Avaya Orchestration Designer Developer's Guide

Release 8.0
Issue 1
September 2020

READ THIS CAREFULLY BEFORE ELECTRONICALLY ACCESSING OR USING THIS PROPRIETARY PRODUCT!

THIS IS A LEGAL AGREEMENT ("AGREEMENT") BETWEEN YOU, INDIVIDUALLY, AND/OR THE LEGAL ENTITY FOR WHOM YOU ARE OPENING, INSTALLING, DOWNLOADING, COPYING OR OTHERWISE USING THE AVAYA SOFTWARE DEVELOPMENT KIT ("SDK") (COLLECTIVELY, AS REFERENCED HEREIN, "YOU", "YOUR", OR "LICENSEE") AND AVAYA INC. OR ANY AVAYA AFFILIATE (COLLECTIVELY, "AVAYA"). IF YOU ARE ACCEPTING THE TERMS AND CONDITIONS OF THIS AGREEMENT ON BEHALF OF A LEGAL ENTITY, YOU REPRESENT AND WARRANT THAT YOU HAVE FULL LEGAL AUTHORITY TO ACCEPT ON BEHALF OF AND BIND SUCH LEGAL ENTITY TO THIS AGREEMENT. BY OPENING THE MEDIA CONTAINER, BY INSTALLING, DOWNLOADING, COPYING OR OTHERWISE USING THE AVAYA SOFTWARE DEVELOPMENT KIT ("SDK") OR AUTHORIZING OTHERS TO DO SO, YOU SIGNIFY THAT YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT HAVE SUCH AUTHORITY OR DO NOT WISH TO BE BOUND BY THE TERMS OF THIS AGREEMENT, SELECT THE "DECLINE" BUTTON AT THE END OF THE TERMS OF THIS AGREEMENT OR THE EQUIVALENT OPTION AND YOU SHALL HAVE NO RIGHT TO USE THE SDK.

DEFINITIONS.

"Affiliates" means any entity that is directly or indirectly controlling, controlled by, or under common control with Avaya Inc. For purposes of this definition, "control" means the power to direct the management and policies of such party, directly or indirectly, whether through ownership of voting securities, by contract or otherwise; and the terms "controlling" and "controlled" have meanings correlative to the foregoing.

"Avaya Software Development Kit" or "SDK" means Avaya technology, which may include Software, Client Libraries, Specification Documents, Software libraries, application programming interfaces ("API"), Software tools, Sample Application Code, and Documentation.

"Client Libraries" mean any enabler code specifically designated as such and included in a SDK. Client Libraries may also be referred to as "DLLs", and represent elements of the SDK required at runtime to communicate with Avaya products or other SDK elements.

"Change In Control" shall be deemed to have occurred if any person, entity or group comes to own or control, directly or indirectly, beneficially or of record, voting securities (or any other form of controlling interest) which represent more than fifty percent (50%) of the total voting power of or to the Licensee.

"Derivative Work(s)" means: any translation (including translation into other computer languages), port, compiling of Source Code into object code, combination with a pre-existing work, modification, correction, addition, extension, upgrade, improvement, compilation, abridgment or other form in which an existing work may be recast, transformed or adapted or which would otherwise constitute a derivative work under the United States Copyright Act. Permitted Modifications will be considered Derivative Works.

"Documentation" includes, programmer guides, CDs, manuals, materials, and information appropriate or necessary for use in connection with the SDK. Documentation may be provided in machine-readable, electronic or hard copy form.

"Intellectual Property" means any and all: (i) rights associated with works of authorship throughout the world, including copyrights, neighboring rights, moral rights, and mask works, (ii) trademark and trade name rights and similar rights, (iii) trade secret rights, (iv) patents, algorithms, designs and other industrial property rights, (v) all other intellectual and industrial property rights (of every kind and nature throughout the world and however designated) whether arising by operation of law, contract, license, or otherwise, and (vi) all registrations, initial applications, renewals, extensions, continuations, divisions or reissues thereof now or hereafter in force (including any rights in any of the foregoing).

"Permitted Modification(s)" means Licensee's modifications of the Sample Application Code as needed to create applications, interfaces, workflows or processes for use with Avaya products.

"Specification Document" means any notes or similar instructions in hard copy or machine readable form, including any technical, interface and/or interoperability specifications that define the requirements and conditions for connection to and/or interoperability with Avaya products, systems and solutions.

"Source Code" means human readable or high-level statement version of software written in the source language used by programmers and includes one or more programs. Source Code programs may include one or more files, such as user interface markup language (.mxml), action script (.as), precompiled Flash code (.swc), java script (.js), hypertext markup language (.html), active server pages (.asp), C# or C# .Net source code (.cs), java source code (.java), java server pages (.jsp), java archives (.jar), graphic interchange format (.gif), cascading style sheet (.css) , audio files (.wav) and extensible markup language (.xml) files.

"Sample Application Code" means Software provided for the purposes of demonstrating functionality of an Avaya product through the Avaya Software Development Kit.

"Software" means data or information constituting one or more computer or apparatus programs, including Source Code or in machine-readable, compiled object code form

LICENSE GRANT.

SDK License.

A. Provided Licensee pays to Avaya the applicable license fee (if any), Avaya hereby grants Licensee a limited, non-exclusive, non-transferable license (without the right to sublicense, except as set forth in 2.1B (iii)) under the Intellectual Property of Avaya and, if applicable, its licensors and suppliers to (i) use the SDK solely for the purpose of Licensee's internal development efforts to develop applications, interfaces, value-added services and/or solutions, workflows or processes to work in conjunction with Avaya products; (ii) to package Client Libraries for redistribution with Licensee's complementary applications that have been developed using this SDK, subject to the terms and conditions set forth herein; (iii) use Specification Documents solely to enable Licensee's products, services and application solutions to exchange messages and signals with Avaya products, systems and solutions to which the Specification Document(s) apply; (iv) modify and create Derivative Works of the Sample Application Code, Specification Documents and Documentation solely for internal development of applications, interfaces, workflows or processes for use with Avaya products, integration of such applications, interfaces, workflows and processes with Avaya products and interoperability testing of the foregoing with Avaya products; and (v) compile or otherwise prepare for distribution the Sample Application Code with Permitted Modifications, into an object code or other machine-readable program format for distribution and distribute the same subject to the conditions set forth in Section 2.1B.

B. The foregoing license to use Sample Application Code is contingent upon the following: (i) Licensee must ensure that the modifications made to the Sample Application Code as permitted in clause (iv) of Section 2.1A are compatible and/or interoperable with Avaya products and/or integrated therewith, (ii) Licensee may distribute Licensee's application that has been created using this SDK, provided that such distribution is subject to an end user pursuant to Licensee's current end user license agreement ("Licensee EULA") that is consistent with the terms of this Agreement and, if applicable, any other agreement with Avaya (e.g., the Avaya DevConnect Program Agreement), and is equally as protective as Licensee's standard software license terms, but in no event shall the standard of care be less than a reasonable degree of care, and (iii) Licensee ensures that each end user who receives Client Libraries or Sample Application Code with Permitted Modifications has all necessary licenses for all underlying Avaya products associated with such Client Libraries or Sample Application Code.

Your Licensee EULA must include terms concerning restrictions on use, protection of proprietary rights, disclaimer of warranties, and limitations of liability. You must ensure that Your End Users using applications, interfaces, value-added services and/or solutions, workflows or processes that incorporate the API, Client Libraries, Sample Code or Permitted Modifications adhere to these terms, and You agree to notify Avaya promptly if You become aware of any breach of the terms of Licensee EULA that may impact Avaya. You will take all reasonable precautions to prevent unauthorized access to or use of the SDK and notify Avaya promptly of any such unauthorized access or use.

C. Licensee acknowledges and agrees that it is licensed to use the SDK only in connection with Avaya products (and if applicable, in connection with services provided by or on behalf of Avaya).

D. With respect to Software that contains elements provided by third party suppliers, Licensee may install and use the Software in accordance with the terms and conditions of the applicable license agreements, such as “shrinkwrap” or “click-through” licenses, accompanying or applicable to the Software.

No Standalone Product. Nothing in this Agreement authorizes or grants Licensee any rights to distribute or otherwise make available to a third party the SDK, in whole or in part, or any Derivative Work in source or object code format on a standalone basis other than the modifications permitted in Section 2.1B of this Agreement.

Proprietary Notices. Licensee shall not remove any copyright, trade mark or other proprietary notices incorporated in the copies of the SDK, Sample Application Code and redistributable files in Licensee’s possession or control or any modifications thereto. Redistributions in binary form or other suitable program format for distribution, to the extent expressly permitted, must also reproduce Avaya’s copyright, trademarks or other proprietary notices as incorporated in the SDK in any associated Documentation or “splash screens” that display Licensee copyright notices.

Third-Party Components. You acknowledge certain software programs or portions thereof included in the SDK 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 SDK (“Third Party Terms”). Information identifying the copyright holders of the Third Party Components and the Third Party Terms that apply is available in the attached Schedule 1 (if any), SDK, Documentation, or on Avaya’s web site at: https://support.avaya.com/Copyright (or such successor site as designated by Avaya). The open source software license terms provided as Third Party Terms are consistent with the license rights granted in this Agreement, and may contain additional rights benefiting You, such as modification and distribution of the open source software. The Third Party Terms shall take precedence over this Agreement, solely with respect to the applicable Third Party Components, to the extent that this Agreement imposes greater restrictions on You than the applicable Third Party Terms. Licensee is solely responsible for procuring any necessary licenses for Third Party Components, including payment of licensing royalties or other amounts to third parties, for the use thereof.

Copies of SDK. Licensee may copy the SDK only as necessary to exercise its rights hereunder.

No Reverse Engineering. Licensee shall have no rights to any Source Code for any of the software in the SDK, except for the explicit rights to use the Source Code as provided to Licensee hereunder. Licensee agrees that it shall not cause or permit the disassembly, decompilation or reverse engineering of the Software. Notwithstanding the foregoing, if the SDK is rightfully located in a member state of the European Union and Licensee needs information about the Software in the SDK in order to achieve interoperability of an independently created software program with the Software in the SDK, Licensee will first request such information from Avaya. Avaya may charge Licensee a reasonable fee for the provision of such information. If Avaya refuses to make such information available, then Licensee may take steps, such as reverse assembly or reverse compilation, to the extent necessary solely in order to achieve interoperability of the Software in the SDK with an independently created software program. To the extent that the Licensee is expressly permitted by applicable mandatory law to undertake any of the activities listed in this section, Licensee will not exercise those rights until Licensee has given Avaya twenty (20) days written notice of its intent to exercise any such rights.

License Restrictions. To the extent permissible under applicable law, Licensee agrees not to: (i) publish, sell, sublicense, lease, rent, loan, assign, convey or otherwise transfer the SDK; (ii) distribute, disclose or allow use the SDK, in any format, through any timesharing service, service bureau, network or by any other means; (iii) distribute or otherwise use the Software in the SDK in any manner that causes any portion of the Software that is not already subject to an OSS License to become subject to the terms of any OSS License; (iv) link the Source Code for any of the software in the SDK with any software licensed under the Affero General Public License v.3 or similar licenses; (v) access information that is solely available to root administrators of the Avaya products, systems, and solutions; (vi) develop a applications, interfaces, value-added services and/or solutions, workflows or processes that causes adverse effects to Avaya and third-party products, services, solutions, such as, but not limited to, poor performance, software crashes and cessation of their proper functions; and (vii) develop a applications, interfaces, value-added services and/or solutions, workflows or processes that blocks or delays emergency calls; (viii) emulate an Avaya SIP endpoint by form or user interface design confusingly similar as an Avaya product ; (ix) reverse engineer Avaya SIP protocol messages; or (x) permit or encourage any third party to do any of (i) through (x), inclusive, above.

Responsibility for Development Tools. Licensee acknowledges that effective utilization of the SDK may require the use of a development tool, compiler and other software and technology of third parties, which may be incorporated in the SDK pursuant to Section 2.4. Licensee is solely responsible for procuring such third party software and technology and the necessary licenses, including payment of licensing royalties or other amounts to third parties, for the use thereof.

U.S. Government End Users. The SDK shall be classified as “commercial computer software” and the Documentation is classified as “commercial computer software documentation” or “commercial items,” pursuant to FAR 12.212 or DFAR 227.7202, as applicable. Any use, modification, reproduction, release, performance, display or disclosure of the SDK or Documentation by the Government of the United States shall be governed solely by the terms of the Agreement and shall be prohibited except to the extent expressly permitted by the terms of the Agreement.

Limitation of Rights. No right is granted to Licensee to sublicense its rights hereunder. All rights not expressly granted are reserved by Avaya or its licensors or suppliers and, except as expressly set forth herein, no license is granted by Avaya or its licensors or suppliers under this Agreement directly, by implication, estoppel or otherwise, under any Intellectual Property right of Avaya or its licensors or suppliers. Nothing herein shall be deemed to authorize Licensee to use Avaya's trademarks or trade names in Licensee's advertising, marketing, promotional, sales or related materials.

2.10 Independent Development

Licensee understands and agrees that Avaya, Affiliates, or Avaya’s licensees or suppliers may acquire, license, develop for itself or have others develop for it, and market and/or distribute applications, interfaces, value-added services and/or solutions, workflows or processes similar to that which Licensee may develop. Nothing in this Agreement shall restrict or limit the rights of Avaya, Affiliates, or Avaya’s licensees or suppliers to commence or continue with the development or distribution of such applications, interfaces, value-added services and/or solutions, workflows or processes.

Nonassertion by Licensee. Licensee agrees not to assert any Intellectual Property related to the SDK or applications, interfaces, value-added services and/or solutions, workflows or processes developed using the SDK against Avaya, Affiliates, Avaya’s licensors or suppliers, distributors, customers, or other licensees of the SDK.

Feedback and Support. Licensee agrees to provide any information, comments, problem reports, enhancement requests and suggestions regarding the performance of the SDK (collectively, “Feedback”) via any public or private support mechanism, forum or process otherwise indicated by Avaya. Avaya monitors applicable mechanisms, forums, or processes but is under no obligation to implement any of Feedback, or be required to respond to any questions asked via the applicable mechanism, forum, or process. Licensee hereby assigns to Avaya all right, title, and interest in and to Feedback provided to Avaya.

2.12 Fees and Taxes

To the extent that fees are associated with the license of the SDK, Licensee agrees to pay to Avaya or pay directly to the applicable government or taxing authority, if requested by Avaya, all taxes and charges, including without limitation, penalties and interest, which may be imposed by any federal, state or local governmental or taxing authority arising hereunder excluding, however, all taxes computed upon Avaya’s net income. If You move any Software, including the SDK, and as a result of such move, a jurisdiction imposes a duty, tax, levy or fee (including withholding taxes, fees, customs or other duties for the import and export of any such Software), then You are solely liable for, and agree to pay, any such duty, taxes, levy or other fees.

Audit. Avaya shall have the right, at its cost and expense, to inspect and/or audit (i) by remote polling or other reasonable electronic means at any time and (ii) in person during normal business hours and with reasonable notice Licensee’s books, records, and accounts, to determine Licensee’s compliance with this Agreement. In the event such inspection or audit uncovers non-compliance with this Agreement, then without prejudice to Avaya’s termination rights hereunder, Licensee shall promptly pay Avaya any applicable license fees. Licensee agrees to keep a current record of the location of the SDK.

No Endorsement. Neither the name Avaya, Affiliates nor the names of contributors may be used to endorse or promote products derived from the Avaya SDK without specific prior written permission from Avaya.

High Risk Activities. The Avaya SDK is not fault-tolerant, and is not designed, manufactured or intended for use or resale as on-line control equipment or in hazardous environments requiring failsafe performance, such as in the operation of nuclear facilities, aircraft navigation or aircraft communications systems, mass transit, air traffic control, medical or direct life support machines, dedicated emergency call handling systems or weapons systems, in which the failure of the Avaya SDK could lead directly to death, personal injury, or severe physical or environmental damage (“high risk activities”). If Licensee uses the Avaya SDK for high risk activities, Licensee does so at Licensee’s own risk and Licensee assumes all responsibility and liability for such use to the maximum extent such limitation or exclusion is permitted by applicable law. Licensee agrees that Avaya and its suppliers will not be liable for any claims or damages arising from or related to use of the Avaya SDK for high risk activities to the maximum extent such limitation or exclusion is permitted by law.

No Virus. Licensee warrants that (i) the applications, interfaces, value-added services and/or solutions, workflows or processes Licensee develops using this SDK will not contain any computer program file that includes time code limitations, disabling devices, or any other mechanism which will prevent the Avaya product (including other software, firmware, hardware), services and networks from being functional at all times (collectively “Time Bombs”); and (ii) the applications, interfaces, value-added services and/or solutions, workflows or processes Licensee develops using this SDK will be free of computer viruses, malicious or other harmful code, black boxes, malware, trapdoors, and other mechanisms which could: a) damage, destroy or adversely affect Avaya product or services and/or end users; b) allow remote/hidden attacks or access through unauthorized computerized command and control; c) spy (network sniffers, keyloggers); and d) damage or erase such applications, interfaces, value-added services and/or solutions, workflows or processes developed using this SDK or data, or any computer files or systems of Avaya, Affiliates, and/or end users (collectively “Virus”). In addition to any other remedies permitted in the Agreement, if Licensee breaches its warranties under this Section, Licensee will, at its expense, take remedial action to eliminate any Time Bombs and/or Viruses and prevent re-occurrence (including implementing appropriate processes to prevent further occurrences) as well as provide prompt, reasonable assistance to Avaya to materially reduce the effects of the Time Bomb and/or Virus.

Disclaimer. Any software security feature is not a guaranty against malicious code, deleterious routines, and other techniques and tools employed by computer “hackers” and other third parties to create security exposures. Compromised passwords represent a major security risk. Avaya encourages You to create strong passwords using three different character types, change Your password regularly and refrain from using the same password regularly. You must treat such information as confidential. You agree to notify Avaya immediately upon becoming aware of any unauthorized use or breach of Your user name, password, account, API Key, or other credentials as provided by Avaya for use of the SDK, or subscription. You are responsible for ensuring that Your networks and systems are adequately secured against unauthorized intrusion or attack and regularly back up of Your data and files in accordance with good computing practices.

Third Party Licensed Software

A. “Commercial Third Party Licensed Software” is software developed by a business with the purpose of making money from the use of that licensed software. “Freeware Licensed Software” is software which is made available for use, free of charge and for an unlimited time, but is not Open Source Licensed Software. “Open Source Software" or "OSS" is as defined by the Open Source Initiative (“OSI”) https://opensource.org/osd and is software licensed under an OSI approved license as set forth at https://opensource.org/licenses/alphabetical (or such successor site as designated by OSI). These are collectively referred to herein as “Third Party Licensed Software”.

B. Licensee represents and warrants that Licensee, including any employee, contractor, subcontractor, or consultant engaged by Licensee, is to the Licensee’s knowledge, in compliance and will continue to comply with all license obligations for Third Party Licensed Software used in the Licensee application created using the SDK including providing to end users all information required by such licenses as may be necessary. LICENSEE REPRESENTS AND WARRANTS THAT, TO THE LICENSEE’S KNOWLEDGE, THE OPEN SOURCE LICENSED SOFTWARE EMBEDDED IN OR PROVIDED WITH LICENSEE APPLICATION OR SERVICES DOES NOT INCLUDE ANY OPEN SOURCE LICENSED SOFTWARE CONTAINING TERMS REQUIRING ANY INTELLECTUAL PROPERTY OWNED OR LICENSED BY AVAYA OR END USERS TO BE (A) DISCLOSED OR DISTRIBUTED IN SOURCE CODE OR OBJECT CODE FORM; (B) LICENSED FOR THE PURPOSE OF MAKING DERIVATIVE WORKS; OR (C) REDISTRIBUTABLE ON TERMS AND CONDITION NOT AGREED UPON BY AVAYA OR END USERS.

C. Subject to any confidentiality obligations, trade secret or other rights or claims of Licensee suppliers, Licensee will respond to requests from Avaya or end users relating to Third Party Licensed Software associated with Licensee's use of Third Party Licensed Software. Licensee will cooperate in good faith by furnishing the relevant information to Avaya or end users and the requester within two (2) weeks from the time Avaya or end user provided the request to Licensee.

OWNERSHIP.

As between Avaya and Licensee, Avaya or its licensors or suppliers shall own and retain all Intellectual Property rights, in and to the SDK and any corrections, bug fixes, enhancements, updates, improvements, or modifications thereto and Licensee hereby irrevocably transfers, conveys and assigns to Avaya, its licensors and its suppliers all of its right, title, and interest therein. Avaya or its licensors or suppliers shall have the exclusive right to apply for or register any patents, mask work rights, copyrights, and such other proprietary protections with respect thereto. Licensee acknowledges that the license granted under this Agreement does not provide Licensee with title or ownership to the SDK, but only a right of limited use under the terms and conditions of this Agreement.

Grant Back License to Avaya. Licensee hereby grants to Avaya an irrevocable, perpetual, non-exclusive, sublicensable, royalty-free, fully paid up, worldwide license under any and all of Licensee's Intellectual Property rights related to any Permitted Modifications,to (i) use, make, sell, execute, adapt, translate, reproduce, display, perform, prepare derivative works based upon, distribute (internally and externally) and sublicense the Permitted Modifications and their derivative works, and (ii) sublicense others to do any, some, or all of the foregoing.

SUPPORT.

No Avaya Support. Avaya will not provide any support for the SDK provided under this Agreement or for any Derivative Works, including, without limitation, modifications to the Source Code or applications built by Licensee using the SDK. Avaya shall have no obligation to provide support for the use of the SDK, or Licensee's application, services or solutions which may or may not include redistributable Client Libraries or Sample Application Code, to any third party to whom Licensee delivers such applications, services or solutions. Avaya further will not provide fixes, patches or repairs for any defects that might exist in the SDK or the Sample Application Code provided under this Agreement. In the event that Licensee desires support services for the SDK, and, provided that Avaya offers such support services (in its sole discretion), Licensee will be required to enter into an Avaya DevConnect Program Agreement or other support agreement with Avaya.

Licensee Obligations. Licensee acknowledges and agrees that it is solely responsible for developing and supporting any applications, interfaces, value-added services and/or solutions, workflows or processes developed under this Agreement, including but not limited to (i) developing, testing and deploying such applications, interfaces, value-added services and/or solutions, workflows or processes; (ii) configuring such applications, interfaces, value-added services and/or solutions, workflows or processes to interface and communicate properly with Avaya products; and (iii) updating and maintaining such applications, interfaces, value-added services and/or solutions, workflows or processes as necessary for continued use with the same or different versions of end user and/or third party licensor products, and Avaya products.

CONFIDENTIALITY.

Protection of Confidential Information. Licensee acknowledges and agrees that the SDK and any other Avaya technical information obtained by it under this Agreement (collectively, “Confidential Information”) is confidential information of Avaya. Licensee shall take all reasonable measures to maintain the confidentiality of the Confidential Information. Licensee further agrees at all times to protect and preserve the SDK in strict confidence in perpetuity, and shall not use such Confidential Information other than as expressly authorized by Avaya under this Agreement, nor shall Licensee disclose any Confidential Information to third parties without Avaya's written consent. Licensee further agrees to immediately 1) cease all use of all Confidential Information (including copies thereof) in Licensee's possession, custody, or control; 2) stop reproducing or distributing the Confidential Information; and 3) destroy the Confidential Information in Licensee’s possession or under its control, including Confidential Information on its computers, disks, and other digital storage devices upon termination of this Agreement at any time and for any reason. Upon request, Licensee will certify in writing its compliance with this Section. The obligations of confidentiality shall not apply to information which (a) has entered the public domain except where such entry is the result of Licensee's breach of this Agreement; (b) prior to disclosure hereunder was already rightfully in Licensee's possession; (c) subsequent to disclosure hereunder is obtained by Licensee on a non-confidential basis from a third party who has the right to disclose such information to the Licensee; (d) is required to be disclosed pursuant to a court order, so long as Avaya is given adequate notice and the ability to challenge such required disclosure.

Press Releases. Any press release or publication regarding this Agreement is subject to prior written approval of Avaya.

NO WARRANTY.

The SDK and Documentation are provided “AS-IS” without any warranty whatsoever. AVAYA SPECIFICALLY AND EXPRESSLY DISCLAIMS ANY WARRANTIES OR CONDITIONS, STATUTORY OR OTHERWISE, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT AND SATISFACTORY QUALITY. AVAYA DOES NOT WARRANT THAT THE SDK AND DOCUMENTATION ARE SUITABLE FOR LICENSEE'S USE, THAT THE SDK OR DOCUMENTATION ARE WITHOUT DEFECT OR ERROR, THAT OPERATION WILL BE UNINTERRUPTED, OR THAT DEFECTS WILL BE CORRECTED. FURTHER, AVAYA MAKES NO WARRANTY REGARDING THE RESULTS OF THE USE OF THE SDK AND DOCUMENTATION. NEITHER AVAYA NOR ITS SUPPLIERS MAKE ANY WARRANTY, EXPRESS OR IMPLIED, THAT THE SDK OR DOCUMENTATION IS SECURE, SECURITY THREATS AND VULNERABILITIES WILL BE DETECTED OR SOFTWARE WILL RENDER AN END USER’S OR LICENSEE’S NETWORK OR PARTICULAR NETWORK ELEMENTS SAFE FROM INTRUSIONS AND OTHER SECURITY BREACHES.

CONSEQUENTIAL DAMAGES WAIVER.

EXCEPT FOR PERSONAL INJURY CLAIMS, AVAYA SHALL NOT BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH, ARISING OUT OF OR RELATING TO THIS AGREEMENT OR USE OF THE SDK, OR FOR THE LOSS OR CORRUPTION OF DATA, INFORMATION OF ANY KIND, BUSINESS, PROFITS, OR OTHER COMMERCIAL LOSS, HOWEVER CAUSED, AND WHETHER OR NOT AVAYA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

LIMITATION OF LIABILITY.

EXCEPT FOR PERSONAL INJURY CLAIMS, IN NO EVENT SHALL AVAYA'S TOTAL LIABILITY TO LICENSEE IN CONNECTION WITH, ARISING OUT OF OR RELATING TO THIS AGREEMENT EXCEED FIVE HUNDRED DOLLARS ($500). THE PARTIES AGREE THAT THE LIMITATIONS SPECIFIED IN THIS SECTION WILL APPLY EVEN IF ANY LIMITED REMEDY PROVIDED IN THIS AGREEMENT IS FOUND TO HAVE FAILED OF ITS ESSENTIAL PURPOSE.

INDEMNIFICATION.

Licensee shall indemnify and hold harmless Avaya, Affiliates and their respective officers, directors, agents, suppliers, customers and employees “Indemnified Parties”) from and against all claims, demand, suit, actions or proceedings (“Claims”) and damages, losses, liabilities, costs, expenses, and fees (including fees of attorneys and other professionals) (“Damages”) based upon an allegation pertaining to wrongful use, misappropriation, or infringement of a third party’s Intellectual Property right arising from or relating to Licensee’s use of the SDK, alone or in combination with other software, such as operating systems and codecs, and the, direct or indirect, use, distribution or sale of any software, Derivative Works or other products (including but not limited to applications, interfaces, and application programming interfaces) developed utilizing the SDK.

Licensee shall defend, indemnify and hold harmless the Indemnified Parties from and against all Claims and Damages arising out of or related to: (i) personal injury (including death); (ii) damage to any person or tangible property caused, or alleged to be caused by Licensee or Licensee’s application created by using the SDK; (iii) the failure by Licensee or Licensee’s application created by using the SDK to comply with the terms of this Agreement or any applicable laws; (iv) the breach of any representation, or warranty made by Licensee herein; or (v) Licensee’s breach of any obligation under the Licensee EULA.

TERM AND TERMINATION.

This Agreement will continue through December 31st of the current calendar year. The Agreement will automatically renew for one (1) year terms, unless terminated as specified in Section 10.2 or 10.3 below.

Either party shall have the right to terminate the Agreement, upon thirty (30) days written notice to the other party.

Notwithstanding language to the contrary, Avaya may terminate this Agreement immediately, upon written notice to Licensee for breach of Section 2 (License Grant), Section 5 (Confidentiality) or Section 12 (Compliance with Laws). Avaya may also terminate this Agreement

immediately by giving written notice if a Change In Control should occur or if Licensee becomes insolvent, or voluntary or involuntary proceedings by or against Licensee are instituted in bankruptcy or under any insolvency law, or a receiver or custodian is appointed for Licensee, or proceedings are instituted by or against Licensee for corporate reorganization or the dissolution of Licensee, which proceedings, if involuntary, have not been dismissed within thirty (30) days after the date of filing, or Licensee makes an assignment for the benefit of its creditors, or substantially all of the assets of Licensee are seized or attached and not released within sixty (60) days thereafter, or if Licensee has ceased or threatened to cease to do business in the regular course.

Upon termination or earlier termination of this Agreement, Licensee will immediately cease a) all uses of the Confidential Information; b) Licensee agrees to destroy all adaptations or copies of the Confidential Information stored in any tangible medium including any document or work containing or derived (in whole or in part) from the Confidential Information, and certify its destruction to Avaya upon termination of this License. Licensee will promptly cease use of, distribution and sales of Licensee products that embody any such Confidential Information, and destroy all Confidential Information belonging to Avaya as well as any materials that embody any such Confidential Information. All licenses granted will terminate.

The rights and obligations of the parties contained in Sections 2.3, 2.6, 2.7, 2.10, 2.11, 2.12, 3, and 5 through 18 shall survive any expiration or termination of this Agreement.

ASSIGNMENT.

Avaya may assign all or any part of its rights and obligations hereunder. Licensee may not assign this Agreement or any interest or rights granted hereunder to any third party without the prior written consent of Avaya. The term “assign” includes, but is not limited to, any transaction in which there is a Change In Control or reorganization of Licensee pursuant to a merger, sale of assets or stock. This Agreement shall terminate immediately upon occurrence of any prohibited assignment.

COMPLIANCE WITH LAWS.

Licensee shall comply with all applicable laws and regulations, including without limitation those applicable to data privacy, intellectual property, trade secret, fraud, music performance rights and the export or re-export of technology and will not export or re-export the SDK or any other technical information provided under this Agreement in any form in violation of the export control laws of the United States of America and of any other applicable country. For more information on such export laws and regulations, Licensee may refer to the resources provided in the websites maintained by the U.S. Commerce Department, the U.S. State Department and the U.S. Office of Foreign Assets Control.

WAIVER.

The failure to assert any rights under this Agreement, including, but not limited to, the right to terminate in the event of breach or default, will not be deemed to constitute a waiver of the right to enforce each and every provision of this Agreement in accordance with their terms.

SEVERABILITY.

If any provision of this Agreement is determined to be unenforceable or invalid, this Agreement will not be rendered unenforceable or invalid as a whole, and the provision will be changed and interpreted so as to best accomplish the objectives of the original provision within the limits of applicable law.

GOVERNING LAW AND DISPUTE RESOLUTION.

Governing Law. This Agreement and any dispute, claim or controversy arising out of or relating to this Agreement (“Dispute”), including without limitation the formation, interpretation, breach or termination of this Agreement, or any issue regarding whether a Dispute is subject to arbitration under this Agreement, will be governed by New York State laws, excluding conflict of law principles, and the United Nations Convention on Contracts for the International Sale of Goods.

Dispute Resolution. Any Dispute will be resolved in accordance with the provisions of this Section 15. The disputing party shall give the other party written notice of the Dispute in accordance with the notice provision of this Agreement. The parties will attempt in good faith to resolve each controversy or claim within 30 days, or such other longer period as the parties may mutually agree, following the delivery of such notice, by negotiations between designated representatives of the parties who have dispute resolution authority.

Arbitration of Non-US Disputes. If a Dispute that arose anywhere other than in the United States or is based upon an alleged breach committed anywhere other than in the United States cannot be settled under the procedures and within the timeframe set forth in Section 15.2, it will be conclusively determined upon request of either party by a final and binding arbitration proceeding to be held in accordance with the Rules of Arbitration of the International Chamber of Commerce by a single arbitrator appointed by the parties or (failing agreement) by an arbitrator appointed by the President of the International Chamber of Commerce (from time to time), except that if the aggregate claims, cross claims and counterclaims by any one party against the other party exceed One Million US Dollars at the time all claims, including cross claims and counterclaims are filed, the proceeding will be held in accordance with the Rules of Arbitration of the International Chamber of Commerce by a panel of three arbitrator(s) appointed in accordance with the Rules of Arbitration of the International Chamber of Commerce. The arbitration will be conducted in the English language, at a location agreed by the parties or (failing agreement) ordered by the arbitrator(s). The arbitrator(s) will have authority only to award compensatory damages within the scope of the limitations of Section 8 and will not award punitive or exemplary damages. The arbitrator(s) will not have the authority to limit, expand or otherwise modify the terms of this Agreement. The ruling by the arbitrator(s)) will be final and binding on the parties and may be entered in any court having jurisdiction over the parties or any of their assets. The parties will evenly split the cost of the arbitrator(s)’ fees, but Avaya and Customer will each bear its own attorneys' fees and other costs associated with the arbitration. The parties, their representatives, other participants and the arbitrator(s) will hold the existence, content and results of the arbitration in strict confidence to the fullest extent permitted by law. Any disclosure of the existence, content and results of the arbitration will be as limited and narrowed as required to comply with the applicable law. By way of illustration, if the applicable law mandates the disclosure of the monetary amount of an arbitration award only, the underlying opinion or rationale for that award may not be disclosed.

Choice of Forum for US Disputes. If a Dispute by one party against the other that arose in the United States or is based upon an alleged breach committed in the United States cannot be settled under the procedures and within the timeframe set forth in Section 15.2, then either party may bring an action or proceeding solely in either the Supreme Court of the State of New York, New York County, or the United States District Court for the Southern District of New York. Except as otherwise stated in Section 15.3 each party consents to the exclusive jurisdiction of those courts, including their appellate courts, for the purpose of all actions and proceedings arising out of or relating to this Agreement.

Time Limit. Actions on Disputes between the parties must be brought in accordance with this Section within 2 years after the cause of action arises.

IMPORT/EXPORT CONTROL.

Licensee is advised that the SDK is of U.S. origin and subject to the U.S. Export Administration Regulations (“EAR”). The SDK also may be subject to applicable local country import/export laws and regulations. Diversion contrary to U.S. and/or applicable local country law and/or regulation is prohibited. Licensee agrees not to directly or indirectly export, re-export, import, download, or transmit the SDK to any country, end user or for any use that is contrary to applicable U.S. and/or local country regulation or statute (including but not limited to those countries embargoed by the U.S. government). Licensee represents that any governmental agency has not issued sanctions against Licensee or otherwise suspended, revoked or denied Licensee's import/export privileges. Licensee agrees not to use or transfer the SDK for any use relating to nuclear, chemical or biological weapons, or missile technology, unless authorized by the U.S. and/or any applicable local government by regulation or specific written license. Additionally, Licensee is advised that the SDK may contain encryption algorithm or source code that may not be exported

to government or military end users without a license issued by the U.S. Bureau of Industry and Security and any other country's governmental agencies, where applicable.

AGREEMENT IN ENGLISH.

The parties confirm that it is their wish that the Agreement, as well as all other documents relating hereto, including all notices, have been and shall be drawn up in the English language only. Les parties aux présentes confirment leur volonté que cette convention, de même que tous les documents, y compris tout avis, qui s'y rattachent, soient rédigés en langue anglaise.

ENTIRE AGREEMENT.

This Agreement, its exhibits, schedules and other agreements or documents referenced herein, constitute the full and complete understanding and agreement between the parties and supersede all contemporaneous and prior understandings, agreements and representations relating to the subject matter hereof. No modifications, alterations or amendments shall be effective unless in writing signed by both parties to this Agreement.

REDISTRIBUTABLE CLIENT FILES.

The list of SDK client files that can be redistributed, if any, are in the SDK in a file called Redistributable.txt.

SCHEDULE 1 TO AVAYA SDK LICENSE AGREEMENT – THIRD PARTY NOTICES

CODECS: WITH RESPECT TO ANY CODECS IN THE SDK, YOU ACKNOWLEDGE AND AGREE YOU ARE RESPONSIBLE FOR ANY AND ALL RELATED FEES AND/OR ROYALTIES, IF ANY. IT IS YOUR RESPONSIBILITY TO CHECK.

THE H.264 (AVC) CODEC IS LICENSED UNDER THE AVC PATENT PORTFOLIO LICENSE FOR THE PERSONAL USE OF A CONSUMER OR OTHER USES IN WHICH IT DOES NOT RECEIVE REMUNERATION TO: (I) ENCODE VIDEO IN COMPLIANCE WITH THE AVC STANDARD ("AVC VIDEO") AND/OR (II) DECODE AVC VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSED TO PROVIDE AVC VIDEO. NO LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION FOR THE H.264 (AVC) CODEC MAY BE OBTAINED FROM MPEG LA, L.L.C. SEE HTTP://WWW.MPEGLA.COM.

Contents

Chapter 1: Introduction

Purpose

This document describes the details about the user interface of Orchestration Designer and various speech application resources that are used to create speech applications that comply with VoiceXML, call control applications that comply with CCXML specification, and message applications that comply with TextXML.

Intended audience

This document is intended for anyone who wants to gain a high-level understanding of the product features, functionality, capacities, and limitations within the context of solutions and verified reference configurations.

Prerequisite knowledge

The primary users of Orchestration Designer are likely to be highly knowledgeable and skilled in telecommunications and Internet technologies. Therefore, this documentation does not cover topics related to those areas. The users must be proficient and knowledgeable in the following areas:

- The operating systems on which the users develop and deploy Orchestration Designer applications.
- Computer networking concepts and technologies.
- Telecommunications concepts and technologies, including switches and gateways.
- Basic programming logic and practice.

Note:

Although not required to develop applications in Orchestration Designer, knowledge and experience of Java programming is helpful.

Orchestration Designer is built on several existing technologies and tools. Hence, Orchestration Designer users must become familiar with the following technologies:

- Eclipse open-source software
- Java servlet technology
- Servlet engine technologies

- Speech recognition and synthesis technologies
- Database administration
- Web service technologies

For more information about additional resources for these technologies, see Documentation for related products and technologies on page 33.

New in this release

Avaya Orchestration Designer 8.0 includes the following:

- Change of brand name from Avaya Aura® Orchestration Designer to Avaya Orchestration Designer.
- Support for latest versions of third-party application server software:
  - Eclipse 4.14
  - Wildfly 18.0.1
  - Tomcat 9.0.30
- Support for Orchestration Designer applications written for deployment on Experience Portal.
- Support to generate a docker container for the Orchestration Designer runtime.
- Support for additional FIPS compliance enhancements.
- Support for Internet Protocol version 6 (IPv6).
- Support for the new PLDS license version that includes all the current licenses for the third-party components.

Warranty

Avaya Inc. provides a 90-day limited warranty on Orchestration Designer. Refer to your sales agreement or other applicable documentation to establish the terms of the limited warranty. In addition, Avaya's standard warranty language as well as details regarding support for Orchestration Designer, while under warranty, is available on the support website at http://www.avaya.com/support.

Viewing the Eclipse documentation

About this task

The documentation for Eclipse and supporting Eclipse components (GEF and WTP) is available at http://www.eclipse.org/documentation/, and in the form of an online Help.

- On the Eclipse user interface, click **Help > Help Contents**.

  The Eclipse user interface displays the Eclipse documentation.

Viewing the Orchestration Designer documentation

About this task

The *Getting Started with Avaya Orchestration Designer* guide is available on the Orchestration Designer installation ISO image.

You can view the Orchestration Designer documentation on the Avaya support website:

http://support.avaya.com

The Orchestration Designer documentation is also available in the form of an online Help.

- On the Eclipse user interface, click **Help > Help Contents > Avaya Orchestration Designer - Self Service**.

Documentation for related products and technologies

Orchestration Designer depends on the use of several closely related software products and technologies. When using Orchestration Designer, review the documentation of these related products and technologies.

Avaya does not reproduce or package the documentation for these related products and technologies. However, to help locate the appropriate documentation, review the following resources:

Note:

The following URLs were valid at the time of publication of this document. Avaya is not responsible if these URLs have changed. For more updated URLs, perform a search operation online.

- For Eclipse and supporting Eclipse components (GEF and WTP), go to:

  http://www.eclipse.org/documentation/

  For more information, see Viewing the Eclipse documentation on page 33.
- For the Java SDK (Software Developer's Kit), go to:

  http://docs.oracle.com/javase/8/docs/index.html

http://docs.oracle.com/javase/9/docs/index.html

- For Tomcat 7.0, 8.0, 8.5, or 9.0 go to:
  http://tomcat.apache.org/tomcat-7.0-doc/index.html
  http://tomcat.apache.org/tomcat-8.0-doc/index.html
  http://tomcat.apache.org/tomcat-8.5-doc/index.html
  http://tomcat.apache.org/tomcat-9.0-doc/index.html
- For IBM WebSphere or WebSphere Express, go to:
  http://www.ibm.com/websphere
- For WebLogic, go to:
  http://www.oracle.com/technetwork/middleware/weblogic/documentation/index.html
- For Redhat JBoss EAP, go to:
  https://developers.redhat.com/products/eap/overview/
- For JBoss Wildfly, go to:
  http://wildfly.org/
- For databases and JDBC implementation, go to:
  http://www.sql.org/
  http://www.firstsql.com/tutor.htm
  http://java.sun.com/developer/onlineTraining/Database/JDBCShortCourse/jdbc/sql.html
- For Web services, go to:
  http://www.w3.org/TR/wsdl
  http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html
- For the W3C VoiceXML 2.0 Recommendation, go to:
  http://www.w3.org/TR/voicexml20/
- For the W3C VoiceXML 2.1 Recommendation, go to:
  http://www.w3.org/TR/voicexml21/
- For the W3C CCXML 1.0 Recommendation (January 19, 2007), go to:
  http://www.w3.org/TR/ccxml/
- For the Speech Recognition Grammar Specification version 1.0, go to:
  http://www.w3.org/TR/speech-grammar/#AppJ.5

Chapter 2: Getting familiar with the Orchestration Designer user interface

Overview

Avaya Orchestration Designer is a Java-based tool that you can use to create the following:

- Speech applications that comply with VoiceXML version 2.1.
- Call control applications that comply with specifications of CCXML version 1.0, January 19, 2007.
- Message applications that comply with TextXML.

The tool is designed as an Eclipse plug-in, which provides an integrated GUI for the design and implementation of the following:

- Speech applications that can operate with Interactive Response, Voice Portal, Media Processing Server, and Avaya Experience Portal systems
- Message applications that can operate with the Avaya Experience Portal system
- Data only applications that can operate with the Avaya Experience Portal system
- HTML5 applications that can operate with the Avaya Experience Portal system

Orchestration Designer is also a suite of self-service products and Avaya Contact Center products, namely, Avaya Experience Portal, Avaya Interactive Response (IR), Media Processing Server (MPS), and Avaya Aura® Contact Center. As a single tool, you can use Orchestration Designer to design, simulate, and maintain the contact routing scripts with inbound and outbound self-service support. It accelerates service design and deployment, reduces cost, and enhances customer experience.

Orchestration Designer integrates seamlessly with Avaya Breeze® platform. With this integration, the Orchestration Designer application can interact with the Engagement Designer workflows and pass the collected data in several ways:

- Orchestration Designer can start a workflow and pass collected data to that workflow by using Context Store. The workflow receives data from Orchestration Designer to process the information and complete the transaction.
- Orchestration Designer supports integration of Orchestration Designer SMS and Email applications with Engagement Designer workflow. With this integration, the Orchestration

Designer application receives data from Engagement Designer workflow, interacts using one more text messages, and returns the data to the Engagement Designer workflow.

- An Engagement Designer workflow can initiate a new call to the customer and plug Orchestration Designer into the call to provide IVR services.
- An Engagement Designer workflow can also plug Orchestration Designer into an existing call.

Orchestration Designer works with the widely accepted Eclipse.org development framework. It provides a drag-and-drop environment for development and maintenance of speech, touchtone, and message applications.

Multi-Channel Self Service

With the Multi-Channel Self Service (MCSS), you can extend the current Experience Portal or Orchestration Designer product capabilities to include channels other than inbound voice, telephony.

Key capabilities of MCSS include:

- Application processing of inbound SMS and Email.
- Send response to an inbound SMS and email.
- Send outbound SMS and email from cross channel applications.

  For example, a speech application sending an SMS or email confirmation.
- New application type **Web** in Orchestration Designer.
- Application design palettes specific to a channel.
- Generic message flow with custom XML generation for social media and other channels.
- Transfer items to AACC agent for handling with reply using SMS and email.
- Web channel to collect information through HTML5 pages.

Inbound processing of SMS and email adds text channels to Experience Portal. You can build Orchestration Designer applications to process and respond to incoming SMS and email. With the outbound SMS and email, the application on a given channel can provide additional feedback to the user through another channel.

For example, a speech application sends confirmation of a transaction to the user by SMS or email. Items that cannot be handled in Self Service can be transferred to an AACC agent by using a connector.

Before you begin

This section contains more details on Eclipse development environment concepts and terminology used by Orchestration Designer.

In addition, this section contains information about the Orchestration Designer GUI, within the Eclipse framework, including the different areas of the Orchestration Designer workbench, menus, and toolbar options.

Note:

Before you use Orchestration Designer, you must configure the basic settings mentioned in the *Getting Started with Avaya Orchestration Designer* guide to configure the environment and ensuring that the environment is ready to use. The *Getting Started with Avaya Orchestration Designer* guide is available on the Orchestration Designer ISO image.

Accessing the Concepts section of the Eclipse Workbench User Guide

About this task

Eclipse Workbench User Guide contains an overview of many of the same concepts used within Avaya Orchestration Designer, but from the Eclipse development environment framework perspective. You can review this information to get familiar with the Eclipse user interface.

Procedure

1. Open Orchestration Designer using the Eclipse interface.
2. On the **Help** menu, click **Help Contents**.
3. In the **Help - Eclipse SDK** dialog box, in the **Contents** pane, click **Workbench User Guide**.
4. In the right pane, click **Concepts**.

   Review the sections that are available in the *Concepts* section to become familiar with Eclipse.

Orchestration Designer workbench

Note:

Be sure that you are familiar with the concepts and terminology of Eclipse. For more information, see Accessing the Concepts section of the Eclipse Workbench User Guide on page 37.

The Orchestration Designer workbench is designed as speech, message, call control, data, and web project perspectives in Eclipse. The layout of the views and workspace are optimized to assist you in creating speech, message, call control, data, and web application projects.

The Eclipse environmental elements are arranged in Orchestration Designer. However, as with any Eclipse perspective, you can arrange the perspective elements as you want. If this case, the view descriptions mentioned in this section might no longer be applicable.

These views are available in the standard Orchestration Designer **Speech**, **Message**, **Call Control**, **Data**, and **Web** perspectives. For information about menus and the toolbar items, see Orchestration Designer menu and toolbar options on page 47.

Orchestration Designer views

The Eclipse environmental elements are arranged in Orchestration Designer. However, as with any Eclipse perspective, you can arrange the perspective elements as you want. If this case, the view descriptions provided in the topics under this section might no longer be applicable.

These views are available in the standard Orchestration Designer **Speech**, **Message**, **Call Control**, and **Data** perspectives. For information about menus and the toolbar items, see Orchestration Designer menu and toolbar options on page 47.

Avaya OD Navigator view

The Avaya OD Navigator view , which is the default Eclipse view when working with Orchestration Designer, provides multiple hierarchical views of the resources in the Orchestration Designer speech, call control, message flow, and web projects.

The Eclipse Navigator view might get busy because other application developers explore resources and the code elements scattered across different speech, call control, message, data, and web applications. Hence, Avaya has developed the Avaya OD Navigator view for Orchestration Designer. The Avaya OD Navigator view helps you remain focused while exploring and navigating across the code base.

In Avaya OD Navigator view, you can:

- Drag and drop. For example, drop a .dbop file into the call flow editor to create a data node with a database operation defined.
- Start exports, except in the Show Easy Find view.
- Start resource wizards. When you select a resource in one of the views, the selection persists even if you switch to another view.

Go to variables from all view types.

Avaya OD Navigator view toolbar options

The Avaya OD Navigator view contains a hierarchical view of all your applications. The following options are available on the toolbar of the Avaya OD Navigator view:

Name	Description
Collapse All	Collapses all the expanded folders.

Table continues…

Name	Description
Link with Editor 	Links the Avaya OD Navigator item with the opened editor. When multiple editors are open and you click an editor, the system highlights the item that the editor links in the Avaya OD Navigator view. This makes it easier to locate the item. Similarly, when you click an item in the Avaya OD Navigator view, the system shows the editor that is linked to the item. **Link with Editor** synchronizes the tree view in the Avaya OD Navigator view with the opened editor.
Show Projects 	Similar to the Eclipse Navigator view, **Show Projects** displays the physical structure of Orchestration Designer projects.
Show Easy Find 	Shows all elements in your development environment, grouped by resource type, that is, logical structure, and then grouped by application name.
Show Custom 	Shows a hybrid view by project, and under project, by resource type.

Viewing the Avaya OD Navigator view

Procedure

1. On the Eclipse user interface, click **Window** and point to **Show View**.
2. Click **Other**.
3. In the Show View dialog box, double-click **Orchestration Designer**.
4. Click **Avaya OD Navigator**.
5. Click **OK**.

Viewing the properties of a resource

About this task

To view the actual resources within the resource group, you can expand any resource group such as grammars, phrases, phrase sets, and prompts. After expanding the resource group, you can view the properties of the resource.

Procedure

In the Avaya OD Navigator view, right-click a resource and then click **Properties**.

The Eclipse user interface displays a context-sensitive Properties dialog box.

Customizing the Avaya OD Navigator view

Procedure

1. On the Avaya OD Navigator view toolbar, click the down arrow, and then click **Filters and Customization**.
2. In the Available Customizations dialog box, select the customization options that you want.
3. Click **OK**.

Navigator view

The Navigator view is a standard Eclipse view that provides a hierarchical view of the resources in the Orchestration Designer speech, call control, and the message flow projects. For greater control over your resources while designing applications, see Avaya OD Navigator view on page 38.

Features of Navigator view and Avaya OD Navigator view

The following are the main features of the Navigator view and the Avaya OD Navigator view:

- When you create a speech, call control, message flow, or web project in Orchestration Designer, Orchestration Designer automatically creates the element folders within the project folder. For more information, see Folder structure of speech, message flow, data, and web projects on page 40 and Folder structure of a call control project on page 42.
- You can open the editors for the project resources, such as phrases, prompts, and grammars by double-clicking the file in the Navigator view or in the Avaya OD Navigator view. For example, to open a prompt in the prompt file editor, double-click the `*.prompt` file.
- You can also perform other actions on the project resources by right-clicking the resources, and then selecting the appropriate action from the context menu.

  Tip:

  You can use the Generate option to generate individual files and the project resources. For example, to regenerate a project grammar without regenerating the entire project, in the Navigator view or in the Avaya OD Navigator view, right-click the `*.gram` file, and then click **Orchestration Designer** > **Generate**.

