gateway (ASG)

Dial-in lines on the Experience Portal system can be protected by an Avaya-developed solution called Access Security Gateway (ASG). The ASG package is integrated into the Experience

Portal system and provides secure authentication and auditing for all remote access into the maintenance ports.

ASG authentication is based on a challenge/response algorithm using a token-based private key-pair cryptographic authentication scheme. Secure auditing is also provided. Logs are available that include information such as successful logins, failed logins, errors, and exceptions.

Although Experience Portal dial-in lines are protected by ASG, concerns might exist about the potential security risks of having a modem connected to the Experience Portal system at all times. If this is an issue, secure the modem by turning it off and only turning it on when service is required. However, this approach makes it more difficult for Avaya technicians to respond to trouble escalations and service requests.

If you want to turn off the modem, contact Avaya so that the appropriate information can be put in special handling notes in the Avaya support organization's tracking database. Also ensure that a local support contact is available who can activate remote access if an Avaya technician needs to troubleshoot or service the Experience Portal system.

System recovery

As with any other application running on a server, being prepared to do a partial or complete Experience Portal system restoration if a disaster occurs is important.

Experience Portal systems should be backed up regularly. Experience Portal includes backup scripts that can be run automatically as a Linux `chron` job and that can perform either full or partial backups on a regular basis.

You should also document the components and settings for the Experience Portal system to facilitate the efforts required to restore the systems. These system records should include the following information:

- Experience Portal customer identification number (CIN), installation location (IL), IP address of the network interface card (NIC), dial-up number of the modem, telephone numbers for test calls, and sample account numbers for testing.
- Server names and IP addresses of all Experience Portal system servers, speech servers, database servers, and Application servers.
- A current list of all software, including versions, installed on the system. The software itself should be stored in a safe and easily accessible location.
- Disk partitioning information, so that applications can be restored to the correct locations.
- Information about what needs to be done to restore each application package. All values and parameters that must be entered should be recorded.

- Changes to system defaults.
- Contact information for Avaya as well as for any application vendors, speech vendors, and database vendors that may have provided components used on or with the Experience Portal system.

The Avaya Business Continuity Services can help design and implement disaster recovery plans to support rapid recovery from outages caused by unforeseen circumstances such as natural disasters or other emergency situations. A well designed disaster recovery plan can help reduce expenses by proactively identifying potentially costly issues related to topology, hardware, software, security, network performance, and business resiliency. For more information about Avaya Business Continuity Services, contact Avaya Support.

Chapter 5: Designing speech applications to run in Avaya Aura Experience Portal

Speech applications in Avaya Aura® Experience Portal

Speech applications are the “directors” of Avaya Aura® Experience Portal system operations. When a caller dials in to the system, the Media Processing Platform (MPP) accesses the appropriate speech application to control the call. From that point on, the speech application directs the flow of the call until the caller hangs up or the application is finished.

Experience Portal systems can have more than one application active and available at a time. The MPP that takes the call accesses the appropriate application based on the Dialed Number Identification Service (DNIS).

Note:

An application does not have to have a DNIS assigned to it. In this case, such an application handles any call that comes in to the system by means of a DNIS that is not assigned to any other application on the system. However, you can only have one such application on the system. If you attempt to configure a second application without a DNIS, the system generates an error.

In addition, if the speech application requires Automatic Speech Recognition (ASR) or Text-to-Speech (TTS) resources, the MPP contacts the appropriate ASR or TTS server through an Media Resource Control Protocol (MRCP) proxy server.

Call flow example

This call flow example shows how the Experience Portal system interacts with other systems to handle an automated telephone transaction.

1. A caller from the Public Switched Telephone Network (PSTN) dials a telephone number.
2. The PSTN routes the call to the Private Branch Exchange (PBX) associated with that number.

3. Using Voice over IP (VoIP), the PBX breaks the voice data into packets and sends them over the LAN to a Media Processing Platform (MPP) server in the Experience Portal system.
4. The MPP server looks at the Dialed Number Identification Service (DNIS) for the incoming call and uses the configuration information downloaded from the EPM server to match the number to a speech application that has been added to Experience Portal.
5. The System Manager starts an Avaya Voice Browser session and passes it the Universal Resource Indicator (URI) specified for the selected speech application.
6. The Avaya Voice Browser contacts the application server and passes it the URI.
7. The application server returns a VoiceXML page to the Avaya Voice Browser.
8. Based on instructions on the VoiceXML page, the MPP uses prerecorded audio files, Text-to-Speech (TTS), or both to play a prompt to start interaction with the caller.
9. If the caller responds by:
    - Entering Dual-tone multi-frequency (DTMF) digits, the MPP establishes a connection to a TTS server and the ASCII text in the speech application is forwarded for processing. The TTS server renders the text as audio output in the form of synthesized speech which the MPP then plays for the caller.

      Note:

      This connection requires one TTS license, which is released as soon as processing is complete.
    - Speaking, the MPP establishes a connection to an Automatic Speech Recognition (ASR) server and sends the caller's recorded voice response to the ASR server for processing. The ASR server then returns the results to the application for further action.

      Note:

      This connection requires one ASR license, which is *not* released until the entire call is complete.
10. If errors are encountered during the call, how these errors are handled depend on the type of grammar used by the application. If the application grammar is:
    - Dynamic, or In-line, the speech server gets the grammar directly from Experience Portal and any error messages are passed back to Experience Portal.
    - External or static, the speech server asks for the grammar using the URL specified in the application. If the URL points to an application server, the speech server interacts directly with that application server. Because Experience Portal is not involved in this communication, any error messages passed back by the application server may not be passed back to Experience Portal.

11. The application terminates the call when it finishes execution or when the caller hangs up.
12. When the call ends, the PSTN clears the call from the PBX and releases the ASR license if one was required.

Speech application development tools

Any speech application that is compliant with the VoiceXML Version 2.1 Recommendation or Call Control eXtensible Markup Language (CCXML) will run in an Experience Portal system, regardless of the tool in which the application was created. However, you must create your speech applications with Orchestration Designer.