For more information about the Navigator view in Eclipse, see “Navigator view” in the Eclipse *Workbench User Guide*.

Folder structure of speech, message flow, data, and web projects

- **Connectivity**: Contains the database and Web service operation files that you create as part of your project.

- **Data**: Contains mostly files that are derived from other files. For example, when you create a grammar file, the Grammar File Editor creates a `*.gram` file that contains the metadata for the grammar. When you generate the project, Orchestration Designer creates a ***.grxml** file that contains the XML grammar file. This `*.grxml` file is derived from the `*.gram` file.

  Usually, the derived files in the **Data** folder are hidden, and the system does not show the derived files in the Navigator view.

  Important:

  Do not manually edit these files.

  The **Data** folder also contains:

  - **log** folder. The system writes the log files that are created during application testing to the **log** folder. You can view these log files for debugging your applications.
  - **temp** folder. The **temp** folder stores any messages that you record during application testing.

  The **Data** folder is available only in the Navigator view.

- <***Language name***>: Contains the resource files of the speech, or the message flow, or the data, or the web projects. For example, prompt files and grammar files. When you create or import a project resource file, such as a prompt file or grammar file, Orchestration Designer stores the file in the appropriate subfolder within the <***Language name***> folder.

  The name of this folder is the name that you assign to the project primary language. For more information, see New Speech Project wizard: Specify Language Parameters page field descriptions on page 67.

- **Flow**: Contains the **main.flow** file. The **main.flow** file is the core file of the speech, or message flow, or data, or web project.

  In a speech project, the **main.flow** file contains the call flow of the speech application. In a message flow project, the **main.flow** file contains the message flow of the message application. In a data project, the **main.flow** contains the data flow of the data application. In a web project, the **main.flow** file contains the web flow of the web application. You can use the call flow editor to create and edit the call flow, the message flow editor to create and edit the message flow, the data flow editor to create and edit the data flow, and the web flow editor to create and edit the web flow.

- **Icons**: Contains several icons used for Orchestration Designer.

  Important:

  Do not delete or edit these icons.

- **WEB-INF**: Contains all output files that the system creates when you generate or build the project. These files are packaged when preparing the application for deployment.

- **Work**: The Tomcat servlet engine uses the **Work** folder as a temporary folder. The system creates the **Work** folder when you simulate the project for the first time.

  Important:

  Do not manually edit the contents of the **Work** folder.

Folder structure of a call control project

- **ccxml**: Contains the default `start.ccxml` file of the call control project. The **ccxml** folder also contains the `.ccxml` files that you create for the call control project.
- **connectivity**: Contains the database operation files and Web service operation files that you create for the call control project.
- **jsp**: Contains the `.jsp` files that you create for the call control project.
- **vxml**: Contains the VoiceXML files of the speech applications that the call control project associates.
- **WEB-INF**: Contains the output files that the system creates when you generate or build the call control project. The system packages these files when preparing the call control project for deployment.

Editor view

By default, the Editor view is located in the upper-right area of the window. The main area of the Editor view is called the “workspace”. The workspace is where you do most of the development work for building an Orchestration Designer application project.

Within the Editor view, you can start numerous subeditors to create, update, and manage an Orchestration Designer application.

For more information about the Editor view, see Editor view tabs on page 42.

Editor view tabs

Orchestration Designer (Eclipse) uses two types of tabs in the Editor view:

- **Editor tabs**: The Editor tabs are located on the top of the Editor view. When you open an editor, the Editor view displays the tab for that editor in the view toolbar. Each tab displays an icon representing the type of editor associated with the tab with the name of the project element that is currently open.
- **Page tabs**: The Page tabs are located at the bottom of the Editor view. Some editors are multiple page editors. That is, these editors have more than one workspace page. In such cases, the Editor view displays a page tab at the bottom of the view for each workspace page available in the active editor.

  For example, when you open a prompt file of a speech project in the prompt file editor, the Prompt Main and <*prompt level number*> tabs are available in the bottom toolbar of the Editor view.

Outline view

By default, the Outline view is located in the lower-left corner of the window. The Outline view is a standard Eclipse view. In Orchestration Designer, the Outline view is available only in the following conditions:

- When the call flow editor or the message flow editor is the active editor.
- When you edit a Java (`*.java`) file.

You can select any of the three views in the upper-right corner of the Outline view. For more information, see Outline view option descriptions on page 43.

Outline view option descriptions

Name	Description
Node List view 	Shows the nodes of a call flow or a message flow in an alphabetical order and also shows all symbolic node references.
Thumbnail view 	You can navigate around easier within a call flow or a message flow if the call flow or the message flow is complicated. You can drag the shaded area in the Outline view. The system updates the main workspace simultaneously to show the full size view of the shaded area.
Bookmark view 	Shows a list of all bookmarks in a call flow or a message flow. You can use this list to navigate between bookmarks in a call flow or a message flow, which is useful in large and complex call flows and message flows. For more information, see Navigating to a bookmarked node on page 149.

Application Simulator view

You can test your applications by simulating the applications in the **Application Simulator** view tab.

The **Application Simulator** view tab offers controls to the Avaya Application Simulator to test applications by simulation.

By default, Orchestration Designer shows the Application Simulator view in a tabbed notebook with the Problems view and the Tasks view, in the center of the lower area of the window. For information about the Application Simulator view options, see Application Simulator view: Application tab option descriptions on page 397.

For information about more controls, see Configuring Application Simulation preferences on page 519.

Problems view

The Problems view tab shows the errors, warnings, and informational messages that are generated while saving a project or any of the project elements. Simultaneously, Orchestration Designer displays a Code Generation error message, unless you click **Do not show me this message in the future**. The Problems view also shows any errors that are generated while compiling the Java code.

If you double-click the error, warning, or informational message in the Problems view, Orchestration Designer navigates to the exact location where the problem occurs. This feature makes it easier to debug an application and resolve the problem.

By default, Orchestration Designer shows the Problems view in a tabbed notebook with the Tasks view and the Application Simulator view, in the center of the lower area of the window.

Tasks view

The Tasks view is a standard Eclipse view. The Tasks view shows the tasks related to various types of errors that might occur, such as Java syntax errors. You can also manually add tasks to this list for things to remember to take care of.

For more information about the Tasks view in Eclipse, see the “Tasks view” section in the Eclipse *Workbench User Guide*.

By default, Orchestration Designer shows the Tasks view in a tabbed notebook with the Problems view and the Application Simulator view, in the center of the lower area of the window.

Snippets view

The Snippets view contains shortcut snippets in JSP or CCXML that you can add onto your CCXML or JSP page while building call control projects.

Properties view

The Properties view is a change of the standard Eclipse Properties view. In a layout optimized for Orchestration Designer, the Properties view displays property names and values for nodes, palette options, or other items. The properties available for editing vary according to the editor, node, or the item you are working with. For more information about the properties you can edit for a particular node, option, or other item, see the Help topic for that node, option, or item.

For more information about the standard Properties view in Eclipse, see the Eclipse *Workbench User Guide*.

By default, Orchestration Designer shows the Properties view in a tabbed notebook with the Console view, in the lower-right area of the window.

Search view

The **Avaya OD Search** displays the search results in the Search view. For more information, see Avaya OD Search on page 156.

For information about the toolbar options available in the Search view, see Search view option descriptions on page 45.

Search view option descriptions

Name	Description
Show Next Match	Highlights the next match in the Search view and opens the associated editor.
Show Previous Match	Highlights the previous match in the Search view and opens the associated editor.
Remove Selected Matches	Deletes the search results that you select.
Remove All Matches	Deletes all the search results.
Expand All	Expands the search result tree view.
Collapse All	Collapses the search result tree view.
Run the Current Search Again	Reruns the current search so that removed search results reappear or changes are reflected.
Cancel Current Search	Cancels the current search.
Show Previous Searches	Browses previously conducted searches and repeats a previous search. You can select a previous search from the drop-down menu and also clear the search history.
Pin the Search View	Retains the Search view that you pin and opens a new Search view for the next search.

Console view

The Console view shows information about the status and activity of the Tomcat server.

- If you run simulations, the system writes the CCXML, VoiceXML, TextXML, and connector logs to the Console view window.
- If you enable the debug output for the tracing function, then the Console view shows the VoiceXML output generated by the speech application and the TextXML output generated by the message application. This information is read-only, but it can be helpful in debugging

applications, especially if you can read and understand VoiceXML and TextXML codes. To enable the debug output for tracing function, see Enabling tracing for simulation on page 392.

By default, Orchestration Designer shows the Console view in a tabbed notebook with the Properties view, in the lower-right area of the window. The Console view displays in the foreground if you start Tomcat or click the **Console** tab.

Email Preview view

The Email Preview view shows a preview of the email message contained in an email file in an HTML format.

Reorganizing views or tab groups

About this task

Use this procedure to reorganize views or tab groups using the Eclipse user interface.

Procedure

1. On the Eclipse user interface, right-click tab label.
2. Select the required options to reorganize views or tab groups.

   For more information, see “Options for reorganizing views or tab groups”.

Options for reorganizing views or tab groups

The following table lists the options for reorganizing views or the tab groups:

Name	Description
Fast View	Minimizes the selected tab view, which is accessible through a clickable icon in the lower-left corner, for a “fast view” access later.
Detached	Use **Detached** for the selected tab view to detach from **Speech** or **Message** perspective, the same way that the **Move** option does.
Restore	Restores the default settings of the selected tab view. You can also click **Reset Perspective** on the **Window** menu to restore the tab view settings to the default position and size.

Table continues…

Name	Description
Move	Use **Move** to move the tab view or the complete tab group, if more than one, outside the perspective. For example, to the Desktop area outside the Eclipse window.
Size	Adjusts the size of the panel border relative to the option selected. Only borders that can be adjusted, per selected tab, are active options. Options are **Left**, **Right**, **Top**, and **Bottom**. For example, on the Avaya OD Navigator tab, to make the area wider, select **Size** > **Right**, and then drag the dark blue highlighted border to make the view wider.
Minimize	Minimizes the size of the tab view that you select.
Maximize	Maximizes the size of the tab view that you select.
Close	Closes the tab view that you select.

Orchestration Designer menu and toolbar options

Several main menu and the toolbar options are specific to Orchestration Designer. The following table shows a quick reference and the summary of these options.

Note:

This table shows the options that are specific to Orchestration Designer. This table does not include generic Eclipse options.

Table 1: Orchestration Designer menu and toolbar options

Icon	Descriptor	Description	Navigation
	Speech Project	**Speech** perspective only. Opens the wizard to create a speech project.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view
	Message Flow Project	**Message** perspective only. Opens the wizard to create a message flow project.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view

Table continues…

Icon	Descriptor	Description	Navigation
	Data Project	**Data** perspective only. Opens the wizard to create a data flow project.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view
	Web Project	**Web** perspective only. Opens the wizard to create a web flow project.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view
	Subflow	**Speech** and **Message** perspective only. Opens the wizard to create a subflow file.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view
	CCXML File Editor	**Call Control** perspective only. Opens an editor to edit CCXML files.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view
	JSP File Editor	**Call Control** perspective only. Opens an editor to edit JSP files.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view
	Call Control Project	**Call Control** perspective only. Opens a wizard to create a call control project.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view
	Database Operation File	**Call Control**, **Speech**, and **Message** perspective. Opens the wizard to create a database operation file, and you can use in either a speech application, call control application, or message application.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view

Table continues…

Icon	Descriptor	Description	Navigation
	Grammar File	**Speech** and **Message** perspective only. Opens the wizard to create a grammar file.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view
	Media file	**Speech** perspective only. Opens a wizard to create media files such as SMIL file containing audio, video, prompts, text, and graphics. You can also use the media elements in call control applications.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view
	Email	**Speech** and **Message** perspective only. Opens the wizard to create an email file.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view
	SMS	**Speech** and **Message** perspective only. Opens the wizard to create an SMS file.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view
	AACC message file	**Message** perspective only. Opens the wizard to create an AACC message file.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view
	Phraseset file	**Speech** perspective only. Opens the wizard to create a phraseset file.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view
	Prompt file	**Speech** and **Message** perspective only. Opens the wizard to create a prompt file.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view

Table continues…

Icon	Descriptor	Description	Navigation
	Textset file	**Web** perspective only. Opens the wizard to create a prompt file.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view
	Web Service Operation File	**Call Control**, **Speech**, and **Message** perspective. Opens the wizard to create an Axis Web service operation file, and you can use it either in speech application, call control application, or message application.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view
	Web Service Operation File (Axis2)	**Call Control**, **Speech**, and **Message** perspective. Opens the wizard to create an Axis2 Web service operation file, and you can use it either in speech application, call control application, or message application.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view
	Web Service Operation File (REST)	**Speech** and **Message** perspective. Opens the wizard to create a REST Web service operation file, and you can use it in a speech application or a message application.	• **File** > **New** • Main toolbar • Right-click anywhere in the Navigator view or in the Avaya OD Navigator view
	Start Tomcat	**Call Control**, **Speech**, and **Message** perspective. Starts the Tomcat servlet engine. You can use the Tomcat servlet engine to simulate applications in Orchestration Designer.	• **Tomcat** menu • Main toolbar
	Stop Tomcat	**Call Control**, **Speech**, and **Message** perspective. Stops the Tomcat servlet engine. You can use the Tomcat servlet engine to simulate applications in Orchestration Designer.	• **Tomcat** menu • Main toolbar

Table continues…

Icon	Descriptor	Description	Navigation
	Restart Tomcat	**Call Control**, **Speech**, and **Message** perspective. Restarts the Tomcat servlet engine. You can use the Tomcat servlet engine to simulate applications in Orchestration Designer.	• **Tomcat** menu • Main toolbar
No icon	Update Context option	**Call Control**, **Speech**, and **Message** perspective. Updates the Tomcat server with the current application information, known as the context definition. This action is required when you update the application, but the changes are not communicated to the Tomcat server. One indication that you might need to update the context is if you get a “General Error 404” message.	• In the Navigator view or in the Avaya OD Navigator view, right-click the <`project name`> folder, and then click **Orchestration Designer** > **Update Context**.
No icon	Generate Project option	**Call Control**, **Speech**, and **Message** perspective. Generates the project code.	• In the Navigator view or in the Avaya OD Navigator view, right-click the <`project name`> folder, and then click **Orchestration Designer** > **Generate Project**
No icon	Validate option	**Speech** and **Message** perspective. Validates the call flow, message flow, and prompts.	• In the Navigator view or in the Avaya OD Navigator view, expand the <*project name*> folder, and then expand the **flow** folder. Right-click the **main.flow** file, and then click **Orchestration Designer** > **Validate**. • In the Navigator view or in the Avaya OD Navigator view, expand the <*project name*> folder, expand the <*language name*> folder, and then expand the **prompts** folder. Right-click a .prompt file, and then click **Orchestration Designer** > **Validate**.

Table continues…

Icon	Descriptor	Description	Navigation
	Event Type Editor	**Speech** and **Message** perspective. Opens the Event Type Editor.	• Main toolbar • In the Navigator view or in the Avaya OD Navigator view, expand the <`project name`> folder, expand the `flow` folder, and then double-click **project.events**.
	Add row after	Grammar File Editor only. Adds a row after the row that you select in a grammar table.	Main toolbar
	Add row before	Grammar File Editor only. Adds a row before the row that you select in a grammar table.	Main toolbar
	Add column after	Grammar File Editor only. Adds a column after the column that you select in a grammar table.	Main toolbar
	Add column before	Grammar File Editor only. Adds a column before the column that you select in a grammar table.	Main toolbar
	Delete row	Grammar File Editor only. Deletes the row that you select, from a grammar table.	Main toolbar
	Delete column	Grammar File Editor only. Deletes the column that you select, from a grammar table.	Main toolbar

Chapter 3: Process for an Orchestration Designer application development

Checklist for developing an Orchestration Designer application

Orchestration Designer simplifies the process of creating:

- VoiceXML-compliant speech applications that you can use with Interactive Response (IR), Media Processing Server (MPS), Voice Portal, and Avaya Experience Portal systems.
- TextXML-compliant message applications that you can use with Avaya Experience Portal systems.
- HTML5-compliant web applications that you can use with Avaya Experience Portal systems.

This topic helps you better understand the process and provides practical ideas and best practices for creating your own Orchestration Designer applications. It also provides a useful checklist for the application design and development process.

The following checklist contains the recommended approach to design and build projects by using Orchestration Designer:

Step number	Process step	Reference
1	Plan the application: 1. Envision the user experience. 2. Foresee potential problems. 3. Plan the flow. 4. Plan for any modules that can be reused. 5. List application resources that must be created or imported.	Plan before you build on page 54. Orchestration Designer speech application resources on page 62. Orchestration Designer message application resources on page 83. Orchestration Designer data application resources on page 89. Orchestration Designer web application resources on page 104.

Table continues…

Step number	Process step	Reference
2	Create the speech, or message flow, or data, or web project.	Creating a speech project on page 64 Creating a message flow project on page 84. Creating a data project on page 90. Creating a web project on page 106.
3	Create or import the required application resources.	Orchestration Designer speech application resources on page 62 Orchestration Designer message application resources on page 83. Orchestration Designer data application resources on page 89. Orchestration Designer web application resources on page 104.
4	Build the application: 1. Add the applicable nodes to the flow editor workspace. 2. Connect the nodes as required. 3. Edit the nodes as required.	Building an Orchestration Designer application on page 60. Adding nodes and other items to the workspace on page 145.
5	Test the application before deploying.	Testing an Orchestration Designer application on page 60
6	Deploy the application to the run-time location.	Deploying an Orchestration Designer application on page 60

Plan before you build

Before building an Orchestration Designer application, you must carefully plan and design the application. The time that you invest in planning can greatly reduce the development time. This investment can also help you avoid problems later on.

Envision the user experience

To start planning the speech application, first envision exactly what do you want the caller to experience when calling into your system. Without being concerned with how to set it up, just ask and attempt to answer as many questions as possible.

Similarly, to start planning the message application, first envision exactly what do you want the customer to experience when sending or receiving messages from your system.

Example questions to envision the caller experience in a speech application:

- What options do you want to offer to the callers?
- Do you want to offer callers the opportunity to interact in more than one language?
- What means do you want to offer to the callers to provide their response to options? For example, by voice, by touchtone keys on the telephone, or by recording their answers or other short messages.
- Do you want to provide a visual guide to the callers in addition to voice?
- What voice gender do you want to use in your prompts?
- Do you need to use text-to-speech (TTS) technology so that the caller can hear text-based information?
- How do you want to cater to the special needs of hearing-impaired or speech-impaired callers?
- In which scenarios or situations do you think the callers require help?
- Do you want to create a multichannel speech application?
- In which scenarios do you want to transfer the call to Avaya Aura® Contact Center for assisted care?
- Do you want to create a speech application to process requests from Avaya Aura® Contact Center?

Example questions to envision the customer experience in a message application:

- Do you want to create a message application to receive and interpret inbound messages and send outbound message responses? Or do you want to create a message application to send outbound messages only?
- Do you want to offer the customers the opportunity to interact in more than one language?
- Do you want to create a multichannel message application?
- In which scenarios do you want to forward the inbound message to Avaya Aura® Contact Center for assisted care?

These questions are examples. Try to be as comprehensive as possible.

Example questions to envision the caller experience in a data application:

- Will this application need to be used by multiple channels?
- How will this application be invoked?
- Will this application run standalone or as a submodule or both?

Example questions to envision the customer experience in a web application:

- Do you foresee many customers with mobile devices when interacting with your system?
- Will the user get a text message, either SMS, or an Email to start a conversation?
- Will it be more convenient for customers to enter answers on a mobile web browser?

- Will interactions with customers be enhanced if you display information visually on a mobile web browser?

Anticipate potential problems and issues

Another important step in planning a speech application, message, and data application is to anticipate potential problems and error conditions. How the system must respond when one of these problem situations arises?

Example questions to anticipate potential problems in a speech application:

- What are the technical or hardware limitations?

  What if a caller does not have a touchtone telephone? How your system must respond to TTY or TDD requests?

- Is accessibility for callers with physical limitations required?

  Have you allowed for the extra time it takes for callers with physical handicaps or other limitations?

- What language limitations will be incurred?

  Some of your callers may need to interact using a language other than the primary language? Do you have the necessary automatic speech recognition (ASR) and TTS servers and software to accommodate them?

- What personal preferences are needed for callers?

  Have you allowed for the personal preferences of your callers? Some people prefer interacting with the system verbally, while others prefer to use dual-tone-multi-frequency (DTMF) responses. And some callers prefer to interact with a live attendant no matter how good your interactive voice response (IVR) speech application is.

  Such preferences are among the reasons that you must always provide callers with the option to speak with a live attendant. Also, your system cannot deal with all eventualities and situations in which a caller needs help. Times are there when only a live attendant can take care of the need.

Example questions to anticipate potential problems in a message application:

- Have you ensured that the message application does not get into an infinite loop?

  If you create a message application to receive inbound messages from customers, notification messages, and delivery status receipts, ensure that you design the message application in such a way that the message application first verifies the message type of the inbound message before processing the message. Otherwise, the message application might get into an infinite loop.

  For example, if you assign a prompt to a No Match event and if your message application processes the inbound messages without verifying the message type, then for every inbound message, including the notification messages and delivery status receipts, for which no

match is found, the system executes the prompt and sends an outbound message to the customer. The outbound message in turn generates additional notification messages.

Ensure that the message application does not get into an infinite loop.

Example questions to anticipate potential problems in a data application:

The following are the two common approaches to map the flow for a data application:

- Describe the flow verbally. Verbally talk through all scenarios. Record these verbal walk-throughs.
- Use a flow diagram.
- Have you ensured that the data application does not get into an infinite loop?

Example questions to anticipate potential problems in a web application:

- What if the user is not using a mobile device?
- Do you have an application to send out a link (to launch the web application) through text messages?
- How does the application fit into your multi-channel strategy?

Map the flow

While envisioning the experience that you want your callers or customers to have, try to foresee and plan for any problem contingencies that can arise. Begin to map the flow of the speech application, or the message application, or the data application. Be as complete and as comprehensive as possible. Try to foresee every possibility, and plan how the system must respond.

The following are the two common approaches to map the flow for a speech application:

- Describe the flow verbally. Verbally talk through all the scenario. Note the location where the prompts occur and what the callers must say or do. Record these verbal walk-throughs.
- Use a flow diagram. Create a flow diagram to show the major points in the call flow. Use this diagram to show where in the call flow:
  - Offer options to the caller.
  - The caller must listen to the entire prompt and where can the caller interrupt, or “barge in,” and cut off the prompt.
  - The speech application must provide a visual context.
  - A response from the caller is required and what should be the valid responses.
  - The speech application must send an SMS or email message to the caller.
  - The speech application must transfer the inbound call to Avaya Aura® Contact Center for assisted care.
  - The speech application must access a database to retrieve or record customer data.

- The speech application must access a web service to respond to a customer request.

The following are the two common approaches to map the flow for message applications:

- Create a list of the messages that the system must send to the customers and the expected responses from the customers.
- Use a flow diagram. Create a flow diagram to show the major points in the message flow. Use this diagram to show where in the message flow:
  - Offer options to the customer.
  - You require a response from the customer and what must be the valid response.
  - The message application must send an SMS or email message to the customer.
  - The message application must forward the inbound message to Avaya Aura® Contact Center for assisted care.
  - The message application must access a database to retrieve or record customer data.
  - The message application must access a web service to respond to a customer request.

The following are the common approaches to map the flow for web (HTML5) applications:

- Identify the data and its types to collect from the customers.
- Categorize input fields and organize them into different web pages.
- Decide the data processing and visual feedback that you want to provide to the users between each data collecting web page.
- Design the flow in a sequence to collect and process information. Finally, give the feedback to the user for confirmation.

Create a modular design

When planning a speech, or a message, or a web application, identify the parts of the application that can be reused. When designing and building the application, create these parts as speech, or message, or web project modules that you can reuse wherever that functionality is needed.

If you plan these modules in advance, you can develop these models before developing the main application project file. You can use these modules when creating your main application.

For example, you want to collect the bank account number or credit card number from the callers at several instances in the call flow. You can create a reusable module that collects the bank account number or the credit card number and use the reusable module at multiple instances in the call flow. You can also use the reusable module in other speech applications.

Advantages of modular design

Using a modular approach to application design has the following advantages:

- Develop one time and use many times. Using a modular approach to application design has tremendous efficiency advantage when certain actions or options are offered to callers or customers at several places within the application.

- Maintenance ease. The application is easier to maintain, even if you are not reusing many codes. When using a modular approach, you can change the code in one part of the application without necessarily requiring a rebuild of the entire application.
- Debugging ease. A modular approach makes it easier to debug an application because you can isolate the trouble areas where errors occur.
- Concurrent engineering capabilities. A team of developers can work on the different pieces of an application separately.

Identify and list application resources

When planning a speech application or a message application, identify and list all required application resources. For a list and description of the types of application resources available in Orchestration Designer, see Orchestration Designer speech application resources on page 62 and Orchestration Designer message application resources on page 83.

After you plan the flow completely, keeping in mind all required application resources, list these resources before actually developing the application. You can then create the resources or import the resources as required within the application.

For example, if you know the phrases that are required in a speech application and you want to get these phrases recorded by a professional talent, you can list all phrases and get the phrases recorded as `*.wav` files before starting the development. Then, when developing the speech application, you can import and use the prerecorded files in the speech application.

Conventions for naming Java components

The following conventions and restrictions apply when naming Java components such as projects, folders, and files.

Important:

Do not use Java reserved words such as Try, Catch, and Switch as identifiers for project name or grammar name.

Java component names:

- Cannot contain spaces
- Are case-sensitive
- Can contain any of the following characters as the initial character:
  - An alphabetic character, either lowercase or uppercase
  - An underscore character (_)
- Can contain the following characters as subsequent characters:
  - Alphabetic characters, either lowercase or uppercase

- Underscore characters (_)
- Numbers

Caution:

Do not use double-byte or extended ASCII characters when naming Java components.

Building an Orchestration Designer application

Building an application in Orchestration Designer involves:

1. Placing nodes in the workspace area.
2. Connecting nodes to create a call flow or a message flow.
3. Configuring nodes.

The call flow represents the possible paths a caller can follow while interacting with your interactive voice response (IVR) system.

The message and the data flow represents the possible paths a customer can follow while interacting with your text processing system.

The web flow represents the possible paths a user can follow while interacting with your web application.

Testing an Orchestration Designer application

Avaya recommends that you test your application while developing the application and before deploying the application. Orchestration Designer offers an application simulation and debugging environment in which you can exhaustively test your application before going live. By using the simulation and debugging tools, you can verify that the application works the way you want. You can also eliminate most of the problems with your application before you deploy the application to a live system.

For more information, see Testing applications by simulation on page 386.

Deploying an Orchestration Designer application

The final step in developing a project is to deploy the application, so that you can use the application as intended. The deployment of the project depends on the objective you set and your expectations from the project.

Before deploying the project, you must build the project and generate the code. Depending on what application execution environment you have created the application for, the deployment process creates a Web archive (WAR) file or an enterprise archive (EAR) file.

If you have designed the project to be a subdialog or a reusable module, which you can use in a larger application, you must select the option that deploys the project to a directory for reusable modules. You can then import the module into another speech, or message flow, or data flow, or web project and use the project as a module node.

Note:

You can import a reusable module only in a project that is created for a channel that is same as that of the reusable module. For example, you can import a reusable speech application module only in a speech project.

If you have designed the project to be the primary application or a standalone project, you can transport the WAR or EAR files to the server on which you want to execute the application.

For more information about deploying applications, see Module nodes on page 143 and Application deployment on page 416.

Chapter 4: Speech applications

Orchestration Designer speech application resources

Many nodes and other flow items in an Orchestration Designer speech application use various application resources, such as phrases, prompts, and grammar. You can create, configure, and modify these application resources by using the resource-specific wizards.

For example, before the **Announce** node is fully functional, you must:

1. Create one or more prompts by using the New Prompt wizard.
2. Assign the prompts to the **Announce** node.
3. Populate the prompts in the **Announce** node.
4. Define the next node that the speech application must execute after the **Announce** node.

Tip:

Carefully plan the speech application and list the application resources that are required. Create or import the application resources before building the call flow. That way, the resources are ready and available when needed.

The following is a list of the application resources available in Orchestration Designer:

- Phrases. Phrases are prerecorded audio files that are called and played back to callers, or text strings that are synthesized into audible form by using text-to-speech (TTS) technology.

  For more information, see Phrases on page 165.
- Phrasesets. Phrasesets are used to group phrases that are usually related to a particular speech application. The main advantage of grouping phrases, besides better organization, is that phrasesets improve file resource efficiencies, which translates into quicker build times.

  For more information, see Phrasesets on page 167.
- Prompts. Prompts are speech elements that are used to announce information to a caller or prompt the caller to provide a specific type of response. Prompts are composed of one or more segments. These segments can consist of phrases, audio variables, text variables, or TTS text. You can also use conditional statements to determine which prompt segments must be used in certain circumstances.

  For more information, see Prompts on page 198.
- Media. Media elements include audio, video, static image, and text files. To create media prompts, you can reuse the existing audio prompts editor with the media palette items and the settings. You can now intermix the existing audio elements with the new media elements

within the same prompt to achieve synchronization between the respective servers or players.

For more information, see Media on page 185.

- Variables. Variables in Orchestration Designer can take any of several different specialized forms. Some of the specialized forms include audio variables and text variables used for TTS, and the standard programming variable types.

  For more information, see Variables on page 226.

- Grammars. Grammars are speech elements used in conjunction with the automated speech recognition (ASR) technology. Grammars are lists of possible responses that the callers provide when responding to the prompts using spoken replies. Grammars define which words or phrases the ASR engine can recognize and respond to.

  For more information, see Grammars on page 242.

- Languages. In Orchestration Designer, you can assign more than one language to a speech project. You can create applications that offer callers the option to select languages, provided those languages are available on your system.

  For more information, see Language and localization on page 276.

- Database operations. You can use database operations to connect and interact with SQL databases. Using database operations, you can write the information collected from a caller to the database. You can also retrieve information from the database to make the information available to the caller.

  For more information, see Database operations on page 295.

- Web service operations. Web services are Internet-based applications that you can use to perform a wide variety of functions. In Orchestration Designer, you can start web services by using a Web-service operation file.

  For example, you can use a web service to enable callers to get stock quotes or weather forecasts for their region. Or you can use Web services to enable callers to find out the availability of airline flights or hotel reservations. Almost any Web service in which the information can be presented using audio files or text-to-speech can be used in an Orchestration Designer application.

  For more information, see Web Services on page 320.

- Event types. Orchestration Designer includes most of the common types of event handlers as part of the options available from the palette. In addition, you can use the Event Handler Editor to create your own custom event types.

  For example, you can create a custom event handler to throw an exception when the amount of a transaction request exceeds the currently available limit on a credit card account belonging to the caller. Then, you can add this event type to the application so that, when the application throws this event, the application goes to a node that informs the caller that the credit limit does not allow this transaction to be completed.

  For more information, see Events on page 314.

- SMS files. Use SMS files in a speech application to send SMS messages, also called as notifications.

  For more information, see SMS files on page 347.
- Email files. Use email files in a speech application to send email messages, also called as notifications.

  For more information, see Email files on page 351.
- AACC treatments. Use AACC treatments to process the calls that Avaya Aura® Contact Center transfers to the Orchestration Designer speech application. The Orchestration Designer speech application processes the calls based on the treatment type that Avaya Aura® Contact Center requests.

  For more information, see AACC treatments on page 371.

Speech project management

Creating a speech project

About this task

Use the New Speech Project wizard to create the basic framework for the speech application project.

Before you begin

Before you create a speech application using Orchestration Designer, you must plan the speech application by envisioning exactly what do you want the caller to experience when calling into your system.

For information about planning an application, see Plan before you build on page 54.

Procedure

1. On the Eclipse user interface, click **File** > **New** > **Project**.
2. In the New Project wizard, double-click **Avaya OD Development**.
3. Click **Speech Project**
4. Click **Next**.

   The Eclipse user interface displays the New Speech Project wizard.
5. On the Create a Speech Project page, specify a name and location for the project and click **Next**.
6. On the Specify Project Parameters page, specify the project parameters and click **Next**.
7. On the Specify Language Parameters page, set the parameters for the primary language to be used in your project.

You can specify only one primary language for each language parameter.

8. Click **Finish**.

   The Eclipse user interface opens the project flow editor, where you can design and build your project call flow.

   For more information about the project flow editor, see Overview of Orchestration Designer project flow editor on page 129.

   For information about setting the project properties, see Orchestration Designer project properties on page 490.

New Speech Project wizard: Create a Speech Project page field descriptions

You can use the **Create a Speech Project** page to specify a name and location for the Orchestration Designer project.

Note:

The **Create a Speech Project** page is mandatory.

Name	Description
Project name	Name for the project. **Caution:** Do not use double-byte or extended ASCII characters or Java reserved words such as Try, Catch, and Switch when naming the project. **Note:** Project names must follow Java naming conventions. For more information, see Conventions for naming Java components on page 59.
Use default location	Check box to use the default project location for generating and saving the project files. Avaya recommends that you use the default location.
Location	Full path of the directory in which you want to generate and save the project files. To use a different location, clear the **Use default location** check box.

New Speech Project wizard: Specify Project Parameters page field descriptions

You can use the **Specify Project Parameters** page to specify meta information and a description for your project. Orchestration Designer does not use this information in the application, nor does the information appear anywhere in the output. This information is primarily for your internal use. Orchestration Designer saves and displays this information only as part of the project properties. However, this information can be useful.

For example, if you are developing similar applications or slight variations of an application for different clients, you can use these fields to specify the client name, information about the purpose of the application and how does the application vary, and so on.

You can also specify information about the VXML compatibility and category for your project on the **Specify Project Parameters** page.

Name	Description
Project	
VXML Compatibility	The VXML version that is compatible with the run-time platform on which the application is targeted to run. By default, Orchestration Designer sets the project VXML compatibility to **2.1**. **Tip:** To run video-enabled application (IVVR), you must set the project VXML compatibility to **2.1 + IVVR**.
Meta Information	
Vendor	The vendor name or the name of vendor company.

Table continues…

Name	Description
Category	The category name for the application or the project module. For example, if you are building a banking application and this module handles savings account queries, you can type **Savings query module**. **Tip:** If you intend to use this project as a reusable module, what you specify in the **Category** field determines where in the Call Flow Editor palette this module appears. If you leave this field blank, the module appears in the **Modules** group. If you specify a category name, the module is grouped with other modules that have the same category name.
Description	Brief description for the project. You can provide information such as the variable values that are accepted and returned from this module, and other similar information.

New Speech Project wizard: Specify Language Parameters page field descriptions

You can use the **Specify Language Parameters** page to set the parameters for the primary language to be used in your project. You can specify only one (primary) language for each language parameter.

You can add more languages later. For more information, see Language and localization on page 276.

Note:

Specify the language parameters on the **Specify Language Parameters** page only if you want to set the language options.

Name	Description
Language Package Name	A logical name for the primary language for your project. The name does not impact the run-time language settings. Orchestration Designer uses this name for the <*language name*> directory in the Navigator view. **Tip:** Specify a descriptive name such as **US_Eng** for U.S. English or **Can_Fr** for Canadian French. For information about file-naming conventions and restrictions, see Conventions for naming Java components on page 59.
Speech Recognition (ASR) Language	The name of the ASR package that you want to use for speech recognition. This field contains a list of ASR languages supported in Orchestration Designer. **Note:** See the documentation of your ASR run-time engine and use of the engine-supported language codes configured on the voice platform. For example, some engines or platforms do not accept “en-us.” Instead, these engines or platforms require “en-US.”
Text-to-speech (TTS) Language	The name of the TTS package that you want to use for speech synthesis. This field contains a list of TTS languages supported in Orchestration Designer. **Note:** See the documentation of your TTS run-time engine and the use of engine-supported language codes configured on the voice platform. For example, some engines or platforms do not accept “en-us.” Instead, these engines or platforms require“en-US.”

Table continues…

Name	Description
Localization Bundle Language	The localization bundle that you want to use. Localization bundles are packages of classes in a *.jar file that the servlet engine uses to convert the run-time application data to the form of audio variables. The VoiceXML interpreter, Avaya Application Simulator, translates this run-time data to standard phrases and plays the phrases back in the form of prerecorded audio files. This field contains a list of localization bundles supported in Orchestration Designer.

VXML subdialog management

Using VXML subdialogs as reusable modules in Orchestration Designer

You can import a VXML subdialog as a reusable module in an Orchestration Designer speech project, and then use the reusable module in the Orchestration Designer speech project.

You can import:

- VXML subdialogs from a .war file created by using a different software or provided by a third-party vendor.
- A .war file of an Orchestration Designer reusable module. You can use this feature when you want to import an Orchestration Designer reusable module that is developed by using Orchestration Designer installed on some other computer and is deployed on some other application server.

When you specify a .war file that you want to import, Orchestration Designer automatically detects whether the .war file is of an Orchestration Designer reusable module or the .war file contains Nuance OSDM or NDM VXML subdialogs.

Orchestration Designer cannot detect third-party VXML subdialogs, except Nuance OSDM and NDM, contained in the .war file. To import other third-party VXML subdialogs, you must specify the number of VXML subdialogs contained in the .war file and the module definitions for the VXML subdialogs.

Note:

Nuance is planning end of life for Open Speech Dialog Module (OSDM) in March 2014. Nuance does not certify OSDMs against Apache Tomcat 7. Therefore, you should upgrade to

Nuance Dialog Module (NDM) for future support and changes. For more information, refer to Nuance documentation and support.

When you want to integrate a third-party VXML subdialog in an Orchestration Designer speech application without using a .war file, specify a module definition for the third-party VXML subdialog and use the VXML subdialog as a reusable module in Orchestration Designer.

You can reuse the reusable modules in more than one speech applications. You can use the reusable modules when common project elements are needed in more than one speech applications and also when you use a speech module that uses the same phrases as the parent application.

Importing a VXML subdialog

About this task

Use this procedure to import a VXML subdialog as an Orchestration Designer reusable module, or Nuance OSDM or NDM VXML subdialogs.

Procedure

1. On the Eclipse user interface, click **File** > **Import**.
2. In the Import wizard, double-click **Avaya OD Development**.
3. Click **Import VXML Subdialog as Reusable Module** and then click **Next**.
4. On the Specify Import Details page, click **Import VoiceXML subdialog from Web Archive (WAR) file**.
5. Select the `.war` file from which you want to import an Orchestration Designer reusable module or other VXML subdialogs.

   Orchestration Designer automatically detects if the `.war` file contains an Orchestration Designer reusable module, or Nuance OSDM or NDM VXML subdialogs. However, it cannot detect third-party VXML subdialogs contained in the `.war` file.
6. If the `.war` file contains an Orchestration Designer reusable module, or Nuance OSDM or NDM VXML subdialogs, do the following:
   a. Click **Next**.
   b. On the Verify Module Information page, verify the parameters that you want to import.
   c. Click **Finish**.
7. If the `.war` file contains third-party VXML subdialogs, other than Nuance OSDM or NDM VXML subdialogs, do the following:
   a. In the **Number of reusable dialogs contained in archive** field, type the number of VXML subdialogs in the `.war` file.
   b. Click **Next**.

c. On the Set information about this module page, click a module and specify a module definition for the module.

   The number of modules in the module list depends on the number of VXML subdialogs that you specify in the **Number of reusable dialogs contained in archive** field.

   You must specify module definitions for all modules in the module list.

d. Click **Finish**.

Result

The Eclipse user interface imports the Orchestration Designer reusable module and VXML subdialogs as reusable modules in the `<eclipse>/Orchestration Designer/Modules` directory, and displays the reusable modules in the form of module items in the Modules Palette of the project flow editor.

Integrating a third-party VXML subdialog

About this task

Use this procedure to integrate a third-party VXML subdialog without importing the VXML subdialog.

Procedure

1. On the Eclipse user interface, click **File** > **Import**.
2. In the Import wizard, double-click **Avaya OD Development**.
3. Click **Import VXML Subdialog as Reusable Module** and then click **Next**.
4. On the Specify Import Details page, click **Manually define module definition**.
5. Click **Next**.
6. On the Set information about this module page, specify the module definition for the third-party VXML subdialog that you want to use in Orchestration Designer without importing the VXML subdialog.
7. Click **Finish**.

Result

After you specify the module definition for a VXML subdialog, the Eclipse user interface displays the reusable module in the form of a module item in the Modules of the Palette pane of the Call Flow Editor.

Import VXML Subdialog as Reusable Module wizard: Specify Import Details page field description

Name	Description
Import VoiceXML subdialog from Web Archive (WAR) file	Option to import an Orchestration Designer reusable module or other VXML subdialog from a Web Archive (WAR) file.
Select WAR file	Directory path of the .war file from which you want to import an Orchestration Designer reusable module or other VXML subdialog.
Type	Type of the VXML subdialog contained in the .war file. The system automatically detects whether the .war file is of an Orchestration Designer reusable module or the .war file contains Nuance OSDM or NDM VXML subdialogs. The following are the available types: • **Orchestration Designer reusable dialog (module)**: If the .war file contains an Orchestration Designer reusable module. • **Nuance NDM/OSDM**: If the .war file contains Nuance OSDM or NDM VXML subdialogs. • **Other/Unknown**: If the .war file contains third-party VXML subdialogs, other than Nuance OSDM or NDM VXML subdialogs.
Number of reusable dialogs contained in archive	Number of VXML subdialogs contained in the .war file that you want to import as reusable modules. If the .war file is of an Orchestration Designer reusable module, the system displays **1**. If the .war contains Nuance OSDM or NDM VXML subdialogs, the system displays the number of VXML subdialogs contained in the .war file. If the .war file contains third-party VXML subdialogs, other than Nuance OSDM or NDM VXML subdialogs, then the system cannot detect the number of VXML subdialogs contained in the .war file. You must type the number of VXML subdialogs contained in the .war file. For more information, see Import VXML Subdialog As Reusable Module wizard Set information about this module page field description on page 74.

Table continues…

Name	Description
Manually define module definition	Option to create a module definition for a VXML subdialog without importing a .war file. You can use this option to integrate third-party VXML subdialogs without importing the VXML subdialogs.

Import VXML Subdialog As Reusable Module wizard: Verify Module Information page field description

Name	Description
Module	The name of the Orchestration Designer reusable module, or the name of the Nuance OSDM or NDM VXML subdialog contained in the .war file.
Name	The name of the Orchestration Designer reusable module, or the Nuance OSDM or NDM VXML subdialog that you select in the **Module** field. This field is read only.
Version	The version number of the Orchestration Designer reusable module, or the Nuance OSDM or NDM VXML subdialog that you select in the **Module** field. This field is read only.
URL	The URL of the Orchestration Designer reusable module, or the Nuance OSDM or NDM VXML subdialog that you select in the **Module** field. This field is read only.
Category	The category name of the Orchestration Designer reusable module, or the Nuance OSDM or NDM VXML subdialog that you select in the **Module** field. The system uses the category name for grouping the module in the **Palette** pane of the Call Flow Editor. This field is read only.
Vendor	The name of the third-party integration vendor. For OSDMs and NDMs, the vendor is Nuance. This field is read only.
Description	The description of the Orchestration Designer reusable module, or the Nuance OSDM or NDM VXML subdialog that you select in the **Module** field. This field is read only.

Table continues…

Name	Description
Languages	The language of the Orchestration Designer reusable module, or the Nuance OSDM or NDM VXML subdialog that you select in the **Module** field. This field is read only.
Dependencies	Any other modules on which the Orchestration Designer reusable module, or the Nuance OSDM or NDM VXML subdialog that you select in the **Module** field has resource dependency. This field is read only.
Input Parameters	The input parameters of the Orchestration Designer reusable module, or the Nuance OSDM or NDM VXML subdialog that you select in the **Module** field. This field is read only.
Output Parameters	The output parameters of the Orchestration Designer reusable module, or the Nuance OSDM or NDM VXML subdialog that you select in the **Module** field. This field is read only.

Import VXML Subdialog As Reusable Module wizard: Set information about this module page field description

Name	Description
Drop-down list	The VXML subdialog for which you want to create a module definition. On the Specify Import Details page: • If you specify a .war file that contains third-party VXML subdialogs, except Nuance OSDM or NDM VXML subdialogs, and then specify the number of VXML subdialogs contained in the .war file, then the system automatically creates names for the VXML subdialogs and displays the names in this field. • If you select **Manually define module definition**, then the system automatically creates a name for the VXML subdialog and displays the name in this field.

Table continues…

Name	Description
Name	Name for the VXML subdialog that you want to create. For OSDMs on external servers, the name must start with OSDM_ and the record must be OSDM_Record.
Version	Version number for the VXML subdialog that you want to create.
URL	The URL of the Servlet/JSP/VXML subdialog for the module being created. This URL is mandatory. **Note:** After you specify the URL, you can modify the URL in the **module.<*module name*>.entry point** context parameter or in the Java code. For more information about modifying the URL by using the **module.<*module name*>.entry point** context parameter, see: • Configuring Orchestration Designer project development, runtime, and Java servlets settings on page 511 • Application tab field descriptions on page 511 • Context parameters for an Orchestration Designer project on page 514.
Category	A category name for the VXML subdialog, if applicable. The system uses the category name for grouping the VXML subdialog in the **Palette** pane of the call flow editor.
Vendor	The name of the third-party integration vendor. For OSDMs and NDMs, the vendor must be Nuance.
Description	A description for the VXML subdialog that you want to create.
GET	To apply the Get method to the VXML subdialog call.
POST	To apply the Post method to the VXML subdialog call.
Input Parameters	**Add** to add and define input parameters, value, and description for the VXML subdialog. For OSDM support, the add parameters are specified in the OSDM documentation.
Send as namelist	Check box to pass parameters through the namelist, such as OSDMs or as child parameter items, rather than params.

Table continues…

Name	Description
Output Parameters	**Add** to add and define output parameters, value, and description for the VXML subdialog. For OSDM support, the add parameters are specified in the OSDM documentation.
Return Multi-part Data	Check box to post data of different media types from the subdialog call. For example, MIME or multi-part such as OSDM record.

Creating module definition for a Nortel SCE application

About this task

Use this procedure to create Orchestration Designer module definition for a Nortal SCE application.

Procedure

1. On the Eclipse user interface, click **File** > **Import**.
2. In the Import wizard, double-click **Avaya OD Development**.
3. Click **Import VXML Subdialog as Reusable Module** and then click **Next**.

   The Eclipse interface displays the Import VXML Subdialog as Reusable Module wizard.
4. Click **Manually define module definition**.
5. Click **Next**.
6. On the Set information about this module page, specify the module definition for the Nortel SCE application.
7. Click **Finish**.

Result

After creating a module definition for the SCE application:

- The Eclipse user interface displays the module in the Palette pane of the project flow editor.
- Orchestration Designer creates a folder in `<eclipse home>/OrchestrationDesigner/Modules` directory in the SCE application. To share this folder with other users, move the folder to the `<eclipse>/OrchestrationDesigner/Modules` directory and restart Orchestration Designer.

Module definition for a Nortel SCE application field descriptions

Name	Description
Name	A name for the SCE application.
Version	A version number for the SCE application.
URL	The URL in the following format: http://<*sce web server*>:8080/CallRouter/callrouter Where <*sce_web_server*> is the IP address of the SCE Web server.
Category	A category name for the SCE application.
Vendor	The name of the vendor, that is, Nortel.
Description	A description for the SCE application.
Get	To apply the Get method to call the SCE application.
Post	To apply the Post method to call the SCE application.
Input Parameters	**Add** to add the following parameters: • **SERVICE**: The name of the SCE application as the default value for the SERVICE parameter. • **SUBDIALOG**: **True** as the default value for the SUBDIALOG parameter. • **CALL_ID**: The call id (session:sessionlabel) or the variable that identifies the call as the default value for the CALL_ID parameter. In addition to the above input parameters, you can add other input parameters that you want to pass to the SCE application.
Send as namelist	Check box to pass parameters through the namelist instead of params.
Output Parameters	The output parameters to receive values from the SCE application.
Return Multi-part Data	Check box to post data of different media types from the subdialog call.

Using a Nuance NDM in an Orchestration Designer speech application

After you import a Nuance NDM (Nuance Dialog Module) as a VoiceXML subdialog in an Orchestration Designer speech application, the Nuance NDM VoiceXML subdialog is available in the Palette pane of the call flow editor.

If you want to use a Nuance NDM 5.0 or 5.2 VoiceXML subdialog in an Orchestration Designer speech application, add the NDM VoiceXML subdialog from the Palette pane to the call flow editor.

If you want to use a Nuance NDM 5.2.x or later VoiceXML subdialog in an Orchestration Designer speech application, then you must first import the WAR file, which Nuance provides, that contains the Nuance core dialog modules in the Orchestration Designer speech application. The Nuance core dialog modules are prepackaged, reusable modules. In addition to other Nuance core dialog modules, the WAR file also contains the SessionStart and SessionEnd core dialog modules.

NDM 5.2.x and later uses the SessionStart module to start the NDM session and the SessionEnd module to end the NDM session.

For more information about the WAR file that contains the Nuance core dialog modules, see the Nuance documentation.

After you import the WAR file in the Orchestration Designer speech application, the **NDM_Sessionstart** and **NDM_Sessionend** items are available in the Palette pane of the call flow editor.

To use the NDM 5.2.x or later VoiceXML subdialogs in the Orchestration Designer speech application, you must add the **NDM_Sessionstart** item from the Palette pane to the call flow editor. You must then add the NDM 5.2.x or later VoiceXML subdialogs that you want to use from the Palette pane to the call flow editor. After you add all the NDM VoiceXML subdialogs that you want to use, add the **NDM_Sessionend** item from the Palette pane to the call flow editor.

Orchestration Designer uses the **NDM_Sessionstart** item to start the NDM session and the **NDM_Sessionend** item to end the NDM session.

Inserting a custom VoiceXML code

About this task

Use this procedure to insert a custom VoiceXML code to return VoiceXML to the VoiceXML browser.

Caution:

Use the VXML Servlet node with caution as this can cause your applications to stop responding or display run-time errors.

Procedure

1. From the Palette pane, drag a VXML Servlet node into the project flow editor.
2. Designate the next node for the VXML Servlet node to proceed to, when finished.
3. In the **Avaya OD Navigator** view, select the project and then press `Control+Shift+S` to save the project.

   Note:

   Before editing the Java class, connect the VXML Servlet node to the next in the call flow and save the speech project. Orchestration Designer does not allow editing of the Java class until Steps 2 on page 79 and 3 on page 79 are completed.
4. After the project is built, right-click the VXML Servlet node, and select **Edit <VXMLservlet>.java** from the pop-up context menu. <*VXMLservlet*> is the name of your VXML servlet node.
5. In the Java class editor, add the method `public void markupLanguageGeneration(PrintStream out, Submit submit, SCESession session) {` along with your custom VoiceXML code.

   Note:

   The ability to integrate variables into and from the VXML Servlet node is not easily supported. Do not use this node to pass variable values.