Orchestration Designer is an Eclipse plug-in that provides an integrated GUI for application design and implementation. It creates speech applications that automatically conform to the Experience Portal requirements and recommendations.

In addition, Experience Portal automatically includes all Orchestration Designer applications in the Application Summary report and Application Detail report. If you want these reports to display messages and status information from an application developed in a third-party tool, you must manually log the messages and status information from that application using the Application Logging web service.

Deploying a speech application

About this task

You must deploy a speech application to an application server connected to your Experience Portal system before you can add it to Experience Portal.

Procedure

1. Create the speech application.

   For design guidelines, see Design for user experience on page 61.
2. Package the application for deployment.

   For more information about packaging applications for deployment on Apache Tomcat or IBM WebSphere, see Tomcat and WebSphere speech application deployment guidelines on page 60.
3. Copy the speech application package file to the Application server from which the application will run.

Security alert:

Remember to exercise and observe appropriate security measures when transporting application package files to the application server.

4. If necessary, take steps to deploy the application on the application server.

   Some application server environments require that you take additional steps to deploy the speech application after the application files have been copied to the application server. This might include installing any run-time support files that the application requires. For more information about support files required by your application server, see the documentation for your server.

Related topics:

Tomcat and WebSphere speech application deployment guidelines on page 60

Tomcat and WebSphere speech application deployment guidelines

You can deploy Orchestration Designer speech applications on a Tomcat or WebSphere application server. If you want to use a different application server, consult your server documentation for deployment requirements.

Apache Tomcat deployment guidelines

To deploy a speech application to an Apache Tomcat application server, you must package the application within a standard Web Archive (WAR) file. A WAR file is a compressed set of files, similar to a ZIP file. The WAR file format is specified by the J2EE specification and all J2EE-compliant application servers should support this format. The Tomcat servlet engine is optimized to handle WAR files. For more information about WAR file requirements for speech applications on your Apache Tomcat system, see the documentation for your system.

When you transport the WAR files to your Apache Tomcat application server, copy them to the directory *TomcatHome*\webapps\, where *TomcatHome* is the directory in which your Apache Tomcat application server software is installed.

Then, when you next start Tomcat, Tomcat automatically installs and deploys the application.

Important:

If you are redeploying an existing application, make sure the original application is not running before you deploy the new version on the server. If the original application is running, there could be conflicts with the log files.

IBM WebSphere deployment guidelines

In order to deploy a speech application on an IBM WebSphere or WebSphere Express application server, you must package the application within a standard Enterprise ARchive (EAR) file or Web ARchive (WAR) file with the JDK source level set to 15. These files are compressed sets of files, similar to a ZIP file. IBM WebSphere servlet engines can use either

EAR or WAR file formats. For more information about EAR or WAR file requirements for speech applications on your IBM WebSphere system, see the documentation for your system.

When you transport the EAR or WAR files to your IBM WebSphere application server, make note of the directory to which you copy them. Then later, use the WebSphere Administrative Console to actually deploy the application from this location. For more information, see your IBM WebSphere documentation.

Important:

If you are redeploying an existing application, make sure the original application is not running before you deploy the new version on the server. If the original application is running, there could be conflicts with the log files.

Speech application design guidelines

Best practices for speech application design

Sound files

High quality stereo sound files may appear to be the perfect way to communicate with your customers, but you must remember that these files will be played over a telephone line which has only 8 KHz bandwidth. Most of the higher frequencies are lost when a recording is played over the telephone.

When you record sound files:

- Record monaural rather than stereo.
- Record at and even multiple of 8 KHz. For example, you could record at 8 KHz, 16 KHz, or 24 KHz,.
- Use 16 bit-depth A-to-D conversion.
- Use a quality audio editing program to normalize the amplitude of your various phrases and convert to mu-LAW or A-LAW PCM.

When you have finished recording, listen to each recording in its final format. If you use the above guidelines, the way they sound at this point will be very close to how they sound over the telephone.

Design for user experience

The best first step in planning an application is to envision exactly what you want the caller to experience when calling in to your system. Do not consider how to set up the application. Simply ask and try to answer as many questions as you can.

For example, ask and try to answer the following questions:

- What options do you want to offer callers?
- Do you want to offer callers the opportunity to interact in more than one language?
- What means do you want to offer callers to respond to options? By voice? By using touchtone keys on the telephone? By recording their answers or other short messages?
- What voice gender do you want to use in presenting your prompts?
- Do you need to use Text-to-Speech (TTS) technology to provide the caller a way to hear text-based information?
- What about hearing- or speech-impaired callers? How do you want to provide for their special needs?

Again, the idea is to ask as many questions as you possibly can. Create sample scenarios for the various situations you think callers might require help with. Try to be as comprehensive as possible.

Design for potential problems

One of the most important steps in planning a good speech application is to plan for any potential problem and error condition you can think of and to include error handlers that can deal with these issues. For example, how should the system respond when one of these problem situations arises?

- Technical or hardware limitations. What if the caller does not have a touchtone telephone? How does your system respond to TTY or TDD requests?
- Accessibility for callers with physical limitations. Have you allowed for the extra time it can take for callers with physical handicaps or other limitations?
- Language limitations. Is it likely that a caller will need to interact using a different language than the primary language? Do you have the necessary Automatic Speech Recognition (ASR) and Text-to-Speech (TTS) servers and software to accommodate them?
- Personal preferences. Have you allowed for the personal preferences of your callers? Some people would rather interact with the system verbally, while others can prefer to use touchtone, or Dual-tone multi-frequency (DTMF) , responses. Other people prefer to interact with a live attendant no matter how good your Interactive Voice Response (IVR) speech application is. How easy is it for such users to get to a live attendant?

If an application encounters an error that is not handled by its own error handlers, or if an application cannot be started due to problems with the application server or the speech servers, Experience Portal uses the error handlers installed on the MPP server. In addition to designing the speech applications, you should also design the error handlers that Experience Portal uses in these cases.

For example, you want your default event handler to:

1. Play a prompt explaining that there was a problem and that the customer is being redirected to an agent immediately.
2. Transfer the call to a special number reserved for such issues.

A call coming in on this special number alerts the agent that the caller has encountered and error in Experience Portal, and that the agent should find out what the customer was doing when the error occurred. The call center can then track these exceptions and fix areas that encounter frequent problems.

For more information, see Experience Portal event handlers on page 63.

Related topics:

Experience Portal event handlers on page 63

Experience Portal event handlers

When a CCXML speech application tries to access an HTML page that cannot be found or a VoiceXML application encounters an unexpected event, the application responds with an exception error message. A well-designed speech application includes exception handlers that deal with these messages and help the application recover so that it can continue processing the call.

Experience Portal uses the error handlers defined in the application whenever possible. However, if an exception error message occurs that is not handled by the application, or if there is a problem running the application due to issues with the application server or the speech servers, Experience Portal uses one of the event handlers installed on the MPP server.

The event handler Experience Portal uses depends on the state of the speech application. If an application:

- Was successfully started and there is a call in progress, Experience Portal uses the event handler associated with that application when it was added to the Experience Portal system.
- Could not be started, Experience Portal looks at the type of application that was requested and uses the appropriate default CCXML or VoiceXML event handler.

When you install the software, Experience Portal automatically installs default event handlers for CCXML and VoiceXML, as well as an event handler prompt that is played by the default event handlers.

If you want to customize the way Experience Portal reacts to a problem, you can add your own event handlers and prompts and then designate which ones Experience Portal should use as the default.

For example, you want your default event handler to:

1. Play a prompt explaining that there was a problem and that the customer is being redirected to an agent immediately.
2. Transfer the call to a special number reserved for such issues.

A call coming in on this special number alerts the agent that the caller has encountered and error in Experience Portal, and that the agent should find out what the customer was doing

when the error occurred. The call center can then track these exceptions and fix areas that encounter frequent problems.

Design for application flow

You have envisioned the experience you want your callers to have. You have tried to foresee and plan for any problem contingency that might arise. Now you are ready to start actually mapping the flow of your speech application. A number of methods can serve you well in this effort, but here are a couple of the more common approaches:

- Describe the flow verbally. Talk through each of your scenarios verbally. Make sure you take note of where the prompts occur and what you want callers to say or do. Record these verbal “walkthroughs”.
- Use a flow diagram. As you work through your scenarios, you can create a flow diagram to show the major points in the call flow. Use this diagram to show such things as:
  - Where you want to offer options to callers
  - Where you want them to listen to the entire prompt and where they can interrupt, or "barge in", and cut the prompt off
  - Where you require a response from callers and what the valid responses will be
  - Where you want or need to access databases to retrieve or record customer data
  - How and where you want to access a Web service to respond to a customer request

Again, the idea is to be as complete and comprehensive as possible. Try and foresee every eventuality, and map how the system will respond.

Design for modularity

When you are planning your speech application, be alert to places where you can reuse parts of the application in two or more places. Then, when designing and building the application, you can create these parts as modules that you can reuse wherever you need that functionality.

If you plan for these modules ahead of time, you can also develop them before developing your main application project file. That way, they are already available when you create your main application.

For example, you might want to collect bank account or credit card numbers from callers at several points in the call flow. You have figured out that it is the same basic process each time you need to collect such numbers. Therefore, you might want to create a speech project module that you can reuse in your master application whenever you need to collect this type of information from callers.

Using this modular approach to application design has several advantages:

- You can "develop once, use many times." This can be a tremendous advantage, especially if you have certain actions or options you want to offer in several places to your callers.
- It is easier to maintain the overall application, even if you are not reusing much of the code. When you use a modular approach, you can change one part of the application without necessarily having to rebuild the entire application.
- It can make it easier to debug your applications, by making it possible to isolate the trouble spots where errors are occurring.

A team of developers can work on separate pieces of an application separately and then merge their efforts.

Design for application resources

As a final step in planning your speech application, you must attempt to identify and list all the application resources you will need to develop the application. Application resources include components such as Automatic Speech Recognition (ASR) and Text-to-Speech (TTS) servers, prerecorded phrases that the system plays back as prompts, and so on.

If you have planned the flow completely, keeping in mind all the resources you will need, you can list those application resources before you actually start to develop the application. Then you can create or import those resources before creating the call flow.

In the case of phrases, for example, if you know exactly what phrases you want to use, and if you plan to have those phrases recorded by a professional talent, you can list all the phrases and have them recorded as WAV files before you start to develop the actual application. Then, when you get to the points in the application where you need the phrases, you can import the prerecorded files.

CCXML and VoiceXML considerations

Keeping CCXML application sessions active until the MPP grace period expires

About this task

When you stop, restart, or halt an MPP, any CCXML application currently running on that MPP is sent a `ccxml.kill` event. This event notifies the application that the MPP is shutting down when the grace period expires.

If you do not define an error handler for this event in your CCXML application, the application exits immediately, even if it still processing a call.

Procedure

To keep the CCXML application active until the grace period has expired or until the application encounters an `</exit>` tag, add the following event handler to your CCXML application:

```
<transition event="ccxml.kill">
<!-- Do nothing.  Platform will kill session
after grace period-->
</transition>
```

Restrictions for dialogs attached to CCXML conference calls

Dialogs attached to conference calls may only play audio or text-to-speech prompts. If an attached dialog attempts to perform speech recognition processing, each attempt will result in a `no resource` error.

If you need speech recognition capabilities, you must attach the dialog to one of the calls that is part of the conference. In this case, however, the dialog can only process the speech that comes from the call to which it is attached.

Speech recognition results and VoiceXML applications

VoiceXML provides shadow variables accessible within a page to access recognition results such as `application.lastresult$`. While this includes a mechanism for presenting multiple possible semantic matches (n-best results whose presence can be determined by inspecting `application.lastresult$.length`), unfortunately the specification does not describe how to present a result that contains multiple interpretations for a given semantic match.

Avaya Aura® Experience Portal addresses the issue of a result containing multiple interpretations for a given semantic match by exposing the shadow variable `application.lastresult$.interpretation$`. To test for the presence of multiple interpretations in a result, examine `application.lastresult$.interpretation $.length`.