Chapter 5: Message applications

Message applications overview

Create and use a message application to automate the process of interpreting an inbound message and sending an automated outbound message in response. You can also create a message application to send proactive outbound messages to customers.

For example, you frequently receive SMS messages from customers requesting the price catalog of your products. You can create a message application for the SMS channel to automatically process and interpret the inbound SMS messages, and then send an automated SMS message response to the customers.

You can create the following two types of message applications:

- SMS channel message application
- Email channel message application

In a message application, you can create a message flow that determines the flow of the message application the same way in which you create a call flow in a speech application. Some application items, nodes, and resources that are available in speech applications are not available in message applications. Also, the purpose of and the configuration required for some items, nodes, and resources in a message application are different from that in a speech application.

For example, an Announce node in a speech application uses speech to announce a prompt to the caller. Whereas, an Announce node in a message application uses text to send a message that is contained in the prompt.

You must host a message application as a managed application on Avaya Experience Portal.

Unlike speech applications that use a voice browser to parse speech data, message applications use a text browser to parse text data. Text browser is the text processing engine for the Orchestration Designer message applications. The system uses the same text browser to run a message application on the platform as well as in a simulation environment. Text browser supports only the message applications that are created by using Orchestration Designer.

Alternatively, you can also use an external text processing engine to analyze and interpret the inbound message. If you use an external text processing engine, then do not use the Collect node in the message application. Instead, use the Servlet node to write a Java code to integrate the external text processing engine to Orchestration Designer. You can integrate an external text processing engine instead of using the built-in text processing with grammars in the TextXML browser.

Orchestration Designer uses the message gateway to receive the inbound SMS message or email message, determine the appropriate message application to handover the inbound message, and then launch the text browser to run the message application. The text browser is installed on VPMS. The text browser is the link between the message gateway and the Orchestration Designer message application.

The text browser supports TextXML schema. The text browser validates each input document against the TextXML schema. TextXML starts with the <TextXML> tag and follows a structure similar to the VoiceXML structure. Unlike speech applications in which you can create or modify VoiceXML, you cannot create or modify TextXML in message applications. The Orchestration Designer runtime generates the TextXML.

The text browser uses grammars to interpret the body of the inbound SMS message. For an email message, the text browser uses grammars to interpret the subject and the body of the inbound email message.

The text browser supports single-slot SRGS literal grammars as well as built-in grammars. For more information about grammars used in message applications, see Grammar compatibility and compliance for speech application on page 244 and Supported built-in grammar types in message applications on page 269.

The text browser invokes a message application from the application server by using the HTTP or HTTPS Get and Post requests. When invoking the initial page of the message application, the text browser passes all session related variables, including the message properties and the UCID to the message application. If you use an external text processing engine, then the text browser also passes the text contained in the inbound SMS or email message to the message application. The text browser also passes information related to administrative variables, reporting, and outbound Web services to the Orchestration Designer runtime.

Orchestration Designer supports exit variables in message applications the same way that it supports in speech applications. In addition to the exit variables used by the speech applications, message applications also use the exitParentId variable. Unlike speech application, which considers the entire conversation in a call as one single session, a message application considers the process of receiving an inbound SMS or email message and sending one or more outbound message responses as one session. The message application processes the other inbound messages as separate sessions even if the inbound messages are correlated to the first inbound message. To bind the correlated messages together for reporting, message applications use the exitParentID variable.

The text browser captures the exit variables from the Orchestration Designer runtime and passes the exit variables to the message gateway.

The structure of the TextXML that is returned by the application server is similar to a subset of VoiceXML. Currently, only <*prompt*>, <*field*>, <*grammar*>, <*goto*>, <*submit*>, and <*catch*> elements are supported in the TextXML that is returned by the application server. The system validates the TextXML that is returned by the application server against the TextXML schema.

SMS channel message applications

Create and use an SMS channel message application to automate the process of interpreting an inbound SMS message and sending an automated outbound SMS message in response. You can also create an SMS channel message application to send proactive outbound SMS messages to customers.

To send an automated SMS message from an SMS channel message application, create an SMS file and specify the SMS message that you want to send in the SMS file. You can then use the SMS file in the SMS channel message application to send the SMS message.

You can receive an inbound SMS message only in an SMS channel message application.

You can forward an inbound SMS message from an SMS channel message application to Avaya Aura® Contact Center for agent assistance.

You can also send an automated email message from an SMS channel message application.

You can also launch an outbound voice call from an SMS channel message application by using the LaunchVXMLCall or the LaunchCCXMLCall items.

However, you cannot receive an inbound voice call or an email message in an SMS channel message application.

Email channel message applications

Create and use an email channel message application to automate the process of interpreting an inbound email message and sending an automated outbound email message in response. You can also create an email channel message application to send proactive outbound email messages to customers.

To send an automated email message from an email channel message application, create an email file and specify the email message that you want to send in the email file. You can then use the email file in the email channel message application to send the email message.

You can receive an inbound email message only in an email channel message application.

You can forward an inbound email message from an email channel message application to Avaya Aura® Contact Center for agent assistance.

You can also send an automated SMS message from an email channel message application.

You can also launch an outbound voice call from an email channel message application by using the LaunchVXMLCall or the LaunchCCXMLCall items.

However, you cannot receive an inbound voice call or an SMS message in an email channel message application.

Orchestration Designer message application resources

Many nodes and other flow items in an Orchestration Designer message application use various application resources, such as prompts, SMS files, email files, and grammars. You can create, configure, and modify these application resources by using the resource-specific wizards.

For example, before the **Announce** node is fully functional, you must:

1. Create one or more prompts by using the New Prompt wizard.
2. Assign the prompts to the **Announce** node.
3. Populate the prompts in the **Announce** node.
4. Define the next node that the message application must execute after the **Announce** node.

Tip:

Carefully plan the message application and list the application resources that are required. Create or import the application resources before building the message flow. That way, the resources are ready and available when needed.

The following is a list of the message application resources available in Orchestration Designer:

- Prompts. Use prompts to send automated SMS messages from an SMS channel message application and email messages from an email channel message application.

  Prompts are composed of one or more segments. In an SMS channel message application, these segments can consist of **SMS**, **Text**, or **Text Variable** items. In an email channel message application, these segments can consist of **Email**, **Text**, or **Text Variable** items.

  You can also use conditional statements to determine which prompt segments should be used in certain circumstances.

  For more information, see Prompts on page 198.

- Variables. Variables in Orchestration Designer can take any of several different specialized forms. For example, standard programming variable types.

  For more information, see Variables on page 226.

- Grammars. Use grammars in conjunction with the automated text processing technology. Grammars are lists of possible text contained in the inbound SMS or email messages. Grammars define which text the text processing engine can recognize and respond to.

  For more information, see Grammars on page 242.

- Languages. In Orchestration Designer, you can assign more than one language to a message flow project. You can create message applications that can process inbound SMS and email messages in different languages, provided those languages are available on your system.

  For more information, see Language and localization on page 276.

- Database operations. Use database operations to connect and interact with SQL databases. You can use database operations to write the information collected from an inbound SMS or

email message to the database. You can also retrieve information from the database and send the information in an outbound SMS or email message.

For more information, see Database operations on page 295.

- Web service operations. Web services are Internet-based applications that you can use to perform a wide variety of functions. In Orchestration Designer, you can invoke Web services by using a Web service operation file.

  For example, you can use a Web service to get stock quotes or weather forecasts for a region. You can then send this information in the form of an SMS or email message to the customers.

  For more information, see Web Services on page 320.

- Event types. Orchestration Designer includes most of the common types of event handlers as part of the options available from the Palette pane. The built-in events for message applications include error.semantic, error.badfetch, noinput, nomatch, and Internal Error. In addition, you can use the Event Handler Editor to create your own custom event types.

  For example, you can create a custom event handler that generates an exception whenever the hotel room that is requested in the inbound SMS message is already reserved. Then, you can add this event type to the SMS channel message application so that when the SMS channel message application generates this event, the SMS channel message application goes to a node that informs the customer that the requested hotel room is already reserved.

  For more information, see Events on page 314.

- SMS files. Use SMS files in a message application to send SMS messages.

  For more information, see SMS files on page 347.

- Email files. Use email files in a message application to send email messages.

  For more information, see Email files on page 351.

- AACC message files. Use AACC message files to send additional information along with the inbound SMS or email message to Avaya Aura® Contact Center for assisted care.

  For more information, see AACC message file on page 365.

Message flow project management

Creating a message flow project

Procedure

1. On the Eclipse user interface, click **File** > **New** > **Project**.
2. In the New Project wizard, double-click **Avaya OD Development**.

3. Select **Message Flow Project** and then click **Next**.
4. On the Create a Message Flow Project page, specify the name and location for the message flow project.
5. Click **Next**.
6. On the Specify Project Parameters page, specify the project parameters, and then click **Next**.
7. On the Specify Language Parameters page, set the parameters for the primary language that you want to use for the message flow project.
8. Click **Next**.
9. On the Specify Channel page, select the channel that you want to use for the message flow project.
10. Click **Finish**.

New Message Flow Project wizard: Create a Message Flow Project page field descriptions

Name	Description
Project name	Name for the message flow project. **Caution:** Do not use double-byte or extended ASCII characters or Java reserved words such as Try, Catch, and Switch when naming the project. **Note:** Project names must follow Java naming conventions. For more information, see Conventions for naming Java components on page 59.
Use default location	Check box to generate and save the project files in the default project location. Avaya recommends that you use the default location.
Location	Full path of the directory in which you want to generate and save the project files. To use a different location, clear the **Use default location** check box.

New Message Flow Project wizard: Specify Project Parameters page field descriptions

You can use the Specify Project Parameters page to specify meta information and a description for your project. The system does not use this information in the application, nor does the information appear anywhere in the output. This information is primarily for your internal use. The system saves and displays this information only as part of the project properties. However, this information can be useful. For example, if you are developing similar applications or slight variations of an application for different clients, you can use these fields to specify the client name, information about the purpose of the application, and how does the application vary.

Name	Description
Meta Information	
Vendor	The name of the vendor.
Category	A category name for the application if you want to deploy the application as a reusable module. For example, if you are building a banking application and this module handles savings account queries, you can type Savings query module. **Tip:** When you deploy the application as a reusable module, the system shows the reusable module in the form of a module item under the category name in the Palette pane of the message flow editor. If you do not specify a category name, then the system shows the reusable module in the form of a module item under Modules in the Palette pane of the message flow editor.
Description	A brief description for the project. You can provide information such as the variable values that are accepted and returned from the application.

New Message Flow Project wizard: Specify Language Parameters page field descriptions

Name	Description
Default Language Name	The language that you want to set as the default language for the message flow project. The default is **english**.
Language ID	The ID of the language that you select in the **Default Language Name** field. The default ID is **en-us**.

New Message Flow Project wizard: Specify Channel page field descriptions

Name	Description
Available Channels: • **Avaya Email Channel** • **Avaya SMS Channel**	The text channel for which you want to create the message flow project. You can select any one channel.

Importing a reusable message application module

About this task

You can import:

- Only those reusable message application modules which are created and exported using Orchestration Designer.
- The .war file of a reusable SMS channel message application only in a SMS channel message application.
- The .war file of a reusable email channel message application only in an email channel message application.

Procedure

1. In the Avaya OD Navigator view, click the message flow project in which you want to import the reusable message application module.
2. On the **File** menu, click **Import**.
3. In the Import wizard, on the Select page, double-click **Avaya OD Development**.

4. Click **Import Message Module** and then click **Next**.
5. On the Specify Import Details page, in the **Select WAR file** field, click **Browse**.
6. In the Open dialog box, navigate to the location and select the .war file of the reusable message application module that you want to import.
7. Click **Open**.
8. On the Specify Import Details page, click **Finish**.

Result

The Eclipse user interface imports the reusable message application module in the `<eclipse>/Orchestration Designer/Modules` reusable modules directory, and displays the reusable module in the form of a module item under **Modules** in the Palette pane of the message flow editor.

Chapter 6: Data applications

Orchestration Designer data application resources

Data applications do not require an incoming channel and typically perform some type of business logic. Because of this, they can be used as a submodule to any type of parent application (except call control). This allows you to write the business logic in only one place for all incoming communication. They can also operate standalone using a web service call, allowing you to write applications that are not tied to any type of incoming communication and can perform back office type of work.

For example, a call can come in through a speech port or an SMS channel. In both cases, you want to perform a series of database operations to obtain and update information about the caller. A data application allows you to write those database operations in one place as a submodule to both incoming channel types.

In a data application, you can create a flow much like you would in a speech or message application. You have a very limited subset of nodes that are available to you including and more important the data node. Any type of node that responds back to the browser such as a flow or menu node are not available.

Data applications do use many of the PDC's that are available to other application types. Through these PDC's, data applications have the ability to generate an outbound communication such as SMS, email and voice to an intended party. These PDC's use the platform web service rather than the browser to generate the communication.

When a data application is used as a submodule to a speech or message application, it does have the ability to accept inputs and return outputs back to the calling application. The data application will take on the identity of the parent application in this case and can interact with the respective browser. However, that is extent of its interaction with the browser: Start and Return and only for the purpose of communicating input and output data. No other nodes that generate output to the browser are usable in a data application.

The following is a list of the application resources available in Orchestration Designer:

- Variables: Variables in Orchestration Designer can take any of several different specialized forms. Some of the specialized forms include audio variables and text variables used for TTS,and the standard programming variable types. For more information, see Variables on page 226.
- Languages: In Orchestration Designer, you can assign more than one language to a data project. You can create applications that offer callers the option to select languages, provided

those languages are available on your system. For more information, see Language and localization on page 276.

- Database operations: You can use database operations to connect and interact with SQL databases. Using database operations, you can write the information collected from a caller to the database. You can also retrieve information from the database to make the information available to the caller. For more information, see Database Operation on page 721.
- web service operations: web services are Internet-based applications that you can use to perform a wide variety of functions. In Orchestration Designer, you can invoke web services by using a web-service operation file. For example, you can use a web service to enable callers to get stock quotes or weather forecasts for their region. Or you can use web services to enable callers to find out the availability of airline flights or hotel reservations. Almost any web service in which the information can be presented using audio files or text-to-speech can be used in an Orchestration Designer application. For more information, see web Services on page 320.
- The SMS and email are available if you enabled the Notification PDC as that will allow you to send outgoing SMS or email using the platform web service call.

Data project management

Creating a data project

About this task

Use this procedure to create the basic framework for the data application project. After you plan the application, you can use Orchestration Designer to create the application. For information about planning the application, see Plan before you build on page 54.

Use the New Data Project wizard to create the basic framework for the data application project.

Procedure

1. On the **File** menu, click **New** > **Project**.
2. In the New Project wizard, double-click **Avaya OD Development**.
3. Click **Data Project**, and then click **Next**.
4. In the New Data Project wizard, on the Create a Data Project page, specify a name and location for the project, and then click **Next**.
5. On the Specify Project Parameters page, specify the project parameters, and then click **Next**.
6. On the Specify Language Parameters page, set the parameters for the primary language to be used in your project.

   You can specify only one (primary) language for each language parameter.

7. Click **Finish**.

Result

After you click **Finish**, the system opens the call flow editor, where you can design and build your project call flow.

For more information, see Overview of Orchestration Designer project flow editor on page 129.

For information about setting the other project properties later, see Orchestration Designer project properties on page 490.

New Data Project wizard: Create a Data Project page field descriptions

You can use the **Create a Data Project** page to specify a name and location for the Orchestration Designer project.

Note:

The **Create a Data Project** page is mandatory.

Name	Description
Project Name	Name for the project. **Caution:** Do not use double-byte or extended ASCII characters or Java reserved words such as Try, Catch, and Switch when naming the project. **Note:** Project names must follow Java naming conventions. For more information, see Conventions for naming Java components on page 59.
Use default location	Check box to use the default project location for generating and saving the project files. Avaya recommends that you use the default location.
Location	Full path of the directory in which you want to generate and save the project files. To use a different location, clear the **Use default location** check box.

New Data Project wizard: Specify Project Parameters page field descriptions

You can use the **Specify Project Parameters** page to specify meta information and a description for your project. Orchestration Designer does not use this information in the application, nor does the information appear anywhere in the output. This information is primarily for your internal use.

Orchestration Designer saves and displays this information only as part of the project properties. However, this information can be useful.

For example, if you are developing similar applications or slight variations of an application for different clients, you can use these fields to specify the client name, information about the purpose of the application and how does the application vary, and so on.

You can also specify information about the category for your project on the **Specify Project Parameters** page.

Name	Description
Meta information	
Vendor	The vendor name or the name of vendor company.
Category	The category name for the application or the project module. For example, if you are building a banking application and this module handles savings account queries, you can type **Savings query module**. **Tip:** If you intend to use this project as a reusable module, what you specify in the **Category** field determines where in the Data Flow Editor palette this module appears. If you leave this field blank, the module appears in the **Modules** group. If you specify a category name, the module is grouped with other modules that have the same category name.
Description	Brief description for the project. You can provide information such as the variable values that are accepted and returned from this module, and other similar information.

New Data Project wizard: Specify Language Parameters page field descriptions

You can use the **Specify Language Parameters** page to set the parameters for the primary language to be used in your project.

Name	Description
Default Language Name	A primary language that was selected for your project.
Language ID	The locale name of the default language.

Chapter 7: Call control applications

CCXML, Orchestration Designer, and call control applications

Orchestration Designer simplifies the process of creating CCXML call control applications for use on Avaya Experience Portal systems.

This section helps you to understand the process and provides practical ideas and "*best practices*" for creating your own call control applications.

Orchestration Designer supports the creation of call control applications based on Call Control eXtensible Markup Language (CCXML), which is the standard language for managing telephony call control state-based applications. A call control CCXML application is a collection of CCXML and JSP pages that manage call control objects such as connections, dialogs, and CCXML sessions.

Orchestration Designer provides a lower level of support for CCXML as compared to VXML speech applications created in Orchestration Designer. By simple, standard definition, CCXML provides a more detail-oriented and finer-grained control in caller application development, when needed.

Orchestration Designer is based on CCXML v1.0, January 19, 2007, specification.

The purpose of this documentation is to provide direction and guidance on working with and implementing CCXML while developing your Orchestration Designer CCXML call control applications. The documentation does not contain information about CCXML itself.

You can study the sample call control applications that are available with Orchestration Designer. To evaluate, review, and test the sample call control applications in Orchestration Designer, open the **Call Control** perspective. For more information, see Opening the Call Control perspective on page 94. The **Call Control** perspective is similar to the **Speech** and **Message** perspectives in appearance, but it has been created specifically for call control projects, and optimized for text-based editing of CCXML and JSP pages, which is the crux of call control applications.

Many tabs are the same in **Call Control**, **Speech**, and **Message** perspectives, but the Snippets tab is unique to the **Call Control** perspective. For more information, see Building a call control application on page 97.

Opening the Call Control perspective

About this task

- On the Eclipse user interface, click **Window** > **Perspective** > **Open Perspective** > **Call Control**.

Call control projects

After you create the base call control project using the **New Call Control Project** wizard, the base components of the new call control project appear in the Navigator view with a red colored Orchestration Designer project icon. The call control project has the following file structure to organize the contents of the project:

- A *ccxml* folder to store the CCXML pages including the default **start.ccxml** page.
- A *connectivity* folder to store the database and Web service operation files.
- A *jsp* folder to store the JSP pages.
- A *vxml* folder to store the static VXML pages.
- Other folders and files that are similar to the folders and files in speech projects.

The preview pane, that is the CCXML editor, shows the initial content to be edited for your call control project.

Call control project management

Creating a call control project

About this task

Use this procedure to create a basic framework for the call control application project using the New Call Control Project wizard. Plan your call control application before using Orchestration Designer to create the actual application.

Call control projects can be made up of VXML, CCXML, and JSP pages.

Procedure

1. On the **File** menu, click **New** > **Project**.
2. In the New Project wizard, double-click **Avaya OD Development**.
3. Click **Call Control Project** and then click **Next**.

   The Eclipse user interface displays the New Call Control Project wizard.

4. On the **Create a Call Control Project** page, specify a name and location for the project.
5. Click **Next**.
6. On the **Specify Project Parameters** page, specify the project parameters.
7. Click **Next**.
8. On the **Select ccxml Template** page, select the ccxml template that you want to use for your call control project.

   Alternatively, you can create a JSP page for the call control project by using the JSP FIle wizard. For more information, see Creating a JSP file and selecting a JSP template on page 101.
9. **(Optional)** If you have already created a call control project, do the following:
   a. Click **Create new CCXML file** on the main toolbar in the Call Control perspective.

      The Eclipse user interface displays the New CCXML page.
   b. On the **Create a CCXML** page, select the project and then click **Next** to navigate to the Select CCXML Template page.
10. Click **Finish**.

New Call Control Project wizard: Create a Call Control Project page field descriptions

You can use the **Create a Call Control Project** page to name the call control project and specify a location where you want save the project.

Note:

The **Create a Call Control Project** page is mandatory.

Name	Description
Project name	Name for the project. **Caution:** Do not use double-byte or extended ASCII characters or Java reserved words such as Try, Catch, and Switch when naming the project. **Note:** Project names must follow Java naming conventions.
Use default location	Check box to use the default project location for generating and saving the project files.

Table continues…

Name	Description
Location	Full path of the directory in which you want to generate and save the project files. To use a different location, clear the **Use default location** check box.

New Call Control Project wizard: Specify Project Parameters page field descriptions

You can use the **Specify Project Parameters** page to specify meta information and a description for your project.

Note:

The **Specify Project Parameters** page is optional.

Orchestration Designer does not use this information in the application, nor does this information appear anywhere in the output. This information is primarily for your internal use. Orchestration Designer saves and displays this information only as part of the project properties. However, this information can be useful.

For example, if you are developing similar applications or slight variations of an application for different clients, you can use these fields to specify the client name, information about the purpose of the application and how does the application vary, and so on.

Name	Description
Project	
Mode	Unavailable in the current release of Orchestration Designer.
Type	Unavailable in the current release of Orchestration Designer.
Meta Information	
Vendor	The vendor name or the name of vendor company.
Category	A category name for the application or the project module.
Description	A brief description for the application or the project module.

New Call Control Project wizard: Select CCXML Template page field descriptions

You can use the **Select CCXML Template** page to start building and defining your call control project by using a CCXML template. This template is a CCXML page that contains initial content for a default start page.

Note:

The **Select CCXML Template** page is optional.

You can use the templates to begin with the CCXML content and JSP content, if you are using a JSP template, as well as the elements in your CCXML pages, and ultimately your call control application. The context of the template determines where the CCXML shows up.

After you define the template, you can open, review, and edit the CCXML template by using the Orchestration Designer CCXML Templates preferences page. For more information, see CCXML template management on page 544.

Name	Description
Use ccxml template	Check box to use a CCXML template for your call control project.
Templates are 'New ccxml' templates found in the ccxml templates page	Link to create, edit, or remove CCXML templates. For more information, see information, see CCXML template management on page 544.
Name / description	The CCXML template that you want to use for your call control project. The following are the options: • **Basic**: Contains basic CCXML content and structure. • **Accept call**: Contains basic CCXML content with transitions to accept a call.
Preview	Shows the preview of the CCXML template that you select.

Building a call control application

The basic process of building a call control application in Orchestration Designer involves:

- Creating, editing, and evolving CCXML templates in the CCXML editor.
- Creating, editing, and evolving JSP pages as required in your application.

CCXML editor

You can use the CCXML editor for editing the CCXML and JSP templates and pages to create the logic for the call control project.

After you create a base call control project by using the New Call Control Project wizard, the Eclipse display of the base call control project shows tabs to assist you in editing and evolving the call control application.

The type of content that you can add or edit is controlled by the ccxml.dtd that governs allowable content supported by the Avaya Experience Portal platform. The ccxml.dtd file is located in the Eclipse/plugins/com.avaya.cc.core/dtd directory.

Important:

Orchestration Designer does not support all CCXML commands that are specified in the CCXML specification. For more information about the supported CCXML command set, see the ccxml.dtd file.

CCXML editor tab descriptions

Name	Description
Problems	Shows any illegal content as a warning or an error. The tab is located in the bottom-center area of Eclipse.
Outline	Shows the CCXML code in a hierarchical tree. You can edit the content within the tab. This tab is located in the upper-right column of Eclipse.
Properties	Shows the CCXML code in a table view. You can edit the content within the tab. This tab is located in the lower-right column of Eclipse.
Design	Shows a hierarchical tree view. You can edit the content within the tab. This tab is located in the lower-left area of the main window.
Snippets	Contains clickable *drawers* of JSP or CCXML code snippets that you can add to the CCXML or JSP pages. This tab is located in the center of the lower area. For information about adding a snippet to the CCMXML code, see Adding a snippet to the CCXML code on page 99. The **Snippets** tab provides easy-to-use CCXML fragments and content to invoke database operations and Web service operations through proxies. For more information, see About using database operations and Web service operations in a call control project on page 99.
Avaya Application Simulator	You can use this tab to test your call control application in a simulated environment.

Formatting the code syntax

About this task

Use this procedure to format the code syntax. The Eclipse WebTools platform provides content aware CCXML editor which is an XML editor. In the CCXML editor, you can adjust the display format of the code.

Procedure

1. On the Eclipse user interface, click **Window** > **Preferences**.
2. On the Preferences page, in the left pane, do one of the following:
   - Click **XML** > **XML Files** > **Editor** > **Syntax Coloring**.
   - Click **Web** > **JSP Files** > **Editor** > **Syntax Coloring**.
3. In the **Syntax Coloring** pane, set the display format.
4. Click **Apply and Close**.

Adding a snippet to the CCXML code

About this task

- Do one of the following:
  - Drag the snippet to the location where you want to add the snippet on the CCXML page. The snippet is added to your code.
  - Move the cursor to the location where you want to add the snippet on the CCXML page. Click the snippet type. The snippet is added to your code.

    Depending on the snippet type, a subsequent dialog box appears in which you can specify the variable data to complete the snippet before it is added to your code.

    Because call control projects do not have variables, you can specify a variable name that you want to map. The system then creates the variable in the script code.

Using database operations and Web service operations in a call control project

In a call control project, you can use database operations and Web service operations in a similar way as in the speech applications. You can also define these operations in the same manner and by using the same wizards.

You can use database operations to connect and interact with SQL databases. You can also use database operations to collect information from a caller and write the information to the database, and retrieve information from the database and make the information available to the caller.

Web services are Internet-based queries and applications that perform a wide variety of functions. In Orchestration Designer, you can invoke Web services by using a Web service operation file.

An example Web service is to enable callers to get stock quotes or weather forecasts for their regions. Another example is to enable callers to find out the availability of airline flights or hotel reservations. In an Orchestration Designer call control application, you can use almost any Web service where information is made available by using audio files or text-to-speech.

At run time, Orchestration Designer uses Dbproxy and Wsproxy proxy classes to invoke database operations and Web service operations respectively, from within your CCXML content. You can write the code, either manually or by using call control projects, to invoke the database operations or Web service operations. To automatically invoke the proxy classes, Avaya recommends that you use the database operations and Web service operations snippets that are available on the **Snippets** tab within the CCXML editor.

Example transition for invoking a database operation

The following transition invokes a DBOP by using the DbProxy class in a **`fetch`** command. An input variable called *id* is created, assigned, and then passed in on the namelist as well as the classname of the database operation that is to be invoked.

```
<transition event="connection.connected" state="init">
    <log expr="'-- ' + event$.name +' -- [' + state +']'"/>
        <log expr="'   eventdata... \n' + objectToString(event$)"/>
        <assign name="state" expr="'invokeDB'"/>
        <var name="classname" expr="'connectivity.db.operations.getSayings'"/>
        <var name="id" expr="getRandomNumber(10)"/>
        <fetch next="'http://localhost:8080/InvokeDatabase/DbProxy'" type="'text/
ecmascript'"  namelist="id classname"/>
    </transition>
```

After the fetch is complete, run the database operation as follows:

Create a variable to obtain the output of the database operation and the results of the invocation. The **`<script>`** command invokes the script, runs the database operation, and sets the output variables as well as the dbResult. The following example then uses a *Saying* from the database to prepare a prompt for the caller.

```
<transition event="fetch.done" state="invokeDB">
<log expr="'--In Event FETCH.DONE--'"/>
        <var name="Saying" expr="''"/>
        <var name="dbResult" expr="'OKAY'"/>
        <script fetchid="event$.fetchid"/>
        <if cond="dbResult == 'OKAY'" >
            <assign name="state" expr="'response'"/>
            <log expr="'Saying is:' + Saying"/>
           <dialogprepare type="'application/voicexml+xml'" connectionid =
"in_connectionid" src="'http://localhost:8080/InvokeDatabase/vxml/Saying.vxml'"
namelist="Saying" />
       <else/>
             <log expr="'--Error in DB call:' + dbResult"/>
              <exit expr="'DB error'"/>
        </if>
    </transition>
```

For an array, you have to pass in the index:

```
<log expr="'Saying is : ' + Saying"/>
<log expr="'Saying is : ' + Saying[0]"/>
 namelist="Saying" />
 namelist="Saying[0]" />
```

Creating a JSP file and selecting a JSP template

About this task

Use this procedure to create a JSP code using base JSP file templates or additionally created JSP template files that contain reusable blocks of code.

You can use the JSP templates to begin with the JSP content and elements that are used in your CCXML code and call control application. You can use the context of the template to determine where the CCXML shows up.

Procedure

1. On the **File** menu, click **New** > **JSP File**.

   The Eclipse user interface displays the New JSP Page page.

2. On the Create a JSP page page, in the Available Projects pane, select the CCXML project for which you want to create a JSP file and use a JSP template.

3. In the **File name** field, type a name for the JSP file and then click **Next**.

   The Eclipse user interface displays the **Select a template as the initial content in the jsp page** page.

4. Select the **Use JSP Template** check box.

5. Select the JSP template that you want to use from the list of templates displayed.

   You can select any one of the JSP templates. Of these templates, two are Orchestration Designer JSP templates and three are Eclipse JSP templates. Some of the templates are:

   - **Basic template**: Orchestration Designer template that contains basic JSP/CCXML content and structure.
   - **Accept call**: Orchestration Designer template that contains basic JSP/CCXML content with transitions to accept a call.
   - **New JSP File (html)**: Eclipse JSP template with HTML markup.
   - **New JSP File (xhtml)**: Eclipse JSP template with xHTML markup.
   - **New JSP File (xhtml, xml syntax)**: Eclipse JSP template with xHTML markup and XML style syntax.
   - **New JSP File (xhtml, xml syntax, JSP 2.0)** : Eclipse JSP template with xHTML markup, XML style syntax, and JSP 2.0.

Note:

You can click the **JSP Templates** link to navigate to the **Templates** page of preferences.

6. Click **Finish**.

   The Eclipse user interface adds the code from the template file to the JSP project file within Eclipse. You can then modify the JSP file as per your requirement.

Adding and modifying a JSP template

About this task

Use this procedure to add or modify a JSP template. The JSP file editor is a text-based editor. The JSP templates page and Eclipse displays additional JSP tag code segments.

Procedure

1. On the Eclipse user interface, click **Window** > **Preferences**.

   The Eclipse user interface displays the **Preferences** page.

2. In the left pane, double-click **Web** > **JSP Files** > **Editor**.
3. Click **Templates**.
4. In the **Templates** pane, click **New** or **Edit** to add or modify the JSP template. For more information, see the *Java Development User Guide*.

Deploying a call control application

The final step in developing a call control application project is to deploy the application. How you deploy the project depends on what you intend the project to do.

Depending on what application execution environment you have created the application for, the deployment process creates a Web archive (WAR) file or an enterprise archive (EAR) file. You can then transport the WAR or EAR file to the server on which you want to execute the application.

For more information about deploying applications, see Module nodes on page 143 and Application deployment on page 416.

Chapter 8: Web applications

Web applications

A Web Channel allows Avaya Experience Portal developers to continue using the Avaya Experience Portal drag-and-drop paradigm to develop projects that interact with users on the HTML browser that exists on smart-phones and mobile devices. In the Avaya Experience Portal design environment, you can create new projects that are marked for Web Channel. New palette items in the flow editor provides the capability to create nodes that are translated to web forms and pages for collecting and displaying user information.

Similar to a speech application development, you can create a flow sequence that collects inputs from the user, performs logic on the input data and displays the data to the user.

The difference between a web application and a speech application is that, the speech application interacts with users through voice prompts while the web applications does that visually through HTML elements rendered by the web browser on a user's mobile device.

You can use the same look and feel in the development environment, which includes the flow editor, navigator view, simulation view and various other existing tools in the Eclipse environment, to create web channel projects. The web channel projects are supported by the similar resource structure as other channels, plus few additional resources for HTML5 elements such as css, js, jsp, and html files. Deployment and common runtime library files are same for projects of all channels.

The web feature in Orchestration Designer does not provide an HTML editor. The web UI of a web channel project is auto generated, and the web elements are displayed in a single-column layout. Due to its pure HTML nature, a web application developed in Orchestration Designer can be viewed by end users using any HTML5 web browser even from large screen computing devices. HTML5 applications also support customization of different look and feels. Developers can customize the applications by editing the generated resources without affecting the functionality provided by the Orchestration Designer tools.

As an Orchestration Designer project type, the web channel projects, like others, use the Orchestration Designer session and variables to pass input data from one flow node to another. A simple flow entails that the form node is used to prompt and collect input data from the end user, passes the data to the data node for processing and displays the processed data back to the end user.

You must host a web application as a managed application on Avaya Experience Portal to take advantage of the reporting and multi-channel capabilities.

Web channel is an eclipse plugin with extensions that support the following functionalities:

- Web Perspective
- New Project Wizard
- Import, upgrade, and export
- New palette items for flow editor
- Code generator to produce runtime code for the flow
- Simulate or test run the application from the design environment
- Project properties that help with customization

Web Perspective is a new Eclipse perspective that contains the navigator, simulator, editors, console and properties that enable developers to create, design, test and maintain all the web channel projects. Developers can also view and integrate other projects of different channels using **Web Perspective**.

Developers can create a new web project using the button in the toolbar or by using an option in the **New** menu to launch the New Web Project wizard. The web project created by the wizard has the structure and initial resources that set up a working scaffold for the developers. Developers can add some minimal content to the main flow and run the application from the simulator.

Flow editor and Code Generation

The main feature of Orchestration Designer is to allow developers to design an automated self-service application using the flow editor. Developers use the items provided in the palette of the flow editor to design forms that collect inputs, add logic to process the collected data, and then add elements to additional forms to display the processed data. The flow editor converts the graphically created flow of developers into a runtime code that in turn produces an interactive UI and logic at runtime.

The flow editor generates JSPs that present the UI to end users. These JSPs then render HTMLs to the browsers of end users on mobile devices. Any instance of the **Form** node, including the **Menu**, **Collect**, and **Announce** templates in the flow editor results in a JSP file in the project's `jsp` folder.

Palette Items

The flow editor supports new palette items for developing input and display elements that are generally required for web forms. For more information on new Palette items supported by web applications, see Detailed Palette option descriptions on page 671.

Orchestration Designer web application resources

Many nodes and other flow items in an Orchestration Designer web application use various application resources, such as prompts, textset files. You can create, configure, and modify these application resources by using the resource-specific wizards.

For example, before the **Announce** node is fully functional, you must:

1. Create one or more prompts by using the New Prompt wizard.
2. Assign the prompts to the **Announce** node.
3. Populate the prompts in the **Announce** node.
4. Define the next node that the web application must execute after the **Announce** node.

Tip:

Carefully plan the web application and list the application resources that are required. Create or import the application resources before building the application flow. That way, the resources are ready and available when needed.

The following is a list of the application resources available in Orchestration Designer:

- Prompts. Prompts are web elements that are used to display information to a user or prompt the user to provide a specific type of response. Prompts are composed of one or more segments. These segments can consist of text, location, picture, or a video. You can also use conditional statements to determine which prompt segments must be used in certain circumstances.

  For more information, see Prompts on page 198.

- Variables. Variables in Orchestration Designer can take any of several different specialized forms. For example, standard programming variable types.

  For more information, see Variables on page 226.

- Languages. In Orchestration Designer, you can assign more than one language to a web project. You can create web applications in different languages, provided those languages are available on your system.

  For more information, see Language and localization on page 276.

- Database operations. You can use database operations to connect and interact with SQL databases. Using database operations, you can write the information collected from a user to the database. You can also retrieve information from the database to make the information available to the user.

  For more information, see Database operations on page 295.

- Web service operations. Web services are Internet-based applications that you can use to perform a wide variety of functions. In Orchestration Designer, you can start web services by using a Web-service operation file.

  For example, you can use a web service to enable callers to get stock quotes or weather forecasts for their region. Or you can use Web services to enable callers to find out the availability of airline flights or hotel reservations. Almost any Web service in which the information can be presented using audio files or text-to-speech can be used in an Orchestration Designer application.

  For more information, see Web Services on page 320.

- Textset files. Use textset files in a web application to display text, labels to the user.

  For more information, see About textset files on page 125.
- CSS. By default, `css` directory contains the css files that come with jQuerymobile. You can change or replace them if you have your own packages or versions to use.
- JS. JS are the Javascript files of the application. These include the jQuerymobile scripts and any other Javascript files that the web application uses. You can add or remove the JS files using the project settings.
- HTML. These are the static HTML files of the web application. For example, the `camera.html` file is used to take pictures and display them on the screen.
- JSP. The JSP files contain the main code used to render the UI of the web application. The JSP files are generated based on the web application flow designed in the flow editor.

  The JSP files are responsible for the HTML output that the end users interact with, in the web browser.
- Custom directory. The `custom` directory contains JSP template files that you can use to customize the auto-generated JSP files in the `jsp` directory.

Web Project Management

Creating a web project

About this task

After you plan the application, you can use Orchestration Designer to create the application. For information about planning the application, see Plan before you build on page 54.

Use the New Web Project wizard to create the basic framework for the web application project.

Procedure

1. On the **File** menu, click **New** > **Project**.
2. In the New Project wizard, double-click **Avaya OD Development**.
3. Click **Web Project**, and then click **Next**.
4. In the New Web Project wizard, on the Create a Web Project page, specify a name and location for the project, and then click **Next**.
5. On the Specify Project Parameters page, specify the project parameters, and then click **Next**.
6. On the Specify Language Parameters page, set the parameters for the primary language to be used in your project.

   You can specify only one (primary) language for each language parameter.

7. Click **Finish**.

Result

After you click **Finish**, the system opens the web application flow editor, where you can design and build your project web application flow.

For more information, see Overview of Orchestration Designer project flow editor on page 129.

For information about setting the other project properties later, see Orchestration Designer project properties on page 490.

New Web Project wizard: Create a Web Project page field descriptions

You can use the **Create a Web Project** page to specify a name and location for the Orchestration Designer project.

Note:

The **Create a Web Project** page is mandatory.

Name	Description
Project Name	Name for the project. **Caution:** Do not use double-byte or extended ASCII characters or Java reserved words such as Try, Catch, and Switch when naming the project. **Note:** Project names must follow Java naming conventions. For more information, see Conventions for naming Java components on page 59.
Use default location	Check box to use the default project location for generating and saving the project files. Avaya recommends that you use the default location.
Location	Full path of the directory in which you want to generate and save the project files. To use a different location, clear the **Use default location** check box.

New Web Project wizard: Specify Project Parameters page field descriptions

You can use the **Specify Project Parameters** page to specify meta information and a description for your project. Orchestration Designer does not use this information in the application, nor does the information appear anywhere in the output. This information is primarily for your internal use.

Orchestration Designer saves and displays this information only as part of the project properties. However, this information can be useful.

For example, if you are developing similar applications or slight variations of an application for different clients, you can use these fields to specify the client name, information about the purpose of the application and how does the application vary, and so on.

You can also specify information about the category for your project on the **Specify Project Parameters** page.

Name	Description
Meta information	
Vendor	The vendor name or the name of vendor company.
Category	The category name for the application or the project module. For example, if you are building a banking application and this module handles savings account queries, you can type **Savings query module**. **Tip:** If you intend to use this project as a reusable module, what you specify in the **Category** field determines where in the Web Flow Editor palette this module appears. If you leave this field blank, the module appears in the **Modules** group. If you specify a category name, the module is grouped with other modules that have the same category name.
Description	Brief description for the project. You can provide information such as the variable values that are accepted and returned from this module, and other similar information.

New Web Project wizard: Specify Language Parameters page field descriptions

Name	Description
Default Language Name	The language that you want to set as the default language for the web project. The default is **english**.
Language ID	The ID of the language that you select in the **Default Language Name** field. The default ID is **en-us**.

Web application Security

You must consider security as the topmost priority when you deploy a web application that gives access to users on internet. The Orchestration Designer runtime framework and generated code which make up each Orchestration Designer application are designed to mitigate well-known threats and attacks. However, since you use Orchestration Designer primarily for application development, you must ensure you deploy and configure the application properly and that all the custom resources in the application are protected.

You can use below recommendations and standards to ensure application security.

Security Scanner

You can use a security scanner called Zed Attack Proxy (ZAP). ZAP is an open-source tool by a group named Open Web Application Security Project (OWASP). The scanner looks for security holes in the target application by automating attacks. It generates reports and recommendations to mitigate certain types of attacks. Use this application before the application is ready to be deployed. For more information on ZAP and its creator OWASP, see https://www.owasp.org and https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project.

Setting HTTP headers

The following HTTP headers and settings are considered the most useful and supported by modern web browser for mitigating well-known attacks:

- **X-Content-Type-Options**: nosniff
- **X-Frame-Options**: **DENY -or- X-Frame-Options**: SAMEORIGIN
- **X-XSS-Protection**: 1; mode=block

For more information on headers, see https://www.owasp.org/index.php/List_of_useful_HTTP_headers.

The header value **“X-Frame-Options: SAMEORIGIN”** is added by default in the JSP pages generated by Orchestration Designer. You can configure these headers in most web servers without changing the application for different types of resources. Alternatively, you can implement and deploy a filter on application server. For more information on implementing and deploying Java filter, see https://www.owasp.org/index.php/ClickjackFilter_for_Java_EE.

Web application customization

The Orchestration Designer design environment in Eclipse is sufficient in producing working HTML5–based data collecting and manipulating applications and allowing easy integration with other Avaya products. This is done by providing developers with the drag-n-drop GUI in the flow editor, wizards, and pluggable data connectors. Customers can customize these applications to make them look unique according to their business needs. Orchestration Designer has custom functions that you can integrate by doing customizations in the web application.

JSP

By using the flow editor's drag-and-drop capability, you can quickly add commonly used data widgets onto webpages and create a flow sequence that the end users can see in the web browser.

Orchestration Designer automatically converts the Announce nodes or Collect nodes, or Form nodes into JSP files and stores these JSPs in the jsp folder under the project root. These JSPs render HTMLs to the web browser at runtime.

By default, Orchestration Designer generates these JSP files based on a set of system templates. The JSP file in the jsp folder gets regenerated or overwritten whenever you edit and save the flow editor, or when you select the Generate Project menu option. Therefore, ensure you do not change these files, unless you need to change them temporarily for testing purpose.

You can use JSPs to create a dynamic layout of the HTML UI components. You can customize a JSP if you need to add an HTML-based UI widget that is not present in the flow node palette. Orchestration Designer provides a set of customizable JSP template files in the project's custom folder. When you create a new project, Orchestration Designer automatically creates a custom folder and populates the folder with four sample template files. Each of these template files correspond to a particular type of JSP page. These files form the base for creating working templates for customizing a flow node.

CSS and JavaScript

The unique capability of Orchestration Designer is to generate standard HTML5 based on the items that you use in the flow editor to design the application. In HTML-based applications, CSS and JavaScript play a key role in giving HTML5 elements a unique look and feel and making them interactive. Orchestration Designer applications are also highly interactive and have a unique look and feel. By default, Orchestration Designer leverages JQuery Mobile, one of the most popular open-source web framework in the industry, for its mobile style look-and-feel and responsiveness.

Note:

Orchestration Designer is not a web design tool and does not provide HTML, CSS, or JavaScript editor.

Orchestration Designer provides a tool to manage CSS and JavaScript files and directly make them work on the application's webpages with a few clicks. If you do not have experience with CSS or don't have any existing CSS to work with, then you can leverage the JQuery Mobile framework and its tool to create themes.

To create a theme for JQuery Mobile, see http://themeroller.jquerymobile.com.

Customizing CSS

About this task

Orchestration Designer provides a tool to manage CSS and JavaScript files and directly make them work on the application's webpages with a few clicks. If you do not have experience with CSS or don't have any existing CSS to work with, then you can simply leverage the JQuery Mobile framework and its tool to create themes.

To create a theme for JQuery Mobile, see http://themeroller.jquerymobile.com. After you create a theme, download the package, and follow the below procedure to use the theme in an Orchestration Designer application.

Procedure

1. Unzip the theme package into any directory.
2. Right-click the project in Eclipse and click **Properties**.
3. Go to the **Orchestration Designer** > **JS/CSS/Theme** tab.
4. Click the **Add File** button to add the css files from the `theme` directory of your theme package.
5. Enter the name of the theme in **Theme Name** field. Ensure the theme name matches the name of the `<theme>.css` file.
6. Select the swatch from the theme.
7. Click **OK**.

   If you add a `CSS` file using the **Add File button**, then Orchestration Designer copies the file into the `css` folder of the project root and creates a link in every webpage of the application.

   You can use the **JS/CSS/Theme** property tab to add the `CSS` or JavaScript that you create. Orchestration Designer copies the `JavaScript` files into the `js` folder and creates a link in every webpage of the application. If the file already exists, then use **Add Link** button. This adds only the link to the webpages.

   By default, the **Use JQuery Based Framework** checkbox is selected. Clear this checkbox to remove all the JQuery links from the webpages.

Next steps

On running the application, if the images are missing in the buttons, then copy the `images` directory from the `theme` package into the `images` folder of the project's `css` directory.

Customizing JSP

About this task

You can have the same look and feel across all the forms in a web application. For example, you can have header and footer on all pages of your web application matching the company's website. The header will have the company logo and footer will have the links to general information.

Follow the below steps to customize an application to have a common look-and-feel across multiple pages:

Procedure

1. Right click the project that you want to customize and click **Properties**. click **Orchestration Designer** > **JS/CSS/Theme** tab.

2. In the Properties dialog, select **Orchestration Designer** > **JS/CSS/Theme**.
3. Click the **Use custom templates instead of system templates to generate jsp files** check box.
4. Click **Ok**.

   Orchestration Designer automatically adds four new template files - `AppRoot_Sample_template.jsp`, `Form_Sample_template.jsp`, `Menu_Sample_template.jsp`, and `Return_Sample_template.jsp`, to the `custom` directory. These new files are the working template files that Orchestration Designer uses to generate `JSPs` instead of the system template files.
5. Add the HTML code in each of the four files, and save all the changes.

   Modify the `Approot_templat.jsp` file to use `Start` page as your front page. You can ignore this file if the **Show Web Content** property of the **AppRoot** node is set to “No”.

   Modify the `Return_templat.jsp` file to create a unique looking ending page. This file works for the **Return** node.

   Similarly, modify the **Menu_template.jsp** and **Form_template.jsp** for **Menu** node and **Form** node respectively.
6. Right click the project and then select **Orchestration Designer** > **Generate Project**. Alternatively, you can make changes in the web flow editor and save.

   Orchestration Designer regenerates all `JSP` files in the `jsp` folder based on the four new template files.

Customizing a single node

About this task

In a web application, there might be a scenario where you would want to customize only one node in the flow, or in case of multiple nodes, each node needs different customization. You can use the following steps to customize a single node:

Procedure

1. Right click the node that you want to customize in the flow editor, and select **Customize…** from the context menu.
2. Orchestration Designer creates a new template file with the node’s name - <nodeName>_template.jsp , in the `custom` folder, and automatically opens up the file in the `JSP` file editor.
3. Add code to the `JSP` file wherever necessary, and save the changes.
4. Right-click on the project in the Navigator and select **Orchestration Designer** > **Generate Project** from the context menu.

   The `JSP` file in the `jsp` folder of the node gets regenerated based on the template file.

5. You can continue to make changes iteratively to the template file. The `JSP` file in the `jsp` folder of the node gets regenerated whenever you select the **Generate Project** menu option, or save any changes in the flow editor.

   Note:

   If you get an error while generating runtime `JSP` files in a project with custom templates, see Resolving error on **ODWebJet** project.

Resolving error on ODWebJet project

About this task

If you get an error while generating runtime `JSP` files in a project with custom templates, then ensure there is no error in the hidden Java project named **ODWebJet**.

ODWebJet is a helper project that Orchestration Designer automatically generates for custom `JSP` generation. By default, Orchestration Designer sets a filter to hide the **ODWebJet** project from the developer.

Follow the below steps to display the **ODWebJet** project in the navigator pane:

Procedure

1. Click on the **View Menu** drop down on the right end of the navigator toolbar and select **Customize View...**.
2. Uncheck the **Hide Web Jet Project** checkbox in the Available Customizations window.
3. Click **OK**.

   If you see an error mark on the **ODWebJet** project in the navigator pane, delete the project. Orchestration Designer automatically generates a new **ODWebJet** project when you regenerate any HTML5 project with custom templates.

Hiding Back button

Procedure

1. Right-click the project in Eclipse.
2. Click **Properties**.
3. Go to the **Orchestration Designer** > **JS/CSS/Theme** tab.
4. In **Navigation Buttons** section, select **Hide Back button**.
5. Click **OK**.

Generating dynamic text for Menu options

Procedure

1. Drag and drop a **Menu** node in your flow editor.
2. Drag and drop a **Choice** item inside the **Menu** node.
3. In the**Text Variable for Choice** property, select the variable that contains the text ID of the textset.

   Note:

   Ensure that the variable that you specify in **Text Variable for Choice** property contains the value as Textset text id in the <textset id>:<text id> format.
4. In the flow, before the **Menu** node, assign the Textset text id to the variable that you specify in **Text Variable for Choice** property using logic or data operations.

Chapter 9: Workflow Integration

Orchestration Designer workflow integration

Orchestration Designer integrates seamlessly with Avaya Breeze® platform and allows an Orchestration Designer application to interact with the Avaya Engagement Designer workflows and pass the collected data in several ways. Avaya Engagement Designer workflow integration allows Orchestration Designer applications to interact with Avaya Breeze® platform and Avaya Engagement Designer workflows. Orchestration Designer supports starting a workflow and also being brought in to an existing Avaya Breeze® platform call to provide voice services. Starting a workflow is useful when an IVR application receives a call from a customer and that call triggers a longer running business process. Orchestration Designer collects the initial data and passes it to a workflow to complete the transaction. Bringing Orchestration Designer in an existing Avaya Breeze® platform call is useful to support a rich VUI environment.

Workflow PDC

About Workflow PDC

A Pluggable Data Connector(PDC), Workflow PDC, supports integrations in Orchestration Designer. The Workflow PDC supports following operations:

- Operations for context store access
- An operation to start a workflow
- An operation to send an event to a workflow
- Automatically add the *avayaContextId* variable to the application
- At runtime, automatically store the context id from either UUI or request parameter into the *avayaContextId* variable

Orchestration Designer using Context Store

Avaya Engagement Designer workflows bring Orchestration Designer into an existing customer call, or the workflow can create a new outgoing call to a customer, and then Orchestration Designer is added to that call. Data for the Orchestration Designer application is placed into a context in the context store and the ID of the context is passed in the shared UUI. Orchestration Designer retrieves data from the context store and updates and/or adds new data in the context. Once Orchestration Designer is added into the call, it interacts with the caller and then places the

return data in the context. When the Orchestration Designer application is finished and drops from the call, the Avaya Engagement Designer workflow retrieves the return data from the context in the context store.

Orchestration Designer applications with the Workflow PDC, automatically copy the data passed in the shared UUI as application data into the *avayaContextId* variable. The workflow PDC assumes that the shared UUI application data is a context ID.

A sample Orchestration Designer application that retrieves the context passed when ID is added to a call, along with instructions to create a workflow that will add an Orchestration Designer application into the call, is available.

In previous release, the PDC supported only **Simple Variable**, or a **Variable Field** when interacting with the Context Store. In current release, a new set of operations, **Set Context From Variable**, and **Get Variable From Context** support reading and writing both, simple, and complex variables with and without collections. Assuming variables of simple, simplecollection, complex, complexcollection, complex variable, and complex collection have 2 fields, field1 and field2, then, when you use the **Set Context From Variable** operation, the data is serialized into JSON as shown in the following table:

Variable	Data serialization in JSON
Simple Variable no collection	{"simple":{"simple":"item1"}}
Simple Variable with collection	{"simplecollection":[{"simplecollection":"item1"}, {"simplecollection":"item2"}, {"simplecollection":"item3"}]}
Complex Variable no collection	{"complex":{"field1":"value1","field2":"value2"}}
Complex Variable with collection	{"complexcollection": [{"field1":"value1","field2":"value2"}, {"field1":"value3","field2":"value4"}, {"field1":"value5","field2":"value6"}]}

The same formats are applicable when using the **Get Variable From Context** operation.

All Context Store operations support a TouchPoint variable, or constant. TouchPoints associate a text tag with a particular context store operation that is stored in a context audit trail, a feature in the Context Store. The **Create Context** operation supports *Persist* parameter that takes a true or false value. When *Persist* is true, and the Context Store is configured appropriately, then the context data and audit trail data are persisted to an external database.

While creating a context, if a context id is not provided, then the **Create Context** operation creates and returns a context ID.

SMS and Email Orchestration Designer

In order to provide Avaya Breeze® platform with two way SMS and Email capabilities, an Avaya Engagement Designer workflow initiates the sending of an SMS or Email message using Orchestration Designer. Orchestration Designer manages the replies received for these messages. An Avaya Engagement Designer flow needs to supply the application name, from address and context id when invoking an Orchestration Designer SMS or Email application. The receiving Orchestration Designer application retrieves input data, if any, from the context and also

places the result in the context. The context ID is passed in the query string to the Orchestration Designer application and automatically stored into the *avayaContextId* application variable. A sample Orchestration Designer application that retrieves the context passed is available along with instructions to create a workflow that sends an SMS or email message and receives a reply from Orchestration Designer.

The Experience Portal Dynamic tasks allow the Avaya Engagement Designer workflow access to the SMS and Email services. The following tasks are added to the Avaya Engagement Designer Palette under Experience Portal menu group to support various integrations:

- Launch SMS Service: Task to start an Orchestration Designer SMS application.
- Launch Email Service: Task to start an Orchestration Designer Email application.

Note:

Orchestration Designer does not support the integration of Orchestration Designer SMS and Email applications with Avaya Engagement Designer. So, the dynamic tasks to launch the SMS and Email applications are disabled in the current release.

Orchestration Designer workflow configuration

Configuration on Experience Portal

The following Experience Portal Configuration tells Orchestration Designer applications where the Context Store and Avaya Breeze® platform are located:

- **Platform Host Address**: The Avaya Breeze® platform SIP entity address.
- **Context Store Address**: Context Store cluster IP address on Avaya Breeze® platform.
- **Client Timeout**: The maximum time an Orchestration Designer application should wait for Avaya Breeze® platform or Context Store to respond to its request. Enter a number between 15 and 300. The default value is 60 seconds.

Configuration on Avaya Breeze® platform

The connection information will be configured on the Avaya Breeze® platform, including the primary address for the Experience Portal, a secondary address for the Experience Portal, and the User ID and password for the Experience Portal services. This identifies the default zone, or any zone configured on Experience Portal. Only one zone will be configured for SMS or Email access. The Avaya Engagement Designer workflow needs to provide the name of the Orchestration Designer application to launch, and a string to identify the context for this request.

The Avaya Engagement Designer tasks need the location of the Context Store. To provide this location, set the Primary Cluster IP address of the Context Store. For details of setting the location of Context Store, refer the online documentation of Experience Portal.

Workflow Management

Starting a Workflow

Creating an event to start the workflow

About this task

Orchestration Designer starts a workflow by sending an event to Avaya Breeze® platform. On receiving the event, Avaya Breeze® platform searches for workflows that have that event as the “start” event and creates a new instance of that workflow.

The start event must be defined on the Avaya Breeze® platform platform as a custom event and then added as the start event for a workflow. To define the event that will start the workflow:

Procedure

1. Log in to System Manager.
2. Click **Elements** > **Avaya Breeze™**.
3. In the left pane, click **ConfigurationEvent Catalog**.
4. On the Event Catalog Configuration page, click **New**.
5. On the Event Catalog Editor page, enter information in the fields as seen in the table below and click **Commit**.

Field	Value
Family	ODEvent
Family Display Name	ODEvent
Type	STARTWFEVENTONLY
Type Display Name	An arbitrary string
Version	1.0
Schema Name	STARTWFEVENTONLY
Schema Type	JSON
Schema	{"title" : "STARTWFEVENTONLY", "type" : "object", "properties" : { "type" : {"type" : "string"}, "vehicle" : {"type" : "string"}, "action" : {"type" : "string"}, "phone" : {"type" : "string"}, "param5" : {"type" : "string"}, "param6" : {"type" : "string"}, "param7" : {"type" : "string"}, "param8" : {"type" : "string"}}}

The event family is fixed as “ODEvent”. You can use different event types to start different workflow applications. When creating an event, by default Orchestration Designer assumes the names param1 through param8. You can change the names of the parameters to more meaningful values, but the number of parameters should be 8.

Setting up the Client keystore

About this task

To send an event to Avaya Breeze® platform, proper certificates need to be in place. The steps to create a key store and point your OD environment to it are:

Procedure

1. Configure the Client Certificate Challenge:
    a. Log into the System Manager.
    b. Click **Elements** > **Avaya Breeze™**.
    c. In the left pane, click **Configuration** > **HTTP Security**.
    d. Select **Client Certificate Challenge Enabled** checkbox.
    e. Click **Commit**.
2. Download Breeze Trusted Certificate from the System Manager:
    a. Log into the System Manager.
    b. Click **Services** > **Inventory**.
    c. In left pane, click **Manage Elements**.
    d. Select a checkbox for an element with type **Avaya Breeze**.
    e. In **More Actions** dropdown, select **Configure Trusted Certificates**.
    f. On Trusted Certificates page, select checkbox for **Store Type**: SECURITY_MODULE_HTTP and **Subject Name** : CN=default, OU=MGMT, O=Avaya.
    g. Click **Export**.
    h. Save as “trust-cert.pem”.
3. Import Trusted Certificate into a keystore:
    a. Login into Orchestration Designer runtimeconfig web application.
    b. Select **Certificates**.
    c. Click **Add**.
    d. Enter the name for the certificate as “breeze-trusted-cert”.
    e. Click **Choose File** and locate the “trust-cert.pem” exported from step 2.
    f. Click **Continue**.
    g. Click **Save**.

Importing workflow into Avaya Engagement Designer

Procedure

1. Login into the System Manager.
2. Click **Elements** > **Avaya Breeze™**.
3. In the left pane, click on **Cluster Administration**.
4. Select your cluster. In **Service URL** dropdown, select “Designer Access URL”.

   This launches Avaya Engagement Designer.
5. In Avaya Engagement Designer, click **Import**.
6. Select the workflow file and click **Import**.
7. Click **Save As**.
8. Enter the file name as “StartClaimEventOnly” and click **Save**.
9. Click **Deploy**.
10. Accept the default values and click **OK**.

Orchestration Designer application

Orchestration Designer application

In the Orchestration Designer application, when the workflow Pluggable Data Connector(PDC) is added, you have an option to disable mutual authentication by selecting the **Skip mutual authentication for Breeze Platform and Context Store** checkbox. This option simplifies the initial application testing. It is not recommended to use this option.

In the LaunchWokflow tab, note the following points in the **Start Workflow** item:

1. **Event Type** must match the event defined while creating an event to start the workflow.
2. **Workflow Name** must match the name of the deployed workflow while importing the workflow in Avaya Engagement Designer.
3. **Parameter names** must match the names given while creating an event to start the workflow.

When you are creating new applications, you can change the event type to different values to start different applications. Also, by default the Orchestration Designer uses param1 through param8 as the parameter names. These can be changed to more meaningful names, though the number of parameters is fixed to 8. Lastly, the values used in the **Start Workflow** item must match what is deployed in an event and workflow on Avaya Engagement Designer. You must deploy the Orchestration Designer application to an application server and configure Experience Portal.

HTML application URL generator PDC

An outbound application may need to send an email or SMS that contains a URL for the message receiver. The message receiver can click this URL to launch the HTML application. You can use the HTML application URL generator PDC to generate a URL to an Orchestration Designer HTML application that is configured on Experience Portal. The PDC provides a palette item in the **Data** node editor, which you can drag and drop into the **Data** node logic tree. This allows the node to generate the URL dynamically when it is executed. You must configure the **Generate URL** item with parameters such as **Application Name**. **Application Name** is the name of the application that you configured on Experience Portal, and not the project name.

If a conversation has been started, then the HTML URL generator PDC adds the conversation ID to the query string. The HTML URL generator PDC uses the configured **HTML Redirector** address in the URL. It also appends the session:ucid to the query string.

Note:

Configure Experience Portal Manager only from a single zone while using the HTML Redirector's utility. For multiple zones, deploy the Redirector for each zone separately, where each setup of the Redirector consists of Experience Portal Manager from the corresponding zone only.

Redirector

A Redirector is a web archive file (war) that must be deployed on a Tomcat server. It is based on Java and requires a JRE and Tomcat servlet container. The main purpose of the Redirector is to hide Experience Portal information. This information is used for initializing the Orchestration Designer application session when the user sends the first request from the browser.

You must deploy the Redirector component as an application only on the Tomcat application server. This component starts the HTML5 applications that are configured as managed applications on Experience Portal Manager (EPM). A sample Redirector application is provided as a reference on how the component must be implemented. You can export and use this sample application from Orchestration Designer. For more information about application installation, see the Redirector setup document that you get with the `redirector.war` package.

Chapter 10: Oceanalytics

Oceanalytics Realtime events

Using Orchestration Designer (OD), an application developer can send real time events to Avaya Analytics™. Such events are referred to as custom events. The events are TextMessages with a JSON string payload. The payload for custom events are either a variable or a set of attribute — value pairs. The variable can be simple, complex, or a collection.

To send events, you must enable the **Publish Realtime Oceanalytics Event** PDC for an application. The variable *avayaSendRTE* is added to the application when the **Publish Realtime Oceanalytics Event** PDC is enabled for that application. This variable is a Configurable Application Variable (CAV).

The OD application does not have control over the contents of the automatic events other than to alter the session exit values. Once you enable the **Publish Realtime Oceanalytics Event** PDC, the “START” and “RESULT” events are always sent for all application types.

In addition to adding the **Publish Realtime Oceanalytics Event** PDC to an application, you must set the global CAV *AvayaBreezeBrokerList* to the address of the Avaya Breeze® platform Reliable Event Framework (REF) broker. The *AvayaBreezeBrokerList* can be a single address, or multiple addresses separated by a vertical bar (|). For example, “`ssl://148.147.10.221:61617`”, or “`ssl://148.147.10.221:61617| ssl://148.147.10.222:61617`”. The OD runtime uses the first broker that works. In case of a communication failure, OD uses a different broker if multiple addresses are supplied. OD discards the event if it cannot contact any broker. OD does not make any attempt to save and replay the real time events, because by definition, these events are transient.

Sample events

- `OD_EVENT_START with body :`
  `{"od_apptype":"speech","od_appname":"TestOceanalytics","od_vpappname":"TestOceanalytics","od_language":"english",`
  `"od_ucid":"undefined","to":"5551212","from":"5551234"}`
- `{"od_apptype":"speech","od_appname":"TestOceanalytics","od_vpappname":"TestOceanalytics","od_language":"english",`
  `"od_ucid":"undefined","to":"5551212","from":"5551234","exitCustomerId":"","exitInfo1":"","exitInfo2":"","exitReason":"","exitTopic":"","exitPreferredPath":"1","exitParentId":""}`

A typical OD event includes:

- sourceId: The IP address of the sending application server.
- sourceVersion: The current version of the OD runtime common.
- snapinVersion: The version of the OD runtime common using which the plugin was built.
- sequenceNumber: The sequenceNumber is maintained for each event type, for each application server and increases by 1 for each event sent. In case of a connection issue, the

sequence numbers might have a gap or might be reset to 0. The receiver can use the sequence number to detect connection issues.

- eventTimeStamp: The system time when the event is sent. The eventTimeStamp is in milliseconds.
- The fields eventFamily, eventAction, eventCategory, eventVersion, sourceType, and snapinID having fixed values.
- <JSON Payload>: OD adds the following values:
  - od_apptype: Type of the OD application such as email, SMS, speech, and so on.
  - od_appname: The name of the OD application.
  - od_vpappname: The name for the OD application, configured on the Experience Portal Manager.
  - od_language: Language selected for the OD application.
  - od_ucid – The ucid if present.

Built-in events

For applications, Orchestration Designer (OD) sends the “OD_EVENT_START” event at the start of a call. In addition to the standard fields, the “to” and “from” values are sent in the event payload. The to and from values vary depending on the channel. For example, on the speech channel, the to and from values are DNIS and ANI, whereas on SMS, the values are the phone numbers of the SMS.

When the application ends, OD sends the “OD_EVENT_RESULT” event. In addition to the standard fields, the *to*, *from*, *exitCustomerId*, *exitInfo1*, *exitinfo2*, *exitParentId*, *exitPreferredPath*, *exitReason*, *exitTopic* values of the session variable are sent in the event payload.

To send events, you must enable the **Publish Realtime Oceanalytics Event** PDC. When the PDC is added to the application, a Configurable Application Variable (CAV), *avayaSendRTE* is automatically created with the default value of “true”. You can enable or disable the sending of the automatic events through an application by toggling this field between true and false. Events are not sent if there is any value in *avayaSendRTE* other than “true”.

Note:

The *avayaSendRTE* variable only affects the built-in events. Custom events are always sent.

User events

An application developer can send a custom event and give a name to that event. All events of that name are published to the same topic. For the payload of the event, the developer can select a variable and let Orchestration Designer (OD) serialize the variable into JSON. The selected variable can be a complex variable, a simple variable, or a collection.

Example

If the developer selects a variable *myeventdata*, then the following payload is generated:

```
   "aaodEvent" : {
        "myeventdata":{
                    "status":"ACTIVE",
                    "item":"Insurance Claim IVR"
                       },
                  <standard fields:>
               }


Alternatively, the developer can choose to specify attribute value pairs.   Both the
attribute name and value can be either constants or a variable.   Using the attribute/
value event type the following payload might be generated:

    "aaodEvent" : {"attributeone":"valueone",
                    "attributetwo":"valuetwo",
                 <standard fields>
                   }
```

Alternatively, the developer can specify attribute - value pairs. Both the attribute name and the value can be either a constant or a variable. Using the attribute - value event type, the following payload is generated:

```
    "aaodEvent" : {"attributeone":"valueone",
                    "attributetwo":"valuetwo",
                 <standard fields>
                   }
```

Chapter 11: Textsets

Textset files

To complete the configuration of an input field in a **Collect** node or a choice in a **Menu** node, you must choose a label or a text id from a textset resource file. The textset file is designed to contain all the language specific text information such as names of the input fields and names of the menu options that the users are able see in the web browser.

Textset file management

Creating a Textset file

Procedure

1. On the **File** menu, click **New** > **Other**.
2. In the New wizard, double-click **Avaya OD Development**.
3. Click **Textset File**, and then click **Next**.
4. In the New Textset wizard, on the Create a Textset page, specify the project name, and a name for the textset file.
5. Click **Finish**.

Result

The system displays the textset editor of the default language. If multiple languages are configured for the project, a textset file gets automatically created for each language in the `language` folder.

Next steps

You can add the labels or text that you want to display on the screen, in the textset editor.

Creating a Textset file from an Input item

Procedure

1. Drag and drop any Input item such as **Text Input** in the flow editor.
2. Right click the input item.
3. In the context menu, select the **Create New Textset File** option.
4. In the New Textset wizard, select the project name from **Available Projects**.
5. Specify the Textset file name.
6. Click **Finish**.

Result

A new textset file, <`textsetfilename.textset`> gets created in the `textsets` folder of your project.

New Textset wizard: Create a Textset page field descriptions

Name	Description
Available Projects	The web project for which you want to create a textset file. You must select a project.
Textset file name	A unique and relevant name for the textset file.

Textset File Editor field descriptions

Name	Description
Texts	List of all labels or textset entries that you created in the textset file.
Add New	Use this button to add a new label or a textset entry.
Delete	Use this button to delete a label or a textset entry.
Textset Text Data	
Name	A unique name for the textset entry.
Value	The actual text to be displayed on the screen.
Comments	Any specific comments or information about the textset entry.
Set Name	Use this button to edit the name of a textset entry selected from the Texts section.

Note:

If a user configures multiple languages, then the new label name automatically populates in all textset files of different languages. User must fill in the language specific value manually in each textset file.

Editing a Textset file

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the web project directory.
2. Double-click ***<language name>*** > **textsets**.
3. Right-click the .textset file that you want to edit, and then click **Open With** > **Textset Editor**.
4. In the textset editor, modify the text you want to display.

Editing the Textset file from an Input item

Procedure

1. Drag and drop any Input item such as **Text Input** in the flow editor.
2. In the context menu, select the **Edit Textset:<textset filename>** option.
3. In the textset editor, modify the text in the **Value** field to change the label that you want to display.
4. Save your changes by pressing `CTRL+S`.

Creating a new Textset item from an Input item

About this task

Use this procedure to create a new textset item from any Input item, such as **Text Input** and **Location Input**.

Procedure

1. Drag and drop any Input item such as **Text Input** in the flow editor.
2. In the Properties window, click  in the **Value** column of the **Label ID** property.

   If you are using a **Location Input** item, then click  in the **Value** column of the **Text in textset** property.
3. In the Select an Id from a Textset window, right click on a textset or a textset ID.

4. Click **Add Text**.
5. Enter the text name.
6. Enter the text value.
7. Click **OK**.
8. In the Select an Id from a Textset window, click **OK**.

Chapter 12: Project flow editor

Overview of Orchestration Designer project flow editor

The Orchestration Designer project flow editor provides the primary workspace and options to create flows in the application projects. The project flow editor provides the means for setting global application properties and editing application flows in a node editor.

You can do most of the basic work for creating an application in a project flow editor. The project flow editor contains the workspace where you can add nodes and connect these nodes to create a project flow. The project flow editor also provides a palette of nodes and options to build simple project flows.

You can use the nodes in a project flow editor as the basic building blocks for creating Orchestration Designer applications. Each node represents a piece of code and a functionality. The Palette pane of the project flow editor provide access to the nodes and action options that you can use to build project flows. For more information, see Components of Orchestration Designer project flow editor on page 130.

Note:

The primary purpose of the project flow editor is to create simple project flow logic. For more complex flows, such as looping or calculations, you must write a Java code that supports the complex functions.

Note:

Be sure that you are familiar with Eclipse concepts and terminology. For more information, see Accessing the Concepts section of the Eclipse Workbench User Guide on page 37.

In the default perspective layout for Orchestration Designer, the project flow editor occupies the Editor view space. For more information, see Editor view on page 42.

Important:

By default, the project flow editor maintains a backup of the flow document, in the event that the main flow is corrupted. Prior to saving the contents of the main flow, the old main flow file is copied to a backup folder using a round-robin file numbering system to maintain multiple backups. The default is 2 and the maximum is 10. The backup files are numbered, for example, `main.flow.1`, `mainflow.2`, but after the maximum number of backup is reached, the numbering sequence starts over. The Call Flow Editor Preferences panel controls the call flow editor and message flow editor backup configurations, that is, enabling or disabling the

feature, and setting the number of backup files maintained. For more information, see Backing up a call flow on page 554.

Components of Orchestration Designer project flow editor

In the default perspective layout for Orchestration Designer, the Orchestration Designer project flow editor occupies the Editor view space. For more information, see Editor view on page 42.

The following table lists the components of a project flow editor:

Name	Description
Workspace	Located in the upper-right area, workspace is the largest area within a project flow editor. Before you build an Orchestration Designer application, the workspace is blank. You can add and connect nodes in the workspace. You can also print a project flow by using the **File** > **Print**.
<project name> [main.flow] tab	Located at the top of the workspace, shows the main project flow.

Table continues…

Name	Description
Palette	Located in the left of the workspace, contains the nodes that you can drag into the workspace to create a project flow. The nodes are grouped as follows: • **Templates** node: Contains predefined system nodes that you can use in an application. • **Application Items**: Contains nodes that you can use for creating custom nodes that are to be used in an application. In addition, the following functional options are also available within the **Palette** pane: • **Select**: Selects one or more items in the workspace. For more information, see Selecting objects in a workspace on page 147. • **Connection**: Connects the nodes to create the project flow. For more information, see Connecting nodes in a workspace on page 147. • **Label**: Adds notes and comments to the workspace. For more information, see Adding comments to a workspace on page 148. • **Bookmark**: Creates markers to make it easier to locate specific nodes in larger, more complex applications. For more information, see Adding a bookmark to an Orchestration Designer project flow on page 149. When you open a node in a node editor, the **Palette** pane shows the options that are available for that node. For more information, see Node management on page 151.
Node editor tabs	Located at the bottom of the workspace, shows the node editors that are open. You can use these tabs to quickly switch between node editors and the main flow.
Application Flow tab	Located at the bottom of the workspace, shows the main call flow or message flow workspace. For more information, see Node management on page 151.

Accessing the project flow editor

Procedure

Perform one of the following actions:

- Create a new project. For more information, see Creating a speech project on page 64, Creating a message flow project on page 84, and Creating a web project on page 106.
- When multiple editors are open, in the Editor view, click the <***project name***> [main.flow] tab.
- In the Navigator view or in the Avaya OD Navigator view, expand the <***project name***> folder, expand the **flow** folder, and then double-click **main.flow**.

  The system displays the respective flow editor for the project.

AppRoot node

When you create a speech, or a message flow, or a web flow project, Orchestration Designer automatically creates the **AppRoot** node as the initial node for the call flow, or message flow, or web flow. The **AppRoot** node is the starting point for all call flows and message flows, and cannot be renamed or deleted.

You can also set global capabilities for the speech, or message flow, or web flow project from the **AppRoot** node. That is, you can use the **AppRoot** node to set event handlers that the system recognizes and responds to, from anywhere in the call flow, or message flow, or web flow. For more information, see Setting global application properties on page 144.

The following are the examples of the types of parameters and capabilities that you can set in the **AppRoot** node of a speech application:

- You want the system to perform some type of *cleanup* at the end of each call, irrespective of where, when, or how the call terminates. You can use a call disconnect event handler in the **AppRoot** node. The call disconnect event handler redirects the application to a form that is set up to record data about the call. This data can include how the call was terminated, whether by the caller or by the system, and where the caller was in the call flow when the call was terminated.
- You can determine how the system functions whenever a caller does not respond as expected, either by not responding at all or by responding with something that does not match what is expected. You can also assign the system a generic prompt to play to callers when an error situation occurs.
- You want to provide a set of caller commands to which the system can respond regardless of where the caller is in the call flow. For instance, you want to provide access to the general help information from anywhere or allow the caller to cancel transactions.
- You can set a variety of properties, such as time-outs and speech recognition confidence settings, that are then applied system wide.

The following are the examples of the types of parameters and capabilities that you can set in the **AppRoot** node of a message application:

- You can determine how the system functions whenever a customer does not respond as expected, either by not responding at all or by responding with something that does not match what is expected. You can also assign the system a generic prompt that sends a message to the customers when an error situation occurs.
- You can configure the **AppRoot** to send the generated exceptions to Avaya Experience Portal Media Processing Platform (MPP). The default event handlers on MPP can then handle the exceptions.
- You can capture ECMA data from the session and store it in the session variables.

The following are the examples of the types of parameters and capabilities that you can set in the **AppRoot** node of a web application:

- You want the system to perform some type of cleanup at the end of each session. A web application supports only On Session Timeout event handler, which is responsible for clean up and reporting, in the **AppRoot** node. Only the **Data** nodes led off by the Next item of the event handler are executed. In the subsequent Data nodes, you can add logic items and report items to complete the cleanup and/or reporting before the Orchestration Designer session is finally destroyed.

  The On Session Timeout event handler handles the HTTP session out event that an application server triggers, after the end user closes the web browser, or leaves the web browser idle in the middle of an interaction with the application.

  Note:

  Every application server supports the time-out setting for ending an HTTP session which has been idle for a specified amount of time. The administrator determines this time-out value based on the business need.

- You can also use the **AppRoot** of a web application as the welcome page of the application. By default, **AppRoot** is not used as the first web page of the application. To turn the feature on, set the **Show Web Content** property of the **AppRoot** node to true. To make it a welcome page, set the **Graphic**, **Graphic Style**, **Title Message** and **Message Style** properties which show a graphic and a title message underneath on the welcome page.

You can set the following properties for the **AppRoot** of a web application:

- **Comments**: Type any comment that you want to add as a description of the purpose or content of the node. Orchestration Designer uses the text in the **Comments** property for the popup hint that appears when you move the pointer over the node icon in the workspace. If you leave the **Comments** property blank, Orchestration Designer uses the name of the node for the pop-up hint.
- **Graphic**: Enter the path and name of the graphic file that you want to display on top of the web page. For example, `css/images/avaya-logo.png`.
- **Graphic URI Variable**: Select the variable that contains the path of the graphic file. Ensure that you copy the image files in the directory that is accessible using url or uri, and that the value of the variable assigned to the **Graphic URI Variable** property matches the url or uri.

- **Graphic URI Variable Field**: Select the field of the variable that contains the graphic file path.
- **Graphic Style**: Enter the CSS style property or properties for the image. For example, to make the image as wide as the browser window, you can enter the property `width:100%`.
- **Title Message**: Enter the message that you want to display on the web page, below the graphic.
- **Message Style**: Enter the CSS style property or properties for the way you want to display the message. For example, to make the message bold, you can enter the property `fontweight: bold`.
- **Show Web Content**: Set this property to “Yes” if you want to use the **AppRoot** node as the first page of the application.

The following are the examples of the types of parameters and capabilities that you can set in the **AppRoot** node of a data application:

- You can capture ECMA data from the session and store it in the session variables.

The options and parameters that can be set in the **AppRoot** node are available in the Palette pane, and are classified in the following two basic types:

- Application Items. Includes the parameters and options, other than the event handlers, that control how an application responds on a global level. For more information, see AppRoot node Application items on page 134.
- Event Handlers. Includes the options that determine how the application handles any exceptions that are not defined within a specific node or field. For more information, see AppRoot node event handlers on page 137.

AppRoot node Application items

Application items of the **AppRoot** node include the parameters and options, other than the event handlers, that control how an application responds on a global level. In many cases, these items support the use of the event handlers or perform actions that event handlers are not designed to handle.

The following table lists the **Application Items** of the **AppRoot** node:

Name	Description
Input Parameter on page 745	Creates a placeholder for input expected from another module. Use this item when you are creating the project to be used as a module. When you drag this item into the workspace, Orchestration Designer automatically creates a variable with the same name.
Capture Expression on page 690	Captures the data through an ECMA script into the Orchestration Designer variables.

Table continues…

Name	Description
Prompt on page 797	In a speech application, the **Prompt** item indicates what prompt to play when the conditions of the event to which the prompt is attached are met. In a message application, the **Prompt** item indicates what prompt the system must execute when the conditions of the event to which the prompt is attached are met. To use the **Prompt** item, you must attach the **Prompt** item to another item, specifically, any of the AppRoot node event handlers on page 137 except the **On Disconnect** item in a speech application. **Prompt** items cannot stand alone in the **AppRoot** node as they can in other nodes. After you attach the **Prompt** item to another item, you must assign a prompt to the **Prompt** item in the Properties view. For more information, see Prompts on page 198.
Grammar on page 737	In the **AppRoot** node, the **Grammar** item works only in conjunction with the **Link** item. You can use the **Grammar** item to communicate with the **Link** item about which ASR or DTMF grammar to use. **Note:** The **Grammar** item is available in the **AppRoot** node of speech applications only. For more information, see Grammars on page 242.
Link on page 762	Informs the speech application which node to go to when the system recognizes a specific spoken or DTMF response from the caller. To use the **Link** item, you must specify either a DTMF or ASR response or both to be recognized. To use the ASR option, you must attach a **Grammar** item to the **Link** item. To use the DTMF option, specify the required DTMF combination of key presses in the **Link** item or attach a DTMF grammar item. **Note:** The **Link** item is available in the **AppRoot** node of speech applications only.

Table continues…

Name	Description
Goto on page 736	Informs the application which node to go to when the conditions of the event to which the **Goto** item is attached are met. To use the **Goto** item, you must attach it to another item, specifically, one of the AppRoot node event handlers on page 137 except the **On Disconnect** item in a speech application. You cannot attach a **Goto** item to an event handler that has a **Throw** item already attached.
Throw on page 844	Informs the application which event to throw when the conditions of the event to which the **Throw** item is attached are met. To use the **Throw** item, you must attach it to another item, specifically, one of the AppRoot node event handlers on page 137 except the **On Disconnect** item in a speech application. You cannot attach a **Throw** item to an event handler that has a **Goto** item already attached.
Property on page 799	Sets properties for ASR, DTMF, and transitional audio prompts at a global level. The following are examples of the types of properties that you can set: • Recognition timeouts • Confidence and sensitivity levels • Terminating key presses **Note:** The **Property** item is available in the **AppRoot** node of speech applications only.
External Property item on page 729	Sets custom properties for ASR. These custom properties are required in some cases. For example, in conjunction with some ASR engines such as some of the Nuance ASR software. For more information, refer to the documentation of your ASR software. **Note:** Although the **External Property** item is available in the **AppRoot** node of both speech and message applications, the current release of Orchestration Designer does not support the use of the **External Property** item in message applications.

Table continues…

Name	Description
Flush Prompts on page 734	Causes queued output to be played before transitioning to the next node in a speech application. When executed, the **Flush Prompts** item directs the voice browser to play all queued output immediately. **Note:** If you use the **Flush Prompts** item in an **AppRoot** node, the system plays the prompts that are configured in the node that is currently executed, and then fetches the next node. Avaya recommends that you use the **Flush Prompts** item on individual nodes, where you want the system to fetch and play prompts, rather than applying the **Flush Prompts** item globally in the application. **Note:** The **Flush Prompts** item is available in the **AppRoot** node of speech applications only.

AppRoot node event handlers

Event handlers of the **AppRoot** node include options that determine how the application handles exceptions that are not defined within a specific node or field.

The **AppRoot** node contains the following types of event handlers:

Name	Description
Catch on page 693	Determines how an event that meets specific criteria is to be handled when the event is thrown. For this event handler to function, you must assign to it at least one **Prompt**, **Goto**, or **Throw** item.
Exit on page 727	Exits the application or returns the events from a module to the calling application. If there are more serious errors, use the **Exit** event handler.
No Input on page 769	Determines what the application does whenever the caller or the customer fails to respond. For the **No Input** event handler to function, you must assign at least one **Prompt**, **Goto**, or **Throw** item to the **No Input** event handler.

Table continues…

Name	Description
No Match on page 770	Determines what the application does whenever the system cannot match the response from a caller or a customer with any expected response. For the **No Match** event handler to function, you must assign at least one **Prompt**, **Goto**, or **Throw** item to the **No Match** event handler.
On Disconnect on page 775	Determines how the speech application responds when the caller hangs up, no matter where in the call flow the disconnection occurs. **Note:** The **On Disconnect** event handler is available in the **AppRoot** node of speech applications only.
Return Event on page 820	Returns an event to the calling application. You can use this event handler to propagate an event caught in a module to the calling application, or to throw a different event back to the calling application.
Transfer on page 848	Special transfer that you can add as a child of an event handler (Catch) and transfer the caller to an extension, that is, to an agent, in the event that the speech application experiences unrecoverable errors. For example, the application server stops responding. The **Transfer** item generates a Blind Transfer in a destination that is hard coded or comes from a variable. **Note:** The **Transfer** event handler is available in the **AppRoot** node of speech applications only.

For more information, see AppRoot node Application items on page 134.

Nodes

Each node is represented by an icon and label in the Palette pane. When you drag the node into the workspace, the node represents a functional piece of code that performs some action.

For example,

- In a speech application, you can use nodes to:
  - Play an announcement to the caller.

  - Play a prompt to the caller and collect input from the caller. The caller can respond by either speaking or by pressing the touchtone keys on the telephone.
  - Present a menu to the caller. The caller can select from the options on the menu, and respond either by speaking or by pressing the touchtone keys on the telephone.
  - Perform a data transaction that sends the data collected from a caller to a customer database. The data transaction then retrieves the information from that database to present to the caller.
  - Perform a Web service operation to connect to a Web service, collect the information that is requested by the caller, and then present the information to caller.
  - Send SMS or email messages to the customers.
- In a message application, you can use nodes to:
  - Send SMS or email messages to the customers.
  - Collect user input from the inbound messages.
  - Perform a data transaction that sends the data collected from the inbound message to a customer database. The data transaction then retrieves the information from that database and sends this data in an outbound message to the customer.
  - Perform a Web service operation to connect to a Web service, collect the information that is requested in the inbound message, and then send the information to the customer in an outbound message.
- In a data application, you can use nodes to:
  - Perform a data transaction that sends the data collected from a caller to a customer database. The data transaction then retrieves the information from that database to present to the caller.
  - Perform a Web service operation to connect to a Web service, collect the information that is requested by the caller, and then present the information to caller.
  - Send SMS or email messages to the customers.
- In a web application, you can use nodes to:
  - Collect user input from the HTML input widgets.
  - Perform a data transaction that sends the data collected from a user to a customer database. The data transaction then retrieves the information from that database to present to the user.
  - Perform a Web service operation to connect to a Web service, collect the information that is requested by the user, and then present the information to user.
  - Send SMS or email messages to the customers.

Orchestration Designer offers three types of nodes to build projects. All three types of nodes are available in the Palette pane of the call flow editor, message flow editor, and web flow editor. These nodes are grouped into two or three sets depending on whether the module nodes have been imported into the Orchestration Designer workbench.

The following are the different types of nodes that are used to build speech, message, and web applications. These node types are described in detail in the subsequent sections.

- Templates (Composite nodes). Composite nodes contain preset functionality and options, which are used as templates. These nodes are easier to use than the Application Items (basic nodes), but they are also intended to be more limited in scope.
- Application items (Basic nodes). Simpler, more generic nodes. These nodes offer greater flexibility than the other two types of nodes. However, to use these nodes effectively, you should be familiar with programming concepts and terminology.
- Module nodes. Available only if you import reusable speech or message project modules into the Orchestration Designer workbench. In speech projects, these modules can be Orchestration Designer projects, Nuance Open Speech Dialog Module (OSDM) modules, Nuance Dialog Module (NDM) modules, or any other Web application that can be invoked as a VXML <*subdialog*>. These modules are custom-built modules and can vary widely in simplicity, complexity, and flexibility.

  Note:

  Orchestration Designer supports the following OSDM and Nuance Dialog Module (NDM) versions:

  - OSDM versions: Address OSDM 2.0.3, Core OSDM 2.0.4., and Name OSDM 2.0.1
  - NDM versions 5.0 and later

  In message flow projects, you can import only those reusable message application modules which are created and exported by using Orchestration Designer.

  You can import a reusable module only in an Orchestration Designer project that is created for a channel that is same as that of the reusable module. For example, you can import a reusable speech application module only in a speech project.

Templates (Composite nodes)

Templates (composite nodes) contain preset functionality and options, which are used as templates. Template nodes are easier to use than the Application Items (basic nodes), but are also intended to be more limited in scope.

Composite nodes are templates for a predetermined use. These nodes are actually Form nodes that Orchestration Designer populates with one or more form items, which in turn makes it easier to create speech applications or message applications. For example, a **Prompt and Collect** node in a speech application consists of:

- A **Form** node with a nested **Prompt** item
- A speech **Input** item
- A selection of event handlers

Orchestration Designer displays the composite nodes under **Templates** in the Palette pane of the call flow editor and message flow editor. Composite nodes have a predefined structure in the node editor.

For more information about composite nodes (Templates), see Detailed node descriptions on page 633.

Note:

A deleted node has a marker in the Task tab and the associated Java class is marked with a warning marker. The task may be deleted if you want to retain the Java code in the project.

Important:

Be aware that Orchestration Designer does not regenerate the Java code and you may encounter Java errors in the future upgrades if the Orchestration Designer run-time API changes. In such cases, you have to manually rectify the Java errors.

Templates (Composite nodes) properties

Depending on the composite node that you select, composite nodes contain some or all of the following properties:

Name	Description
Name	A name that reflects the central purpose of the node. Use naming conventions for Java components. For more information, see Conventions for naming Java components on page 59.
Preferred Path	True to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has Preferred Path set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to IQ for reporting.
Secure Fetch	True if you want the next node in the call flow or message flow to get and post data by using HTTPS. If you set this property to true, the system generates the URL for the next node in the call flow or message flow as HTTPS and uses the configured fetch secure port.

Table continues…

Name	Description
Comments	A comment that you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank. Orchestration Designer uses the text that you enter in the **Comments** property for the pop-up hint that appears whenever you move the pointer over the node in the workspace. If you leave the **Comments** property blank, Orchestration Designer uses the name of the node for the pop-up hint.

Application items (Basic nodes)

The Application items (basic nodes) are relatively open-ended nodes with limited built-in functionality. Usually, a basic node performs a single function, although you can modify the node to make the node more complex.

Orchestration Designer displays the basic nodes under **Application Items** in the Palette pane of the call flow editor or message flow editor.

Some basic nodes have default subflows in the node editor.

For more information about the basic nodes (Application Items), see Detailed node descriptions on page 633.

Note:

A deleted node has a marker in the Task Tab and the associated Java class is marked with a warning marker. The task may be deleted if you want to keep the Java code in the project.

Important:

Be aware that Orchestration Designer does not regenerate the Java code and you may encounter Java errors in the future upgrades if the Orchestration Designer run-time API changes. In such cases, you have to manually rectify the Java errors.

Application items (Basic nodes) properties

Except the **Symbolic** node, basic nodes contain some or all of the following properties depending on the basic node that you select:

Name	Description
Name	A name that reflects the central purpose of the node. Use naming conventions for Java components. For more information, see Conventions for naming Java components on page 59.

Table continues…

Name	Description
Preferred Path	True to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has Preferred Path set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to IQ for reporting.
Secure Fetch	True if you want the next node in the call flow or message flow to get and post data by using HTTPS. If you set this property to true, the system generates the URL for the next node in the call flow or message flow as HTTPS and uses the configured fetch secure port.
Comments	A comment that you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank. Orchestration Designer uses the text that you enter in the **Comments** property for the pop-up hint that appears whenever you move the pointer over the node in the workspace. If you leave the **Comments** property blank, Orchestration Designer uses the name of the node for the pop-up hint.

Module nodes

In a speech application, a module node is a speech project that is treated as a secondary call flow or module. In a message application, a module node is a message flow project that is treated as a secondary message flow or module. In a web application, a module node is a web flow project that is treated as a secondary web flow or module. The module node invokes the speech project module or message flow project module, or web flow project module and transfers the application focus to that module.

Note:

The name under which the system groups these modules varies depending on the source of the module. For instance, any modules that you create in Orchestration Designer are grouped together under the category name that you specify in the **Category** field in **Orchestration Designer** properties in the **Properties for <*project name*>** dialog box. For more information, see Orchestration Designer project properties on page 490. Other modules, such as Nuance OSDM modules or other custom modules, are grouped together under **Modules** or other label that is specified in the appropriate category fields.

To use a module node, you must first create a speech project or message flow project or a web project to be used as the module. You must then deploy it into the Orchestration Designer workspace.

For more information, see Deploying a speech, or a message flow, or a web project as an Orchestration Designer reusable module on page 431.

Note:

You can import a reusable module only in an Orchestration Designer project that is created for a channel that is same as that of the reusable module. For example, you can import a reusable speech application module only in a speech project.

Note:

A deleted node has a marker in the Task Tab and the associated Java class is marked with a warning marker. The task may be deleted if you want to keep the Java code in the project.

Important:

Be aware that Orchestration Designer does not regenerate the Java code and you may encounter Java errors in the future upgrades if the Orchestration Designer run-time API changes. In such cases, you have to manually rectify the Java errors.

Tip:

You can provide a description for the module in the **Comments** property. For example, provide information about what the module does or why you are using the module. Orchestration Designer uses the text that you enter in the **Comments** property for the pop-up hint that appears whenever you move the pointer over the node in the workspace. If you leave the **Comments** property blank, Orchestration Designer uses the name of the node for the pop-up hint.

Setting global application properties

About this task

The properties that you set for the AppRoot node application items and the AppRoot node event handlers in the **AppRoot** node are applied globally in the speech, or the message, or the web application.

Procedure

1. Open the call flow editor or message flow editor or the web flow of the speech application or message application or web application respectively, in which you want to set the global application properties. For more information, see Accessing the project flow editor on page 132.
2. Double-click **AppRoot**.

   The system opens the AppRoot in the AppRoot tab.

3. From the Palette pane, drag the AppRoot node application items and AppRoot node event handlers that you want to add to the AppRoot tab workspace.
4. In the AppRoot tab workspace, select the AppRoot node application item or the AppRoot node event handler.
5. In the Properties view, set the properties.
6. Similarly, set the properties for all the AppRoot node application items and AppRoot node event handlers that you drag to the AppRoot tab workspace.

Project flow management

Adding nodes and other items to the workspace

About this task

You can click-and-drop or drag the nodes and other items into the Editor view workspace.

Procedure

1. In the Palette pane, click the node or item that you want to add to the project flow.
2. In the workspace, click the location where you want to add the node or item.

   For more information about creating nodes and node items, see Creating nodes and node items quickly on page 145.

   For more information about the types of nodes available in Orchestration Designer, see Nodes on page 138.

Creating nodes and node items quickly

About this task

You can drag items from the Avaya OD Navigator view into the flow area to create nodes quickly. For example, dragging a prompt into the flow creates an Announce node, and dragging a grammar creates a Prompt and Collect node.

You can also quickly create node items from within a node. For example, you can create a prompt item in a Prompt and Collect node or a Web service operation file in a Data node.

Note:

You can quickly create the following node items from within their respective node editors:

- Prompts
- Grammars

- Database
- Web services
- Pluggable connector items

Procedure

1. Open the appropriate node editor.
2. Perform one of the following actions:
   - Right-click the existing node item and click **Create New** > ***node item name***.

     For example, right-click a **Prompt** node item and click **Create New Prompt File**.
   - Click the node item and press `Control`+`Shift`+`N`.

Result

After you create a new node item, the system automatically updates the call flow, or message flow, or data flow to refer to the new node item.

Note:

If you create new nodes or node items by using the toolbar icons, you must manually update the call flow, or message flow, or data flow to refer to the new nodes or node items.

Configuring nodes and other items

About this task

You can configure almost any node or item in the Editor view workspace. You must configure the nodes or other items to ensure that the nodes and other items work the way they are intended to work.

For more information about editing and configuring nodes and other flow items, see Adding nodes and other items to the workspace on page 145 and Nodes and Palette options on page 633.

You also need to add one or more application resources, such as phrases, prompts, and grammars, to some nodes. For more information, see Orchestration Designer speech application resources on page 62 and Orchestration Designer message application resources on page 83.

Procedure

1. In the workspace, double-click the node or item that you want to configure.
2. In the Properties view, set the properties for the node or the item. You can edit some properties directly in the node editor for that node.

Selecting objects in a workspace

About this task

Use the **Select** option to select one or more objects in the workspace. Objects include both nodes and connectors. Use the **Select** option to move nodes or edit connector lines that are already in the workspace.

Procedure

1. In the Palette pane of the call flow editor or message flow editor, click **Select**.
2. Perform one of the following actions. To select:
   - A single object, click the object.
   - A group of nodes, click and drag to enclose the nodes you want to select.

     Note:

     The click-and-drag method selects only nodes and does not select connectors. To add connectors to a selected group, use the following Control-click method.
   - Multiple objects that are not grouped together in the workspace, or to select connectors, press and hold the Control key, and then click each object that you want to select.

Fine tuning the position of the connector dots or a node in a workspace

Procedure

1. In the Palette pane of the call flow editor or message flow editor, click **Select**.
2. In the workspace, click the connector or node that you want to reposition.
3. Press the Period (.) key.
4. Use the arrow keys to move the object to the location you want.
5. After you reposition the object, press the Period (.) key.

Connecting nodes in a workspace

About this task

You can use the **Connection** option to connect the nodes with a connector and create the flow of the speech application or message application. The arrows, on the nodes and on the connector, indicate the direction of the flow.

Tip:

To quick-enable the **Connection** option, use the `Shift` key.

Procedure

1. In the Palette pane of the call flow editor or message flow editor, click **Connection**.
2. In the workspace, click the outgoing connection point of the first node.
3. Click the incoming connection point of the node to which you want to connect the first node.

   After you connect the two nodes, you can use the **Select** option to reposition the connector line. Repositioning connector lines can help you to reduce workspace clutter. For more information, see Fine tuning the position of the connector dots or a node in a workspace on page 147.

Adding comments to a workspace

About this task

You can use the **Label** option to add comments, instructions, or other text messages directly to the workspace, much like a virtual *sticky note*.

You can use labels to:

- Insert explanations of what a certain group of nodes does.
- Add comments for fellow team members.
- Add comments in the workspace.

Labels are not part of the generated code.

Procedure

1. In the Palette pane of the call flow editor or message flow editor, click **Label**.

   Note:

   If the **Label** item is hidden, click the arrow next to the **Bookmark** item to view the **Label** item.

2. In the workspace, click the location where you want to add the label.
3. Type the text that you want to add inside the label.

   Note:

   When you drag a label into the workspace and type the text in it, the label box expands to accommodate the text. You can drag the handles to adjust the height and width of the label. After you manually adjust the height and width of the label, the label no longer expands automatically. Text that does not fit into the space, is hidden until you manually adjust the size of the label.

Adding a bookmark to an Orchestration Designer project flow

About this task

You can use the **Bookmark** option to create bookmarks or location markers in an Orchestration Designer project flow. Bookmarks are useful primarily in large, complex project flows. Such project flows can have several key sections that you want to navigate between. You can add bookmarks to the key sections to navigate quickly and easily between these sections.

After you add the bookmarks, you can use the Outline view to navigate between the bookmarks.

Bookmarks are not part of the generated code.

Procedure

1. In the Palette pane of a project flow editor, click **Bookmark**.

   Note:

   If the **Bookmark** item is hidden, click the arrow next to the **Label** item to view the **Bookmark** item.

2. In the workspace, click the location where you want to add the bookmark.
3. Type the text that you want to add inside the bookmark.

Navigating to a bookmarked node

Procedure

1. In the upper-right corner of the Outline view, click **Bookmark**.
2. In the Bookmark view, click the bookmarked node.

   The system highlights the bookmarked node in the main workspace of the call flow editor or the message flow editor.

Calling a reusable speech module using local call

About this task

You can use local call to transfer data from a complex variable as well as from a collection of simple and complex variables, between the reusable speech modules. The local call does not use the VoiceXML subdialog to invoke a reusable speech module.

The VoiceXML subdialog uses the Avaya Voice Browser (AVB) to keep a track of the context, create a new context, and handle the return. Hence, the VoiceXML subdialog passes only simple text values in between the reusable speech modules. By default, Orchestration Designer calls the reusable speech modules by using the VoiceXML subdialog.

In local call, the Orchestration Designer runtime invokes the reusable speech module on the application server and uses a redirect to move to the new Web application. The Orchestration Designer runtime manages the context switch and the return. This ensures efficient transfer of data from a complex variable as well as from a collection of simple and complex variables, between the reusable speech modules.

You can set the call type to subdialog or local call.

Procedure

1. From the **Palette** pane, drag the reusable speech module into the call flow editor. The system displays the **Module Call Type** dialog box.

   Note:

   If you call an OSDM or a NDM module, Orchestration Designer enforces the subdialog call. If the reusable speech module that you call uses a complex variable as an input parameter, Orchestration Designer enforces the local call. In both the cases, the **Module Call Type** dialog box is not displayed.

2. Perform one of the following actions:
   - Click **Subdialog** to set the call type to subdialog call.
   - Click **Local** to set the call type to local call.

   Note:

   After you set the call type, you cannot edit it. If you want to edit the call type, you can delete the reusable speech module from the call flow editor, and then drag a new instance of the reusable speech module into the call flow editor.

Branching the call flow, or message flow, or data flow based on date and time intervals

About this task

You can use the datetime functions and the compound conditions to build call flow, or message flow, or data flow branches based on date and time variables of any format. The **Datetime Functions** item consists of **Convert Date String**, **Set OD Date/Time**, and **Convert OD Date/Time** functions. These functions are available in the Palette pane of a data node. You can also convert values based on time zone.

You can use these functions to route your call flow, or message flow, or data flow based on the date and time. For example, you have an organization level calendar that includes holidays, weekends, and business hours. You want to compare these days with the date and time provided by the caller or contained in the inbound message, and announce the exceptions. To do this, perform the following actions:

Procedure

1. Use the **Convert Date String** function to convert the date string of the holiday into a Java date object. You can get this date into the system from any data source.

2. Use the **Set OD Date/Time** function to split the input date and time Java date object, for example, year, month, day of the month, day of the week, hour, minute, and second, and populate the OD date and time variables. You can then use these variable field values individually for comparison.
3. Use the **Convert OD Date/Time** function to convert the split OD date and time variable values into a single Java date object.
4. Use the conditional operators that support numeric values to compare both the Java date objects and verify if the input date given by the caller or contained in the inbound message is a holiday.

   When comparing the two Java date objects, the conditional operators that you select must support numeric in either the **If** or the **Expression** item. The operators that support numeric values are =, !=, >, =>, <, <=.

   Java date objects make comparison accurate because of their numeric value with millisecond precision.

Node management

For a node to be of any practical use, you must modify or edit the node. Each node has an associated node editor where you can modify existing flows or create new ones.