The following shows a sample grammar in Nuance OSR grammar format:

```
<grammar xmlns="http://www.w3.org/2001/06/grammar" xml:lang="en-US" version="1.0"
mode="voice" root="mult">
<rule id="mult" scope="public">
  <item>
    <one-of>
      <item> Stargazer <tag>SWI_meaning='eyes'</tag></item>
      <item> Stargazer <tag>SWI_meaning='look'</tag></item>
      <item> starchaser <tag>SWI_meaning='legs'</tag></item>
      <item> starchaser <tag>SWI_meaning='run'</tag></item>
    </one-of>
  </item>
```

```
</rule>
</grammar>
```

If the VoiceXML page containing that grammar uses the `maxnbest` property of `2`, then the recognition results arising from a caller saying “star gazer” would be accessed as follows in the VoiceXML specification:

```
application.lastResult$.length              2
application.lastResult$[0].confidence       0.83
application.lastResult$[0].utterance        Stargazer
application.lastResult$[0].interpretation   eyes
application.lastResult$[1].confidence       0.12
application.lastResult$[1].utterance        starchaser
application.lastResult$[1].interpretation   legs
```

Even though the recognition results contain the additional interpretations “look” and “run”, the VoiceXML specification makes no provisions for making those additional interpretations available to the application. To accomplish this, Avaya Aura® Experience Portal extends this list of shadow variables as:

```
application.lastResult$[0].interpretation$.length   2
application.lastResult$[0].interpretation$[0]       eyes
application.lastResult$[0].interpretation$[1]       look
application.lastResult$[1].interpretation$.length   2
application.lastResult$[1].interpretation$[0]       legs
application.lastResult$[1].interpretation$[1]       run
```

Privacy feature support for VoiceXML applications

In VoiceXML applications, you can declare a form or field to be private using the following property statement:

```
<property name="private" value="true"/>
```

While a private form or field is executing, Experience Portal does not write any speech recognition results, DTMF results, or TTS strings into the session transcription logs or into any alarms that may be generated during execution. Once the private form or field has finished executing, Experience Portal resumes logging these items as normal.

Note:

If tracing is turned on for the MPP, Experience Portal ignores the privacy property and writes the requested debugging information into the MPP trace logs.

Call classification in speech applications

Call classification overview

Call classification, or call progress, is a method for determining who or what is on the other end of a call by analyzing the audio stream. When the analysis determines a probable match, the CCXML page receives a `connection.signal` event which contains the result in the `event$.info.callprogress` variable. Depending on the results of the analysis, call classification may continue for the duration of the call. In that case, the CCXML page will may receive multiple `connection.signal` events.

Call classification falls into two categories:

- Tone-based classification. This category has a high degree of accuracy because it is easy for the system to detect busy signals or fax machine signals.
- Speech-based classification. This category as a much lower degree of accuracy because it can be difficult to tell a human being's speech from an answering machine's recorded message.

Experience Portal uses speech-based classification when it detects an amount of energy in a frequency consistent with human speech. When it detects human speech, it differentiates between a live human being and an answering machine based on the length of the utterance. Generally, if a live human being answers, the utterance is relatively short while answering machine greetings are usually much longer.

For example, a live human might just say “Hello.” while an answering machine message might say “Hello. I can't come to the phone right now. Please leave your message after the beep.”

However, some people may answer the phone with a much longer greeting while others have recorded a very short answering machine greeting. In both of these cases, the call classification algorithm will incorrectly identify the call and assume the long greeting is a recorded message while the short greeting is a live human being.

If you want to use call classification in your applications, make sure you design the application so that it takes such mistakes into account.

Call classification analysis results

When Experience Portal classifies the call, it sends a `connection.signal` event to the CCXML page. This event includes the results of the classification in the `event $.info.callprogress` variable.

Value of event $.info.callprogress	Applies to	Description
`live_voice`	Outbound calls only	The call is probably connected to a human being. No further classifications will be sent for this call.
`busy_tone`	Outbound calls only	A busy tone was received. This is commonly referred to as the “slow busy” tone. No further classifications will be sent for this call.
`reorder`	Outbound calls only	A switch error, such as all circuits being busy, has occurred. This is commonly referred to as the “fast busy” tone. No further classifications will be sent for this call.
`sit_tone`	Outbound calls only	A special information tone was received which indicates that the call could not be completed. This is the three frequency tone that is often followed by a spoken message. No further classifications will be sent for this call.
`recorded_msg`	Outbound calls only	An answering machine was detected and a recorded message has just started. This classification will always be followed by either a `msg_end` or `timeout` classification.
`msg_end`	Outbound calls only	The end of a recorded message, such as that played by an answering machine, was detected. No further classifications will be sent for this call.
`fax_calling_tone`	Inbound calls only	A fax machine was detected as the initiator of an inbound call. No further classifications will be sent for this call.
`fax_answer_tone`	Outbound calls only	A fax machine was detected as the recipient of an outbound call. No further classifications will be sent for this call.
`ringing`	Outbound calls only	A “ring back” tone was detected. One or more of these classifications may be received by the application, but they are always followed by one of the other applicable classifications.
`timeout`	Inbound and outbound calls	The classification algorithm failed to classify the call before the allotted time ran out.

Value of event $.info.callprogress	Applies to	Description
		By default, the allotted time is 20 seconds (or 20000 milliseconds). The default value can be overridden for outbound calls by setting the `call_classification_timeout` parameter in the `hints` attribute on the `<createcall>` tag to the desired number of milliseconds before call classification analysis should time out. No further classifications will be sent for this call.
`error`	Inbound and outbound calls	An internal error occurred during the classification analysis and the call could not be properly classified. No further classifications will be sent for this call.
early_media	Outbound calls only	Early media refers to media that is exchanged before a particular session is accepted by the called user. For example, for outbound calls the color ring tone will be detected as early_media by call classification.

Call classification for inbound calls

The only call classification provided for inbound calls is the ability to detect an incoming fax. It is extremely difficult to differentiate between a live human being and a recorded message for an incoming call because you cannot assume the initial greeting will be as short for an incoming call as it is for an outgoing call. Therefore, any classification for an incoming call other than `fax_calling_tone` should be treated as if a live human being was detected.