The node editor for each node has its own set of palette options which is available in the **Palette** pane. Most nodes also include a default structure in the node details editor which you can modify.

Note:

A deleted node has a marker in the Task tab and the associated Java class is marked with a warning marker. The task may be deleted if you want to keep the Java code in the project.

Important:

Be aware that Orchestration Designer does not regenerate the Java code and you may encounter Java errors in the future upgrades if the Orchestration Designer run-time API changes. In such cases, you have to manually rectify the Java errors.

Specifying a name for a node

About this task

When you drag a node into the workspace, the system automatically assigns a generic name to the node. Each node must have a unique name within the project.

Caution:

Do not use double-byte or extended ASCII characters or Java reserved words such as Try, Catch, and Switch when naming nodes.

Note:

Node names must follow Java naming conventions. For more information, see Conventions for naming Java components on page 59.

Tip:

Specify a name that communicates the central purpose or function of the node.

When you name a node for the first time and save a project, the system automatically names the Java class, that is associated with that node, with the same name as that of the node. Later, when you rename the node, the system renames the associated Java class, only if you save the call flow or message flow after renaming the node.

Note:

Be sure to save the call flow or message flow after you rename a node. Otherwise, the associated Java class can get out of sync with the rest of the application.

Perform one of the following actions:

- In the workspace, slowly double-click the node name that you want to rename and specify a new name for the node.
- In the workspace, click the node that you want to rename. In the Properties view, in the **Name** field, specify a new name for the node.

Opening a node editor

About this task

You can edit a node in the node editor.

All node editors open in the same workspace as the project flow editor. When you open a node editor, the system displays the node editor tab at the bottom of the flow editor view. You can use this tab to switch quickly between the node editors and the main flow in the workspace.

Each node editor has a set of palette options which is available in the Palette pane.

Note:

A deleted node has a marker in the Task Tab and the associated Java class is marked with a warning marker. The task may be deleted if you want to keep the Java code in the project.

Important:

Be aware that Orchestration Designer does not regenerate the Java code and you may encounter Java errors in the future upgrades if the Orchestration Designer run-time API changes. In such cases, you have to manually rectify the Java errors.

- In the call flow editor, double-click the node that you want to open.

Adding items to a node

About this task

The procedure for adding items to a node editor is similar to that of adding nodes to the main call flow or message flow, except for the following key differences:

- In a node editor, use the drag-and-drop operation to move the items to the location you want, in most cases.
- In a node editor, you can add an item only to a location where the item is allowed to be added.

  To know if you can add an item to a location in the node editor, drag the item to the location. If the pointer becomes a plus sign (+), you can add the item.
- Many node items depend on other items. For example, a **Grammar** item cannot be used by itself. You must attach a **Grammar** item as a sub-item to another item.

  For more information about the dependencies and restrictions on items, see Nodes and Palette options on page 633.
- You need not connect items in the node editor. Usually, items are executed in the order in which the items appear, though there are exceptions. For example, event handlers are executed only when the conditions for their execution are met.

Procedure

1. Open the node editor in which you want to add an item.
2. From the Palette pane, drag the item to the location in the node editor where you want to add the item.

Adding white space to a data node

About this task

You can use the **Separator** item to add white space between the *sections* of a data node.

Procedure

1. In the Palette pane of the data node editor, click **Separator**.
2. In the data node editor, click the item after which you want to add a white space.

Adding event handlers to a module node

About this task

You can use the event handlers to handle any event that is returned from a subdialog directly within the VoiceXML or TextXML that invoked the subdialog, as opposed to from the AppRoot (global) handler. For more information, see AppRoot node event handlers on page 137.

For more information about event handlers, see Events on page 314.

Procedure

1. In the workspace, double-click the module node to which you want to add an event handler.
2. From the Palette pane, drag the event handlers that you want to add to the module node editor.
3. In the module node editor, select the event handler.
4. In the Properties view, set the properties for the event handler.

   Similarly, set the properties for all the event handlers that you drag to the module node editor.

Copying, cutting, and pasting elements in Orchestration Designer applications

About this task

Orchestration Designer supports copy and paste operations for elements of an expanded node, such as logic, and use of resources from one project to another. Orchestration Designer also supports copying, cutting, and pasting a set of nodes and supporting resources, or a set of resources including prompts, phrases, and grammars in the following scenarios:

- From one project to another project.

  Note:

  You can copy, cut, and paste elements only between projects that are created for the same channel. For example, you can cut, copy, and paste elements only from one SMS channel message application to another SMS channel message application.

- Within a project, from one part to another part, within the same flow.

Upon using the copy, cut, and paste operations, the names of the objects are preserved unless conflicts are discovered, in which case modified similar names are auto-generated.

Before you cut, copy, or paste an object, review Caveats for copying, cutting, and pasting elements on page 155.

Note:

Use the Shift key to copy or cut multiple consecutive nodes in a flow, or multiple consecutive items in an editor. Use the Control key to copy or cut multiple nonconsecutive nodes in a flow, or multiple nonconsecutive items in an editor.

Procedure

1. Right-click a node, resource, prompt, phrase, or grammar, and then perform one of the following actions:
   - Click **Cut** to cut the selected object.

- Click **Copy** to copy the selected object.

2. Right-click the location where you want to paste the object, and then click **Paste**.

   The copied object appears with the same name or modified name depending on whether the operation is performed in the same project or between different projects respectively.

Caveats for copying, cutting, and pasting elements

Before using the cut, copy, and paste operations within Orchestration Designer, review the following caveat statements to understand the functionality and limitations:

- When cutting from a call flow or a message flow, nodes or items from the source call flow or message flow are deleted. Any variables that are defined by the call flow or message flow are also deleted.
- When copying across projects, the entire resource dependency tree is copied to the target project including prompts, grammars, phrases, variables, and so on.
- When the resource dependency tree is copied, certain variables are not copied because these variables are implicitly created by other project elements. For example, a **Prompt & Collect** node in a speech application. So if the project element is not part of the clipboard contents, then the variable is not copied to the target project.
- When copying across projects with multiple languages, the system attempts to determine where language resources belong upon pasting. If there are different languages in the source or target project, after-paste manual steps are required to align things properly.
- To undo a copy and paste operation, press **Control+Z**.

  Important:

  You cannot undo a copy and paste operation after you save the changes.

- When splitting large call flows or message flows into multiple call flows or message flows, use **Cut** to move nodes from one flow to the other rather than **Copy**. **Copy** creates variables. **Cut** moves the nodes from one flow to the other.

Note:

You can copy, cut, and paste elements only between projects that are created for the same channel. For example, you can cut, copy, and paste elements only from one SMS channel message application to another SMS channel message application.

Editing the Java class of a node

Before you begin

Before you edit the Java class of a node, ensure that you rectify all the errors in the speech or message application, and save the speech or message application.

About this task

You can add or edit Java code only in those methods which are allowed to be manually edited. Do not add or edit Java code in methods which are not allowed to be manually edited. If you add or

edit Java code in methods which are not allowed to be manually edited, the system overwrites the Java code that you specify with the system-generated code the next time you save the application.

The system-generated code contains comments which specify whether you can manually edit the code.

Procedure

1. In the call flow editor or message flow editor, right-click the node for which you want to edit the Java class, and then click **Edit <*node name*>.java**.
2. In the <node name>.java Java class editor, edit the Java class of the node.

Avaya OD Search

In addition to the other Eclipse search plug-ins, you can use **Avaya OD Search** to search for the speech or message project element references, nested inside the project resource files, without manually opening different editors to locate these elements.

You can use the search plug-in to:

- Search the speech or message project element references that you want to modify or delete.
- Quickly determine where resource elements, such as prompts, grammars, variables, phrases, and TTS entries, are referenced in other container resource files such as database operation, Web service operation, prompt, and flow files.

With the **Avaya OD Search** plug-in, you can use wildcards within your searches to search for project element references within the defined scope. You can use the asterisk (*) and question mark (?) wildcards for searching partial element names.

Searching a project element

About this task

Note:

You can right-click the resource file in the Avaya OD Navigator view and click **Orchestration Designer** > **Search**. The system automatically highlights the searched element in the Search view. You can double-click the highlighted element to open the element in the appropriate editor.

Procedure

1. On the **Search** menu, click **Search**.

   The system displays the Search dialog box.
2. Click the **Avaya OD Search** tab.

3. In the **Search by item name** field, type the name of the element that you want to search.

   Note:

   - If you select an element in Avaya OD Navigator and open the Search dialog box, the system automatically lists the selected element name in the **Search by item name** field.
   - The system retains the values specified in the **Search by item name** field as a list and the list appears when you reopen the Search dialog box to perform another search.
   - You can use the following wildcards to search partial element names:
     - Use the asterisk (*) to represent any series of characters.
     - Use the question mark (?) to represent any single character.
4. **(Optional)** Select the **Case Sensitive** check box to search the entries that exactly match the capitalization specified in the **Search by item name** field.
5. **(Optional)** Select the **Search for unused items** check box to search the elements that are not used in any container resource such as the call flow or message flow. If you select the **Search for unused items** check box, the **Search by item name** field is unavailable.

   Note:

   To search unused elements within a project, in the Avaya OD Navigator view, right-click the *<project name>* folder and use the Search Unused option.
6. In the Search For Type area, select the check box corresponding to the type of element that you want to search.
7. In the Scope area, click the scope of the search. Click one of the following options:
   - **Workspace**: To search in all the projects and files in the workspace.
   - **Selected resources**: To search in the project where the selected resource is located.
   - **Enclosing projects**: To search in any number of projects selected in the workspace.
   - **Working set**: To search only in the resources that belong to the selected working set.
8. Click **Search**.

   The system displays the search results and their corresponding locations as a tree in the Search view. The results are also highlighted in the tree. The tree is part of the project > file > node > item hierarchy. You can double-click the highlighted searched element to open the element in the appropriate editor.

Working with the search results

Procedure

1. On the **Search** menu, click **Search**.

Note:

You can right-click the resource file in the Avaya OD Navigator view and click **Orchestration Designer** > **Search**.

The system displays the Search dialog box.

2. Click the **Avaya OD Search** tab and specify the search parameters, and then click **Search**. For more information, see Searching a project element on page 156.

   The system displays the search result in the Search view.

3. You can perform the following actions in the Search view:
   - View the search results and the corresponding locations of the searched element references. The results are listed as a tree in the Search view, which represents the project > file > node > item hierarchy for the searched element.
   - Double-click an element in the search result to open the element in the appropriate editor.
   - Right-click an element in the search result and click **Show In** > **Avaya OD Navigator** to locate and highlight the element in the Avaya OD Navigator view.
   - Drag an element from the search result into the appropriate editor to create a corresponding node. For example, you can drag a prompt into the call flow editor to create an **Announce** node, drag a database operation element into the call flow editor to create a data node referencing that database operation element, or drag a phrase element into a prompt file.

Subflow management

Subflows

Creating and working with subflows is similar to creating and working with main flows. However, the following are the differences:

- Palettes are similar with all modules listed. However, return items are different.
- Subflows have a Begin node which is the entry point to the flow. There is no AppRoot in a subflow. AppRoot is defined in main.flow.
- Subflows support multiple return points, forwarding the request to the next node in the calling call flow or message flow. Main flows allow only one return, that is, exit.

The following are the considerations about subflows:

- Subflows are always contained within the same application scope as the main flow.
- Subflows share project variables.

- Subflows share resources such as prompts, grammars, DB operations, and WS operations.
- Subflows share application context and session objects at run time.
- Subflows can be nested, that is, you can call other subflows or modules.
- There is no limit to the number of subflows in a project. However, the subflows are not intended to replace modules. Therefore, you should design applications, particularly large applications, appropriately.
- Implicit variable item names, such as names of Menu, Prompt & Collect, and Transfer items, must be unique across all flows in a project. When adding or renaming these items, the call flow editor, message flow editor, and project variables enforce uniqueness.
- Subflow code is generated in a separate package. For example, MySubflow.flow generates Java code in the flow.subflow.MySubflow package.
- Servlet names/mappings changed (web.xml). For example, MySubflow.flow has servlet names starting with MySubflow-<*node name*> and servlet mappings starting with / MySubflow-<*node name*>.
- With subflows, there should be no impact on the run-time performance. Servlets are part of the same Web application, and requests are forwarded when entering or returning from a subflow.

Subflows vs. modules

When designing a speech application or a message application, take design considerations into account. For example, decide when and where to create subflows or reusable modules in your application. The following table shows the difference between modules and subflows:

Modules	Subflows

• Project resources are self-contained.
• Reusable across projects.
• Better for multiple developers.
  - You can work separately on your modules without affecting the work of other developers.
• Separate application contexts at run time.
  - Session data is independent.
  (Orchestration Designer runtime has mechanisms to share Java objects across modules.)
  - The voice browser fetches the speech application module as a VoiceXML <*subdialog*>.
  - The text browser fetches the message application module as a TextXML <*subdialog*>.
	• Shared project resources such as prompts, grammars, and variables.
	• Reusable only within a project.
	• Not good for multiple developers.
	  - Multiple project files are shared and updated by various editors, such as variables and web.xml. Trying to merge updates to these files is difficult.
	• Same application context at run time.
	  - Session data is shared.
	  - Requests are forwarded while entering or exiting subflows. There is no round trip HTTP request or response.
	  - In a speech application, all VoiceXML is part of the same dialog.
	  - In a message application, all TextXML is part of the same dialog.

Creating a subflow

Procedure

1. On the **File** menu, click **New** > **Other**.
2. In the New dialog box, double-click **Avaya OD Development**.
3. Click **Flow File**, and then click **Next**.
4. In the New Call Flow wizard, in the **Available Projects** pane, click the project in which you want to add a subflow.
5. In the **Name** field, type a unique name for the subflow.

   The system creates a new `<subflow name>.flow` file in the `flow` folder.

Invoking a subflow

Procedure

1. From the Palette pane of a flow editor, drag **Sub Flow** into the flow workspace.

   Note:

   You can invoke a subflow from any flow within your application.
2. In the Properties view of the subflow node, in the **Flow Name** field, click the name of the subflow that you want invoke from the flow.

To set the other properties of the subflow node, see Sub Flow Begin node on page 664.

Note:

When you select a subflow, the system adds return points to the node. Each return point is used to divert the flow to the next node when returning from the subflow. You can add, rename, or delete return points from a subflow.

Note:

Multiple return points allow you to return from a subflow and execute different parts of the flow depending on the return point that is used. Similar to a “switch” statement in programming.

Considerations for copying, cutting, and pasting subflows

When you copy, cut, and paste subflows, remember the following:

- Copying, cutting, and pasting is supported in the main flow and subflows.
- Copying or pasting from one flow to another flow in the same project moves the nodes, but it does not delete or rename the variables.
- You cannot cut and paste within the same flow.
- Cutting or pasting from one flow to another flow in the same project duplicates the nodes, but:
  - It creates copies of nodes and variables.
  - To avoid name conflicts, if you rename a variable after pasting the variable, then all references to the variable in the copied selection are updated to reference the name of the new variable.

    Important:

    When you copy other nodes that reference a variable that was renamed during an earlier copy and paste operation, then the copied nodes may not reference the right variable when pasted. For example, if you copy a Prompt & Collect node named AccountNum and rename it to AccountNumCopy, then all nodes in the selection change to reference the AccountNumCopy node.

    If you later copy and paste other nodes that reference AccountNum, then these nodes continue to reference AccountNum. This is because at this point the project has both the AccountNum and AccountNumCopy variables.
- Copying a subflow reference node to another project copies the subflow and all nested resources to the target project.

Note:

You can copy a subflow from one project to another project only if the two projects are created for the same channel. For example, you can copy a subflow from one speech project to another speech project only.

Guidelines for refactoring a project to use subflows

Before refactoring projects to use subflows, review the following:

- Move nodes by cutting and pasting them between the main flow and the subflows. This preserves variable names and references, and does not create duplicates, that is, clones.
- Some nodes cannot be moved to a subflow. AppRoot and Return (main.flow) are not allowed on subflows. Do not include them in the selection when moving nodes to the subflow.
- Symbolic node references may be cleared if symbolic node is copied, but the reference node is not.
- While planning the return points of the subflow, review the following:
  - A subflow can utilize multiple return points in order to handle different return paths in the calling flow.
  - When selecting a group of nodes to move to a subflow, often a symbolic node indicates the need for a return point when the reference node is not in the selection.

    Important:

    Back up your project so that you can revert if you encounter problems.
- Although renaming is supported, do not rename a subflow. Never rename a main.flow.
- If you copy or cut from a main flow to a subflow, the Java code is also copied and moved. If you reference a code in the flow package without a fully qualified name, such as project variables, the user-defined code can have compile errors.
- You must organize imports to fix import statements. Perform one of the following actions:
  - Press **Control**+**Shift**+**O**. This opens Java class: Source > Organize Imports.
  - From the Package/Project Explorer view, right-click Java class: Source > Organize Imports.
- Cutting and pasting in the same flow is no longer supported because it is just a move operation.

Important:

Do not create infinite loops. Do not create circular references to subflows.

Orchestration Designer project documentation

You can generate documentation for an Orchestration Designer speech project or message flow project.

You can view the flow map of the main flow and modules that are used in the main flow. You can also view the flow details as well as information about the variables, prompts, phrases, and grammars that are used in the project.

The system generates a PDF and an HTML output for the project documentation and saves the output at the directory location that you specify. If you use the default `\data` directory to save the generated documentation, the system creates a `docs` folder within `\\workspace\<project name>\data` folder, and saves the generated documentation in the `docs` folder. The `workspace` folder is the folder that you specify to save the project when you start Eclipse.

Generating documentation for an Orchestration Designer project

About this task

Note:

You can generate documentation only for an Orchestration Designer speech project or message flow project.

Procedure

1. In the Navigator view or the Avaya OD Navigator view, right-click the speech project or message flow project for which you want to generate the documentation.
2. Click **Orchestration Designer** > **Generate Documentation**.

   The system highlights the project for which you want to generate the documentation in the **Available Projects** area of the Generate Project Documentation dialog box.

   You can also select another project if you want to generate documentation for that project.
3. In the **Enter a directory** field, type or browse to the directory path where you want to save the generated documentation.

   Note:

   The default directory path is `\\workspace\<project name>\data`.
4. Click **OK**.

Result

The system displays the main.html page of the generated HTML output in the editor workspace.

Next steps

In the left navigation pane of the generated .html page, click the links to view the related information.

On the upper-right side of the generated .html page, click **Print** to view the contents of the .html page in a PDF format.

Chapter 13: Phrases and phrasesets

Phrases

One of the key elements to create speech applications with Orchestration Designer is the phrase. You can use phrases to build prompts, which in turn are used to guide the caller through the call flow.

In speech and interactive voice response (IVR) applications, phrases are among the most basic building blocks. Phrases consist of prerecorded audio files.

Phrases can be as short as one word or as long as several complete sentences. Phrases can even be music recordings, which is often the case with transitional audio prompts. For more information, see Transitional audio prompts in speech applications on page 205.

Although you can use text-to-speech technology to render audible output to callers, prerecorded audio phrases are usually preferable whenever possible. Audio phrases give a finished speech application a more professional presentation and a more satisfying experience to the caller.

Important:

You can use phrases and phrasesets only in speech applications. Call control, message, and data applications do not incorporate phrases or phrasesets.

Example usage of phrases

Usually, phrases are used as prompt segments. You can assemble phrases to build prompts. For example, your speech application uses the following prompt to report to a caller the balance in the checking account of the caller:

“You have $528.48 in your checking account.”

This one prompt, however, can consist of eight or more separate phrases. For example:

- “You have”
- “five hundred”
- “twenty-eight”
- “dollars”
- “and”

- “forty-eight”
- “cents”
- “in your checking account.”

Because a return statement from the system often must use variable values, for example, the balance amount in the checking account in the preceding example, the number and currency words can be contained within and used as variables.

Assembling phrases

You can gain access to phrases only from the Prompt File Editor. In this editor, you can assemble phrase files and other prompt segments into the prompt for use in the speech application.

For more information, see Prompt file editor in a speech application on page 207.

Types of phrases

Orchestration Designer uses two types of phrases: standard and custom.

Standard phrases

Standard phrases are included in Orchestration Designer as part of the language phrase bundles that you purchase. Standard phrases are not, however, installed automatically. To use standard phrases, you must first install the standard phrases.

Note:

Although there is nothing to prevent this, standard phrases are not intended for general use in a speech application. Their primary intended use is to be associated with the localization bundle and audio variables. If standard phrases are used and the localization bundle is later altered, unexpected results are encountered.

Standard phrases consist of common types of phrases that are used in prompts. Standard phrases include numbers in various formats, currency words, names of months, days of the week, years, and similar words and phrases.

You can easily identify standard phrases in the Navigator view because the names of standard phrase audio files start with the **std_** characters.

Note:

Because Orchestration Designer uses the **std_** prefix to distinguish standard phrases, do not use the **std_** prefix when naming custom phrase files.

For information about installing standard phrases, see Installing the standard phrasesets of a localization bundle on page 293.

Note:

Standard phrases were converted into a phraseset with Dialog Designer 3.1.

Custom phrases

You can create custom phrases by using the audio files recorded by you or by a professional voice talent. Custom phrases can be long or short and can also include periods of silence.

You can record custom phrases by using Orchestration Designer. To record phrase file audio of higher quality, Avaya recommends that you engage the services of a professional voice talent and recording studio. With professional studio recordings, audio recordings of phrases are of higher quality, and can be better tuned to sound better and to trim silence from the beginning and end of the recorded files. Also, a professional talent can better match the standard phrase voice, which enables a more seamless prompt to callers.

Note:

Professionally recorded audio files must be transcoded to one of the formats required by the VoiceXML 2.0 standard. For more information, see the “Audio File Formats” appendix in the W3C VoiceXML 2.0 Recommendation.

Phrasesets

You can use phrasesets to group phrases, especially phrases that are related to a particular speech application. Phrases are used to build prompts, which in turn are used to guide the caller through the call flow.

The main advantage of grouping phrases, other than better organization, is that phrasesets greatly improve file resource efficiencies which translates into quicker build times. There are four files per phrase: a .phrase file, a .java file, a java .class file, and a .wav file. As a result, if your speech project contains 1000 phrases, then the system actually needs to build and manage 4000 files. In a phraseset, you can group upto 1000 phrases. Hence, in the preceding example, there are only 1003 files to manage, which are, a .phraseset file, a .java file, a Java .class file, and 1000 audio files.

Phrasesets also support referencing audio files stored on an external server.

Important:

You can use phrases and phrasesets only in speech applications. Call control, message, and data applications do not incorporate phrases or phrasesets.

Phraseset file editor

You can use the Phraseset file editor to modify the phraseset files. When you open a phraseset file, the Phraseset File Editor for that file is displayed in the main editor workspace of Orchestration Designer. The Phraseset File Editor has the following pane and tabs:

- **Phrases** pane: Use this pane to create or delete phrases from an already created phraseset file.
- **Phrase Data** tab: Use this tab to set or modify text descriptions, comments, search keywords, and other attributes for the phrases within the phraseset file.
- ***<Audio>*** tab: Use this tab to set or modify parameters for an associated audio file, or to record a new audio file. The name of the audio tab changes depending on the phrase audio file location that you specify in the **Phraseset Audio Files Location** area.

Phraseset management

You can add and delete phrases from a phraseset file. You can also specify the audio file location for a phrase within a phraseset file.

When you open the Phraseset File Editor, all phrases that are currently included within the phraseset are shown in the **Phrases** pane.

Creating a phraseset

About this task

You can use phrasesets to group phrases, especially phrases that are related to a particular speech application. To create a phrase, see Phraseset file editor on page 168.

After you create a phraseset file, use the Phraseset File Editor to record an audio file or to set other properties of the phraseset file.

Procedure

1. On the **File** menu, click **New** > **Other**.

   The system displays the New dialog box.

2. Double-click **Avaya OD Development**.
3. Click **Phraseset File**, and then click **Next**.

   The system displays the New Phraseset wizard.

4. On the Create a Phraseset page, specify the speech project, language, and name for the phraseset file, and then click **Next**.

5. On the Specify Audio File Directory page, in the **Audio file directory** field, type the directory path or click **Browse** to navigate to the directory that contains the audio files that you want to add to the phraseset file.
6. Click **Finish**.

   The system creates the phraseset file and opens the phraseset file in the Phraseset File Editor. You can use the Phraseset File Editor to add or delete phrases from the phraseset file, set other properties for the phrase files within the phraseset file, and record an associated audio file. For more information, see Phraseset file editor on page 168.

New Phraseset wizard: Create a Phraseset page field descriptions

Name	Description
Available Projects	The speech project for which you want to create the phraseset file. You must select a speech project.
Available Languages	Check box to select the language in which you want the phraseset to be available. You must select a language. This pane contains all the languages that exist in your workspace. For more information, see Project languages on page 277.
Phraseset file name	A name for the phraseset file. Type a name that intuitively describes the content of the phraseset file. The system automatically adds a ".phraseset" extension to the file name when the phraseset file is created. **Note:** Phraseset file names must follow Java naming conventions. For more information, see Conventions for naming Java components on page 59. **Caution:** Do not use double-byte or extended ASCII characters or Java reserved words such as Try, Catch, and Switch when naming the phraseset. If you do not specify a name for the phraseset file, the system generates and assigns a generic name to the phraseset file.

Adding a phrase to a phraseset file

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project in which you want to add a phrase to the phraseset file.
2. Double-click **<*language name*>** > **phrasesets**.
3. Right-click the *.phraseset file to which you want to add a phrase, and then click **Open With** > **Phraseset Editor**.
4. In the **Phraseset Audio Files Location** area, perform one of the following actions:
   - Click **Local** if the audio files for the phrase are within the project resources.
   - Click **External** if the audio files for the phrase are external.
   - Click **MMF (MPS only)** if you are using an MPS.

   For more information, see Specifying the phrase audio file location on page 171.
5. Click **Add New**. The system displays the Add new phrase dialog box.

   Note:

   If you click **External**, the system adds a new phrase with a default name in the **Phrases** pane without displaying the Add new phrase dialog box.
6. In the Add new phrase dialog box, specify the information, and then click **OK**.

   You can click the phrase in the **Phrases** pane and specify basic identification properties for the phrase on the Phraseset File Editor: Phrase Data tab field descriptions on page 173.

Add new phrase dialog box field descriptions

Name	Description
Phrase name	A name for the phrase that you want to add to the phraseset.
Audio Information Properties	

Table continues…

Name	Description
Audio file: • New • Existing	The source of the audio file. The following are the options: • **New** to create a new audio file. If you click **New**, you must also specify the audio file format extension. • **Existing** to use an existing audio file. If you click **Existing**, then in the **File name** field, specify the directory path where the audio file is stored. You must select either **New** or **Existing**.
Extension: • .wav • .au	The appropriate audio format for the audio file. These options are available and mandatory when you click **New** as the **Audio file** type. The default **.wav** file format is used on Windows-based computers. The . **au** format is used on Linux-based and UNIX-based computers. **Note:** Professionally recorded audio files must be transcoded to one of the formats required by the VoiceXML 2.0 standard. For more information, see the “Audio File Formats” appendix in the W3C VoiceXML 2.0 Recommendation.
File name	The directory path where the audio file is stored. This field is mandatory when you click **Existing** in **Audio file**.

Specifying the phrase audio file location

About this task

You should specify the audio file location for a phrase within a phraseset file. The audio files can be local or external.

Orchestration Designer supports MPS, which uses a proprietary audio format called MMF in the prompt play back. Orchestration Designer has the ability to export speech applications to the MPS platform that uses the MMF format. Orchestration Designer does not store the MMF files within the speech application nor can it parse the MMF files to determine the MMF element names. Hence, you must know the MMF file names and element names. MMF file name and element name are case sensitive.

Note:

The Orchestration Designer ISO image contains an external tool called `listeaps.exe`. This tool lists the contents of an MMF file. The documentation is included with the tool.

You should specify the location of the MMF file. An MMF file can contain multiple elements. These elements are the actual audio files. You can specify if the system should use the phrase name or the audio file name to name the element.

For more information, see Phraseset File Editor: Audio tab field descriptions on page 176.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project that contains the phrase for which you want to specify the audio file location.
2. Double-click **<*language name*>** > **phrasesets**.
3. Right-click the `*.phraseset` file that contains the phrase for which you want to specify the audio file location, and then click **Open With** > **Phraseset Editor**.
4. In the Phrases pane, click the phrase for which you want to specify the audio file location.
5. In the Phraseset Audio Files Location area, perform one of the following actions:
   - Click **Local** if the audio files are within the project resources.
   - Click **External** for external audio files. In the **URL base** field, type the URL base where the audio files are located.
   - Click **MMF (MPS only)** if you are using an MPS. Perform the following actions:
     a. In the **Filename** field, specify the location of the MMF file.
     b. In **Element names**, perform one of the following actions:
        - Click **Use Phrase Name** to use the phrase name to name the MMF element.
        - Click **Use Audio File Name** to use the audio file name to name the MMF element. If you click **Use Audio File Name**, the system does not append the file extension when naming the MMF element.

          Note:

          You can click **Use Audio File Name** when you use MPS audio conversion tools, such as WAV2MMF, because these tools use the audio file name to name the MMF element.

Setting the basic identification properties for a phrase

About this task

You can set the basic identification properties for a phrase within a phraseset file by using **Phrase Data** tab in the Phraseset File Editor.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project that contains the phrase for which you want to set the basic identification properties.
2. Double-click **<*language name*>** > **phrasesets**.

3. Right-click the `*.phraseset` file that contains the phrase for which you want to set the basic identification properties, and then click **Open With** > **Phraseset Editor**.
4. In the **Phrases** pane, click the phrase for which you want to set the basic identification properties.
5. Click the **Phrase Data** tab.
6. On the **Phrase Data** tab, set the basic identification properties for the phrase.

Phraseset File Editor: Phrase Data tab field descriptions

Name	Description
Phrase Name	The name of the phrase that you select in the **Phrases** pane. You can click **Set Name** to specify a new name for the phrase. If the Use Phrase Name option is selected, MMF files use this name to name the element within the MMF file.
Voice Talent	The information about the professional individual or studio that recorded the phrase audio file.
Phrase Text	The exact text for the phrase. This field serves three purposes: • It identifies the intended wording and purpose of the phrase. • If no audio file is selected or exists for the phrase, the system automatically uses this text to render the phrase by using text-to-speech (TTS) technology. • If you plan to record the audio file later, this text serves as a script for the individual or company that records the audio. **Note:** To disable the TTS generation missing audio file backup, add the **runtime-Disable-TTS-Backup** context parameter with the value set to “true” to the project `Web.xml` file. For more information, see Disabling the TTS backup on page 174. Although this field is not mandatory, you must specify the exact text for the phrase in this field.

Table continues…

Name	Description
Base Language Text	The exact text for the phrase in the default project language. This field contains the same text as in the **Phrase Text** field. However, when you create applications that you want to convert or translate into another language, the **Base Language Text** field contains text in the language in which you create the application. The **Phrase Text** field, however, contains the text in the language in which you want to translate the application. For example, you develop an application in English, and you want to translate it to German. In this case, the **Base Language Text** contains the text in English. For example, “Good morning. Thank you for calling the German Federal Bank.” The **Phrase Text** field, however, contains the German translation of the English **Base Language Text** phrase, that is, “Guten morgen. Danke fuer Ihre Anruf zum Bundesbank Deutschland.”
Comments	A description for the phrase. This text is displayed only in the Phraseset File Editor.
Search Keywords	The keywords that you want to use for searching phrases. **Note:** The current release of Orchestration Designer does not support the ability to search phrases based on keywords.

Disabling the TTS backup

About this task

To disable the TTS generation missing audio file backup, add the **runtime-Disable-TTS-Backup** context parameter with the value set to “true” to the project `Web.xml` file.

If the **runtime-Disable-TTS-Backup** context parameter already exists, then set the value to “true”.

Procedure

1. On the **Project** menu, click **Properties** > **Orchestration Designer**.
2. Click the **Web Descriptor** tab.
3. If the “runtime-Disable-TTS-Backup” context parameter does not exist, click **Add** to add the context parameter.

The system displays the Add context parameter dialog box.

4. In the **Name** field, type `runtime-Disable-TTS-Backup`.
5. In the **Value** field, type `true`.
6. In the **Description** field, type a description for the context parameter.
7. Click **OK**.

Configuring audio for a phrase in a phraseset file

About this task

You can use the audio tab to select an audio file, record a new audio file, play back the audio, and simulate audio for files that are in an MMF format.

Note:

To record an audio file, you must plug in a microphone to your computer. Before you record an audio file, properly calibrate the automatic record levels for the microphone. Also, use headphones during recording rather than speakers connected to the computer. For more information about configuring Microsoft SAPI Speech for microphone input, see the *Getting started with Orchestration Designer* guide.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project that contains the phrase for which you want to configure the audio.
2. Double-click **<*language name*>** > **phrasesets**.
3. Right-click the `*.phraseset` file that contains the phrase for which you want to configure the audio, and then click **Open With** > **Phraseset Editor**.
4. In the Phrases pane, click the phrase for which you want to configure the audio.
5. Click the audio tab.
6. The name of the audio tab changes depending on the phrase audio file location that you specify in the Phraseset Audio Files Location area:
   - If you click **Local**, the name of the audio tab changes to Local Audio.
   - If you click **External**, the name of the audio tab changes to External Audio.
   - If you click **MMF (MPS only)**, the name of the audio tab changes to Simulation Audio.
7. On the audio tab, configure the audio.

Phraseset File Editor: Audio tab field descriptions

Name	Description
File Name	The name of the file that contains the audio recording assigned to the phrase. For local audio, browse to locate a new audio file or record over the existing audio file. For external audio, test play the audio from the external server. For MMF, the Avaya Application Simulator cannot play back the files that are in an MMF format. You can convert the actual MMF file outside Orchestration Designer by using the mmf2wav MPS tool, or you can select or record your own filler audio to be used for simulation. The simulation audio file name indicates a file that is played back instead of the MMF file.
Browse	The **Browse** button is available only if you click **Local** or **MMF (MPS only)** in the **Phraseset Audio Files Location** area. For local audio, browse to change the currently assigned audio file. Use this button to locate a different prerecorded audio file. The audio file must be in either a .wav or an .au format. For MMF, you can locate the file that you want to play back for simulation instead of the MMF file.
Test	The **Test** button is available only if you click **External** in the **Phraseset Audio Files Location** area. For external audio, click **Test** to test play the audio from the external server.
Format	The audio format of the selected audio file. This field is read only. **Note:** The current release of Orchestration Designer supports only 8KHz 8-bit mono formats.
Position	The position of the playback cursor when an audio file is being played back. This field is read only.
Length	The length, in seconds, of the audio file. This field is read only.

Table continues…

Name	Description
Play button 	To play the audio file. **Note:** This button is available only after you record or import an audio file. In case of external audio, this button is available after you get an external audio file by clicking the **Test** button.
Stop button 	To end a recording or stop an audio file that is being played back. **Note:** This button is available only when you record an audio or play back a recording.
Record button 	To record an audio file. **Note:** This button is unavailable for external audio phrases.

Deleting a phrase from a phraseset file

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project that contains the phrase that you want to delete.
2. Double-click ***<language name>*** > **phrasesets**.
3. Right-click the `*.phraseset` file from which you want to delete a phrase, and then click **Open With** > **Phraseset Editor**.
4. In the Phrases pane, click the phrase that you want to delete.
5. Click **Delete**.

   Note:

   If you delete a phrase that is used in a prompt, the system generates an error when rebuilding the prompt and shows the error in the Problems view.

Recording script report management

You can use the Orchestration Designer localization features to switch between different languages to play a speech application. You can generate a report for a recording studio to record all phrases of a speech application in a new language.

You can use the New Recording Script wizard to quickly generate a cleanly formatted HTML or XML report of all phrases in a speech application for submitting to a recording studio. The report contains the following details:

- Project name
- Language
- Audio file format
- Any specified comments or notes to the recording studio about the phrases
- Date
- A list of phrases including:
  - The audio file name of each phrase
  - The text to be recorded
  - The pronunciation of the phrase

  You can generate a report of all the phrases, user-defined phrases, and standard phrases.

Creating a recording script report and specifying an audio file format

Procedure

1. On the **Project** menu, click **Reports** > **Recording Script**.

   The system displays the New Recording Script wizard.
2. On the Create a Recording Script page, specify the project, language, phrase type, output file type, and location for the recording script report.
3. Perform one of the following actions:
   - Click **Finish** to generate the recording script report without specifying the audio file format or comments for the report.

     The system generates and saves the recording script report file. By default, this file contains Raw (headerless) 8KHz, 8-bit, mono mu-law audio files. If you have generated recording script reports earlier, this file contains the audio file format that you chose for the last generated report.
   - Click **Next**. On the Specify Script Parameters page, specify the audio file format and comments for the recording script report, and then click **Finish**.

New Recording Script wizard: Create a Recording Script page field descriptions

You can set the basic settings such as project name, language, phrase type, output file type, and location to generate and save the recording script report.

Name	Description
Available Project	The project for which you want to generate the recording script report.
Language	The language in which you want to generate the recording script report.
Phrase Type	The phrase type for which you want to generate the recording script report. The following are the options: • **All Phrases** • **User Defined Phrases** (These phrases are also called as Custom Phrases.) • **Standard Phrases**
Output File Type	The format in which you want to generate the recording script report. The options are: • **HTML**: If you want to generate the recording script report in an HTML format. • **XML**: If you want to generate the recording script report in an XML format.
Select the report destination directory	The directory path where you want to save the recording script report. Alternatively, click **Browse** to navigate to an appropriate directory. The file name of the saved report uses the following format: **ProjectName_SelectedLanguage_recordingscript.html**

New Recording Script wizard: Specify Script Parameters page field descriptions

You can specify the audio file format and comments for the recording script report on the Specify Script Parameters page.

Name	Description
Audio File Format	The audio file format for the phrases that you want to include in the recording script report. The following are the options: • Raw (headerless) 8kHz 8-bit mono mu-law • Raw (headerless) 8kHz 8-bit mono A-law • WAV (RIFF header) 8kHz 8-bit mono mu-law • WAV (RIFF header) 8kHz 8-bit mono A-law The default is **Raw (headerless) 8kHz 8-bit mono mu-law**. If you have generated recording script reports earlier, this file contains the audio file format that you chose for the last generated report.
Recording Comments	Comments or instructions for the recording studio.

Phrase file import and export

You can use phrases from one speech application project in other speech application projects. This is useful when you want to use a common set of phrases in two or more applications, or when you have a speech module that uses the same phrases as the parent application.

Exporting a phrases .Zip file

About this task

You can export a phrases .zip file from a speech application project. You can then import the exported phrases .zip file in other speech application projects and reuse the phrases.

Important:

Be careful when exporting phrase files. Prompts rely on phrases, grammars, and other project resources that may not exist in the speech project where you import the exported phrase files. If you import phrase files that rely on resources located in other speech projects, Orchestration Designer generates errors. Be sure that you either import the missing resources in the speech project where you import the phrase files, or redirect the imported phrases to make use of the resources from the other speech projects.

Procedure

1. On the **File** menu, click **Export**.

   The system displays the Export wizard.

2. Double-click **Avaya OD Development**.
3. Click **Phrases Zip File**, and then click **Next**.

The system displays the Export Phrases wizard.

4. Specify the parameters to export the phrases .zip file.
5. Click **Finish**.

Export Phrases wizard field descriptions

Name	Description
Available Projects	The speech project from which you want to export the phrase files.
Available Languages	The languages for which you want to export the phrase files. The Available Languages pane shows only those languages in which phrases exist in the speech project that you select.
Zip file	Directory path and file name for the .zip file that you want to export.
Include directory structure	Check box to include the directory path information in the .zip file.
Include project name	Check box to include the name of the speech project, from where you export the phrases, in the .zip file.

Importing a phrases .Zip file

About this task

You must first export a phrases .zip file from a speech project, and then import the exported file into the speech project where you want to reuse the phrases.

Important:

Be careful when importing phrase files. Prompts rely on phrases, grammars, and other project resources that may not exist in the speech project where you import the exported phrase files. If you import phrase files that rely on resources located in other speech projects, Orchestration Designer generates errors. Be sure that you either import the missing resources in the speech project where you import the phrase files, or redirect the imported phrases to make use of the resources from the other speech projects.

Note:

You can use the following procedure only for the phrase files that are exported to a .zip file by using the Export Phrases wizard in Orchestration Designer. For information, see Exporting a phrases .Zip file on page 180.

Procedure

1. On the **File** menu, click **Import**.

   The system displays the **Import** wizard.
2. Double-click **Avaya OD Development**.
3. Click **Phrases Zip File**, and then click **Next**.

   The system displays the Import Phrases wizard.
4. Specify the parameters to import the phrases .zip file.
5. Click **Finish**.

Import Phrases wizard field descriptions

Name	Description
Phrase zip file	The directory path and name of the .zip file that you want to import.
Available Projects	The speech project in which you want to import the .zip file.
Overwrite existing resources without warning	Check box to overwrite, without warning, any existing phrase files having the same name as that of the phrase files you import.

Importing a delimited phrase data text file

About this task

Note:

Before you import a phrase data file, you must create a delimited phrase data file as per the specifications mentioned in Setting up a delimited phrase data text file for import on page 183.

Procedure

1. On the **File** menu, click **Import**.

   The system displays the Import wizard.
2. Double-click **Avaya OD Development**.
3. Click **Phrase Data**, and then click **Next**.

   The system displays the Import Phrase Data wizard.
4. Specify the import criteria, and then click **Finish**.

Setting up a delimited phrase data text file for import

You can import phrase data from a delimited text file to update the existing phrase data and add new phrase data to a phraseset within your speech application project.

Before importing a phrase data file, you must first create a delimited phrase data text file. The column layout of the fields in the text file is configurable during the import. The rows, however, must have a consistent number of columns, whether or not each column in a row is defined with data or left blank.

As a delimiter, you can use a tab or another consistently used character in your file that does not conflict with the data content. For example, a pipe symbol (|) or a comma.

The delimited phrase data text file can contain some or all of the following accepted data content within its columns, on a row-by-row basis:

- Audio file name
- Phrase name
- Phrase text
- Comments
- Voice talent
- Base language text
- Search keywords

The following is an example of a short delimited phrase file:

```
MainMenu.wav|Main Menu|This phrase describes the main menu|James Earl
Jones Directions.wav||This phrase includes directions to the store|
StoreHours.wav|Hours||Valerie Snow
```

With the preceding example delimited phrase file, the columns are configured as follows within the Import Phrase Data wizard. The remaining data content types are not used.

```
audio file name|phrase name|phrase text|voice talent
```

You can also add or ignore new phrases from the text file. You can control and configure the import by specifying the text row of the file from where you want to begin importing the data, and by defining a zero-based index of the columns. Specifying the text row of the file from where you want to begin importing the data is especially useful if you use the same delimited phrase file repeatedly over time and you want to import only the new data and do not want to overwrite the existing data.

Import Phrase Data wizard field descriptions

Name	Description
From file	The delimited phrase data file that you want to import.
Target Phraseset	
Project	The speech project in which you want to import the delimited phrase data file.
Language	The language of the phrases based on which you want to update the phrases with the data from the delimited phrase data file.
Phraseset	The phraseset that you want to update with the data from the delimited phrase data file.
Options	
New phrases	**Ignore** to ignore the new phrase data in the delimited phrase data file that you import. **Add** to add the new phrase data to the phraseset from the delimited phrase data file that you import.
Delimiter	**Tab** if a tab is used as a delimiter in the delimited phrase data file. **Other** to specify the delimiter that is used in the delimited phrase data file.
Advanced Options	The options to configure the column layout of the delimited phrase data file.
Start import at row	**0** to import the data from the beginning of the phrase data file, or enter the the row number of the phrase data file from where you want to begin importing the data.
<Other fields>	Values in the column name fields, which contain phrase information, based on the data that you define in the columns of the delimited phrase data file. The index to be defined is a zero-based index, zero being the first column, and counts up for the column order for data included in the text file. If a particular type of data is not included or if you do not want to import the data, type the value `-1`.

Chapter 14: Media

Media

Media makes speech recognition applications more effective by providing a visual guide besides audio features. You can use voice and video integration to enable standard Avaya Experience Portal audio processing with video.

Orchestration Designer provides a set of media extensions to the standard Orchestration Designer tools. You can use these media elements to create Interactive Voice and Video Response (IVVR) applications. You can use the IVVR applications in:

- Call centers or help desks
- Travel reservations
- Banking applications

Note:

You can use the media elements and resources only in speech applications.

Media includes audio, video, text, and static image files. The Orchestration Designer speech project structure supports these media elements with additional project resources. The additional palette items and settings are an enhancement to the existing audio prompts editor.

Media pages are the basic containers of media content within a prompt. You can include existing audio elements and new media elements within the same prompt to synchronize the respective servers or players.

At run time, Orchestration Designer converts the media pages into Synchronized Multimedia Integration Language (SMIL) content. SMIL is a standard W3C approved, XML-like language. Use SMIL for simple authoring of audio-visual presentations.

When you integrate IVVR into Orchestration Designer, you can create media elements to use with Avaya Experience Portal. You can use this feature to:

- Reduce operating expenses by having fewer operators.
- Enhance customer satisfaction by providing visual context to cell phone users.
- Achieve quicker and more accurate transactions.

Media Page item

A media page contains media elements. It defines the SMIL content which is passed to the Avaya Voice Browser (AVB). You can use conditional statements on a media page to define a logic for processing. However, audio segments or SSML cannot exist on a media page.

Note:

The **Media Page** item is available in the Palette pane of the prompt level tab for speech applications only.

The following media elements are added to the existing audio prompt palette:

- Media type:
  - Media on page 764
  - SMIL Link on page 833
- Media items:
  - Region on page 814
  - Parallel on page 792
  - Sequence on page 831
  - Coordination Parallel on page 713
  - Coordination Sequence on page 714
  - Text item in a speech application on page 836
  - Text Block on page 839
  - Media on page 764

You must add a new media page to the prompt before adding media items. You can either generate the media content by using the palette items or link it to an external SMIL file.

Adding a media page to the prompt gives you a basic media item. You can then populate the media contents into the media page. The media page consists of two sections:

- **Head**: Definition area
- **Body**: Content

On a media page, the elements from the body use elements defined in the head. Currently, you can use only region elements in the head. Changes, such as renaming, deleting, and adding, to a region element have a dynamic effect on the body elements.

The regions are valid only within the media page. The media page object maintains an internal list of regions and contains a list in memory uncommitted regions. The elements in the body can access this memory list when required.

Note:

All elements on a media page, for example, Region, Parallel, Sequence, Coordination Parallel, Coordination Sequence, Text, Text Block, and Media, are VXML 2.1 + IVVR tags. By default, Orchestration Designer sets the speech projects to VXML 2.1 compatibility. Therefore, if you use these palette items, you must update the compatibility to VXML 2.1 + IVVR.

Media file management

Creating a media file

About this task

You can create a media file only in speech projects.

Procedure

1. On the **File** menu, click **New** > **Other**.

   The system displays the New wizard.

2. Double-click **Avaya OD Development**.
3. Click **Media File**, and then click **Next**.

   The system displays the New Media wizard.

4. On the Create a Media resource page, specify the name of the speech project, language, media type, and name for the media file, and then click **Next**.

   The system displays a page depending on the media type that you select on the Create a Media resource page.

5. Specify the information, and then click **Finish**.

New Media wizard: Create a Media resource page field descriptions

Name	Description
Available Projects	The speech project for which you want to create a media file. You must select a speech project.

Table continues…

Name	Description
Available Languages	The language for the media file. You must select a language for the media file.
Media type	The type of media file that you want to create. The options are **audio**, **video**, **image**, and **text**. You must select a media type.
Media file name	A unique and relevant name for the media file. You must specify a name for the media file

New Media wizard: Create an audio resource page field descriptions

If you select the **Media type** as **audio** on the **Create a Media resource** page, the system displays the Create an audio resource page. You can create a new audio file, use an existing audio file such as a phrase audio file, or use an external audio file.

Name	Description
Audio file	Specifies the source of the audio file. The following are the options: • **New** to create a new audio file. • **Existing** to use an audio file that is stored in the speech project resources. • **External** to use an audio file that is stored on another server. You must specify the source of the audio file.
Extension	If you are creating a new audio file, the appropriate audio format for the audio file. The options are **.wav** and **.au**. You must specify an extension for the audio file if you click **New** in **Audio file**.
File name	If you are creating a new audio file, a name for the audio file. If you are using an audio file that is stored in the speech project resources, browse to navigate to the audio file that you want to use. If you are using an external audio file, the name of the external audio file. This field is mandatory.

Table continues…

Name	Description
URL Base	If you are using an external audio file, the URL where the external audio file is stored. The default is **http://localhost:8080**. This field is mandatory if you click **External** in **Audio file**.

New Media wizard: Create a video resource page field descriptions

If you select the **Media type** as **video** on the Create a Media resource page, the system displays the **Create a video resource** page. You can use a video file that exists in the speech project resources or use an external video file. You cannot create a video file by using Orchestration Designer.

Name	Description
Video file	Specifies the source of the video file. The following are the options: • **Existing** to use a video file that is stored in the speech project resources. • **External** to use a video file that is stored on another server. You must specify the source of the video file.
File name	If you are using a video file that is stored in the speech project resources, browse to navigate to the video file that you want to use. If you are using an external video file, the name of the external video file. This field is mandatory.
URL Base	If you are using an external video file, the URL where the external video file is stored. The default is http://localhost:8080. This field is mandatory if you click **External** in **Video file**.

New Media wizard: Create an image resource page field descriptions

If you select the **Media type** as **image** on the Create a Media resource page, the system displays the Create an image resource page. You can use an image file that exists in the speech project

resources or use an external image file. You cannot create static images by using Orchestration Designer.

Name	Description
Image file	Specifies the source of the image file. The following are the options: • **Existing** to use an image file that is stored in the speech project resources. • **External** to use an image file that is stored on another server. You must specify the source of the image file.
File name	If you are using an image file that is stored in the speech project resources, browse to navigate to the image file that you want to use. If you are using an external image file, the name of the external image file. This field is mandatory.
URL Base	If you are using an external image file, the URL where the external image file is stored. The default is http://localhost:8080. This field is mandatory if you click **External** in **Image file**.

New Media wizard: Create a text resource page field descriptions

If you select the **Media type** as **text** on the Create a Media resource page, the system displays the Create a text resource page. You can use a text file that exists in the speech project resources or use an external text file.

Name	Description
Text file	Specifies the source of the text file. The following are the options: • **Existing** to use a text file that is stored in the speech project resources. • **External** to use a text file that is stored on another server. You must specify the source of the text file.

Table continues…

Name	Description
File name	If you are using a text file that is stored in the speech project resources, browse to navigate to the text file that you want to use. If you are using an external text file, the name of the external text file. This field is mandatory.
URL Base	If you are using an external text file, the URL where the external text file is stored. The default is http://localhost:8080. This field is mandatory if you click **External** in **Text file**.

Editing a media file

About this task

Each media type, that is, audio, video, image, and text, has its own editor. You can use the editors to:

- Define and view properties related to the media and file location.
- Setup an external location.
- Preview or edit the content of the file.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project that contains the `.media` file that you want to edit.
2. Double-click **<*language name*>** > **media**.
3. Right-click the `.media` file that you want to edit, and then click **Open With** > **Media Editor**.

   The system opens the media file in the appropriate media editor.
4. Modify the media file in the media editor.

Media Audio Editor field descriptions

You can use the Media Audio Editor to modify, record, and play audio files. You can also use this editor to define and view file properties, file locations, setup an external location, and preview and edit the contents of the audio files. You cannot record audio for external audio files.

The Media Audio Editor shows the settings that you configure on the Create an audio resource page of the New Media wizard. For more information, see Creating a media file on page 187. You can modify these settings by using the Media Audio Editor.

Note:

To record an audio file, you must plug in a microphone to your computer. Before you record an audio file, properly calibrate the automatic record levels for the microphone. Also, use headphones during recording rather than speakers connected to the computer. For more information about configuring Microsoft SAPI Speech for microphone input, see the *Getting started with Orchestration Designer* guide.

Name	Description
Media Information	
Comments	Comments related to the audio file. The comments that you type are displayed only in the Media Audio Editor.
Audio Information	
Local	The file type that you specify in the New Media wizard. **Local** to modify the file type and use an audio file that is stored in the speech project resources. You must select either **Local** or **External**.
External	The file type that you specify in the New Media wizard. **External** to modify the file type and use an audio file that is stored on another server. You must select either **Local** or **External**.
URL base	The URL of the external audio file that you specify in the New Media wizard. To modify the URL, type the URL where the external audio file is stored. This field is available and mandatory if you click **External**.
File Name	The file name that you specify in the New Media wizard. To select some other audio file that is stored in the speech project resources, click **Browse**. To specify some other external audio file, type the name of the external audio file. This field is mandatory.

Table continues…

Name	Description
Test	To verify the external audio file on the server on which the external audio file is stored. **Note:** This option is available only for external audio files.
Format	The audio format of the audio file that you select. This field is read only. **Note:** Orchestration Designer supports only 8KHz 8-bit mono format.
Play button 	To play the audio file. **Note:** This button is available only after you record or import an audio file, or after the system verifies the external audio file when you click **Test**.
Stop button 	To stop recording or playing an audio file. **Note:** This button is available only when you record or play an audio file.
Record button 	To start recording an audio file. If you record over an audio file that already contains recorded audio, the new recording overwrites the existing recording. **Note:** This button is unavailable for external audio files.

Media Video Editor field descriptions

You can use the Media Video Editor to modify video file information and play the video files at run time as a part of your media page. You cannot preview or record videos by using the Media Video Editor.

The Media Video Editor shows the settings that you configure on the Create a video resource page of the New Media wizard. For more information, see Creating a media file on page 187. You can modify these settings by using the Media Video Editor.

Name	Description
Media Information	
Comments	Comments related to the video file.
Video Information	
Local	The file type that you specify in the New Media wizard. **Local** to modify the file type and use a video file that is stored in the speech project resources. You must select either **Local** or **External**.
External	The file type that you specify in the New Media wizard. **External** to modify the file type and use a video file that is stored on another server. You must select either **Local** or **External**.
URL base	The URL of the external video file that you specify in the New Media wizard. To modify the URL, type the URL where the external video file is stored. This field is available and mandatory if you click **External**.
File Name	The file name that you specify in the New Media wizard. To select some other video file that is stored in the speech project resources, click **Browse**. To specify some other external video file, type the name of the external video file. This field is mandatory.

Media Image Editor field descriptions

You can use the Media Image Editor to modify image file information and preview image files that are stored in the speech project resources. You cannot preview external image files. You cannot capture images by using the Media Image Editor.

The Media Image Editor shows the settings that you configure on the Create an image resource page of the New Media wizard. For more information, see Creating a media file on page 187. You can modify these settings by using the Media Image Editor.

Name	Description
Media Information	

Table continues…

Name	Description
Comments	Comments related to the image file.
Image Information	
Local	The file type that you specify in the **New Media** wizard. **Local** to modify the file type and use an image file that is stored in the speech project resources. You must select either **Local** or **External**.
External	The file type that you specify in the **New Media** wizard. **External** to modify the file type and use an image file that is stored on another server. You must select either **Local** or **External**.
URL base	The URL of the external image file that you specify in the **New Media** wizard. To modify the URL, type the URL where the external image file is stored. This field is available and mandatory if you click **External**.
File Name	The file name that you specify in the **New Media** wizard. To select some other image file that is stored in the speech project resources, click **Browse**. To specify some other external image file, type the name of the external image file. This field is mandatory.

Media Text Editor field descriptions

You can use the Media Text Editor to modify text file information, and view and modify the content of the text files that are stored in the speech project resources. You cannot view and modify the content of external text files.

The Media Text Editor shows the settings that you configure on the Create a text resource page of the New Media wizard. For more information, see Creating a media file on page 187. You can modify these settings by using the Media Text Editor.

Name	Description
Media Information	
Comments	Comments related to the text file.

Table continues…

Name	Description
Text Information	
Local	The file type that you specify in the New Media wizard. **Local** to modify the file type and use a text file that is stored in the speech project resources. You must select either **Local** or **External**.
External	The file type that you specify in the New Media wizard. **External** to modify the file type and use a text file that is stored on another server. You must select either **Local** or **External**.
URL base	The URL of the external text file that you specify in the New Media wizard. To modify the URL, type the URL where the external text file is stored. This field is available and mandatory if you click **External**.
File Name	The file name that you specify in the New Media wizard. To select some other text file that is stored in the speech project resources, click **Browse**. To specify some other external text file, type the name of the external text file. This field is mandatory.
Text	The content of the text file that is stored in the speech project resources. In the **Text** field, you can edit the content of a text file that is stored in the speech project resources. You cannot view and edit the content of an external text file in the **Text** field.

Previewing a Media Page

About this task

The Media Preview feature uses a third-party tool, Ambulant player 2.1 embedded in an Eclipse view, to play and preview a media file. Though Ambulant player 2.1 consists of all the menu and toolbar options and controls, do not use the options in the player.

Before you preview a media file, start Tomcat. If you do not start Tomcat, the system automatically starts Tomcat when previewing the media file.

Previewing a media page allows you to play back any media page in a prompt during designing.

Procedure

1. Open the speech project.
2. In the call flow editor, double-click the node that contains the prompt in which the media file is added. For more information, see Prompt on page 797.
3. Right-click the prompt in which the media file is added and click **Edit Prompt: <*prompt name*>**.

   The system displays the **<*prompt name*>.prompt [<*language name*>]** prompt editor.
4. Right-click the media page in the prompt, and then click **Preview Media Page**. The system does the following:
   - Starts Tomcat if it is not already started.
   - Displays the Media Preview dialog box. You can use the Media Preview dialog box to hard code a value for any variables or conditions found on the media page. For more information, see Media Preview dialog box field descriptions on page 197.
   - Generates a SMIL file with a unique name and stores it in `<project>/data/<language>/prompts/cache`.
   - Invokes Ambulant player 2.1 and sends the SMIL file to Ambulant player 2.1 for play back.
5. After the play back, manually close the player. Otherwise, the player restarts the next time you preview a media page.

   You can make changes to the media file and repeat steps 1 and 2 to preview your changes.

Media Preview dialog box field descriptions

Name	Description
Name	The names of the variables set on the media page. This field is read only.
Value	The value of the variable field when you preview the media page for the first time. The next time you preview the media page, the system displays the set variable value. This field is mandatory.
Condition	The conditions set on the media page. This field is read only.
Evaluates to	Set to **true** or **false** as appropriate. This field is mandatory.

Chapter 15: Prompts

Prompts

One of the key elements that is used to create speech, message, data, and web applications with Orchestration Designer is the prompt. In a speech application, prompts guide a caller through a call flow. In a message application, prompts guide a customer through a message flow. In a data application, prompts guide a customer through a data flow. In a web application, prompts guide a customer through a web flow.

You can use prompts in:

- A speech application to play announcements to the caller, prompt the caller to provide a specific type of response, or respond to the request of a caller for information.
- An SMS channel message application to send SMS messages to the customers.
- An email channel message application to send email messages to the customers.
- A web application to collect and display information to the users.

Prompts are among the most important building blocks in speech, message, web and Interactive Voice Response (IVR) applications. A prompt consists of a set of one or more **prompt segments** that you can assemble to communicate information to a caller or a customer and ask the caller or the customer to respond to a request for input.

Note:

You can use variables for the prompts, so that prompts vary dynamically. If you copy a prompt variable from one speech application or message application or web application to another, then the system copies the variable, but does not copy the source to which the variable references.

Important:

You can use prompts only in speech, message, and web applications. You cannot use prompts in call control and data applications.

Types of prompt segments

Prompts can consist of a combination of prompt segments.

Types of prompt segments that are available in speech applications:

- **Phrase**: Phrases are prerecorded digital audio files that the system plays back. For more information, see Phrases on page 165 and Phrase Variable on page 794.
- **Audio variable**: Audio variables are special variables that analyze the content of the variable, parse the content, and play the content using special prerecorded phrase files. For more information, see Audio Variable on page 676.
- **Audio constant**: By using Audio constant, you can specify a constant value that the system converts into phrases. This segment type is similar to the Audio Variable, except that the value is a constant and not a variable. For more information, see Audio Constant on page 677.
- **Phrase variable**: Phrase variables are special variables for:
  - Playing project phrases where the variable value is a phrase name or an audio file URL.
  - Supporting failover TTS that is hard coded or stored in a variable.
  - Supporting collections where the variable is a collection of phrase names.
  - Supporting DTMF tones by converting the string value of a variable into a list of DTMF recordings.
- **Text-to-Speech (TTS)**: Text entered in the **TTS Text** field of the TTS prompt item that renders the content as synthesized speech by using TTS technology. For more information, see TTS on page 851.
- **Expression**: ECMAScript expression segments that you type. The system renders these into a generated VXML page, which is useful for playing certain VXML page variables in a prompt.

  To use a valid expression in a prompt, see the W3C VoiceXML 2.0 Recommendation. This is important because different expressions can be legal or illegal at different points in a VXML page.

Types of prompt segments that are available in message applications:

- **SMS**: Use the **SMS** item to assign an SMS file to the prompt. When the system executes the prompt, the system sends the SMS message that is contained in the SMS file to the text browser that is located on Avaya Experience Portal Media Processing Platform (MPP) for further processing and delivery. For more information, see SMS on page 833.

  Note:

  The **SMS** item is available only in an SMS channel message application.
- **Email**: Use the **Email** item to assign an email file to the prompt. When the system executes the prompt, the system sends the email message that is contained in the email file to the text browser that is located on Avaya Experience Portal Media Processing Platform (MPP) for further processing and delivery. For more information, see Email on page 725.

  Note:

  The **Email** item is available only in an email channel message application.

- **Text**: If you want to send a simple SMS message only from a single prompt of an SMS channel message application, then you can use the **Text** item instead of creating an SMS file.

  If you want to send a simple email message only from a single prompt of an email channel message application, then you can use the **Text** item instead of creating an email file.

  For more information, see Text on page 835.

Types of prompt segments that are available in web applications:

- **Web Element**: A **Web Element** is used to display input values of different types such as map, picture, and text. For more information, see Web Element on page 838.
- **Click To Call**: Using the **Click To Call** button, you can call on a constant number or on a number that is dynamically determined at runtime. For more information, see Click To Call on page 865.
- **HTML Text**: You can use a **HTML Text** item to display any text on your web form in the way it is styled. For more information, see HTML Text on page 837.
- **Loop Variable Collection**: You can use a **Loop Variable Collection** item to loop through the collection in a variable. For more information, see Loop Variable Collection on page 871.

Types of prompt segments that are available in speech, message, and web applications:

- **Text Variable**: In prompts for speech applications, text variables are variables that contain text which the system can render as synthesized speech by using TTS technology.

  In prompts for SMS channel message applications, you can use text variable to send variable text in the outbound SMS messages.

  In prompts for email channel message applications, you can use text variable to send variable text in the outbound email messages.

  In prompts for web applications, you can use text variable to display the value of a variable as plain text. You can wrap the **HTML Text** item around this item to add style to the text. For example,

  ```
  HTML Text <div style =”font-weight:bold”>
  Text VariablefirstName
  HTML Text</div>
  ```

  For more information, see Text Variable on page 840.

- **If**: Use the **If** condition item to select and execute different prompt or data flow segments based on the condition being tested. The **If** item works in conjunction with the **Else** item to test conditions and return a result. You can use the **If** item to compare the value of a selected variable, the **Left Variable** or **Left Variable Field** with:

  - A condition
  - A constant
  - The value of another variable or variable field

  Then, based on the result of the comparison, the system responds according to what is assigned to the **If** item, or continues further. For more information, see If on page 739 and Condition on page 699.

- **Else**: Use the **Else** condition item to select and execute different prompt or data segments based on the condition being tested. The **Else** item works in conjunction with the **If** item to test conditions. The **Else** item defines a default for cases where conditions of an **If** item are not met.

  You can use the **Else** item in the call flow editor and message flow editor to conditionally execute different operations based on the evaluation of logical statements. The **Else** item works in conjunction with an **If** item to execute operations when the “If” condition evaluates to a “false” value. For more information, see Else on page 722 and Condition on page 699.

- **Else If**: Use the **Else If** item to build complex If/Else logic. You can use the **Else If** condition item to select and execute different prompt or data flow segments based on various conditions. The **Else If** item introduces a new “nested if” condition. If the “If” condition fails, the **Else If** item makes it possible to compare the value of a selected variable, the **Left Variable** or **Left Variable Field** with:

  - A condition
  - A constant
  - The value of another variable or variable field

  Based on the result of the comparison, the system responds according to what is assigned to the **Else If** item, or continues further. For more information, see Else If on page 723 and Condition on page 699.

- **AND**: Use the **AND** Boolean operator to evaluate if all of its child expressions are true. This item sets a compound condition that evaluates to true only if all of its child expressions are true. This item is used as a parent item and can be nested under **If** or any other Boolean operator. This item connects all its child expressions logically. Its child expressions can be either a single expression or a compound condition. For more information, see AND on page 675.

- **OR**: Use the **OR** Boolean operator to evaluate if one of its child expressions is true. This item sets a compound condition that evaluates to true only if one of its child expressions is true. The item is used as a parent item and can be nested under **If** or any other Boolean operator. This item connects all its child expressions logically. Its child expressions can be either a single expression or a compound condition. For more information, see OR on page 790.

- **NOT**: Use the **NOT** Boolean operator to evaluate the opposite of what its child expressions or compound conditions evaluate. This item creates a condition that evaluates to the opposite of what its child expressions or compound conditions evaluate. This item is used as a parent item and can be nested under **If** or any other Boolean operator. It can have only one child expression which can be either a single expression or a compound condition. For more information, see NOT on page 772.

- **Expression**: Use the **Expression** Boolean operator to create Boolean expressions under the **AND**, **OR**, and **NOT** Boolean operators. This item creates a Boolean expression that contains two operands with a comparison operator in between. This item evaluates to either true or false. This item is used as a child of the AND, OR, and NOT Boolean operators to create a compound condition. For more information, see Expression on page 728.

Prompt controls

In addition to selecting prompt segments from among the available types, you can also select different prompt controls to monitor different aspects of a prompt that is in use.

Example prompt controls in a speech application:

- Use prompt levels to vary the prompt that is played to the caller, based on how many times the prompt is repeated. For more information, see Prompt levels in speech applications on page 202.
- Determine the order in which you want the prompt levels to play back. For more information, see Play order for prompt levels in speech applications on page 203.
- Use condition settings to control the specific prompt segments that you want to play and specify the circumstances under which you want to play the prompt segments. For more information, see Conditions in prompts on page 204.
- Control the rendering of TTS by using SSML tags. For more information, see SSML controls in prompts for speech applications on page 205.
- Determine whether and how to use the barge-in feature with the prompt. For more information, see Prompt File Editor: Prompt Main tab field descriptions on page 214.

Example prompt controls in a message and a web application:

- Use condition settings to control the specific prompt segments that you want to execute and specify the circumstances under which you want to execute the prompt segments. For more information, see Conditions in prompts on page 204.

Prompt levels in speech applications

Note:

You can use prompt levels only in speech applications.

When a caller does not respond to a prompt at all or the ASR server does not recognize the response, the system usually repeats the prompt. You can also configure the system to repeat the prompt content and prompt the caller second time by using a different version of the first prompt.

For example, a speech application has a prompt that says:

“Please say your 7-digit account number. The account number is located in the upper-right corner of your bill.”

If the caller says the account number too quickly for the ASR server to recognize, you can set the prompt to return a “No Match” event and play the following prompt:

“I’m sorry, I did not understand you.”

At this point, the system repeats the first prompt, but instead of the full-length version of the first prompt, you can set a second, shorter version. For example:

“What is your 7-digit account number?”

If the system still cannot recognize what the caller says, it can play a second version of the “No Match” prompt. For example:

“I’m sorry. I still did not understand you.”

You can set a third version of the main prompt. For example:

“Please speak slowly and clearly, and say all 7 digits of your account number.”

To use different versions of a prompt, you can designate each version as a different level of the prompt. The system automatically creates the first level of a prompt when the prompt is created. You must add the additional levels of the prompt. For example, to use three different versions of a prompt, you must add two new prompt levels within the **Prompt Main** tab of the prompt file editor. For more information, see Prompt File Editor: Prompt Main tab field descriptions on page 214.

The system displays each prompt level on a separate tab in the prompt file editor. When you add a level, the system appends a new tab to the prompt file editor. The system uses the prompt level number to name the tab. For example, 1st, 2nd, 3rd, and so on, up to 25.

The definition and settings for each prompt level are independent of all other prompt levels. That is, each level of the prompt uses its own segments, conditions, and SSML controls, regardless of any other level of prompt. For more information about defining prompts in levels, see Prompt File Editor: Level tabs on page 217.

Play order for prompt levels in speech applications

Note:

You can use prompt levels only in speech applications.

In an Orchestration Designer speech application, you can control the order in which different versions of prompts play. Prompt levels facilitate different versions of a prompt when a prompt is repeated for a caller. For more information, see Prompt levels in speech applications on page 202.

Depending on the location of the prompt, the **Play Order** settings affect the selection of prompt levels:

- For a prompt which is at the topmost level of a node, the only **Play Order** setting that affects the order or selection of the prompt level is **Random**. In this case, each time the call flow enters the node, the system randomly selects the prompt level.

  The other **Play Order** settings have no effect on this prompt.
- For a prompt which is at a level other than the topmost level within a node, the play order setting that you select in the **Play Order** field determines how to use the prompt level.
  - **Standard**: This setting uses the number value of the level to determine when to play the prompt. When the system reaches the highest numbered level, it uses the highest numbered level for all the repeats that follow.

    For example, you want the system to use the first level of a prompt twice before using the second version of the prompt. In other words, you want the second version of the prompt to play only at the third repetition of the prompt.

In this case, add one new level to the prompt. In the **Prompt Count** dialog box, in the **Please enter the attempt count** field, type `3` instead of typing `2`. The system assigns the name **Third** to the level tab of this level.

At run time, the system plays the **First** prompt level twice before it plays the **Third** prompt level.

- **First**: This setting plays only the **First** or the lowest numbered prompt level, no matter how many times the system repeats the prompt. The system ignores all prompt levels other than **First** level. If there is no **First** level, the system plays the lowest numbered prompt level.
- **Random**: This setting randomly selects the prompt level to play. Hence, the order in which the system plays the prompt levels is not determined.
- **Sequential**: This setting plays each prompt level one time, in order from the lowest to the highest numbered prompt level. When the system reaches the highest numbered prompt level, it uses the highest numbered prompt level for all the repeats that follow.

  For example, if there are three prompt levels numbered **First**, **Fourth**, and **Fifth**, the system plays the **First** level on the first pass, the **Fourth** level on the second pass, and the **Fifth** level on the third and all the passes that follow.

Example on prompt level play order

The following example describes how the **Play Order** settings work. For example, there are four prompt levels numbered **First**, **Third**, **Fifth**, and **Sixth**, and you want the prompt to repeat indefinitely. The following table shows the order in which these four prompt levels are played for each **Play Order** setting:

Play Order	Order in which the prompt levels are played
Standard	**First**, **First**, **Third**, **Third**, **Fifth**, **Sixth**, **Sixth**, **Sixth**, **Sixth**...
First	**First**, **First**, **First**, **First**, **First**, **First**...
Random[1].	**Fifth**, **Third**, **Sixth**, **Third**, **First**, **First**, **First**, **Fifth**, **Sixth**...
Sequential	**First**, **Third**, **Fifth**, **Sixth**, **Sixth**, **Sixth**, **Sixth**...
1. The order shown here is an example of the Random setting.	

Conditions in prompts

You can use **Conditions** to change a prompt depending on the situation.

For example, if you create a speech application that plays a different greeting to the caller based on business hours, you can use the built-in date and time variables to return the current day and time.

To test if the call is coming into the system during business hours, use an **If** item. If the call is during business hours, the system plays the prompt on that branch of the condition tree. Otherwise, the system plays the default prompt. If needed, use the nested **If-Else** conditions.

For more information about using **Condition** items in prompts, see:

- If on page 739
- Else on page 722

SSML controls in prompts for speech applications

Note:

You can use SSML controls only in the prompts that are created for speech applications.

For TTS-based prompt segments, **Text Variable** items, and **TTS** items, use the SSML controls to refine the way in which the TTS engine renders the text.

Note:

For these controls to work, the TTS engine must be SSML compliant. Orchestration Designer uses the Microsoft SAPI Speech TTS engine, which is *not* SSML-compliant. As a result, these controls are ignored during testing.

You can use the SSML controls to enable the following functionalities in prompts for speech applications:

- Insert or eliminate breaks, or pauses, in the flow of the speech. For more information, see Break on page 682.
- Increase or decrease the emphasis on words or phrases. For more information, see Emphasis on page 726.
- Use markers to detect barge-in during TTS play back. For more information, see Mark on page 764.
- Vary and control the pitch, rate, duration, and volume. For more information, see Prosody on page 804.
- Determine the way in which a word or phrase is interpreted. For more information, see Say As on page 821.
- Set the age and gender of the synthesized voice that is used for text rendering. For more information, see Voice on page 853.

Transitional audio prompts in speech applications

Note:

You can create and use transitional audio prompts only in speech projects.

Transitional audio prompts are similar to other prompts in many respects, but are specialized in significant ways. Usually, the system uses transitional audio prompts during transitions. For

example, the system retrieves data while the caller waits for a further response from the system, or during bridge call transfers when the caller waits for the call to be transferred.

Often transitional audio prompts consist of a music selection that the system plays when completing a task.

You can use transitional audio prompts in the following two situations:

- The **Property** form item. This usage is usually when the system requires more than a couple of seconds to process an action or a request by the caller. An audio cue, such as music, indicates to the caller that the system is processing data, and the caller should stay online.

  For more information, see Property on page 799.

- The **Bridged Transfer** application item. This usage is usually when the system requires more than a couple of seconds to complete a call transfer. The audio cue tells the caller that the system is working to complete the call transfer, and the caller should stay online.

  Note:

  Audio prompts in or after a transfer node do not play before the call is transferred. Setting the audio prompt as a transitional prompt when transitioning to the node that performs the transfer ensures that the prompt plays before the transfer happens.

For more information, see the **Transfer Audio** prompt property in Bridged Transfer on page 684.

Difference between transitional audio prompts and other prompts in speech applications

Transitional audio prompts differ from other prompts in a speech application in the following significant ways. In transitional audio prompts:

- You cannot edit the prompt attributes on the **Prompt Main** tab. However, you can edit the level properties, which are **Barge In**, **Timeout Representation**, **Constant Timeout**, **Time Unit**, **Variable**, and **Field** in the Properties view. For more information, see Prompt File Editor: Prompt Main tab field descriptions on page 214 and Prompt level property descriptions on page 213.
- You cannot add or delete prompt levels.
- The Palette pane for the tab of the **1st** level contains two types of prompt segments, which are, **Phrase** and **Phrase Variable**. Also, there are no **SSML** items.
- There can be only one phrase as the prompt, unless you use **Condition** items. Even if you use **Condition** items, you can use only one phrase per **If** or **Else** item.

  A phrase file that is in an audio file format, such as .wav or .au, does not have to be speech. Phrase file can also consist of a musical selection.

Prompt file editor

Prompt file editor in a speech application

Use the prompt file editor in a speech application to define and modify prompt files after you create the prompt files. Use the prompt file editor to define and modify prompts, from very simple announcements to very complex prompts involving variables, conditions, and logic.

You can use the prompt file editor in a speech application to:

- Set conditions that govern when different prompt segments are used. For more information, see Conditions in prompts on page 204.
- Add SSML controls to TTS segments. For more information, see SSML controls in prompts for speech applications on page 205.
- Define the behavior of transitional audio prompts. For more information, see Transitional audio prompts in speech applications on page 205.
- Set barge-in and play order attributes. Add, edit, and delete prompt levels. For more information, see Prompt File Editor: Prompt Main tab field descriptions on page 214.
- Assemble one or more prompt segments to make a complete prompt or announcement. For more information, see Building prompts on page 223.

For more information about accessing the prompt file editor and the associated functional tab in a speech application, see the following sections:

- Editing a prompt file on page 214
- Prompt File Editor: Level tabs on page 217
- Prompt File Editor: Prompt Main tab field descriptions on page 214

Tip:

To set the text that is synthesized for a TTS segment in the prompt file editor, select the TTS segment and set the **TTS Text** property in the Properties view. Alternatively, click twice slowly the TTS segment in the prompt file editor and edit the text.

Prompt file editor in a message application

Use the prompt file editor in a message application to define and modify prompt files after you create the prompt files. Use the prompt file editor to define and modify prompts, from very simple outbound messages to very complex prompts involving variables, conditions, and logic.

You can use the prompt file editor in a message application to:

- Set conditions that govern when different prompt segments are used. For more information, see Conditions in prompts on page 204.
- Assemble one or more prompt segments to make a complete prompt. For more information, see Building prompts on page 223.

For more information about accessing the prompt file editor and the associated functional tab in a message application, see the following sections:

- Editing a prompt file on page 214
- Prompt File Editor: Prompt Main tab field descriptions on page 214
- Prompt File Editor: Prompt Contents tab on page 217

Prompt file editor in a web application

Use the prompt file editor in a web application to define and modify prompt files after you create the prompt files. Use the prompt file editor to define and modify prompts, from very simple web applications to very complex prompts involving variables, conditions, and logic.

You can use the prompt file editor in a web application to:

- Set conditions that govern when different prompt segments are used. For more information, see Conditions in prompts on page 204.
- Assemble one or more prompt segments to make a complete prompt. For more information, see Building prompts on page 223.

For more information about accessing the prompt file editor and the associated functional tab in a web application, see the following sections:

- Editing a prompt file on page 214
- Prompt File Editor: Prompt Main tab field descriptions on page 214
- Prompt File Editor: Prompt Contents tab on page 217

Default timeout for prompts in Orchestration Designer speech projects

The default prompt timeout determines the number of seconds or milliseconds that the speech application must wait for a response from the caller after the speech application finishes playing the prompt and before the speech application generates a `No Input` event.

You can set a default timeout for prompts at the following levels:

- For the new prompts in all Orchestration Designer speech projects.

- For the new as well as existing prompts in a single speech project.
- For a single prompt in a speech project.

To set a default timeout for the new prompts in all speech projects, use the Preferences dialog box. The system uses the default timeout for the new prompts that you create in all the speech projects.

To set a default timeout for the prompts that you create in a single speech project, use the Properties for <project name> dialog box. You can specify whether you want to use the default timeout only for the new prompts that you create in the speech project or the new as well as the existing prompts in the speech project.

If you set the default timeout for prompts in the Properties for <project name> dialog box and create new prompts in the project, then the timeout that you specify in the Properties for <project name> dialog box takes precedence over the timeout that you specify in the Preferences dialog box.

To set a default timeout for a single prompt within a speech project, use the **Prompt Main** tab of the prompt file editor in the speech project.

Prompt management

Creating a prompt file

Procedure

1. On the **File** menu, click **New** > **Other**.

   The system displays the New wizard.

2. Double-click **Avaya OD Development**.
3. Click **Prompt File**, and then click **Next**.
4. In the New Prompt wizard, on the Create a Prompt page, specify the information to create a new prompt.

Next steps

After creating a prompt file, use the prompt file editor to assemble prompt segments or to set other properties for the prompt file. For more information, see Prompt file editor in a speech application on page 207 and Prompt file editor in a message application on page 207.

New Prompt wizard: Create a Prompt page field descriptions

Name	Description
Available Projects	The speech project or message flow project for which you want to create a prompt. You must select a project.
File Name	A name for the prompt file. Type a name that identifies the prompt and intuitively describes the content of the prompt file. For example, you can name a prompt that asks the caller for an account number as **AcctNmbr**. The system automatically appends the **.prompt** extension to the file name when you create the prompt file. For transitional audio prompts in speech applications, the system appends the **.audioprompt** extension. **Note:** You can create a transitional audio prompt only in speech applications. Prompt file names must follow Java naming conventions. For more information, see Conventions for naming Java components on page 59. If you do not specify a name for the prompt file, the system assigns a generic name to the prompt file.

Table continues…

Name	Description
Transitional audio prompt (allows only a single phrase)	Check box to make the prompt a transitional audio prompt. You can use transitional audio prompts in: • **Property** form item For more information, see Property on page 799. • **Bridged Transfer** application item For more information, see the **Transfer Audio** prompt entry in Bridged Transfer on page 684. **Note:** The **Transitional audio prompt (allows only a single phrase)** check box is available only if you create a prompt for a speech application. For more information about transitional audio prompts, see Transitional audio prompts in speech applications on page 205.

Creating a transitional audio prompt in a speech project

About this task

You must define a prompt as a transitional audio prompt when you create the prompt. You cannot convert a prompt that is not originally created as a transitional audio prompt to a transitional audio prompt.

Note:

You can create a transitional audio prompt only in a speech project.

Procedure

1. On the **File** menu, click **New** > **Other**.
2. In the New wizard, double-click **Avaya OD Development**.
3. Click **Prompt File**, and then click **Next**.
4. In the New Prompt wizard, on the Create a Prompt page, specify the information to create a new prompt.
5. Select the **Transitional audio prompt (allows only a single phrase)** check box to create a transitional audio prompt.
6. Click **Finish**.

Adding a prompt level to a prompt in a speech project

About this task

You can add prompt levels only to the prompts that are created for speech projects.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project that contains the .prompt or .audioprompt file in which you want to add a prompt level.
2. Double-click **<*language name*>** > **prompts**.
3. Right-click the `.prompt` or `.audioprompt` file in which you want add a prompt level, and then click **Open With** > **Prompt Editor**.

   The system opens the prompt file in the prompt file editor.
4. On the Prompt Main tab, click **Add**.

   The system displays the Prompt Count dialog box.
5. In the **Please enter the attempt count** field, type the attempt count number for the prompt level.

   By default, the system assigns the next incremental count number for the prompt level.
6. In the **Copy from existing prompt** field, click an existing prompt level from which you want to copy the prompt level contents to the prompt level that you create. The **Copy from existing prompt** field displays all the prompt levels that exist within the prompt. Alternatively, you can click **None** if you do not want to copy from any existing prompt level.
7. Click **OK**.

Editing the prompt level properties in a speech project

About this task

You can create and edit prompt levels only in the prompts that are created for speech projects.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project that contains the .prompt or .audioprompt file in which you want to edit the prompt level properties.
2. Double-click **<*language name*>** > **prompts**.
3. Double-click the `.prompt` or `.audioprompt` file in which you want edit the prompt level properties.
4. In the prompt file editor, click the **Prompt Main** tab.
5. In the **Prompt Levels** area, click the prompt level for which you want to edit the properties.

6. In the Properties view, edit the prompt level properties.

Prompt level property descriptions

Property name	Description
Barge In	Option to enable or disable barge in for the prompt level. The options are: • **true**: To enable barge in for the prompt level. • **false**: To disable barge in for the prompt level. **Note:** The default is **true**.
Timeout Representation	Option to specify whether you want to use a constant timeout value or variable timeout value for the prompt level. The options are: • **constant**: If you want to specify a constant timeout value for the prompt level. • **variable**: If you want to specify a variable timeout value for the prompt level.
Constant Timeout	The default constant timeout value for the prompt level. **Note:** The **Constant Timeout** field is available only if you select **constant** in the **Timeout Representation** field.
Time Unit	The unit of time for the default constant timeout value that you specify in the **Constant Timeout** field. The options are: • **seconds** • **milliseconds** **Note:** The **Time Unit** field is available only if you select **constant** in the **Timeout Representation** field.

Table continues…

Property name	Description
Timeout Variable	The variable that contains the default timeout value that you want to use for the prompt level. Ensure that the variable that you select contains only integer values. The system processes the timeout value contained in the variable as milliseconds. **Note:** The **Timeout Variable** field is available only if you select **variable** in the **Timeout Representation** field. If you select a complex variable in the **Timeout Variable** field, then you must select a complex variable field in the **Timeout Variable Field** field.
Timeout Variable Field	The complex variable field if you select a complex variable in the **Timeout Variable** field. **Note:** The **Timeout Variable Field** field is available only if you select a complex variable in the **Timeout Variable** field.

Editing a prompt file

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the project that contains the .prompt or .audioprompt file that you want to edit.

   Note:

   You can create and edit an .audioprompt file only in speech projects.

2. Double-click ***<language name>*** > **prompts**.
3. Right-click the `.prompt` or `.audioprompt` file that you want to edit, and then click **Open With** > **Prompt Editor**.
4. In the prompt file editor, modify the prompt file.

Prompt File Editor: Prompt Main tab field descriptions

In a speech application, use the Prompt Main tab of prompt file editor to do the following:

- Set barge-in behavior for the prompt.
- Determine the way in which prompt levels are displayed.

- Add, edit, and delete prompt levels. For more information, see Prompt levels in speech applications on page 202.

In message and web applications, the Prompt Main tab shows the name of the prompt file and the default language of the message or the web application.

The Prompt Main tab is located at the bottom of the workspace area of the prompt file editor.

The following table shows fields on the Prompt Main tab of prompt file editor of a speech application:

Name	Description
Attributes	
Display Name	The name of the prompt file that you specify when creating the prompt file. This field is read only.
Language	The default language that is assigned to the application. This field is read only.
Barge-in Type	The barge-in type that you want to use to trigger barge in. The following are the options: • **speech**: If you want any spoken response to trigger barge in and stop the prompt. This is the default setting. • **hotword**: If you want a complete match of an active grammar to trigger barge in and stop the prompt. For more information, see Editing the prompt level properties in a speech project on page 212. The **Barge-in Type** field is mandatory. **Note:** The **Barge-in Type** field is available only in the prompt file editor of a prompt in a speech application.

Table continues…

Name	Description
Play Order	The order in which you want the system to play the prompt levels. The following are the options: • **standard** • **first** • **random** • **sequential** The default play order is **standard**. For more information, see Play order for prompt levels in speech applications on page 203. The system repeats a prompt if the VXML platform, such as Voice Portal, Avaya Experience Portal, or MPS, determines that the prompt needs to be repeated because of a nomatch event or a timeout caused by a noinput event. The **Play Order** field is mandatory. **Note:** The **Play Order** field is available only in the prompt file editor of a prompt in a speech application.
Prompt Levels area: Displays all the levels that are defined for the prompt. **Note:** The **Prompt Levels** area and the options under the **Prompt Levels** area are available only in the prompt file editor of a prompt in a speech application.	
Add	To add a prompt level. For more information, see Adding a prompt level to a prompt in a speech project on page 212.
Edit	To edit the prompt level that you select in the **Prompt Levels** area. The system activates the tab of the prompt level. The maximum prompt level is 25. For more information, see Prompt levels in speech applications on page 202.
Delete	To delete the prompt level that you select in the **Prompt Levels** area.

Prompt File Editor: Level tabs

When using multiple versions of a prompt in a speech application, create a *prompt level* for each version that you want to use. Orchestration Designer displays the definition and settings for each prompt level in a separate tab in the prompt file editor. These tabs are located at the bottom of the workspace area of the prompt file editor.

Orchestration Designer displays prompt levels in numeric order, labeled **1st**, **2nd**, **3rd**, and so on, up to 25. For more information, see Prompt levels in speech applications on page 202.

Note:

The prompt level tabs are available only in the prompt file editor of a prompt in a speech application.

Use the level tabs to:

- Assemble prompt segments into prompts. For more information, see Building prompts on page 223.
- Use conditions to determine when various prompt segments must play. For more information, see Conditions in prompts on page 204.
- Use SSML controls to regulate rendering of TTS content. For more information, see SSML controls in prompts for speech applications on page 205.

Each level tab has its own Palette.

Tip:

To set the text that is synthesized for a TTS segment in the prompt file editor, select the TTS segment and set the **TTS Text** property in the Properties view. Alternatively, click twice slowly the TTS segment in the prompt file editor and edit the text.

Prompt File Editor: Prompt Contents tab

Note:

The Prompt Contents tab is available only in the prompt file editor of a prompt in a message application and a web application.

Use the Prompt Contents tab to:

- Assemble prompt segments into prompts. For more information, see Building prompts on page 223.
- Use conditions to determine when various prompt segments must execute. For more information, see Conditions in prompts on page 204.

Prompt file export and import

In Orchestration Designer, you can reuse prompts of one project in another project, provided that the two projects belong to the same channel. For example, both the projects are for the SMS channel.

This is useful when you use a common set of prompts in two or more applications, or when you have a reusable speech or message module that uses the same prompts as the parent application.

Exporting a prompts .Zip file

About this task

To export a prompts .zip file for reuse in more than one speech project or message flow project, use the Export Prompts wizard.

Important:

Be careful when exporting prompts files. Prompts rely on phrases, grammars, and other project resources that may not exist in the project where you import the exported prompts. If you import prompt files that rely on resources located in other projects, Orchestration Designer generates errors. Be sure that you either import the missing resources in the project where you import the prompt files, or redirect the imported prompts to make use of the resources from the other projects.

Procedure

1. On the File menu, click **Export**.

   The system displays the Export wizard.

2. Double-click **Avaya OD Development**.
3. Click **Prompts Zip File**, and then click **Next**.

   The system displays the Export Prompts wizard.

4. On the Specify Export Parameters page, specify the parameters to export the prompts .zip file.
5. Click **Finish**.

Export Prompts wizard: Specify Export Parameters page field descriptions

Name	Description
Available Projects	The speech project or message flow project from which you want to export the prompts.
Available Languages	The languages for which you want to export the prompts. The **Available Languages** pane shows only those languages in which prompts exist in the project that you select.
Zip file	The directory path and file name for the .zip file that you want to export.
Include directory structure	(Optional) Check box to include the directory path information in the .zip file.

Importing a prompts .Zip file

About this task

You must first export a prompts .zip file from a project, and then import the exported file into the project where you want to reuse the prompts.

Note:

You can import prompts in a project only if the project from which you export the prompts and the project in which you import the prompts belong to the same project type. For example, both the projects are speech projects.

If both the projects are message flow projects, then both the projects must belong to the same channel type. For example, both the message flow projects are for the SMS channel.

Important:

Be careful when importing prompts files. Prompts rely on phrases, grammars, and other project resources that may not exist in the project where you import the exported prompts. If you import prompt files that rely on resources located in other projects, Orchestration Designer generates errors. Be sure that you either import the missing resources in the project where you import the prompt files, or redirect the imported prompts to make use of the resources from the other projects.

Note:

You can use the following procedure only for the prompt files that are exported to a .zip file by using the Import Prompts wizard in Orchestration Designer. For information, see Exporting a prompts .Zip file on page 218.

Procedure

1. On the **File** menu, click **Import**.

   The system displays the Import wizard.
2. Double-click **Avaya OD Development**.
3. Click **Prompts Zip File**, and then click **Next**.
4. In the Import Prompts wizard, on the Specify Import Parameters page, specify the parameters to import the prompts .zip file.
5. Click **Finish**.

Import Prompts wizard: Specify Import Parameters page field descriptions

Name	Description
Prompt zip file	The directory path and name of the .zip file that you want to import.
Available Projects	The speech project or message flow project in which you want to import the .zip file.
Overwrite existing resources without warning	Check box to overwrite, without warning, any existing prompt files having the same name as that of the prompt files you import.

Importing a delimited prompt data text file in a speech project

About this task

Note:

Before you import a prompt data file, you must create a delimited prompt data file as per the specifications mentioned in Setting up the prompt data text file for import on page 221.

You can import a delimited prompt data text file only in a speech project.

Procedure

1. On the **File** menu, click **Import**.

   The system displays the Import wizard.
2. Double-click **Avaya OD Development**.
3. Click **Prompt Data**, and then click **Next**.

   The system displays the Import Prompt Data wizard.

4. On the Prompt Data Import page, specify the criteria to import the delimited prompt data text file.
5. Click **Next**.

   The system displays the Import Preview page.
6. Click **Preview** to preview the data.
7. Click **Finish**.

Setting up the prompt data text file for import

You can use the **Prompt Data** option in the Import wizard to import new prompt data from a delimited text file to update the existing prompts or add new prompts to your speech application project. This capability is helpful to create or update prompts that play pre-recorded audio.

Note:

The **Prompt Data** option does not support multiple prompt levels and multiple simultaneous languages.

You can import a delimited prompt data text file only in a speech project.

Before importing a prompt data file, create a delimited prompt data text file. The text file must have the following format for each row:

```
<prompt>,<phraseset>:<phrase>,<phraseset>:<phrase>, ... n phrases per prompt
```

and/or

```
<prompt>,<phraseset>:<phrase>,<standalone_phrase>, ... n phrases per prompt
```

Each prompt can be made up of *n* number of *phraseset:phrase* pairs or *standalone* phrases, or both. There is no limit to the number of phrases a single prompt can contain.

As a delimiter, you can use a tab or another consistently used character in your file that does not conflict with the data content. For example, a pipe symbol (|) or a comma.

Import Prompt Data wizard: Prompt Data Import page field descriptions

Note:

You can import a delimited prompt data text file only in a speech project.

Name	Description
From file	The directory path of the delimited prompt data file that you want to import.
Target Language	
Project	The speech project in which you want to import the delimited prompt data file.
Language	The language based on which you want to update the prompt data. For example, the delimited prompt data file that you want to import contains prompts defined in English language. If your speech project is multilingual and you select **English** in the **Language** field, then the system updates the prompts that are in English language in your speech project. The system adds new prompts from the delimited prompt data file to your speech project in English as well as in other languages that exist in your speech project, but populates the prompt contents only in the prompts that are in English language. The names of the prompts are the same in all the languages. To update and populate prompt contents in the prompts that are in other languages in your speech project, you must either manually edit the prompts that are in other languages, or import the prompt data in the respective languages. For more information, see Language implementation in Orchestration Designer on page 276.
Options	
Tab	If a tab is used as a delimiter in the delimited prompt data file.
Other	The delimiter that is used in the delimited prompt data file.

Building prompts

Before building a prompt

Before you build a prompt, create the required prompt segments.

For example, in a speech application, you want to create a prompt that reads the address of a caller from a variable and asks the caller whether the address is correct. You want the caller to

hear: “According to our records, you live at <*address record*>. Is this correct?” This prompt requires two phrase segments and one text variable segment:

- Phrase segment: “According to our records, you live at”
- Text variable segment: The address value returned by the database query, which is rendered by using TTS
- Phrase segment: “Is this correct?”

Before you build this prompt, you must use a database operation to read the address from the database. Then, you must assign the value returned by the database operation to a variable, which you must later assign to the text variable in the prompt file editor.

You must also create the required phrases. For information about creating phrases, see Custom phrases on page 167.

Building prompts

Before you build a prompt, create the required prompt segments.

To build a prompt, assemble one or more prompt segments in the order that you want them to execute. You can use any combination of prompt segment types. For more information about prompt segment types, see Types of prompt segments on page 198.

For more information about building a prompt, see the following sections:

- Before building a prompt on page 222
- Building a sample prompt for a speech application on page 223
- Building a sample prompt for an SMS channel message application on page 224
- Building a sample prompt for a web application on page 225

Building a sample prompt for a speech application

About this task

After you create the required prompt segments, you must assemble the prompt.

To build the sample prompt for a speech application as shown in Before building a prompt on page 222, see the following procedure.

Procedure

1. Click an item in the Palette pane of the prompt level tab.
2. In the prompt level tab workspace, click where you want to add the item.
3. Add the required prompt segments, conditions, and SSML controls so that the prompt works the way you want.
4. In the Palette pane of the prompt level tab, click **Phrase**.

5. Click in the prompt level tab workspace.

   The system adds a **Phrase** item to the workspace.
6. In the Properties view, assign the appropriate phrase file to the **Phrase** item.

   In this example, the phrase is: “According to our records, you live at”.
7. In the Palette pane of the prompt level tab, click **Text Variable**.
8. In the prompt level tab workspace, click below the **Phrase** item

   The system adds a **Text Variable** item below the **Phrase** item in the workspace.
9. In the Properties view, assign to the **Text Variable** item the variable that contains the address returned by the database operation.
10. In the Palette pane of the prompt level tab, click **Phrase**.
11. In the prompt level tab workspace, click below the **Text Variable** item.

    The system adds a **Phrase** item below the **Text Variable** item in the workspace.
12. In the Properties view, assign the appropriate phrase file to the **Phrase** item.

    In this example, the phrase is: “Is this correct?”.

Building a sample prompt for an SMS channel message application

About this task

After you create the required prompt segments, you must assemble the prompt.

The following is an example procedure to create a prompt that sends an SMS message.

Similarly, you can use the **Email** item in an email channel message application to build a prompt that sends an email message.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the SMS channel message flow project that contains the `.prompt` file in which you want to assemble the prompt.
2. Double-click **<*language name*>** > **prompts**.
3. Right-click the `.prompt` file in which you want to assemble the prompt, and then click **Open With** > **Prompt Editor**.
4. In the prompt file editor, click the Prompt Contents tab.
5. From the Palette pane of the Prompt Contents tab, drag an **SMS** item to the Prompt Contents tab workspace.
6. In the Properties view of the **SMS** item, assign the appropriate SMS file to the **SMS** item.

7. In the Prompt Contents tab workspace, add other prompt segments and conditions as required so that the prompt works the way you want.

Building a sample prompt for a web application

About this task

After you create the required prompt segments, you must assemble the prompt.

The following is an example procedure to create a prompt in a web application:

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the web project that contains the `.prompt` file in which you want to assemble the prompt.
2. Double-click **<*language name*>** > **prompts**.
3. Right-click the `.prompt` file in which you want to assemble the prompt, and then click **Open With** > **Prompt Editor**.
4. In the prompt file editor, click the Prompt Contents tab.
5. From the Palette pane of the Prompt Contents tab, drag a **Web Element** item to the Prompt Contents tab of the workspace.
6. In the Properties view of the **Web Element** item, select the variable name that you want to display in the **Variable** property.

   Note:

   By default, the **Type** property is set to “Text”. Ensure that the type matches the content of the variable. For example, if the variable contains map coordinates, then set the **Type** property to “Map” to display a map on the screen.
7. In the Prompt Contents tab of the workspace, add other prompt segments and conditions as required so that the prompt works the way you want.

Chapter 16: Variables

Variables

You can use variables in Orchestration Designer applications just like in any other application. Because speech applications, message, data, and web applications are specialized, there are special types and uses of variables in Orchestration Designer.

Important:

You can use variables that are described in this section only in speech applications, message, data, and web applications. Call control application variables are incorporated within a CCXML file and are not similar to speech, message, and data application variables.

Note:

Be sure that you are familiar with the general concept and use of variables in programming. This documentation does not cover the basic concepts and use of variables.

You can use variables in Orchestration Designer to:

- Store data for processing or manipulation.
- Store information about audio file locations in speech applications.
- Pass data to and from other modules or applications.
- Receive data from or pass data to other connected systems. For example, Avaya Interaction Center (IC) or Application Enablement Services (AES) systems in speech applications.

  Note:

  You can set variables to null.

  Tip:

  When naming variables, do not use a "__VP" prefix which is reserved for VP 4.1 and later deployments. If the speech, or message, or web application contains two or more variables with the same name, the system might execute the application incorrectly or might report incorrectly.

Types of variables in Orchestration Designer

Orchestration Designer provides the following different types of variables:

- **Simple variable**: Simple variable holds only one variable value at a time. This is the most widely known type of variable in programming.
- **Complex variable**: Complex variable holds multiple values within a single variable. To hold multiple values within a single variable, Orchestration Designer uses variable fields which are assigned to a complex variable. To make the complex variable functional, at least one field must be assigned to a complex variable. When you create a complex variable, Orchestration Designer automatically creates a field and assigns it to the complex variable.

  Note:

  In Orchestration Designer, variables can also hold multiple values by using *collections*. This happens when a variable, either simple or complex, returns more than one value for the variable or variable field.

  For example:

  - An **N Best** property that causes an operation to return several values for a single variable.
  - A database query that returns more than one record.
  - A Web service operation that returns an array.

Different ways of creating variables

The following are the four basic ways of creating variables:

- **System variables**: System variables are built-in variables and are automatically created in each speech project, message flow, and data flow project that you create in Orchestration Designer. System variables are read-only variables. You can use system variables for common operations in speech applications, message applications, and data applications.

  For a complete list of system variables and information about using system variables, see System variables on page 563.
- **Generated variables**: For some Palette options, Orchestration Designer automatically generates variables. Most of these variables and their associated fields are read only, but for some variables, you can add custom fields.

  Orchestration Designer-generated variables can be either simple or complex variables, depending on what Palette options the variables are associated with. For a list of the Palette options that automatically create variables, see Variables generated by Orchestration Designer on page 240.
- **Custom variables**: You can create simple and complex variables by using the variable editor.

Note:

To make a complex variable functional, at least one variable field must be assigned to the complex variable. When you create a complex variable, the system automatically creates a variable field and assigns it to the complex variable.

Note:

To copy a variable from one variable editor to another, use the Control+C, Control+V option. You cannot drag and drop variables between two variable editors.

You can copy variables from one project to another project only if both the projects belong to the same channel. For example, both the projects belong to the SMS channel.

For more information about creating variables, see Creating a variable on page 231.

- **Local variables**: You can create local simple and complex variables within a **Data** node and use them within that **Data** node. For example, you can create a local variable to store the size of an array, which you can use to do certain operations within that **Data** node. This reduces the overhead to store global, application-wide variables.

  You can view the local variables in the item properties along with the project variables. Orchestration Designer appends the letter “L” to the name of the local variable. Local variables support all the functions that project variables support at the **Data** node level.

  For more information about creating local variables, see Creating a local variable on page 232.

Passing variable values

Variables in Orchestration Designer are always local to an application or project. For example, if you are using an Orchestration Designer speech project as a module in another application, the application cannot access the variables in that module. Any information that passes between the module and the application must be passed as input or output parameters.

Pass variable values between applications and modules

You can pass variable values between an Orchestration Designer:

- Speech application and a reusable speech module.
- Message application and a reusable message application module.
- Web application and a reusable web application module.
- Speech, message, data, or web application and a reusable data application module.