When you add an application to Experience Portal using the EPM Web interface, the following parameters in the **Advanced Parameters** group on the Add Application page determine what happens when an incoming fax machine call is detected:

Parameter	Description
Fax Detection Enabled	The options are: • **Yes**: The application should attempt to identify whether the caller is a fax machine and route any fax machine calls to the telephone number specified in **Fax Phone Number**. • **No**: The application should not attempt to identify whether the caller is a fax machine. The default is **No**.
Fax Phone Number	If **Fax Detection Enable** is set to **Yes**, this is the telephone number or URI to which fax machines calls should be routed.

Call classification for outbound calls

The application designer needs to enable call classification for outbound calls. If the call is going to be invoked by the Application Interface web service`LaunchVXML` method, you can specify the call classification parameters when you invoke the web service, as described in the *Call classification with the LaunchVXML method* topic in the *Administering Avaya Aura® Experience Portal* guide.

Otherwise, the application designer has to set `enable_call_classification=true` in the `hints` attribute of the `<createcall>` tag.

When call classification is enabled, a call will receive one or more `connection.signal` events containing the `event$.info.callprogress` field. This field will have one of the values described in Call classification analysis results on page 69.

Note:

Remember that the page may receive `connection.signal` events that do *not* contain the `callprogress` field. It is up to the page to determine if the`callprogress` field exists and to take the appropriate course of action based on the value of this field.

The default timeout for outbound call classification is 20 seconds (or 20000 milliseconds).The default value can be overridden for outbound calls by setting the `call_classification_timeout` parameter in the `hints` attribute on the `<createcall>` tag to the desired number of milliseconds before call classification analysis should time out.

The following example shows a simple `connection.signal` handler page that first determines whether this is a `connection.signal` event bearing classification data. If it is, the handler assesses the data to determine what action to take. If the classification is `live_voice`, then it returns a status of `success` and the page continues to run. Otherwise, a it returns the failure status `no answer`.

```
<transition event="connection.signal">
    <if cond="typeof(event$.info) != 'undefined'">
        <if cond="typeof(event$.info.callprogress) != 'undefined'">
            <var name="call_classification" expr="event$.info.callprogress"/>
            <var name="status"/>
            <if cond="call_classification == 'live_voice'">
                <assign name="status" expr="'success'"/>
                <send name="'avaya.launchresponse'" targettype="'avaya_platform'"
                        target="session.id" namelist="status"/>
            <else/>
                <assign name="status" expr="'no answer'"/>
                <send name="'avaya.launchresponse'" targettype="'avaya_platform'"
                        target="session.id" namelist="status"/>
            </if>
        </if>
    </if>
</transition>
```

SIP application support

User-to-User Interface (UUI) data passed in SIP headers

When you add an application to Experience Portal using the EPM Web interface, you also specify how User-to-User Interface (UUI) information will be passed to the application if it uses a SIP connection using the **Operation Mode** field in the **Advanced Parameters** group on the Add Application page.

The options are:

- Service Provider: Experience Portal passes the UUI data along as a single block to the application without making any attempt to interpret data.

  If you select this option, the application must handle the UUI data on its own.

- Shared UUI: Experience Portal takes the UUI data and parses it into an array of IDs and their corresponding values. It then passes the application both the fully encoded UUI data and the parsed array with only the values still encoded..

  If you select this option, the UUI data must conform to the Avaya UUI specifications listed below.

UUI data format in CCXML applications

For a CCXML application, the UUI data is organized in a tree format. For example:

```
avaya.ucid = '####################'
avaya.uui.mode = "shared uui"
avaya.uui.shared[0].id = "FA"
avaya.uui.shared[0].value = "################"
avaya.uui.shared[1].id = "BG"
avaya.uui.shared[1].value = "################"
```

Each connection has its own tree associated with it, and the values for that connection can be accessed using the format `session.connections['`*`connection_number`*`'].avaya.`*`element_name`*.

For example, if you want to declare two variables called `ucid` and `mode`, and you wanted to set those variables to be equal to the corresponding values for connection 1234, you would specify:

```
<var name="ucid" expr="session.connections['1234'].avaya.ucid"/>
<var name="mode" expr="session.connections['1234'].avaya.uui.mode"/>
```

UUI data format in VoiceXML applications

In VoiceXML, there is only one active call so the UUI information is organized into a similar tree format and placed into a session variable at the start of the dialog. The tree structure looks like this:

```
session.avaya.ucid
session.avaya.uui.mode
session.avaya.uui.shared[]
```

With the Shared UUI mode, you must send UUI/AAI data as name-value pairs in the following format:

```
<id0>,<value0>;<id1>,<value1>;<id2>,<value2>; ...
```

When you specify the name-value pairs:

- Each name must be set to the hexadecimal encoded ID stored in the `shared[]` array.
- Each value must be set to the encoded value stored in the `shared[]` array.
- Each name in the pair must be separated from its value by a `,` (comma).
- Each name-value pair must be separated from the next name-value pair by a `;` (semi-colon).

For example, if you wanted to send a UCID using UUI/AAI, you might specify: `aai = "FA,2901000246DEE275"`

Related topics:

Universal Call Identifier (UCID) values included in UUI data on page 73
Experience Portal application parameters affecting the UUI data on page 74

Universal Call Identifier (UCID) values included in UUI data

The Universal Call Identifier (UCID) is an Avaya-proprietary call identifier used to help correlate call records between different systems. This identifier can either be generated by the Experience Portal MPP server or it can be passed to Experience Portal through an application's SIP headers if the application uses a SIP connection and the application's **Operation Mode** is set to **Shared UUI**.

Note:

If the application uses an H.323 connection, Experience Portal can receive UCID from Communication Manager. This feature is supported in Communication Manager 5.2.

To enable this feature, you need to administer ucid-info on button 10 on the 7434ND stations used by Experience Portal.

A UCID consists of 20 decimal digits in three groups. Before the UCID is added to the UUI data, Experience Portal encodes each group as a hexadecimal value and concatenates the three groups together. The:

- First group of 5 digits represents a 2 byte network node identifier assigned to the Communication Manager. In the UUI data, this group is encoded as 4 hexadecimal digits.
- Second group of 5 digits represents a 2 byte sequence number. This group is encoded as 4 hexadecimal digits.
- Third group of 10 digits represents a 4 byte timestamp. This group is encoded as 6 hexadecimal digits.

For example, the UCID 00001002161192633166 would be encoded as `000100D84716234E`.

When Experience Portal passes the UCID 00001002161192633166 to the application, it would look like this:

```
avaya.ucid = '00001002161192633166'
avaya.uui.mode = 'shared uui'
avaya.uui.shared[0].id = 'FA'
avaya.uui.shared[0].value = '000100D84716234E'
```

Note:

The identifier for the UCID is always 250, which becomes FA in hexadecimal in the UUI `shared[]` array.

Experience Portal application parameters affecting the UUI data

When you add an application to Experience Portal using the EPM Web interface, the following parameters in the **Advanced Parameters** group on the Add Application page affect the contents of the SIP UUI data:

Parameter	Description
Generate UCID	The Universal Call Identifier (UCID) is an Avaya-proprietary call identifier used to help correlate call records between different systems. The options are: • **Yes**: If the CM does not pass a UCID to Experience Portal, the MPP server generates a UCID. • **No**: The MPP does not generate a UCID.
Operation Mode	The SIP header for each call can contain User-to-User Interface (UUI) information that the switch can either pass on as a single block or attempt to parse so that the information can be acted on. This field determines how Experience Portal treats the UUI data.

Parameter	Description
	The options are: • **Service Provider**: Experience Portal passes the UUI data as a single block to the application without making any attempt to interpret data. If you select this option, the application must handle the UUI data on its own. • **Shared UUI**: Experience Portal takes the UUI data and parses it into an array of IDs and their corresponding values. It then passes the application both the fully encoded UUI data and the parsed array with only the values still encoded. If you select this option, the UUI data must conform to the Avaya UUI specifications described in User-to-User Interface (UUI) data passed in SIP headers on page 72. • **Service Provider**: Experience Portal passes the UUI data as a single block to the application without making any attempt to interpret data. If you select this option, the application must handle the UUI data on its own. • **Shared UUI**: Experience Portal takes the UUI data and parses it into an array of IDs and their corresponding values. It then passes the application both the fully encoded UUI data and the parsed array with only the values still encoded. If you select this option, the UUI data must conform to the Avaya UUI specifications described in User-to-User Interface (UUI) data passed in SIP headers.
Transport UCID in Shared Mode	If **Operation Mode** is set to **Shared UUI** and **Generate UCID** is set to **Yes**, this field determines whether Experience Portal encodes the Experience Portal-generated UCID and adds it to the UUI data for all outbound calls. The default is **No**, which means that a UCID is only passed as part of the UUI information if that UCID was passed to Experience Portal by the application.
Maximum UUI Length	The maximum length of the UUI data that can be passed in the SIP header. If this length is exceeded and Experience Portal generated a UCID for the call, the UCID is removed from the UUI data. If the result still exceeds this value, or if the UCID was passed to Experience Portal by the application, Experience Portal does *not* send any UUI data. Instead, it leaves the entire field blank because it has no way to determine what can be left out.

SIP header support for CCXML and VoiceXML applications

Session Initiation Protocol (SIP) headers can provide additional information about a call that a CCXML or VoiceXML application can use to determine what processing needs to be done for that call. Experience Portal uses a collection of session variables to present this information to the application. However, not all SIP headers are accessible and many accessible headers are read only.

The following table lists the session variables that may accompany an inbound SIP INVITE. In the table, `<sip>` is the variable access string used to access the variables in a particular context. The valid strings are:

Variable access string	Description
`session.connection.protocol.sip`	This access string can be used by either CCXML or VoiceXML applications.
`event$.info.protocol.sip`	This access string can used when variables arrive in the `info` map for a transition. These variables are only valid within the transition in which they arrived.
`session.connections['`*`SessionID`*`'].protocol.sip` where *SessionID* is the session ID.	This access string can be used to retrieve the variables from the connection object in the session variable space. The variables exist between transitions but can be overwritten by new data at any time.

If you want to use a variable in your application, you must use the complete text in your code.

For example, if you want to access the `<sip>.callid` variable in a session with the session ID *1234*, you would code one of the following, depending on the context in which you want to access the variable:

- `session.connection.protocol.sip.callid`
- `event$.info.protocol.sip.callid`
- `session.connections['1234'].protocol.sip.callid`

Session Variable	SIP Header	Notes
<sip>.callid	Call-ID	Uniquely identifies a particular invitation or all registrations of a particular client.
<sip>.contact[array].displayname <sip>.contact[array].uri	Contact	Provides the display name, a URI with URI parameters, and header parameters.
<sip>.from.displayname	From	The initiator of the request.

Session Variable	SIP Header	Notes
<sip>.from.uri <sip>.tag		
<sip>.historyinfo[array].displayname <sip>.historyinfo[array].user <sip>.historyinfo[array].host <sip>.historyinfo[array].optheader	History-Info	This field is typically used to inform proxies and User Agent Clients and Servers involved in processing a request about the history or progress of that request.
<sip>.passertedid[array].displayname <sip>.passertedid[array].uri	P-asserted Identity	The verified identity of the user sending the SIP message. This field is typically used among trusted SIP intermediaries to prove that the initial message was sent by an authenticated source.
<sip>.require	Require	The options that the User Agent Client (UAC) expects the User Agent Server (UAS) to support in order to process the request.
<sip>.supported[array].option	Supported	All extensions supported by the UAC or UAS.
<sip>.to.displayname <sip>.to.host <sip>.to.uri <sip>.to.user	To	The logical recipient of the request.
<sip>.unknownhdr[array].name <sip>.unknownhdr[array].value	Unknown	All headers not understood by Avaya Aura® Experience Portal are passed to the application through this array.
<sip>.useragent[array]	User-Agent	Contains information about the UAC originating the request.
<sip>.via[array].sentaddr <sip>.via[array].sentport <sip>.via[array].protocol <sip>.via[array].branch	Via	The path taken by the request to this point along with the path that should be followed in routing responses.
<sip>.name		This variable returns “sip” when the SIP protocol is used.
<sip>.version		This variable returns the SIP protocol version when the SIP protocol is used.
<sip>.requestmethod <sip>.requestversion <sip>.requesturi <sip>.request.user	“request”	The various components of the request URI (INVITE). For example, this variable includes the parameters, user and host part of the URI, and the request method.