The reusable speech module can be another speech project that is created by using Orchestration Designer. Alternatively, it can be a reusable speech module created by using a different tool, such as the OSDMs created by using Nuance software.

The reusable message application module must be a message flow project that is created by using Orchestration Designer, and must belong to the same channel as that of the parent application. For example, the parent application and the reusable message application module belong to the SMS channel.

The reusable web module can be another web project that is created by using Orchestration Designer. Alternatively, it can be a reusable web module created by using a different tool, such as the Nuance software.

To pass values between a parent Orchestration Designer application and module, you must use the **Input Parameter** and **Output Parameter** items. For more information about using these items to pass variable values, see:

- Input Parameter on page 745
- Output Parameter on page 791

Passing variable values between speech applications and VoiceXML objects

In Orchestration Designer, you can use VoiceXML objects to extend the capabilities of the speech application beyond the normal capabilities of VoiceXML. These objects consist of codes that you write or obtain from someone.

To pass values between an Orchestration Designer speech application and a VoiceXML object, you must use the **Object**, **Object Input**, and **Object Output** items. For more information about passing variable values between Orchestration Designer speech projects and VoiceXML objects, see:

- Object on page 772
- Object Input on page 773
- Object Output on page 774

Passing variable values between speech applications and IC systems

Orchestration Designer has a built-in connector that you can use to establish contact with and pass values between an Avaya IVR system and an Avaya Interaction Center (IC) system. With this connector, you can use the strengths of both the systems within a single speech application.

See IC connector on page 622.

Passing variable values between speech applications and AES systems

Orchestration Designer has a built-in connector that you can use to establish contact with and pass values to and from an Application Enablement Services (AES) system. You can use this connector to extend the capabilities of Orchestration Designer speech applications beyond the limitations of VoiceXML version 2.0-compliant applications.

Passing variable values between speech applications and call control applications

You can use the namelist in the **dialogprepare** or **dialogstart** tag to pass values from a CCXML application to a speech application. In the speech application, use the Capture Expression on page 690 item to store the values into an Orchestration Designer variable.

If the namelist in CCXML is *namelist*="myvar myothervar" />, then the expression in the speech application to access the value that is passed is **session.connection.ccxml.values.myvar** and **session.connection.ccxml.values.myothervar**.

To return data from the speech application to the CCXML application, add an Output Parameter on page 791 item to the Return node on page 662. The returned values are shown in CCXML in **dialog.exit** event **event.values.***.

For the **dialog.exit** event, if **dialogresult** and **numbertocall return** in the transition, you can access the return values as follows:

```
<transition event="dialog.exit" state="in_dialog">
 <log expr="' dialogresult : ' + event$.values.dialogresult"/>
 <log expr="' numbertocall : ' + event$.values.numbertocall"/> ... ... ...
```

Note:

You can pass only simple strings back and forth. You cannot pass objects.

Variable management

When you access the variable editor of a speech application or a message application, the system displays all the variables that currently exist in the speech application or message application. Additional editing functions are also available.

Use the variable editor to create, define, and modify variables. You can also use the variable editor to view the variables that exist in the current project.

You can create variables by using the **Simple Variable**, **Complex Variable**, and **Field** items that are available in the Palette pane of the variable editor. For more information, see Creating a variable on page 231.

Creating a variable

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech, or message, or web flow project directory in which you want to create a variable.
2. Double-click the **flow** subdirectory.
3. Double-click **project.variables**.

   The system displays the variable editor in the main workspace.
4. From the Palette pane, drag **Simple Variable** or **Complex Variable** to the main workspace area.

   The system adds the variable to the main workspace and automatically assigns it a temporary default name.
5. **(Optional)** In the Properties view, in the **Name** field, type a name for the variable.

   Caution:

   Do not use double-byte or extended ASCII characters or Java reserved words such as Try, Catch, and Switch when naming variables.

   Tip:

   When naming variables, do not use a "__VP" prefix which is reserved for VP 4.1 and later deployments. If the speech, or message, or web application contains two or more variables with the same name, the system might execute the application incorrectly or might report incorrectly.

   After you create the variable, you can set the variable properties in the Properties view. For more information, see Simple Variable on page 832 and Complex Variable on page 698.

   Note:

   To make a complex variable functional, at least one variable field must be assigned to the complex variable. When you create a complex variable, the system automatically creates a variable field and assigns it to the complex variable. To add a variable field to a complex variable, from the Palette pane of the variable editor, drag **Field** under the *<complex variable name>* node. For more information about variable field, see Field on page 733.

   Tip:

   To edit the name of a variable or a variable field, in the variable editor, click the variable or the variable field. In the Properties view, in **Name** field, type a name for the variable

or the variable field. Alternatively, in the variable editor, click twice slowly the variable or the variable field and edit the name.

Note:

You cannot edit system variables.

Creating a local variable

Procedure

1. In the call flow editor or message flow editor, double-click the **Data** node under which you want to create a local variable.
2. From the Palette pane, drag **Simple Variable** or **Complex Variable** under **Local Variables** in the data node editor.

   The system adds the variable under the Local Variables node in the data node editor and automatically assigns it a temporary default name.

   Note:

   If you are using a project that is created by using Dialog Designer 5.1 or earlier, drag **Local Variables** from the Palette pane to the data node editor.
3. **(Optional)** In the Properties view, in the **Name** field, type a name for the variable.

   Caution:

   Do not use double-byte or extended ASCII characters or Java reserved words such as Try, Catch, and Switch when naming variables.

   Tip:

   When naming variables, do not use a "__VP" prefix which is reserved for VP 4.1 and later deployments. If your speech application or message application contains two or more variables with the same name, the system might execute the application incorrectly or might report incorrectly.

After you create the variable, you can set the variable properties in the Properties view. For more information, see Simple Variable on page 832 and Complex Variable on page 698.

Note:

To make a complex variable functional, at least one variable field must be assigned to the complex variable. When you create a complex variable, the system automatically creates a variable field and assigns it to the complex variable. To add a variable field to a complex variable, from the Palette pane of the data node editor, drag **Variable Field** under the <*complex variable name*> node.

Deleting a variable

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech application or message application project directory from which you want to delete the variable.
2. Double-click the **flow** subdirectory.
3. Double-click **project.variables**. The system displays the variable editor in the main workspace.
4. Select the variable or the variable field that you want to delete.

   Tip:

   Use `Shift+click` to select multiple adjacent variables, or use `Control+click` to select multiple nonadjacent variables.

5. Press **Delete**.

   Note:

   To delete a local variable or a local variable field, open the data node editor from which you want to delete the local variable or the local variable field. Under **Local Variables**, right-click the local variable or the local variable field that you want to delete, and then click **Delete**.

Converting user-defined variables into Avaya Experience Portal configurable variables

In speech applications and message applications, you can convert user-defined variables into Avaya Experience Portal configurable variables. The configurable variables are administered through the Experience Portal Manager (EPM) interface. You can gain access to the configurable variable value and use the value in an Orchestration Designer application, but you cannot modify the configurable variable value from Orchestration Designer. You can administer the value of the configurable variable through the EPM interface. Every time the configurable variable value is updated, Orchestration Designer retrieves the configurable variable value from the EPM interface.

The advantage of using configurable variables is that you can deploy an Orchestration Designer speech application or message application that uses the configurable variables on different sites in an enterprise and centrally administer the configurable variables through the EPM interface. You can administratively change the functioning of the application to accommodate the local specifics. After you convert a user-defined variable into a configurable variable, you cannot reconvert the configurable variable into a user-defined variable.

Converting a user-defined variable into an Avaya Experience Portal configurable variable

About this task

Note:

The **Change Type to Configurable Variable** option is available only after you enable the **Voice Portal Configurable Application Variables** pluggable data connector. For more information, see Enabling project pluggable data connectors on page 503.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project directory or the message flow project directory in which you want to convert a user-defined variable into a configurable variable.
2. Double-click the **flow** subdirectory.
3. Double-click **project.variables**.
4. In the variable editor, right-click the user-defined variable that you want to convert into an Avaya Experience Portal configurable variable, and then click **Change Type to Configurable Variable**.

Result

After you convert the user-defined variable into an Avaya Experience Portal configurable variable, the system displays the value of the user-defined variable in the **Default Value** property of the Avaya Experience Portal configurable variable.

Different ways of employing variables in applications

You can use variables in the Orchestration Designer speech applications in the following ways:

- Use the palette options and items in the nodes to employ variables. For more information, see Employ variables using node items on page 235.
- Use the prompt file editor to employ audio or text variables as prompt segments within prompts. For more information, see Employ variables in a speech application using the prompt file editor on page 235.
- Employ call session variables programmatically. For more information, see Access a call, or message, or web session variables programmatically on page 237.

You can use variables in the Orchestration Designer message applications in the following ways:

- Use the palette options and items in the nodes to employ variables. For more information, see Employ variables using node items on page 235.

- Use the prompt file editor to employ text variables as prompt segments within prompts. For more information, see Employ variables in a message application using the prompt file editor on page 236.
- Employ message session variables programmatically. For more information, see Access a call, or message, or web session variables programmatically on page 237.

Employ variables using node items

In Orchestration Designer speech projects and message projects, many nodes have palette options in which you can use variables.

For example,

- In a speech project, you can:
  - Use the **Operation** palette option in the **Data** node to assign values to variables, or to perform calculations or other manipulations on variable values.
  - Use a database operation to retrieve a telephone number from a database, and then assign the telephone number to a variable. You can then use the variable as the transfer destination variable in a call transfer.
  - Use a Web service operation to receive data in a variable from a Web service.
  - Include application performance and call information data in variables within the **Report** item. You can send this data to an IVR system such as Avaya Experience Portal. In Avaya Experience Portal, you can use the data to generate reports and analyze application data.
- In a message application, you can:
  - Use the **Operation** palette option in the **Data** node to assign values to variables, or to perform calculations or other manipulations on variable values.
  - Use a database operation to receive data in a variable from a database.
  - Use a Web service operation to receive data in a variable from a Web service.

Other nodes have options in which you can use system variables to set branching conditions. You can also use system variables to pass variable values to other reusable application modules.

In each case, the variable must exist before you can use it in a node item. If the variable does not exist, you must create a custom variable.

For more information about the palette nodes and items in which you can employ variables and variable fields, see Nodes and Palette options on page 633.

Employ variables in a speech application using the prompt file editor

The prompt file editor in a speech application offers two ways to render or play a variable value depending on the type of variable being rendered.

For more information, see the following sections:

- About rendering audio variables in a speech application on page 236
- About rendering text variables in a speech application on page 236

About rendering audio variables in a speech application

In a speech application, audio variables render variable information into audio outputs by parsing the variable content, and then concatenating the prerecorded audio files to play back the content.

To create a variable or variable field that you can use as an audio variable, ensure that the value of the variable matches the format that the localization bundle expects. For example, all localization bundles expect variable data that is to be rendered as a date to be in the *YYYYMMDD* or *YYYY-MM-DD* format, where:

- *YYYY* is the four-digit year
- *MM* is the two-digit month
- *DD* is the two-digit day

For more information, see the following sections:

- Variable formats in localization bundles on page 583
- Audio Variable on page 676

About rendering text variables in a speech application

In a speech application, text variables use text-to-speech synthesis technology to render variable information of various formats as audible output.

You can use most variables as text variables. However, you must exercise caution when selecting a variable that is to be rendered by using TTS. Orchestration Designer takes the value of the variable and passes the value in a string format to the TTS server for rendering. Therefore, if you use a variable whose content consists of a Java object reference, you might get unexpected results when the TTS server renders the content as synthesized speech.

Be aware that not all TTS servers render the same variable value in the same way. For example, one TTS server might render the date 05-25-2005 as “May twenty-fifth, two thousand five.” A different TTS server might render the same date as “oh five dash twenty-five dash two thousand and five.” To use text variables effectively, you must know the formats and functions of the TTS server.

For more information, see Text Variable on page 840.

Employ variables in a message application using the prompt file editor

For more information about using the prompt file editor in a message application to render a variable value, see About rendering text variables in a message application on page 237.

About rendering text variables in a message application

Use the **Text Variable** item in an SMS channel message application to send the content of the variable in the form of an SMS message.

Use the **Text Variable** item in an email channel message application to send the content of the variable in the form of an email message.

For more information, see Text item in an SMS channel message application on page 836 and Text item in an email channel message application on page 837.

Employ variables in a web application using the prompt file editor

For more information about using the prompt file editor in a web application to render a variable value, see About rendering text variables in a web application on page 237.

About rendering text variables in a web application

Use the **Text Variable** item in a web application to display the value of a variable in plain text.

For more information, see Text Variable item in a web application on page 843.

Access a call, or message, or web session variables programmatically

You can access or update call, or message, or web session variables, simple, complex, or system, in any of the generated Java codes or developer-defined codes that have access to a **com.avaya.sce.runtimecommon.SCESession** object. The **SCESession** object stores Orchestration Designer-generated variables and can also store any variables defined by the application developer.

In a typical Orchestration Designer project, you can programmatically access session variables in one or two places, through the Servlet Node or other nodes. For example, Form, Menu, or Data node.

For information about the ways in which you can programmatically access call, or message, or web session variables within Orchestration Designer, see:

- General approach to access call, or message, or web session variables on page 237
- Get a Call, or message, or web session variable values on page 238
- Set a call, or message, or web session variable values on page 239
- About IProjectVariables on page 239

General approach to access call, or message, or web session variables

The general approach to gain access to call, or message, or web session variables, either simple or complex, is to add code to a node.

For a Servlet node, override the following method:

public void servletImplementation(SCESession mySession)

For an other node, for example, Form, Menu, or Data, override the following method:

public void requestBegin(SCESession mySession)

Get a Call, or message, or web session variable values

The method to get call, or message, or web session variables programmatically depends on the type of variable, that is, simple or complex.

For more information, see the following sections:

- Example code to get the value of a simple variable on page 238
- Example code to get the value of a complex variable on page 238

Refer to the *Orchestration Designer Programmer Reference* guide, available in the online Help, for the full set of API for accessing variables and their values.

Example code to get the value of a simple variable

The following code gets the value of the *weather* simple variable as a Java String object.

Note:

For this code, a simple variable *weather* must be defined in the project variable editor. For more information, see About IProjectVariables on page 239.

```
IVariable variableRef = mySession.getVariable(IProjectVariables.WEATHER);
IVariableField variable = variableRef.getSimpleVariable();
String weather = variable.getStringValue();
```

A short-hand way of writing the same code is:

```
IVariableField variable = mySession.getVariableField(IProjectVariables.WEATHER);
String weather = variable.getStringValue();
```

Example code to get the value of a complex variable

The following code gets the value of the *date:year* complex variable field as an integer:

```
IVariable variableRef = mySession.getVariable(IProjectVariables.DATE);
IComplexVariable complexVariable = variableRef.getComplexVariable();
IVariableField field = complexVariable.getField(IProjectVariables.DATE_FIELD_YEAR);
int year = field.getIntValue();
```

A short-hand way of writing the same code is:

```
IVariableField field = mySession.getVariableField(IProjectVariables.DATE,
IProjectVariables.DATE_FIELD_YEAR);
int year = field.getIntValue();
```

Set a call, or message, or web session variable values

The method to programmatically set call, or message, or web session variables depends on the type of variable, that is, simple or complex.

For more information, see the following sections:

- Example code to set a value for a simple variable on page 239
- Example code to set a value for a complex variable on page 239

Refer to the *Orchestration Designer Programmer Reference* guide, available in the online Help, for the full set of API for accessing variables and their values.

Example code to set a value for a simple variable

The following code sets the value of the *weather* simple variable to a Java String object.

Note:

For this code, a simple variable *weather* must be defined in the project variable editor. This code uses the short-hand methods to access variables.

```
String weather = "It is cold outside!";
IVariableField variable = mySession.getVariableField(IProjectVariables.WEATHER);
variable.setValue(weather);
```

Example code to set a value for a complex variable

To set the value of a complex variable field, code format to obtain an IVariableField reference for a complex variable field.

```
SCESession.getVariableField(String variableName, String fieldName)
```

About IProjectVariables

Orchestration Designer generates a class, `IProjectVariables`, which defines the names of all speech, message, and web project variables and their fields as constant Java strings. Orchestration Designer automatically generates this class by using all the variables defined in the **flow** > **project.variables** file in the variable editor. For more information about creating variables, see Creating a variable on page 231.

Avaya recommends that you define your own constants for any variables that are programmatically added to the session. When editing the IProjectVariables class, be sure to make changes *outside* the code generation tags (*//***{{START**: etc.), otherwise, changes might be overwritten by the code generator.

Avaya recommends that you reference variables by using the constants defined in IProjectVariables because it reduces the possibility of application bugs when variables are renamed. For example, you define the variable *weather* once in the project variables, and then elsewhere in the application written code:

```
IVariableField variable = mySession.getVariableField("weather");
```

Another developer modifies the application and renames *weather* to *theWeather* in project variables. Now the "mySession.getVariableField("weather")" code results in a run-time error because the variable *weather* is no longer a project variable.

If you originally wrote the following code:

```
mySession.getVariableField(IProjectVariables.WEATHER)
```

, then when a second developer renames the project variable, the system generates a Java compile error because the “IProjectVariables.WEATHER” constant no longer exists. The “IProjectVariables.WEATHER” constant is changed to “IProjectVariables.THE_WEATHER”.

Variables generated by Orchestration Designer

When you use some of the palette options and items that are available in the Orchestration Designer user interface, Orchestration Designer automatically generates corresponding variables with the same name. For example, if you use a Blind Transfer node in a speech application, Orchestration Designer automatically generates a variable for the Blind Transfer node. This is done because the Blind Transfer node contains a Blind Transfer item. When a Blind Transfer item is used, Orchestration Designer automatically generates a complex variable with the same name.

Orchestration Designer generates an associated read-only variable for the following palette option items in a speech application:

- Blind Transfer on page 678
- Bridged Transfer on page 684
- Call Info (AES) on page 709
- Blind Call (AES) on page 709
- Consultation Call (AES) on page 712
- Dial(AES) on page 715
- Input on page 741
- Input Parameter on page 745
- Module Output on page 767
- Menu node on page 656
- Object Output on page 774
- Record on page 810

In addition to these, if the Interaction Center (IC) connector is enabled for the speech application, Orchestration Designer generates two complex variables: *vdu* and *vdu_cache*. You can add custom fields to the **vdu** variable. You cannot add custom fields to the *vdu_cache* variable.

For more information about the IC connector, see IC connector on page 622

Orchestration Designer generates an associated read-only variable for the following palette option items in a message application:

- Input on page 741

- Input Parameter on page 745
- Module Output on page 767

Orchestration Designer generates an associated read-only variable for the following input items in a web application:

- Choice Input on page 863
- Location Input on page 870
- Picture Input on page 872
- Text Input on page 876
- TextArea Input on page 877
- Video Input on page 878
- Voice Input on page 879

Chapter 17: Grammars

Grammars

The key to recognize input received through inbound voice calls and messages is the use of grammars. In the context of IVR applications, a grammar is a predefined set of words or DTMF tones that a speech application uses to interpret and respond to caller inputs. In the context of text-based applications, a grammar is a predefined set of words that a message application uses to interpret and respond to an inbound SMS or email message.

One of the basic functions of an interactive voice response (IVR) system is to collect and recognize input from the caller. This input can be in the form of either spoken voice responses or touchtone key presses, known as dual tone multi-frequency (DTMF) input. The system uses this input to direct the flow of a call. Similarly, one of the basic functions of a text-processing system is to collect and recognize input from inbound SMS and email messages. The system uses this input to direct the flow of a message application.

You must use a grammar in a speech application or a message application where caller or customer input, that the system can process, needs to be collected. The exceptions to this rule are those nodes and node items in which you can use built-in DTMF, speech, or text recognition.

Note:

You can use variables for grammars, so that grammars can vary dynamically. If you use variables for grammars, then the system does not return the grammar slot values. Instead, the system returns only the complete utterance and interpretation.

Message applications support only single-slotted grammars.

Data applications do not support grammars.

If you copy a grammar variable from one speech application to another speech application or from one message application to another message application, the system copies the variable, but does not copy the source to which the variable references.

The multi-vendor ASR capability is introduced which allows you to dynamically change from using one ASR vendor to another in any VXML node. The platform must know which ASR vendor loads which grammar file. To let the platform know which ASR vendor to use, an External Property to the VXML node is added which can be the Menu, Prompt/Collect, or Form node. The name of the property must be “com.avaya.asr.vendor”, and its value must be one of the following which corresponds to the supported ASR vendors:

- nuance osr

- loquendo
- googleasr

The generated VXML form that is sent to the platform displays the property. On the application side, the property directs the framework runtime to output the grammar file or content specific for the target ASR.

Types of grammars

Types of grammars in speech applications:

- *Voice* (speech) grammars are designed for automated speech recognition (ASR) servers to recognize and interpret speech. You can use these grammars in a speech application project where spoken responses need to be collected.

  To use voice grammars, an ASR server is required to process the input when deploying the speech application on an IVR system. Orchestration Designer uses Microsoft SAPI Speech for testing and simulation. For run-time deployment, you can use any SAPI-compliant ASR server.

- *DTMF grammars* are designed to recognize combinations of one or more touchtone key presses on a telephone keypad. For many Orchestration Designer nodes and items, you need not define DTMF grammars if the system should respond to single key presses. In some node items, such as the Choice item that is used in the Menu node, you can define single *and* multiple key presses, without using a DTMF grammar.

  If you select the **Show AutoVon keys** check box within the Configuring Application Simulation preferences on page 519, the system adds four more DTMF buttons next to the dial keypad for sending A, B, C, and D AutoVon tones for input.

  The term “multiple key presses” in VoiceXML applications does not mean multiple *consecutive* key presses, as it does in some other types of IVR systems. In VoiceXML applications, the term “multiple key presses” means that you have multiple *single* key presses available for the same menu choice.

  For example, in a Choice menu, if the # and 7 DTMF key options are entered, the VoiceXML application interprets the key presses in such a way that either a **#** key press or a **7** key press activates the menu choice.

Types of grammars in message applications:

- Text grammars are designed for automated text recognition to recognize and interpret inbound SMS and email messages. You can use these grammars in a message flow project where inbound SMS and email messages need to be collected.

Grammar compatibility and compliance

Grammar compatibility and compliance for speech application

By default, for speech applications, Orchestration Designer generates grammars that are compliant with the W3C Speech Recognition Grammar Specification (SRGS) version 1.0. Hence, you can use grammars with any ASR server that can process SRGS-compliant grammars, including ASR servers and software produced by companies such as IBM (WebSphere) and Nuance.

In particular, the SRGS supports the following two tag formats:

- semantics/1.0
- semantics/1.0-literals

Semantics/1.0 allows you to use SISR in the tags (ecmascript). Semantics/1.0-literals provides a literal value in the tag.

Note:

Different companies approach and interpret a new standard differently. This fact is evident in the way different companies have approached the use of the <tag> element in the SRGS version 1.0 specification. For more information, visit the Web site: http://www.w3.org/TR/speech-grammar.

Be aware of these differences when creating grammars for speech applications. For more information about how an ASR server implements the SRGS version 1.0 specification, see the documentation for the ASR server.

Grammar compatibility and compliance for message application

Unlike speech application that uses a voice browser to parse speech data, message application uses a text browser to interpret the different attributes of an inbound SMS or email message and parse text data.

The text browser uses grammars to interpret the body of the inbound SMS message. The text browser uses grammars to interpret the subject and the body of the inbound email message. The text browser supports single-slot SRGS literal grammars as well as built-in grammars. The text browser supports the following built-in grammar types: number, digits, nexttoken, and regular expression.

Review the following considerations before using grammars in message applications. In a message application:

- Grammars must be in the literal format. The system does not support semantic interpretation of grammars.

- Grammars must be single slotted.
- Grammars support “repeat.”
- The text browser supports only grammars that are passed inline to the text browser.
- The system does not support grammar caching.
- The grammar tag format must be semantics/1.0-literals.
- The system does not support attachments in email messages.
- The system does not support external grammars.

Grammar planning and designing

You must plan carefully when designing your grammars. Callers or customers who send messages do not always respond the way you expect them to respond. Hence, you must try to anticipate all the possible responses you might get for a particular prompt.

For example, in a speech application, even when you want the caller to respond a simple “yes” or “no,” the actual caller responses might include “yeah,” “OK,” “yep,” and “sure” instead of “yes.” Similarly, negative responses might include “nah,” “nope,” or “uh-uh” instead of “no.”

Multiple grammars

Your speech application or message application might contain overlapping grammars at some point. Overlapping grammars occur when the system has two or more active grammars for which it can accept responses.

For example, there can be a grammar set up in the AppRoot node whose function is to listen for help requests or direct a call or an inbound message to a live attendant. Such grammars are almost always active. Even within the same node there can be multiple grammars, for example, in the Menu node.

You can also use multiple grammars with a single field. For example, in a speech application, you want to offer callers the option to respond with either a spoken response or a DTMF key press. In this case, you can create a grammar for the ASR response and a different grammar for the DTMF response. When used on the same field, either grammar can trigger the recognition. Good application design might even require you to create multiple ASR grammars for the same response field, especially when designing applications that use a natural language speech recognition approach.

Be aware when your application contains overlapping grammars. Also, ensure that you do not have overlapping responses or entries within them.

Basic process of using grammars in a speech and message application

Procedure

1. Create and define a grammar file. To create and define grammars, see:
   - Grammar file management on page 246
   - Editing a grammar file on page 250
2. Perform one of the following actions:
   - If the application is a speech application, then from the Palette pane, drag the **Grammar** item to the node editor where you want to collect and interpret the speech input.
   - If the application is an SMS channel message application, then from the Palette pane, drag the **Grammar** item to the node editor where you want to collect and interpret the inbound SMS message.
   - If the application is an email channel message application, then from the Palette pane, drag the **Grammar** item to the node editor where you want to collect and interpret the inbound email message.

   For more information, see Grammar on page 737.
3. In the node editor, in the Properties view, assign the grammar file to the **Grammar** item.

Grammar file management

In a speech application, a grammar is a predefined set of words or DTMF tones that the speech application uses to interpret and respond to caller inputs.

In a message application, a grammar is a predefined set of words that the message application uses to interpret and respond to the inbound messages.

To create a grammar, you must create a grammar file to hold the grammar entries.

For more information, see the following sections:

- Creating a grammar file on page 247

After you create a grammar file, you must define grammar entries and settings that are to be used by the grammar. To define grammar entries, use the grammar file editor. For more information, see Editing a grammar file on page 250.

Creating a grammar file

Procedure

1. On the **File** menu, click **New** > **Other**.

   The system displays the New wizard.
2. Double-click **Avaya OD Development**.
3. Click **Grammar File**, and then click **Next**.

   The system displays the New Grammar wizard.
4. On the Create a Grammar page, specify the information to create a grammar file.
5. Click **Finish**.

New Grammar wizard: Create a Grammar page field descriptions

Name	Description
Available Projects	The speech project or message flow project for which you want to create a grammar file.
File Name	A name for the grammar file. **Note:** Grammar file names must follow Java naming conventions. For more information, see Conventions for naming Java components on page 59. **Caution:** Do not use double-byte or extended ASCII characters or Java reserved words such as Try, Catch, and Switch when naming a grammar file.

Table continues…

Name	Description
Type	The type of grammar that you want to create. The following are the options: • **Static**: Static grammar is the most common type of grammar and contains a limited number of entries. Static grammars have a ".gram" file extension. For more information, see Static grammar in speech applications on page 251. • **Dynamic**: Dynamic grammar is a Java class grammar that uses an API to build the elements that form a grammar. At run time, a dynamic grammar (the Java class) generates the appropriate grammar content dynamically by retrieving values from a data source such as a database, an LDAP, or a Web service. When you click **Finish** in the New Grammar wizard, the system automatically opens a Java class editor that is populated with a sample code to help you get started. Dynamic grammars have a ".gram-dyn" file extension. For more information, see Editing the dynamic grammar Java class on page 262. • **External**: An external grammar is a grammar that is created, defined, and maintained outside Orchestration Designer. To use an external grammar, you must know the name and path of the grammar file, and the domain host where the grammar file resides. External grammars have a ".gram-extrn" file extension. For more information, see Editing external grammar access properties on page 262. **Note:** The **External** option is available only if you create a grammar file for a speech application.

Table continues…

Name	Description
Mode	The mode of the grammar. The following are the options for creating a grammar file for a speech application: • **DTMF** • **Voice** The following is the option for creating a grammar file for a message application: • **Text**
Style	The style of the grammar. The options are: **Custom** and **Built-in**. Built-in grammars have a ".gram-builtin" file extension. For more information, see Selecting a built-in grammar type in a speech application on page 266. **Note:** The **Style** field is available only if you select **Static** in the **Type** field.
Number of Rows	The number of rows anticipated for the grammar. For each caller response to be defined for recognition, you must define a row in the grammar table. For more information, see Static grammar table rows on page 253 **Note:** The **Number of Rows** field is available only if you select **Static** in the **Type** field and **Custom** in the **Style** field.

Table continues…

Name	Description
Number of Columns	The number of columns anticipated for the grammar. Each column in a static grammar table represents a potential system response to a caller utterance. For more information, see Static grammar table columns on page 254. **Note:** The **Number of Columns** field is available only if you create a grammar file for a speech application. This is because message applications support only single-slot SRGS literal grammars. If you create a grammar file for a speech application, the **Number of Columns** field is available only if you select **Static** in the **Type** field and **Custom** in the **Style** field.
Open file for editing when done	Check box to automatically open the grammar file editor after creating the grammar file. For more information, see Editing a grammar file on page 250. **Note:** The **Open file for editing when done** check box is available only if you select either **Static** or **External** in the **Type** field.
Generate sample grammar	Check box to automatically create a sample grammar as a starting point to help you define the grammar. **Note:** The **Generate sample grammar** check box is available only if you select **Static** in the **Type** field and **Custom** in the **Style** field.

Editing a grammar file

About this task

Use the grammar file editor to define grammar entries in a grammar file. For information about creating a grammar file, see Grammar file management on page 246.

In a speech application, you can create grammar entries to define a predefined set of words or DTMF tones that the speech application uses to interpret and respond to caller inputs.

In a message application, you can create grammar entries to define a predefined set of words that the message application uses to interpret and respond to the inbound messages.

Procedure

1. Perform one of the following actions:
   - Create a grammar file using the New Grammar wizard.

     For more information, see Creating a grammar file on page 247.

     In the New Grammar wizard, if you select the grammar **Type** as **Dynamic** or if you select the **Open file for editing when done** check box, and then click **Finish**, the system automatically opens the Grammar File Editor.
   - In the Navigator view or in the Avaya OD Navigator view, double-click the `*.gram*` file that you want to edit. The grammar file extension depends on the type of grammar. For more information about grammar types and extensions, see New Grammar wizard: Create a Grammar page field descriptions on page 247.

     After you open a grammar file, the system opens the editor depending upon the type of grammar that you edit.
2. Modify the grammar file in the grammar file editor.

Static grammar management

Static grammar in speech applications

The grammar file editor for a static grammar in a speech application uses a tabular format to define the entries that comprise the grammar. Each column is a rule for reference-able column value items. For example, you can rename Column 1 and Column 2 to “animal” and “color” by modifying the name field in the Properties view. For more information, see Setting static grammar table column properties on page 260.

If you add a Grammar on page 737 item under an Input on page 741 item in a Form node on page 655 or a Prompt and Collect node on page 659, the system creates a complex variable that contains additional fields for the column names that you have added to the static grammar, along with the standard confidence, inputmode, interpretation, noinputcount, nomatchcount, utterance, and value fields.

For example, you can set “animal” to “dog” and “color” to “blue.” When the speech recognition is successfully complete, the system stores the interpretation of each column (rule) in the corresponding field name.

The first time when you open the grammar file editor for a static grammar, the table consists of only one active column and row. This means that there is only one active cell, which is initially not populated.

When creating a grammar file, if you select the **Generate sample grammar** check box in the New Grammar wizard, the grammar file editor opens with three populated cells.

With the added support for Google ASR, static grammars for this ASR cannot be created using the grammar editor. When a grammar file is created using the wizard, an ASR specific file for Google is also generated automatically in the **data/<language>/grammars** folder. You must open the file in the text editor and fill the content that works for Google ASR.

For information about editing in the static grammar file editor, see the following sections:

- Static grammar table rows on page 253
- Static grammar table columns on page 254
- Tags on page 256
- Setting static grammar entry properties on page 258

Static grammar in message applications

The grammar file editor for a static grammar in a message application uses a tabular format to define the entries that comprise the grammar. The grammar file editor for a static grammar in a message application contains only one column. This is because message applications support only single-slot SRGS literal grammars. You can rename the column name by modifying the name field in the Properties view. For more information, see Setting static grammar table column properties on page 260.

Note:

You can add rows to the static grammar file editor. You cannot add columns to the static grammar file editor.

If you add the **Grammar** item under the **Input** item in the **Form** node or the **Collect** node, the system creates a complex variable that contains an additional field for the column name contained in the static grammar, along with the standard **confidence**, **inputmode**, **interpretation**, **noinputcount**, **nomatchcount**, **utterance**, and **value** fields.

For example, you can set “animal” to “dog.” When the text recognition is successfully complete, the system stores the interpretation of the column (rule) in the corresponding field name. The first time when you open the grammar file editor for a static grammar, the table consists of only one active column and row. This means that there is only one active cell, which is initially not populated.

When creating a grammar file, if you select the **Generate sample grammar** check box in the New Grammar wizard, the grammar file editor opens with three populated cells.

For information about editing in the static grammar file editor, see the following sections:

- Static grammar table rows on page 253
- Tags on page 256
- Setting static grammar entry properties on page 258

Static grammar table rows

For each caller or inbound message response that you want to define for recognition, you must add a row. For example, you want to offer the following list of food items from which a caller or a customer can order:

- Hamburger
- Cheeseburger
- French fries
- Onion rings
- Salad

To offer these as options, create at least one row entry for each food item. In this example, you can create multiple entries for food items based on expected caller or inbound message responses. For example, “hamburger” and “burger.”

Adding a row to a static grammar table

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project or message flow project that contains the static grammar file in which you want to add a row.
2. Double-click **<*language name*>** > **grammars**.
3. Double-click the `*.gram*` static grammar file in which you want to add a row.
4. In the static grammar file editor, click an active cell in the grammar table.

   Note:

   The rows that are created but not populated contain a single hyphen (-) in each cell.
5. To add a row before the cell that you select, on the main toolbar, click **Add a new grammar row before selection**.

   To add a row after the cell that you select, on the main toolbar, click **Add a new grammar row after selection**. 

   You must add a row for each grammar entry that you want to define.

   Tip:

   Press `Enter` to add a row to the grammar table. The system adds the row after the last row.

Deleting a row from a static grammar table

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project or message flow project that contains the static grammar file from which you want to delete a row.
2. Double-click **<*language name*>** > **grammars**.
3. Double-click the `*.gram*` static grammar file from which you want to delete a row.
4. In the static grammar file editor, click a cell in the row that you want to delete.
5. Click **Delete selected grammar row**. 

Static grammar table columns

Note:

You can add columns only to static grammar files of speech applications. You cannot add columns to static grammar files of message applications because message applications support only single-slot SRGS literal grammars.

In static grammar for speech applications, you can use columns to expand and refine the way an ASR server processes and interprets caller responses. Each column in a static grammar table represents a potential system response to a caller utterance.

Adding columns provide the following capabilities:

- Instructs the ASR server to ignore irrelevant input and listen only for valid responses.

  To do this, in the first row of the column where you want to ignore all the responses except the relevant responses, type the keyword `filler`.

  Note:

  You must type the keyword `filler` in the first row of the column. After you enter the keyword `filler` in the first row, you cannot create entries in any other cell of that column. Orchestration Designer displays **FILLER** in the first cell of the column and clears the contents of any other cells in that column.

  Caution:

  Exercise caution when using this option. When you use the `filler` keyword, you instruct the system to expect irrelevant speech before a recognizable utterance. If the caller does not utter any irrelevant speech, the system can fail to recognize the valid utterance because it expects to receive irrelevant speech first.

- Collects multiple responses with a single grammar.

Each new column represents another utterance to which the system can respond. If you have multiple columns of entries, the speech application expects a caller response that matches an entry in each column. For this reason, exercise caution when creating a grammar with multiple columns. Ensure that your callers provide enough responses to satisfy the expectations of each expected input column.

For example, if you have one column that lists five food items and another column that lists five drinks, the system then expects the caller to order a food item and a drink. If the caller does not order both, the system treats it as a No Match situation and responds accordingly.

- Reuses the contents of one column to recognize a second input.

  If you want to offer the caller a chance to provide two responses from the same list, you can create a column that refers to another column. For example, you offer the following list of food items from which the caller can order:

  - Hamburger
  - Cheeseburger
  - French fries
  - Onion rings
  - Salad

  If you offer this list in only one column, the caller can order only one item. If you want to offer the caller a chance to order two items at one time, you can create a second column that refers to the first column.

  To do this, in the first row of the column that you want to use to refer to another column, type the percent symbol (%) followed by the name of the column from which you want to allow another chance to select. For example, if the column name is **Column0**, then type `%Column0`.

  Note:

  You must type the percent symbol and the column name that you want to refer in the first row of the column. After you type the percent symbol and the column name in the first row, you cannot create entries in any other cell in that column. Orchestration Designer clears the contents of any other cells in that column.

Adding a column to a static grammar table

About this task

You can add columns only to static grammar files of speech applications.

You cannot add columns to static grammar files of message applications because message applications support only single-slot SRGS literal grammars.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project that contains the static grammar file in which you want to add a column.

2. Double-click **<*language name*>** > **grammars**.
3. Double-click the `*.gram*` static grammar file in which you want to add a column.
4. In the static grammar file editor, click an active cell in the grammar table.
5. To add a column *before* the column that you select, on the main toolbar, click **Add a grammar column before selection**. 

   To add a column *after* the column that you select, on the main toolbar, click **Add a grammar column after selection**. 

Deleting a column from a static grammar table

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project that contains the static grammar file from which you want to delete a column.
2. Double-click **<*language name*>** > **grammars**.
3. Double-click the `*.gram*` static grammar file from which you want to delete a column.
4. In the static grammar file editor, click a cell in the column that you want to delete.
5. Click **Delete selected grammar column**. 

Tags

The W3C Speech Recognition Grammar Specification (SRGS) version 1.0 recommends the use of tags with grammar entries. Although tags are optional, you can use tags to greatly simplify the recognition of spoken responses and inbound message, especially when you create multilingual applications.

For example, in a speech application, even when you want the caller to respond a simple “yes” or “no,” the actual caller responses might include “yeah,” “OK,” “yep,” and “sure” instead of “yes.” Similarly, negative responses might include “nah,” “nope,” or “uh-uh” instead of “no.”

You can use a single tag, “YES,” to assign the same value to all the possible affirmative responses. You can also use a single tag, “NO,” for all the possible negative responses.

At run time, when the ASR server receives a spoken response or when the text-processing system receives an inbound message, the ASR server or the text-processing system returns the tag as the *interpretation* and *value* for the input fields. This means that the speech application or message application does not have to identify all of several different returns to determine the next step. Instead, the speech application or message application has to identify only one single return to determine the next step.

The use of tags can be particularly helpful when you want to create multilingual applications. You can accept second-language responses, tag them with the same tag as those in the first

language, and route the call flow or message flow appropriately. For example, in addition to all the English responses tagged as “YES” in the earlier example, you can also take the equivalent responses of “ja,” “ja wohl,” “klar,” “klar doch,” “todsicher,” and so on, and tag them to return the value of “YES.”

Tags facilitate the ability to use the returned interpretation to further direct the inbound call or message because tags make it easier to branch or determine the direction of the speech and message applications. This is because you need to map only one response, in this example “YES,” instead of having to map multiple responses, one for each alternate possibility.

You can set the background and foreground color for the tags that you apply to your grammar entries. You can also set the color for the tag values and cursor. For more information, see Configuring the settings for grammar tags on page 557.

Adding an SRGS tag to a static grammar entry

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project or the message flow project that contains the static grammar file in which you want to add an SRGS tag to a static grammar entry.
2. Double-click **<*language name*>** > **grammars**.
3. Double-click the `*.gram*` static grammar file in which you want to add an SRGS tag to a static grammar entry.
4. In the static grammar file editor, click the cell of the static grammar entry that you want to tag.

   The system highlights the entry and displays the tag assignment field on the right side of the cell.

   Note:

   The system displays the tags that are associated with the grammar entry in the grammar file editor only if you select the **Show tags in editor** check box in **Grammar** preferences. If you clear the **Show tags in editor** check box, then the system displays the associated tag in the grammar editor only if you click the cell that contains the tag. For more information, see Configuring the settings for grammar tags on page 557.
5. Perform one of the following actions:
   a. To create a new tag, highlight the default text **TAG** and type the tag text that you want.
   b. To select an existing tag, in the **TAG** list, click the tag that you want to assign to the grammar entry.

   Note:

   You can view and select a tag from the **TAG** list only if you have already created one or more tags.

In the example mentioned in Tags on page 256, to tag all the affirmative responses as “yes,” assign a new tag, **yes**, to one entry in the grammar table by typing `yes` in the **TAG** field. In the remaining entries, select **yes** from the **TAG** list.

Setting static grammar entry properties

About this task

For each grammar entry in a static grammar table, you can set properties in the Properties view. For more information, see Speech Recognition Grammar Specification Version 1.0.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project or the message flow project that contains the static grammar file in which you want to set the static grammar entry properties.
2. Double-click **<*language name*>** > **grammars**.
3. Double-click the `*.gram*` static grammar file in which you want to set the static grammar entry properties.
4. In the static grammar file editor, in the static grammar table, click the static grammar entry for which you want to set the properties.
5. In the Properties view, set the static grammar entry properties.

Static grammar entry property descriptions

Name	Description
Tag	The tag for the grammar entry. For more information, see About tags on page 256.

Table continues…

Name	Description
Weight	A multiplying factor that indicates how likely it is that the entry is uttered. A weight of 1.0 is considered neutral. That is, an entry with a weight of 1.0 does not bias the recognition one way or the other. A weight of greater than 1.0 biases the grammar entry in a positive way. A weight of less than 1.0 biases the entry in a negative way. The exact application of weights depends on the ASR engine that you use. Therefore, a weight that works well on a particular platform might produce different results on other platforms. Weights have no effect on DTMF grammars. The default for this property is **-1**, which means that no weight is applied at all. **Note:** The **Weight** property is available in a static grammar entry of a speech application only.
Repeat	The number of times you expect that this entry might be repeated before anything in the next column. For example, you have a grammar for which a caller can order a pizza. When ordering, you expect that the caller might say “pizza,” “a large pizza,” or “a very large pizza.” Because you do not know whether or not the caller will say “large” or “very large,” you specify “large” and “very large” entries as possibilities in column 1. You then assign a **Repeat** value of `0-1` to both entries. In the second column, you specify a single entry, “pizza.” A repeat value of **0-** indicates that an item might be repeated zero times, once, or any number of multiple times. A repeat value of **-1** indicates that an item might be repeated once or any number of multiple times. The default for this property is **-1**, which means that this property is not applied at all. **Note:** The **Repeat** property is available in a static grammar entry of a speech application only.

Table continues…

Name	Description
Repeat Probability	The repeat probability. This property indicates how likely it is that this column will be part of the utterance. A value of 1.0 in this field indicates an overwhelming probability that the item will be uttered. A value of 0.0 in this field indicates an overwhelming probability that the item will not be uttered. For example, if you think there is a good chance that the item will be uttered, you can type 0.75 in this field to alert the recognizer that it is likely to encounter this item. The valid range for this property is **0.0** through **1.0**, or **-1**. The default for this property is **-1**, which means that this property is not applied at all. **Note:** The **Repeat Probability** property is available in a static grammar entry of a speech application only.

Setting static grammar table column properties

About this task

For each column in a static grammar table, you can set the properties in the Properties view.

For static grammar tables in speech applications, you can also set the column repeat feature on a column, in addition to a single entry in a column.

The column repeat feature is not supported in message applications because message applications support only single-slot SRGS literal grammars.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project or the message flow project that contains the static grammar file in which you want to set the static grammar table column properties.
2. Double-click **<*language name*>** > **grammars**.
3. Double-click the `*.gram*` static grammar file in which you want to set the static grammar table column properties.
4. In the static grammar file editor, in the static grammar table, click the static grammar column header.

5. In the Properties view, set the static grammar table column properties.

Static grammar table column property descriptions

Name	Description
Name	A logical name for the column. If you add a **Grammar** item under an **Input** item in a **Form** node or a **Prompt and Collect** node, the system creates a complex variable that contains an additional field for the column name that you have added to the static grammar file along with the standard **confidence**, **inputmode**, **interpretation**, **noinputcount**, **nomatchcount**, **utterance**, and **value** fields.
Repeat	**Note:** The **Repeat** property is available only in the static grammar table column for speech applications. The number of times you want to repeat the column. You can assign a **Repeat** value of `3`, `0-3`, `1-5`, and so on. A repeat value of **0** indicates that a column might be repeated zero times, once, or any number of multiple times. A repeat value of **1** indicates that a column might be repeated once or any number of multiple times. The default value is `-1`, which means that this property is not applied at all. For example, you want a caller to provide a DTMF input that ranges between 1 to 12 digits followed by a pound key (#), and then 2 digits followed by a pound key (#). To do this, in the first column of the static grammar table, type `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, and `9` vertically in consecutive cells, and then set the repeat value for this column to `1-12`. In the first cell of the second column, type `#`. In the third column, type `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, and `9` vertically in consecutive cells, and then set the repeat value for this column to `2`. In the first cell of the fourth column, type `#`.

Editing the dynamic grammar Java class

When you create a dynamic grammar, the system automatically creates a skeleton grammar Java class and opens the Java class editor. To complete the Java class that generates the dynamic grammar, you must edit and save this grammar file. For information about creating a grammar file, see Grammar file management on page 246.

You can define column names/slots in the dynamic grammar file editor. When you use the dynamic grammar file in a **Grammar** item, the system creates a complex variable that contains additional fields for the column names/slots that you have defined in the dynamic grammar file, along with the standard **confidence**, **inputmode**, **interpretation**, **noinputcount**, **nomatchcount**, **utterance**, and **value** fields.

To ensure that the system stores data in the additional fields that are created for the column names/slots, you must specify the column names/slots in the Java code that you write for the dynamic grammar. When you specify the column names/slots in the Java code, you must use the same name that you specify for the columns/slots in the dynamic grammar editor.

Note:

You can add only one column name/slot to the dynamic grammar file of a message application because message applications support only single-slot SRGS literal grammars.

If you do not specify column names/slots, then the grammar interpretation returns the standard interpretations, which are, **confidence**, **inputmode**, **interpretation**, **noinputcount**, **nomatchcount**, **utterance**, and **value**.

You can use **Add** and **Delete** to add and delete column names/slots.

You can use **Move Up** and **Move Down** to manage the grammar data in the **Column/Slots Names** box.

For more information about the skeleton Java class, in the online Help, see *Avaya Orchestration Designer - Self Service* > *Programmer Reference* > *API Reference* > *All Classes* > *DynamicGrammar*.

For general information about editing Java classes, see the Eclipse documentation.

External grammar management

Editing external grammar access properties

About this task

Note:

You can use external grammars only in speech applications.

Because an external grammar is not created by using Orchestration Designer and does not reside within the Orchestration Designer environment, if you use an external grammar in your Orchestration Designer speech application, you must specify the location and the type of the grammar.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project that contains the external grammar file for which you want to edit the access properties.
2. Double-click **<*language name*>** > **grammars**.
3. Double-click the `*.gram*` external grammar file for which you want to edit the access properties.
4. In the grammar file editor of the external grammar, modify the access properties of the external grammar file.

External grammar file editor field descriptions

Note:

You can use external grammars only in speech applications.

External grammars can be relative to the speech application and also the parameter substitution. The following variables are allowed in the URL and are replaced at run time. However, if you use these variables, you cannot use the **Test** button.

%DDVAR_<variablename>%

%runtime_Platform%

%runtime_ASR%

Note:

The **Media Type** and **URL** fields support “%DDVAR_<variablename>%” variable substitution.

Name	Description
URI	The name and port number, in URL format, of the application server on which the external grammar resides. The name and port number information must be in the following format: http://localhost:nnnn where: • **localhost** is the IP address or fully qualified domain name of the application server on which the external grammar resides. • **nnnn** is the port number that the Orchestration Designer speech application must use to gain access to the external grammar. The default URI is http://localhost:8080.
Media Type	An Internet mail extension (MIME) type. Enter the media type of the external grammar. To specify another media type, see the documentation of your ASR server for the appropriate media type, and type the media type in this field. **Note:** Currently, Orchestration Designer has only one predefined media type: **application/srgs+xml**. This media type supports the SRGS specification. For Google Speech, media type is **application/avaya-ep-csr**.
Do not encode query string on URI. Must override getURL() in the generated java class file and provide your own encoding.	Check box if you do not want the Orchestration Designer runtime to encode any query string on the external grammar URI. If you select this check box, you must override the getURL() method in the generated Java class file, and add a custom encoding of the query string. If you fail to do this, the voice browser or the text browser gives an error.semantic error.

Table continues…

Name	Description
Test	To test and reference the grammar from the designated application server and display it on a Web page. Be sure that the designated application server is running. **Note:** The **Test** button does not work when you use parameter substitution for the external variable. For example, %DDVAR_<variablename>%; %runtime_Platform%; %runtime_ASR%.
Column/Slots Names	You can define column names/slots in the external grammar file. When you use the external grammar file in a **Grammar** item, the system creates a complex variable that contains additional fields for the column names/slots that you have defined in the external grammar file, along with the standard **confidence**, **inputmode**, **interpretation**, **noinputcount**, **nomatchcount**, **utterance**, and **value** fields. If you do not specify column names/slots, then the grammar interpretation returns the standard interpretations, which are, **confidence**, **inputmode**, **interpretation**, **noinputcount**, **nomatchcount**, **utterance**, and **value**. You can use **Add** and **Delete** to add and delete column/slot names. You can use **Move Up** and **Move Down** to manage the grammar data in the **Column/Slots Names** box.

Configuring external grammar

Note:

You can use external grammars only in speech applications.

When you use external grammars, you must create an interpretation and assign the interpretation to a property of the interpretation. When Orchestration Designer generates grammars, it assigns the interpretation to the “cxtag” property. The Orchestration Designer runtime looks for this “cxtag” and stores it into the “value” field. If you do not define the “cxtag”, the runtime extracts all properties from the interpretation because it does not know which property to use. This might cause unexpected results. Therefore, you must add “cxtag” when you use the “value” field of the recognition result complex variable in external grammars.

For example, you have a grammar that sets two tags:

```
<tag>ORDERNUM=.....;</tag>
<tag>PRICE=.....;</tag>
```

You must define the ORDERNUM and PRICE slots so that the Orchestration Designer runtime extracts these properties and stores them correctly. If you do not set the “cxtag” explicitly, the Orchestration Designer runtime iterates over the interpretation and sets the “value” field to the concatenation of all the properties in the interpretation. With no “cxtag”, the “value” field contains both ORDERNUM and PRICE in it. If you add <tag>cxtag=PRICE;</tag> to your grammar, then the “value” field contains PRICE.