Session Variable	SIP Header	Notes
<sip>.request.host <sip>.requestparams[array].name <sip>.requestparams[array].value		
<sip>.respcode		The results of a transaction. The actual contents varies by transaction type.
<sip>.resptext		The results of a transaction. The actual contents varies by transaction type.

Sample VoiceXML page logging SIP headers

The following VoiceXML page logs various SIP headers using the Experience Portal session variables.

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml" xml:lang="en-us">
<property name="promptgender" value="female"/>
<property name="timeout" value="4s"/>
<property name="maxspeechtimeout" value="15s"/>
 <form id="form0">
  <block>
    <log>session.connection.protocol.sip <value
    expr="session.connection.protocol.sip"/></log>
    <log>session.connection.protocol.name: <value
    expr="session.connection.protocol.name"/></log>
    <log>session.connection.protocol.version: <value
    expr="session.connection.protocol.version"/></log>
    <log>session.connection.protocol.sip.requesturi: <value
    expr="session.connection.protocol.sip.requesturi"/></log>
    <log>session.connection.protocol.sip.requestmethod: <value
    expr="session.connection.protocol.sip.requestmethod"/></log>
    <log>session.connection.protocol.sip.requestversion: <value
    expr="session.connection.protocol.sip.requestversion"/></log>
    <log>session.connection.protocol.sip.request.user: <value
    expr="session.connection.protocol.sip.request.user"/></log>
    <log>session.connection.protocol.sip.request.host: <value
    expr="session.connection.protocol.sip.request.host"/></log>
    <log>session.connection.protocol.sip.requestparams[0].name: <value
    expr="session.connection.protocol.sip.requestparams[0].name"/></log>
    <log>session.connection.protocol.sip.requestparams[0].value: <value
    expr="session.connection.protocol.sip.requestparams[0].value"/></log>
    <log>session.connection.protocol.sip.to.uri: <value
    expr="session.connection.protocol.sip.to.uri"/></log>
    <log>session.connection.protocol.sip.to.displayname: <value
    expr="session.connection.protocol.sip.to.displayname"/></log>
    <log>session.connection.protocol.sip.to.user: <value
    expr="session.connection.protocol.sip.to.user"/></log>
    <log>session.connection.protocol.sip.to.host: <value
    expr="session.connection.protocol.sip.to.host"/></log>
    <log>session.connection.protocol.sip.from.uri: <value
    expr="session.connection.protocol.sip.from.uri"/></log>
    <log>session.connection.protocol.sip.from.displayname: <value
    expr="session.connection.protocol.sip.from.displayname"/></log>
    <log>session.connection.protocol.sip.from.tag: <value
```

```
    expr="session.connection.protocol.sip.from.tag"/></log>
    <log>session.connection.protocol.sip.from.user: <value
    expr="session.connection.protocol.sip.from.user"/></log>
    <log>session.connection.protocol.sip.from.host: <value
    expr="session.connection.protocol.sip.from.host"/></log>
    <log>session.connection.protocol.sip.useragent[0]: <value
    expr="session.connection.protocol.sip.useragent[0]"/></log>
    <log>session.connection.protocol.sip.contact[0].displayname: <value
    expr="session.connection.protocol.sip.contact[0].displayname"/></log>
    <log>session.connection.protocol.sip.contact[0].uri: <value
    expr="session.connection.protocol.sip.contact[0].uri"/></log>
    <log>session.connection.protocol.sip.via[0].sentaddr: <value
    expr="session.connection.protocol.sip.via[0].sentaddr"/></log>
    <log>session.connection.protocol.sip.via[0].protocol: <value
    expr="session.connection.protocol.sip.via[0].protocol"/></log>
    <log>session.connection.protocol.sip.via[0].sentport: <value
    expr="session.connection.protocol.sip.via[0].sentport"/></log>
    <log>session.connection.protocol.sip.via[1].sentaddr: <value
    expr="session.connection.protocol.sip.via[1].sentaddr"/></log>
    <log>session.connection.protocol.sip.via[1].protocol: <value
    expr="session.connection.protocol.sip.via[1].protocol"/></log>
    <log>session.connection.protocol.sip.via[1].sentport: <value
    expr="session.connection.protocol.sip.via[1].sentport"/></log>
    <log>session.connection.protocol.sip.supported: <value
    expr="session.connection.protocol.sip.supported"/></log>
    <log>session.connection.protocol.sip.require: <value
    expr="session.connection.protocol.sip.require"/></log>
  </block>
 </form>
</vxml>
```

Support for unknown headers

If Experience Portal receives an INVITE with a header it does not recognize, it saves the name and value of the header in the `session.connection.protocol.sip.unknownhdr` session variable array.

For example, if Experience Portal receives an INVITE with the following unknown headers:

```
new_header: "helloworld"
another_new_header: "howareyou"
```

Experience Portal adds the following entries to the `session.connection.protocol.sip.unknownhdr` array:

```
session.connection.protocol.sip.unknownhdr.unknownhdr[0].name = "new_header"
session.connection.protocol.sip.unknownhdr.unknownhdr[0].value = "helloworld"
session.connection.protocol.sip.unknownhdr.unknownhdr[1].name =
"another_new_header"
session.connection.protocol.sip.unknownhdr.unknownhdr[1].value = "howareyou"
```

RFC 3261 SIP headers

You can set a limited number of SIP headers independently for applications that initiate an outbound call with one of the following:

- A REFER as a result of a blind or consultative transfer
- An INVITE as a result of bridged transfer or a CCXML outbound call

Note:

Not all headers that are available to be set on an INVITE are available to be set on a REFER.

You can set the following headers with the VoiceXML `<property>` element before `<transfer>` element. In the table, `<sip_prop>` represents `AVAYA_SIPHEADER.session.connection.protocol.sip`.

Note:

If you want to set a header in your application, you must use the complete text in your code. For example, code `<sip_prop>.callid` as `AVAYA_SIPHEADER.session.connection.protocol.sip.callid`.

SIP Header	<property>	Notes
Call-Info	<sip_prop>.callinfo	Provides additional information about the source or target of the call, depending on whether it is found in a request or response.
To.displayname	<sip_prop>.to.displayname	Displays the name of the call's target.
From.displayname	<sip_prop>.from.displayname	Displays the name of the call's source.
P-asserted Identity	<sip_prop>.passerted.displayname <sip_prop>.passerted.uri	The unique identifier of the source sending the SIP message. This identifier is used for authentication purposes if your SIP configuration requires trusted connections. **Note:** If you define the **P-Asserted-Identity** parameter for the SIP connection through the EPM, Experience Portal ignores any attempt by an application to change this identity.
Subject	<sip_prop>.subject	A summary of the call.
Organization	<sip_prop>.organization	The name of the organization to which the SIP element issuing the request or response belongs.
Priority	<sip_prop>.priority	The priority of the request.

Creating a custom header

Procedure

To create a custom header, store a name and value pair in the`session.connection.protocol.sip.unknownhdr` session variable array.

Example

To create a custom header with the name as `new_header` and value as `mycustomheader`, add the following to the `session.connection.protocol.sip.unknownhdr` array:

```
AVAYA_SIPHEADER.session.connection.protocol.sip.unknownhdr[0].name = "new_header"
AVAYA_SIPHEADER.session.connection.protocol.sip.unknownhdr[0].value =
"mycustomheader"
```

Sample VoiceXML page setting SIP headers in a VoiceXML application

The following VoiceXML page sets various SIP headers on a bridged transfer.

```
<?xml version="1.0" ?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml" xml:lang="en-us" >
<form id="form0">
  <property name="AVAYA_SIPHEADER.session.connection.protocol.sip.from.displayname"
   value="kong, king"/>
  <property name="AVAYA_SIPHEADER.session.connection.protocol.sip.to.displayname"
  value="godzilla"/>
  <property
name="AVAYA_SIPHEADER.session.connection.protocol.sip.passertedid.displayname"
  value="authority"/>
  <property name="AVAYA_SIPHEADER.session.connection.protocol.sip.passertedid.uri"
  value="sip:1234@123.321.123.321"/>
  <property
name="AVAYA_SIPHEADER.session.connection.protocol.sip.unknownhdr[0].name"
  value="Random"/>
  <property
name="AVAYA_SIPHEADER.session.connection.protocol.sip.unknownhdr[0].value"
  value="This is an unknown header"/>
  <transfer name="t1" type="bridge" dest="tel:1234"/>
</form>
</vxml>
```

SIP UPDATE method

The SIP UPDATE method, as per RFC 3311, allows you to update parameters of a session. While a call is in a queue, Experience Portal allows the SIP UPDATE method to update the following parameter of the call:

- User-to-User Interface data

You can send multiple UPDATE messages after the initial INVITE is established and before the final response to the INVITE.

Note:

Experience Portal supports SIP UPDATE method only if the Allow header indicates support.

Related topics:

Sample SIP UPDATE Method on page 82

Sample SIP UPDATE Method

The following is an example of the SIP UPDATE method:

```
Via: SIP/2.0/UDP pc33.<domain_name>.com;branch=<branch id>
;received=<ip_address>
To: <sip: email id>;tag=<tag number>
From: <name>
<sip:email id>;tag= <tag number>
 Call-ID: <call_id>
 CSeq: 63104 OPTIONS
 Contact: <sip: email id>
 Contact: <mailto: email id>
 Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, UPDATE
 Accept: application/sdp
 Accept-Encoding: gzip
 Accept-Language: en
 Supported: foo
 Content-Type: application/sdp
 Content-Length: 274
```

Frequently asked questions about using the Orchestration Designer Report control

For each application, the information you specify in the Orchestration Designer Report control determines what information is displayed in the Avaya Aura® Experience Portal Application Detail report and Application Summary report. This topic answers the following questions:

What is an "activity"?

An activity should be comprised of at least two messages, with an **Activity Type** of **Start** and **End** respectively.

The **Start** activity type starts the "duration clock" so that the **Activity Duration** field is populated correctly for each message in the activity. If you do not use a **Start** activity, the duration is always 0.

The **End** activity type stops the "duration clock" so that the **Activity Duration** field is populated correctly for the length of the entire activity.

For example, if you create a Holiday Planning application, it could be broken into 2 main activities: Car Rental and Hotel Booking.

If you have defined a **Start** and **End** time for each activity, you can determine how much time was required to set up the car rental versus book the hotel.

Otherwise, the only information Experience Portal can display is the total length of the call. It cannot help you determine how long it took to process any individual part of that call.

How should "Activity Levels" be used?

The level should be set to something other than **Info** if the message is a result of unusual or undesired flow. The choices are:

- **Fatal**
- **Error**
- **Warning**

For example, when a requested car or hotel is not available, that message should be marked as a **Warning**. If the credit card process did not complete correctly, that message should be marked as either **Error** or **Fatal**, depending on how you want to treat that condition.

This allows you to generate a report showing just those calls in which **Warning**, **Error**, and **Fatal** messages were reported.

Where do I put message details?

The details of a message should be displayed with an associated variable.

For example, if you put the type of car requested in as a variable called `carType` and have a generic Warning-level message stating that a car of that type is not available, Application Detail report users could click on the message to display the value of the associated `carType` variable to find out what specific kind of car was not available.

Furthermore, if you set the activity **Level** filter to **Warning**, the **Variable Name** filter to `carType`, and the **Variable Value** filter to `4Runner`, you can generate an Application Detail report showing all occurrences where a 4Runner was requested but was not available.

How do I specify the value that appears in the Session Label/UCID field in the Application Detail report and Application Summary report?

Assign the value using the **Value** field in the Avaya Properties view.

Note:

This is the only field of the session variable to which you can assign a value. It can contain any arbitrary data which applies to all report messages in the session, such as the account number of a returning customer.

Index

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W