If you do not want to add the “cxtag” tag, then do not use the “value” field.

If you do not want the Orchestration Designer runtime to encode any query string on the external grammar URI, then select the **Do not encode query string on URI. Must override getURL() in the generated java class file and provide your own encoding** check box in the external grammar file. For more information, see Editing external grammar access properties on page 262. If you select the **Do not encode query string on URI. Must override getURL() in the generated java class file and provide your own encoding** check box, you must override the getURL() method in the generated Java class file, and add a custom encoding of the query string. If you fail to do this, the voice browser or text browser gives an error.semantic error.

Built-in grammar management

Built-in grammar management in speech applications

Selecting a built-in grammar type in a speech application

About this task

A built-in grammar is a static DTMF or voice (speech) grammar. You cannot create or edit built-in grammar entries. You can use built-in grammar entries in your speech application.

To use a built-in grammar, when creating a grammar file in a speech application, you must select **Style** as **Built-in** in the New Grammar wizard. For more information, see New Grammar wizard: Create a Grammar page field descriptions on page 247.

Note:

Different speech servers return different values for their supported built-in grammars. If you are using MRCP in the simulator to work with third-party speech, then the supported languages differ based on the third-party speech engine. For information about the implementation of built-in grammars in your speech server, refer to the documentation of your speech server.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project that contains the `*.gram*` built-in grammar file that you want to use.
2. Double-click ***<language name>*** > **grammars**.

3. Double-click the `*.gram*` built-in grammar file that you want to use.
4. In the **Type** field, select the built-in grammar type.

Supported built-in grammar types in speech applications

Orchestration Designer supports the following DTMF and voice built-in grammars in speech applications, as specified by the W3C VoiceXML 2.0 Recommendation.

Type	Supported languages	Description
boolean	English Japanese Chinese	Inputs include affirmative and negative phrases appropriate to the current language. The default DTMF values are 1 and 2. DTMF 1 is affirmative and 2 is negative. For information about editing the DTMF key presses, see Editing the DTMF key presses in a built-in grammar file of a speech application on page 268.
number	English Japanese Chinese	Valid spoken inputs include phrases that specify numbers such as “one hundred twenty-three” or “five point three”. Valid DTMF inputs include positive numbers entered using digits and “*” to represent a decimal point. The result is a string of digits from 0 through 9 and can optionally include a decimal point (.) using the star key (*). **Note:** Do not allow callers to use a leading zero (0) with this option because the interpreter might interpret this as an octal number.
digits	English Japanese Chinese	Valid spoken or DTMF inputs include one or more digits, 0 through 9. The result is a string of digits. You must define a digit length for the expected DTMF key presses. For more information, see Defining a digit length for DTMF or spoken input in a built-in grammar file of a speech application on page 269.
currency	English	Valid spoken inputs include phrases that specify a currency amount. For DTMF input, the star key (*) acts as the decimal point. The result is a string in the UUUmm.nn format, where UUU is the three character currency code according to ISO 4217 standard, mm is the integer part of the currency, and nn is the fractional part of the currency. If a caller does not specify the currency or if the currency cannot be reliably determined, then the system uses the default currency code of the speech project language that you specify. For more information, see Project languages on page 277.

Table continues…

Type	Supported languages	Description
date	English	Valid spoken inputs include phrases that specify a date, including a month, day, and year. DTMF inputs are four digits for the year followed by two digits for the month and two digits for the day. The result is a fixed-length date string with yyyymmdd format, such as “20000704”. If the year is not specified, yyyy is returned as “????”. If the month is not specified, mm is returned as “??”. If the day is not specified, dd is returned as “??”.
phone	English	Valid spoken inputs include phrases that specify a phone number. DTMF star key (*) represents “x”, for extension. For example, a string that contains a telephone number and optionally the character “x” to indicate a phone number with an extension. For example, in North America, the 18005551234*103 entry informs the system to dial 1-800-555-1234, extension 103.
time	English	Valid spoken inputs include phrases that specify a time, including hours and minutes. The result is a five character string in the hhmmx format, where x is either “a” for AM, “p” for PM, “h” to indicate a time specified by using 24 hour clock, or “?” to indicate an ambiguous time. Input can be through DTMF. In the case of DTMF input, the result always ends with “h” or “?” because there is no DTMF convention for specifying AM/PM.

Editing the DTMF key presses in a built-in grammar file of a speech application

About this task

You can edit the DTMF key presses to signify a “yes” response and a “no” response, in the grammar file editor of a DTMF built-in grammar file of a speech application.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project that contains the `*.gram*` built-in DTMF grammar file in which you want to edit the DTMF key presses.
2. Double-click **<*language name*>** > **grammars**.
3. Double-click the `*.gram*` built-in DTMF grammar file in which you want to edit the DTMF key presses.
4. In the **Type** field, click **boolean**.
5. In the **Yes Digit** field, type a DTMF key for a “yes” response.
6. In the **No Digit** field, type a DTMF key for a “no” response.

Defining a digit length for DTMF or spoken input in a built-in grammar file of a speech application

About this task

The valid spoken or DTMF inputs include one or more digits, 0 through 9. The result is a string of digits.

You must define a digit length for the expected DTMF key presses in the built-in DTMF grammar file of a speech application.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project that contains the `*.gram*` built-in DTMF or voice grammar file in which you want to edit the digit length of the DTMF key presses or spoken input.
2. Double-click **<*language name*>** > **grammars**.
3. Double-click the `*.gram*` built-in DTMF or voice grammar file in which you want to edit the digit length of the DTMF key presses or spoken input.
4. In the **Type** field, click **digits**.
5. Perform one of the following actions:

   Click the button next to the **Min Len** field to specify a range of number of the DTMF key presses or spoken input, and then perform the following actions:

   - In the **Min Len** field, type the minimum number of DTMF key presses or spoken input for a match.
   - In the **Max Len** field, type the maximum number of DTMF key presses or spoken input for a match.

   Click the button next to the **Exact Length** field to specify an exact number of key presses or spoken input.

Built-in grammar management in message applications

Supported built-in grammar types in message applications

The following table contains information about the supported built-in grammar types for message applications:

Type	Supported languages	Description
number	English	Valid text inputs include positive numbers that are entered by using digits and Period (.) to represent a decimal point. The result is a string of digits from 0 through 9 and can optionally include a decimal point (.).

Table continues…

Type	Supported languages	Description
digits	English	Valid text inputs include one or more digits from 0 through 9. The result is a string of digits and depends on the digit length that you specify for the expected inbound SMS or email message. If you use the digits built-in grammar type, you must specify a digit length for the expected inbound message. For more information, see Specifying a digit length for built-in grammars in a message application on page 270.
nexttoken	English	Valid text inputs include letters, numbers, digits, and other characters, except the characters that you use as separator characters. The result is a string of letters, numbers, digits, and other characters, except the separator characters, and depends on the separator characters and the token length that you specify for the expected inbound message. If you use the nexttoken built-in grammar type, you must specify a token length for the expected inbound message. For more information about defining the token length, see Specifying token length and separator characters for built-in grammars in a message application on page 271.
regular expression	English	Regular expression to identify and match the inbound message based on patterns. The following are some examples: • The regular expression a*b collects all the words that contain the letters aaab or ab as matches. • The regular expression \$?[0-9]{1,20}.?[0-9]{0,2} collects all words that contain $50.00, $50, 50, 50.00, and so on as matches.

Specifying a digit length for built-in grammars in a message application

About this task

If you use the digits built-in grammar type in a message application, you must specify a digit length for the expected inbound SMS or email message.

Valid text inputs in an inbound message include one or more digits from 0 through 9. The result is a string of digits and depends on the digit length that you specify for the expected inbound message.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the message flow project directory that contains the built-in grammar file in which you want to specify the digit length.
2. Double-click **english** > **grammars**.
3. Right-click the .gram-builtin file in which you want to specify the digit length, and then click **Open With** > **Grammar Built-in Editor**.

4. In the **Type** field, click **digits**.
5. Perform one of the following actions:
   - Select the button next to the **Min Len** field to specify the range of number of digits that you expect in the inbound message, and then perform the following actions:
     a. In the **Min Len** field, type the minimum number of digits that you expect in the inbound message.
     b. In the **Max Len** field, type the maximum number of digits that you expect in the inbound message.
   - Select the button next to the **Exact Length** field to specify the exact number of digits that you expect in the inbound message, and then perform the following action:

     In the **Exact Length** field, type the exact number of digits that you expect in the inbound message.
   - Select the button next to **Implied Length** to collect digits based on the default separator characters. The system collects digits from the inbound message until it encounters a tab, a return, a newline, a formfeed, or a space separator character in the inbound message. The system similarly processes the rest of the text in the inbound message.

About nexttoken built-in grammar type in message applications

If you use the **nexttoken** built-in grammar type in a message application, you must specify a token length for the expected inbound SMS or email message. You can specify separator characters. The system collects the characters from an inbound message depending on the token length or until the system encounters a separator character. The system similarly processes the rest of the text in the inbound message, and then passes these collections of characters as separate input tokens to the text browser. The text browser uses grammars to match these tokens and to return a corresponding semantic interpretation.

If the length of the token that you specify does not match the length of the token that the system collects, the system throws a No Match event.

For example, you specify the exact token length as `nine` characters and specify `tab` as the separator character. When processing the inbound message, if the system comes across a tab after five characters, then the system throws a No Match event.

If you do not specify separator characters, then the system uses tab, return, newline, formfeed, and space as the default separator characters.

Specifying token length and separator characters for built-in grammars in a message application

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the message flow project directory that contains the built-in grammar file in which you want to specify the token length and the separator characters.
2. Double-click **english** > **grammars**.

3. Right-click the .gram-builtin file in which you want to specify the token length and the separator characters, and then click **Open With** > **Grammar Built-in Editor**.
4. In the **Type** field, click **nexttoken**.
5. Perform one of the following actions:
   - Select the button next to the **Min length** field to specify a minimum and maximum length for the token. Perform the following actions:
     a. In the **Min Len** field, type a minimum length for the token.
     b. In the **Max Len** field, type a maximum length for the token.
   - Select the button next to the **Exact Length** field to specify an exact length for the token. Perform the following action:

     In the **Exact Length** field, type an exact length for the token.
   - Select the button next to **Implied Length** to create tokens based on the separator characters.
6. In the **Separator Characters** field, type the characters that you want to use as separator characters to create tokens.

   Note:

   You can use characters such as semicolon (;), slash (/), underscore (_), and dollar sign ($) in between the separator characters if you want to specify more than one separator character.

   Use a separator character that would not conflict with the characters in the inbound SMS or email message. For example, pipe symbol (|).

   If you do not specify separator characters, the system creates tokens based on the default separators, which are tab, return, newline, formfeed, and space. To use tab, return, newline, and formfeed as separators, you must use `\t`, `\r`, `\n` and `\f`.

Grammar file import

Importing grammar files in a speech application

In Orchestration Designer, you can import built-in grammar definitions and external grammar definitions from a delimited text file. However, you cannot import actual grammars.

Note:

You can import built-in grammar definitions and external grammar definitions from a delimited text file only in speech applications.

For more information, see the following section:

Before importing a delimited grammar file in a speech application on page 273

Before importing a delimited grammar file in a speech application

Before you import a grammar data file in a speech application, you must first create a delimited grammar data text file. The column layout of the fields in the text file varies depending on the type of grammar definition that you import. Each row has a fixed number of required columns followed by optional columns.

Requirements for importing an external grammar definition delimited file in a speech application

Note:

You can import an external grammar definition delimited file only in a speech application.

An external grammar requires the columns: name, mode, type, url, mediatype, followed optionally by grammar slot names. For example,

test2|voice|external|http://localhost:8080/eeeeezz|application/srgs+xml|slota|slotb

The following is the column description:

name: Grammar name

mode: DTMF or voice

type: External or builtin

url: URI of the external grammar

mediatype: Usually application/srgs+xml

As a delimiter, you can use a tab or another consistently used character in your file that does not conflict with the data content. For example, a pipe symbol (|) or a comma.

Note:

Before you import a grammar data file, you must create a delimited grammar data file as per the above specifications.

Example of a vertical bar delimited grammar file for external grammar

The following is an example of a vertical bar delimited grammar file for external grammar in a speech application:

```
# comment external voice, default media type is application/srgs+xml
#
# name|mode|type|url|mediatype|[slotname] [slotname]...
#
```

```
# testa|voice|external| http://localhost:8080/xxxxzzz|xxx/srgs+xml|
# testb|voice|external| http://localhost:8080/eeeeezz|application/srgs+xml|slota slotb
```

The “slotname” column after the “mediatype” column is optional.

Requirements for importing a built-in grammar delimited file in a speech application

Note:

You can import a built-in grammar delimited file only in a speech application.

A built-in grammar requires the columns: name, mode, type, subtype, followed optionally by the columns for the options. For example,

test9|dtmf|builtin|digits|minlength=4;maxlength=6

The following is the column description:

name: Grammar name

mode: DTMF or voice

type: External or builtin

subtype: The builtin grammar type: number, time, date, digits, boolean, phone, or currency.

options: Options specific to the built-in grammar. For more information, see the VXML specification for built-in grammar options.

As a delimiter, you can use a tab or another consistently used character in your file that does not conflict with the data content. For example, a pipe symbol (|) or a comma.

Note:

Before you import a grammar data file, you must create a delimited grammar data file as per the above specifications.

Example of a vertical bar delimited grammar file for built-in grammar

The following is an example of a vertical bar delimited grammar file for built-in grammar:

```
# comment dtmf builtin
#
# name|mode|type|subtype|[options]
#
test5|dtmf|builtin|number
test6|dtmf|builtin|time
test7|dtmf|builtin|date
test8|dtmf|builtin|digits
test9|dtmf|builtin|digits|minlength=4;maxlength=6
test10|dtmf|builtin|digits|length=6
test11|dtmf|builtin|boolean|y=1;n=2
test12|dtmf|builtin|phone
```

test13|dtmf|builtin|currency

The columns for the options, after the “subtype” column in the delimited grammar file, are optional.

Importing a delimited grammar data file in a speech application

About this task

Note:

You can import a delimited grammar data file only in speech applications.

Before you import a grammar data file in a speech application, you must create a delimited grammar data file as per the requirements mentioned in the Requirements for importing an external grammar definition delimited file in a speech application on page 273and Requirements for importing a built-in grammar delimited file in a speech application on page 274.

Procedure

1. On the **File** menu, click **Import**.

   The system displays the Import wizard.

2. Double-click **Avaya OD Development**.
3. Click **Grammar Data**, and then click **Next**.

   The system displays the Import Grammar Data wizard.

4. On the Grammar Data Import page, specify the criteria to import the delimited grammar data file.
5. Click **Finish**.

Import Grammar Data wizard: Grammar Data Import page field descriptions

Name	Description
From file	The directory path of the delimited grammar data file that you want to import.
Target Language	
Project	The project in which you want to import the delimited grammar data file.
Language	The language of the grammar based on which you want to update the grammar with the data from the delimited grammar data file.
Options	
Tab	If a tab is used as a delimiter in the delimited grammar data file.
Other	The delimiter that is used in the delimited grammar data file.

Chapter 18: Language and localization

Localization

Localization is the process of adapting an Orchestration Designer project for use with other languages so that the project can be used in different locales around the world.

To facilitate the localization of the Orchestration Designer projects, Avaya has designed Orchestration Designer to separate language-related functions and components from the logic related functions and components. You can create a single application by using a single flow and its corresponding business logic. You can then easily and quickly adapt the application for use with other languages.

Web applications automatically detect the language settings of the users' web browser, and present the localized text of the language to the web browser.

Language implementation in Orchestration Designer

Depending on the type of the project, Orchestration Designer supports language implementation at the following levels:

- **Project languages**: Project language refers to the default starting language assigned when a speech, or a message flow, or a web project is created. In Orchestration Designer, you can create and add project languages. In an Orchestration Designer speech project, you can also assign an ASR language, TTS language, and localization bundle language to the project language when you create and add the project language. Adding project languages also copies all existing prompts and phrases to a new language directory, so that you can translate the prompts and phrases to another language.

  For more information, see Project languages on page 277.

- **Automated Speech Recognition (ASR) languages**: ASR languages are not installed as part of the Orchestration Designer software. The ASR languages available to you depend on what languages are installed and are available on your ASR server. When you add an ASR language to Orchestration Designer, you must ensure that the language exists on the ASR server, otherwise you might get unpredictable results with ASR requests.

  Note:

  You can install and use ASR languages only in speech applications.

For more information, see Automated speech recognition languages on page 284.

- **Text-to-Speech (TTS) languages**: TTS languages are not installed as part of the Orchestration Designer software. The TTS languages available to you depend on what languages are installed and are available on your TTS server. When you add a TTS language to Orchestration Designer, you must ensure that the language exists on the TTS server, otherwise you might get unpredictable results with TTS requests.

  Note:

  You can install and use TTS languages only in speech applications.

  For more information, see Text-to-speech languages on page 287.

- **Localization bundle languages**: Language localization bundles are packaged with Orchestration Designer, but not installed automatically during initial installation and configuration. If you need a localization bundle for your speech project, you can install it.

  The current release of Orchestration Designer includes the *U.S. English* language localization bundle.

  You can obtain and install additional localization bundles as they become available. To receive updated information about the availability of localization bundles, contact your Avaya service representative or visit support.avaya.com.

  Note:

  You can install and use localization bundles only in speech applications.

  For more information, see Localization bundles on page 289.

Project language management

Project languages

Project language refers to the default starting language assigned when a speech, or a message flow, or a web project is created. In Orchestration Designer, you can create and add project languages. In an Orchestration Designer speech project, you can also assign an ASR language, TTS language, and localization bundle language to the project language when you create and add the project language. Adding a project language also copies all existing grammars, phrases, and prompts to the new language directory, so that you can translate the grammars, phrases, and prompts to the new project language.

The default project language is **english**. This language is the starting language that Orchestration Designer loads at run time for the application. However, you can select a different starting language for the application on the **Languages** tab in the Properties for <project name> dialog box. For more information, see Changing the project default language on page 283.

When you add a project language, Orchestration Designer creates a new language directory in the project. At the same time, Orchestration Designer copies the existing grammar, phrase, and prompt files from the language directory on which you are basing the new package, into the new directory. This makes it easier to identify and translate all the grammars, phrases, and prompts to the new language.

Tip:

If you want to run an application in two different languages, you should complete the first language version of the project before adding a second project language. This ensures that all the required grammars, phrases, and prompts are copied into the new language directory.

Important:

To ensure that the call flow and the message flow use the prompt and grammar resources, do not move or rename the prompt and grammar files. Be sure that each project language has the same number of prompt and grammar resource files with the same file names.

For example, you have a speech project that is based on U.S. English and you want to localize the speech project for Canadian French. The first step after finishing the speech project application is to add a project language for Canadian French. For more information, see Adding a project language on page 278.

After adding the project language, you must go through all the grammars, phrases, and prompts in the Canadian French language resource directory and translate them from English to French. If you are using audio variables in the speech project, you must install the localization bundle for Canadian French. For more information, see Installing a localization bundle on page 291.

As the last step, you must instruct Orchestration Designer to use the Canadian French language resources instead of the English resources. For more information, see Changing the project default language on page 283.

Adding a project language

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, click the speech project or message flow project in which you want to add a language.
2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left pane, click Orchestration Designer.
4. Click the **Languages** tab.
5. In the **Project Languages** area, click **Add**.

   The system displays the **Project Languages** dialog box.
6. On the New language page, specify the information to add a project language.

Note:

When you install standard phrases in a speech project, the system copies the phrasesets and optionally the audio files to your speech project. If you want to use the audio files that are stored on an external server, you must configure the phraseset to reference the external audio file location.

7. Click **OK**.

   The system displays the project language that you add in the **Project Languages** area on the **Languages** tab in the Properties for <project name> dialog box.

Project Language dialog box: New language page field descriptions

Note:

The system displays the ASR language, the TTS language, and the Localization language in the respective fields of the Project Language dialog box of a speech project only after you add these languages. For more information, see Adding an ASR language on page 285, Adding a TTS language on page 287, and Adding a localization bundle on page 290.

Name	Description
Language Name	A name for the project language. This name appears only in Orchestration Designer.
Base Language	The project language that you want to use as the base for this project language.
Language ID	The language ID for the project language. **Note:** The **Language ID** field is available only if you add the project language for a message flow project or a web flow project.
ASR Language	The ASR language that you want to use with the project language. Usually, the ASR language is the same as the default project language. You must select an ASR language for your speech project even if your speech project does not use an ASR language. **Note:** The **ASR Language** field is available only if you add the project language for a speech project.

Table continues…

Name	Description
TTS Language	The TTS language that you want to use with the project language. Usually, the TTS language is the same as the default project language. You must select a TTS language for your speech project even if your speech project does not use a TTS language. **Note:** The **TTS Language** field is available only if you add the project language for a speech project.
Localization Language	The localization bundle that you want to use with the project language. Usually, the localization language is the same as the default project language. You must select a localization bundle for your speech project even if your speech project does not use a localization bundle. **Note:** The **Localization Language** field is available only if you add the project language for a speech project.
Localization Bundle Options **Note:** The **Localization Bundle Options** area is available only if you add the project language for a speech project.	
Standard Phrases	**Install** to install the standard phrase audio files that are associated with the localization bundle. **Uninstall** to uninstall the installed standard phrase audio files that are associated with the localization bundle. **Note:** The **Standard Phrases** field is available only if you add the project language for a speech project.

Table continues…

Name	Description
Use standard phrases on an external server	Check box to use the standard phrase audio files that are stored on an external server. If you select this check box, the system does not install the standard phrase audio files that are associated with the localization bundle in your speech project. If you select this check box, then in the phraseset file, you must select **External**, and in the **URL base** field, type the URL base where the audio files are located. For more information, see Specifying the phrase audio file location on page 171. **Note:** The **Use standard phrases on an external server** check box is available only if you add the project language for a speech project.

Editing a project language

About this task

You can edit a project language only in a speech project.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, click the speech project in which you want to edit or uninstall a project language.
2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left pane, click **Orchestration Designer**.
4. Click the **Languages** tab.
5. In the Project Languages area, select the project language that you want to edit or uninstall.
6. Click **Edit**.

   The system displays the Project Language dialog box.
7. On the Edit language page, modify the project language information.

   Note:

   When you install standard phrases in a speech project, the system copies the phrasesets and optionally the audio files to the speech project. If you want to use the audio files that are stored on an external server, you must configure the phraseset to reference the external audio file location.
8. Click **OK**.

Project Language dialog box: Edit language page field descriptions

Note:

You can edit a project language only in a speech project.

The system displays the ASR language, the TTS language, and the Localization language in the respective fields of the **Project Language** dialog box of a speech project only after you add these languages. For more information, see Adding an ASR language on page 285, Adding a TTS language on page 287, and Adding a localization bundle on page 290.

Name	Description
Language Name	The name of the project language. This field is read only.
ASR Language	The ASR language that you want to use with the project language. Usually, the ASR language is the same as the default project language.
TTS Language	The TTS language that you want to use with the project language. Usually, the TTS language is the same as the default project language.
Localization Language	The localization bundle that you want to use with the project language. Usually, the localization language is the same as the default project language.
Localization Bundle Options	
Standard phrases	**Install** to install the standard phrase audio files that are associated with the localization bundle. **Uninstall** to uninstall the installed standard phrase audio files that are associated with the localization bundle.
Use standard phrases on an external server	Check box to use the standard phrase audio files that are stored on an external server. If you select this check box, the system does not install the standard phrase audio files that are associated with the localization bundle in your speech project. If you select this check box, then in the phraseset file, you must select **External**, and in the **URL base** field, type the URL base where the audio files are located. For more information, see Specifying the phrase audio file location on page 171.

Changing the project default language

About this task

You can change the default or starting language of a speech, or message flow, or web project.

You can also create a multilingual application by changing the language within the application at run time. For more information, see Changing the language within an application at run time on page 283.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, click the speech, or message flow, or web project for which you want to change the default language.
2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left pane, click Orchestration Designer.
4. Click the **Languages** tab.
5. In the **Starting Language** field, click the language that you want to use as the default language for the project.
6. Click **Apply** and **Close**.

Changing the language within an application at run time

About this task

You can create a multilingual application by changing the language within the application at run time. The language that you specify at run time is used for the current call. For other calls, the system uses the default or starting project language. For more information, see Changing the project default language on page 283.

Procedure

1. In the call flow, or message flow, or web flow editor, double-click the data node for which you want to change the language.
2. From the Palette pane, drag **Operation** into the data node editor.
3. In the Properties pane, in the **Type** field, click **Set Current Language**.
4. In the **Language** field, click the language that you want to set. If the name of the project language is stored in a variable, then perform the following actions:
   a. In the **Variable** field, select the variable that contains the name of the project language.
   b. If you select a complex variable in the **Variable** field, select the variable field that contains the name of the project language in the **Variable Field** field.

Deleting a project language

About this task

Note:

If the speech or message flow or web project contains only one project language, you cannot delete the project language.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, click the speech or message flow or web project from which you want to delete a project language.
2. On the **Project** menu, click **Properties**.

   The system displays the **Properties for <*project name*>** dialog box.
3. In the left pane, click **Orchestration Designer**.
4. Click the **Languages** tab.
5. In the Project Languages area, click the project language that you want to delete.
6. Click **Delete**.

   The system deletes the project language that you select, including the language directory and all associated language resources.

   Note:

   The **Delete** button is unavailable if the speech or message flow or web project contains only one project language.

Automated speech recognition language management

Automated speech recognition languages

Note:

You can add and use ASR languages only in speech applications.

Automated speech recognition (ASR) languages are not installed as part of the Orchestration Designer software installation. Orchestration Designer speech applications, however, generate the code to use ASR on the designated ASR server. This means that you must know which languages are installed and are available on your ASR server.

By default, Orchestration Designer speech projects use U.S. English (**en-us**) as the ASR language. You can, however, change the default ASR language.

Adding an ASR language

About this task

When you add an ASR language to Orchestration Designer, you must ensure that the language exists on the ASR server. If the language does not exist on the ASR server, you might get unpredictable results for ASR requests.

Note:

You can add ASR languages only in speech applications.

Procedure

1. On the **Window** menu, click **Preferences**

   The system displays the Preferences dialog box.

2. In the left pane, double-click **Avaya** > **Orchestration Designer**.

3. Click **Languages**.

   The right pane of the Preferences dialog box displays the **Languages** options.

4. In the **Automated Speech Recognition** area, click **Add**.

   The system displays the Add Automated Speech Recognition (ASR) Language dialog box.

5. In the field, type the code of the ASR language that you want to add.

   The code that you specify must be in the "*ll-cc* " format, where *ll* represents the two-character language code, and *cc* represents the two-character country code. For example, the code for U.S. English is **en-us** and the code for UK English is **en-uk**.

   Note:

   The ASR language code that you specify must match exactly the ASR language code of the language that is installed on the ASR server. Otherwise, the system cannot process the ASR requests that use this language.

   For a complete list of the two-letter country codes supported by the International Organization for Standardization (ISO 3166), see the ISO Web site countries list.

   For a complete list of the two-letter language codes supported by ISO 639-1, see the ISO Web site languages list.

6. Click **OK**.

7. In the Preferences dialog box, click **Apply**.

Editing a project language to use a different ASR language

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, click the speech project for which you want to specify a different ASR language.
2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left pane, click **Orchestration Designer**.
4. Click the **Languages** tab.
5. In the Project Languages area, click the project language for which you want to use a different ASR language.
6. Click **Edit**.

   The system displays the Project Language dialog box.
7. In the **ASR Language** field, click the ASR language that you want to use.
8. Click **OK**.
9. In the Properties for <project name> dialog box, click **Apply** and **Close**.

Deleting an ASR language

About this task

Note:

If your speech project contains only one ASR language, you cannot delete the ASR language.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left pane, double-click **Avaya** > **Orchestration Designer**.
3. Click **Languages**.

   The right pane of the Preferences dialog box displays the **Languages** options.
4. In the Automated Speech Recognition area, click the ASR language that you want to delete.
5. Click **Delete**.

   The system displays a Question message.
6. To confirm the deletion, click **OK**.

The system displays a Warning message that informs you that deleting the ASR language might invalidate projects.

7. To confirm the deletion and remove the ASR language, click **Yes**.
8. In the Preferences dialog box, click **Apply**.

Text-to-speech language management

Text-to-speech languages

Note:

You can add and use TTS languages only in speech applications.

Text-to-speech (TTS) languages are not installed as part of the Orchestration Designer software installation. Orchestration Designer speech applications, however, generate the code to use TTS on the designated TTS server. This means that you must know which languages are installed and are available on your TTS server.

By default, Orchestration Designer speech projects use U.S. English (**en-us**) as the TTS language. You can, however, change the default TTS language.

Adding a TTS language

Before you begin

When you add a TTS language to Orchestration Designer, you must ensure that the language exists on the TTS server. If the language does not exist on the TTS server, you might get unpredictable results for TTS requests.

Note:

You can add TTS languages only in speech applications.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left pane, double-click **Avaya** > **Orchestration Designer** .
3. Click **Languages**.

   The right pane of the Preferences dialog box displays the **Languages** options.
4. In the Text-to-Speech area, click **Add**.

The system displays the Add Text-To-Speech (TTS) Language dialog box.

5. In the field, type the code of the TTS language that you want to add.

   The code that you specify must be in the *ll-cc* format, where *ll* represents the two-character language code, and *cc* represents the two-character country code. For example, the code for U.S. English is en-us and the code for UK English is en-uk.

   Note:

   The TTS language code that you specify must match exactly the TTS language code of the language that is installed on the TTS server. Otherwise, the system cannot process the TTS requests that use this language.

   For a complete list of the two-letter country codes supported by the International Organization for Standardization (ISO 3166), see the ISO Web site countries.

   For a complete list of the two-letter language codes supported by ISO 639-1, see the ISO Web site languages list.

6. Click **OK**.
7. In the Preferences dialog box, click **Apply**.

Editing a project language to use a different TTS language

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, click the speech project for which you want to specify a different TTS language.
2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left pane, click **Orchestration Designer**.
4. Click the **Languages** tab.
5. In the Project Languages area, click the project language for which you want to use a different TTS language.
6. Click **Edit**.

   The system displays the Project Language dialog box.
7. In the **TTS Language** field, click the TTS language that you want to use.
8. Click **OK**.
9. In the Properties for <project name> dialog box, click **Apply** and **Close**.

Deleting a TTS language

About this task

Note:

If your speech project contains only one TTS language, you cannot delete the TTS language.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left pane, double-click **Avaya** > **Orchestration Designer**.
3. Click **Languages**.

   The right pane of the Preferences dialog box displays the **Languages** options.
4. In the Text-to-Speech area, click the TTS language that you want to delete.
5. Click **Delete**.

   The system displays a Question message.
6. To confirm the deletion, click **OK**.

   The system displays a warning message that informs you that deleting the TTS language might invalidate projects.
7. To confirm the deletion and remove the TTS language, click **Yes**.
8. In the Preferences dialog box, click **Apply**.

Localization bundle management

Localization bundles

Note:

You can add and use localization bundles only in speech applications.

Localization bundles need to be installed only when you plan to use audio variables in your speech project, because the localization bundle is what tells Orchestration Designer how to handle the data input to provide the correct audio output. For more information, see Audio Variable on page 676.

Localization bundles are sets of files for a particular language that include:

- A JAR (Java ARchive) file that contains the logic to tell Orchestration Designer how to perform the conversion from variable data to audio output for that language.
- A metadata file that the prompt file editor of a speech project uses to display the formats that the localization bundle supports.
- The standard phraseset (`*.phraseset`) file and standard phrase audio (`*.wav`) files associated with that language.

These bundles are delivered in the form of JAR files. The bundles that are available at the time of the release of Orchestration Designer are included on your ISO image. You can download additional localization bundles, as they become available, from the Avaya customer support site, support.avaya.com. If you want localization languages that are not available with this release, contact Avaya service representative and check the customer support site frequently.

Tip:

When you receive a new localization bundle, save it to a directory on your hard drive, and make note of where you saved it. You can add the new localization bundle to the Orchestration Designer workbench environment later.

Adding a localization bundle

About this task

Before you can install a language localization bundle, you must add it to Orchestration Designer, if it has not already been added.

Note:

You can add localization bundles only in speech applications.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left pane, double-click **Avaya** > **Orchestration Designer**.
3. Click **Languages**.

   The right pane of the Preferences dialog box displays the **Languages** options.
4. In the **Audio Localization Packages** area, click **Add**.

   The system displays the **Browse for localization package** dialog box.
5. Locate the localization bundle JAR file you want to add and click **Open**.

   Orchestration Designer automatically adds the localization bundle to the workbench environment. The new localization bundle is displayed in the **Audio Localization Packages** list.

6. In the Preferences dialog box, click **Apply**.

Installing a localization bundle

About this task

Before installing a language localization bundle in a speech project, you must add the localization bundle to Orchestration Designer, if it has not already been added. To add a localization bundle, see Adding a localization bundle on page 290.

Note:

The following procedure installs the Java code that contains the logic for language localization. This procedure does not install the standard phrases for the localization language. That is a separate procedure. Both standard phrases and localization logic must be installed for Audio Variables to work in a speech project. For more information, see Installing the standard phrasesets of a localization bundle on page 293.

Procedure

1. In the Navigator view or the Avaya OD Navigator view, click the speech project for which you want to install the localization bundle.
2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left pane, click **Orchestration Designer**.
4. Click the **Languages** tab.
5. In the **Localization Bundles** area, click the localization bundle that you want to install.
6. Click **Install**. The status in the **Installed** column for the localization bundle changes to **true**.
7. Click **Apply** and **Close**.

Uninstalling a localization bundle

About this task

Note:

Unlike the procedure to delete a localization bundle, which completely removes the bundle from the development environment, the following procedure removes only the Java code library that contains the logic for language localization. Hence, the impact of performing the following procedure is not as great as it would otherwise be. Also note that this procedure does not uninstall the standard phrases.

Procedure

1. In the Navigator view or the Avaya OD Navigator view, click the speech project for which you want to uninstall the localization bundle.
2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left pane, click **Orchestration Designer**.
4. Click the **Languages** tab.
5. In the Localization Bundles area, click the localization bundle that you want to uninstall.

   Note:

   Verify that the **Installed** column shows a status of `true` for the selected localization bundle. If the status is `false`, the **Uninstall** button is disabled.
6. Click **Uninstall**.

   The **Installed** column status for the localization bundle changes to `false`.
7. Click **Apply** and **Close**.

Deleting a localization bundle

About this task

Note:

If your speech project contains only one localization bundle, you cannot delete the localization bundle.

Caution:

When you delete a localization bundle, Orchestration Designer removes the bundle completely from the development environment. Deleting localization bundles can also invalidate languages that exist in the speech application because the file that the prompt file editor of a speech application uses for audio variable formats, no longer exists. Finally, this procedure can result in code generation errors because the localization language code no longer exists. Exercise extreme caution when performing this procedure.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left pane, double-click **Avaya** > **Orchestration Designer** .
3. Click **Languages**.

   The right pane of the Preferences dialog box displays the Languages options.

4. In the Audio Localization Packages area, select the localization bundle that you want to delete.
5. Click **Delete**.

   Orchestration Designer displays a Question dialog box that asks you to confirm the deletion.
6. To confirm the deletion, click **OK**.

   The system displays a Warning dialog box that informs you that removing the localization bundle can invalidate projects.
7. To confirm the deletion and remove the localization bundle, click **Yes**.
8. In the Preferences dialog box, click **Apply**.

Installing the standard phrasesets of a localization bundle

About this task

Standard phrasesets are used, in conjunction with language localization bundles, to provide audio variable functionality to Orchestration Designer speech applications. In audio variables, Orchestration Designer takes the value of a variable and parses it in such a way that pre-recorded audio files can be used to play back certain standardized types of information. For more information about audio variables, see Audio Variable on page 676.

Note:

Installing standard phrases involves copying phrasesets and optionally the audio files. If the audio files are stored externally, then it is up to you to store the audio files on the external server and configure the phraseset to reference the external location.

Procedure

1. In the Navigator view or the Avaya OD Navigator view, click the speech project for which you want to install a set of standard phrasesets.
2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left pane, click Orchestration Designer.
4. Click the **Languages** tab.
5. In the **Project Languages** area, click the language for which you want to install the standard phrases.
6. Click **Edit**. The system displays the Project Language dialog box.
7. Click the cascading icon adjacent to **Localization Bundle Options** to display standard phrases installed (or not installed). Perform the following actions:
   a. In the **Standard Phrases** field, click **Install** to install the standard phrase audio files that are associated with the localization bundle.

To uninstall the installed standard phrase audio files that are associated with the localization bundle, click **Uninstall**.

b. Click the **Use standard phrases on an external server** check box to use the standard phrase audio files that are stored on an external server. If you select this check box, the system does not install the standard phrase audio files that are associated with the localization bundle in your speech project.

If you select this check box, then in the phraseset file, you must select **External**, and in the **URL base** field, type the URL base where the audio files are located. For more information, see Specifying the phrase audio file location on page 171.

8. Click **OK**.

Orchestration Designer copies the files based on the specified localization bundle options to the appropriate resource directories.

Chapter 19: Database operations

Database operations

Within Orchestration Designer, databases can be used in call flows and message flows . For example, during a call, a user wants to know the current status of an account, status of which is maintained within an SQL database. A call flow or message flow can be written to query the database for the account information of the user, including current status. Then this information can be assigned to audio or text variables to send the information back to the user.

Orchestration Designer uses database operation files to perform these database operations. To use database operation files, a database operation file must first be created and defined. Before creating a database operation file, however, one or more data sources in which to perform the operations must be configured. After the database operation file has been created and defined, it can be used within a Data node in a call flow or message flow application.

Data source management

Before performing any operations involving a database, you must configure Orchestration Designer to access the database.

Because Orchestration Designer is a Java-based tool, data sources for Orchestration Designer speech and message applications must be JDBC-compliant. If JDBC drivers are not installed on your computer, JDBC drivers must be installed to work with data sources. To install these drivers, see the documentation that came with your database software.

For more information about configuring Orchestration Designer to work with a JDBC database driver, see JDBC driver management on page 549.

When at least one data source is configured for the project, database operations can be created to make use of the data source in the project. To create database operations, use the Database Operation wizard. For more information, see Creating a database operation file on page 300.

Adding a JDBC data source

Procedure

1. In the Navigator view or the Avaya OD Navigator view, click the project in which you want to add a JDBC data source.
2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left pane, click Orchestration Designer.
4. Click the **Pluggable Connectors** tab.
5. In the **Category** field, click **Data Access**.
6. In the lower pane, select the check box corresponding to **Database**.

   The system displays the Database Configuration pane.
7. Click **Add**.
8. In the Create a new datasource dialog box, specify the information to create a JDBC data source, and then click **OK**.
9. In the Properties for <project name> dialog box, click **OK**.

Create a new data source dialog box field descriptions

Name	Description
Name	A name for the data source. **Note:** The data source name is used only within Orchestration Designer.

Table continues…

Name	Description
Driver Class Name	The name of the driver class to use with this data source. For example, **oracle.jdbc.OracleDriver** **Note:** The WAR file for Tomcat contains the context file that is automatically deployed in the Tomcat directory for the application. Tomcat uses this context file to create and manage data source connections. To modify the data source information, such as the driver class and URL without going through the design environment, go directly into this context file. The file exists in the `<Tomcat_Home>\conf\Catalina\localhost\` directory with the <*application_name*>.xml naming. To modify the driver class or URL, locate the Resource element which has the corresponding data source in its dataSourceName attribute, and change the URL or driver attribute of the same element. See http://commons.apache.org/dbcp/configuration.html for more details on DBCP configurations.
Connection URL	The URL to the data source. For example, **jdbc:oracle:thin:@148.147.46.68:1521:orcl** This URL can be either external to the local system or somewhere on the local system. For more information about what to enter in this field, see the documentation for your database software.
Username	A login user ID.
Password	A login password. For more information about what to enter in these fields, see the documentation for your database software.

Editing a JDBC data source

Before you begin

Before you edit a JDBC data source, verify that you have the appropriate JDBC driver installed and configured. For more information about installing and configuring the JDBC driver, see JDBC driver management on page 549.

Procedure

1. In the Navigator view or the Avaya OD Navigator view, click the project in which you want to edit a JDBC data source.

2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left pane, click **Orchestration Designer**.
4. Click the **Pluggable Connectors** tab.
5. In the **Category** field, click **Data Access**.
6. In the lower pane, select the check box corresponding to **Database**.

   The system displays the Database Configuration area.
7. In the JDBC Data Sources pane, select the JDBC data source that you want to edit, and then click **Edit**.

   The system displays the Edit the data source dialog box.
8. Modify the information. For more information, see Adding a JDBC data source on page 296.

   Note:

   You cannot edit the data source name.
9. Click **OK**.

Deleting a JDBC data source

Procedure

1. In the Navigator view or the Avaya OD Navigator view, click the project from which you want to delete a JDBC data source.
2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left pane, click **Orchestration Designer**.
4. Click the **Pluggable Connectors** tab.
5. In the **Category** field, click **Data Access**.
6. In the lower pane, select the check box corresponding to **Database**.

   The system displays the Database Configuration area.
7. In the JDBC Data Sources pane, select the JDBC data source that you want to delete, and then click **Delete**.

Creating a failover data source

Procedure

1. In the Navigator view or the Avaya OD Navigator view, click the project for which you want create a failover data source.
2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left pane, click **Orchestration Designer**.
4. Click the **Pluggable Connectors** tab.
5. In the **Category** field, click **Data Access**.
6. In the lower pane, select the check box corresponding to **Database**.

   The system displays the Database Configuration area.
7. In the JDBC Data Sources pane, select the JDBC data source for which you want to configure a failover data source, and then click **Failover**.

   The system displays the Failover Path dialog box.
8. Click **Add** to select a failover data source.

   The system displays the Data Sources dialog box.

   Note:

   The Data Sources dialog box contains a list of data sources defined in the Database Configuration pane.
9. Click a data source for failover, and then click **OK**.

   Note:

   To designate more than one data source as failover data source, repeat steps 7 and 8. To set priority for the data sources that are used as failovers, in the Failover Path dialog box, select the failover data source, and then click **Up** or **Down** as necessary.
10. In the Failover Path dialog box, click **OK**.

Removing a failover data source

Procedure

1. In the Navigator view or the Avaya OD Navigator view, click the project from which you want remove a failover data source.
2. On the **Project** menu, click **Properties**

   The system displays the Properties for <project name> dialog box.

3. In the left pane, click Orchestration Designer.
4. Click the **Pluggable Connectors** tab.
5. In the **Category** field, click **Data Access**.
6. In the lower pane, select the check box corresponding to **Database**.

   The system displays the **Database Configuration** area.
7. In the **JDBC Data Sources** pane, select the JDBC data source from which you want to remove a failover data source, and then click **Failover**.

   The system displays the Failover Path dialog box.
8. Select the failover data source that you want to remove, and then click **Remove**.
9. Click **OK**.

Database operation management

You can use the database operation files to:

- Query a database and return the query results to one or more variables.
- Add, update, or delete records in a database.
- Run a procedure stored within a database.

Note:

If you try to create a database operation before data sources are configured, Orchestration Designer displays an error message. To configure data sources, see Data source management on page 295.

Creating a database operation file

Procedure

1. On the **File** menu, click **New** > **Other**.

   The system displays the New wizard.
2. Double-click **Avaya OD Development**.
3. Click **Database Operation File**, and then click **Next**.

   The system displays the New Database Operation wizard.
4. On the Create a Database Operation page, specify the data source name, the operation type, and a name for the database operation file, and then click **Next**.

5. On the Select Data Objects page, select the data objects to be included in the database operation file. Check all data objects that are to be included with the database operation file.

   Note:

   If you select the operation type as **Delete** on the Create a Database Operation page, the system does not display the Select Data Objects page.

6. Click **Next**.
7. On the Map to Variables page, map the columns and parameters to the variables.
8. Click **Finish**.

   The system creates the database operation file and stores it in the following directory: `<ProjectName>\connectivity\dboperations`. A suffix of `.dbop` is assigned to the new database operation file.

New Database Operation wizard: Create a Database Operation page field descriptions

Name	Description
Available Projects	The project to which the database operation file will be applied.
Data Source name	The database (data source) to use for this operation.
Operation	The type of operation you want to perform. Options include: • **Add**: Takes the values of selected variables in Orchestration Designer and creates a record in the database, populating the columns with the variable values. • **Delete**: Deletes designated records from the database. • **Execute**: Runs a procedure previously stored in the database. • **Query**: Retrieves the values of selected records in a database and assigns the values to variables in Orchestration Designer. • **Update**: Updates the values of selected records in the database with new values from selected variables in Orchestration Designer.

Table continues…

Name	Description
File name	Name identifying the database operation that describes the function of the database operation. For example, you can name an operation to query a database for the account status of a caller as **QueryAcctStatus**. **Note:** Database operation file names must follow Java naming conventions (see Conventions for naming Java components on page 59).
Open file for editing	Check box to automatically open the Database Operation File Editor after you create the file by using the wizard. After you create a database operation file by using the Database Operation wizard, you must further define and set parameters for the database operation by using the Database Operation File Editor.

New Database Operation wizard: Map to Variables page field descriptions

Use the **Map to Variables** page of the **Map to Variables** wizard to:

- Select the columns or parameters that you want to map to variables in the operation.
- Map columns in the data source to variables. When the **Operation** type is **Add** or **Update**, this mapping determines which variable values will be written to the database.
- Map parameters in the data source to variables. When the **Operation** type is **Query**, this mapping determines which records are retrieved from the database, the values of which the query operation stores in the mapped variables.
- Map parameter of Array data type in a data source to a SQL type name.

To prevent the system from automatically creating variables and mapping the variables to the columns or parameter in the data source, you must create custom variables to map to *before* attempting to map any variables in the Database Operation Wizard. For information about creating custom variables, see Creating a variable on page 231.

The following table describes the options available in the Map to Variables page.

Field	Description	Operation
Column Name	The columns that will perform the operation.	Add Query Update

Table continues…

Field	Description	Operation
Column Type	Displays the database type for each column. This type helps determine what type of variable to use. The database vendor defines the types for columns. For more information about the types, see your database documentation.	Add Query Update
Parameter Name	Displays the parameter names for inputs that the stored procedure expects to receive.	Execute
Parameter Type	Displays the type of the expected input parameter.	Execute
Variable Name	The variable to map the column or parameter to. If the variable is a complex variable, select a **Variable Name**. If a variable is selected, the Auto Create option is not available.	Add Query Update Execute
Variable Field	If the variable selected in the **Variable Name** field is a complex variable, select a field to map the column or parameter.	Add Query Update Execute
Auto Create	This option automatically creates and names a variable to map the column or parameter to. If selected, the **Variable Name** and **Variable Field** fields are disabled.	Add Query Update Execute
Creating one complex variable for all the columns	This option creates a complex variable with a field for each column, which helps keep variables used in the database operation grouped together. If this option is not selected, a simple variable will be created for each column.	Add Query Update Execute
Result set returned	Check box to specify the number of columns expected to return if you are creating the stored procedure operation that returns a resultset. You can click **Next** to open the Map to Variables page to map the **resultset** column to variables.	Add Query Update Execute
SQL Type	The SQL type name when the parameter in a stored procedure is of Array data type. The SQL type name should match with the array name defined in the stored procedure. Qualify the SQL type name with the schema name. For example, if param1 is defined as “param1 out ID_ARRAY” in the stored procedure, you should specify the SQL type name as OD.ID_ARRAY, where OD is the schema name.	Execute

Database Operation File Editor

You can use the Database Operation File Editor to further define database operations. In general, this is done by setting conditions in which the database operation file compares a constant or variable value in Orchestration Designer with a specified value or set of values in a column of the database. The operation then operates only on the records that meet the specified conditions.

The Database Operation File Editor opens in the main editor area of Orchestration Designer. For general information about page tabs, see Editor view tabs on page 42.

The Database Operation File Editor can have two or three tabs associated with it, depending on the type of operation file that was created to perform. Furthermore, the fields and options for each tab vary depending on what type of operation the file was created to perform.

Editing a database operation file

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, right-click the ***.dbop** file that you want to edit, and then click **Open With** > **Database Operation Editor**.

   The system opens the database operation file in the Database Operation File Editor.

2. Modify the information in the database operation file.

Remapping variables to columns in a database operation file

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, right-click the `*.dbop` file in which you want remap the variables.
2. Click **Open With** > **Database Operation Editor**.

   The system opens the database operation file in the Database Operation File Editor.

3. Click the **Database Operation** tab.
4. In the **Variable Name** field, click the variable to map to the corresponding column name or parameter.
5. If the variable is a complex variable, then in the **Variable Field** field, click the complex variable field to map to the corresponding column name or parameter.

Determining the order for the query return results

About this task

With Query operations, you can determine the order in which the return results are presented.

For example, you have a database operation that returns a list of all customers who opened accounts during the past week. The query returns each customer's first name, last name, and account number. You want the results returned and sorted by last name as the first column. If there are two or more customers with the same last name, you want the results then returned and sorted by first name. You do not want the returns sorted by account number at all.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, right-click the `*.dbop` file in which you want determine the order for the return results.
2. Click **Open With** > **Database Operation Editor**.

   The system opens the database operation file in the Database Operation File Editor.
3. Click the **Database Operation** tab.
4. In the **Order By** field for the last name, enter `1`.
5. In the **Order By** field for the first name, enter `2`.
6. Leave the **Order By** field for the account number blank.

Database operation file editor: Database Operation tab field description

For a **Query** or **Insert** operation on a table, you may need to adjust the columns that are affected by the operation without using the New Database Operation wizard to re-create the operation. For the **Select** operation, you can add another table to do a join, or to just add other columns from the same table. For the **Insert** operation, it makes sense to only add other columns from the same table as defined in the operation.

Use the **Database Operation** tab of the Database Operation File Editor to:

- View basic information about the database operation file.
- Remap variables to columns. For more information, see Remapping variables to columns in a database operation file on page 304.
- Determine the order in which return values are to be returned and sorted (**Query** operations only). For more information, see Determining the order for the query return results on page 305.

Note:

Not all of the following information is displayed for all operation types. A **Delete** operation displays only the first four types. Other exceptions are as noted in the field descriptions.

Name	Description
Datasource name	Displays the name of the data source as it appears in Orchestration Designer.
Operation	Displays the type of operation this file performs. For more information about operation types, see New Database Operation wizard: Create a Database Operation page field descriptions on page 301.
Distinct select	(**Query** operations only) Return only one record for all the records that have the same data in the returned columns.
Table name	Displays the name of the table in which the database operation is to operate.
Type	Displays whether the database operation performs within a table or executes a procedure. • Query - creating a select statement on a table • Add - creating an insert statement on a table. • Delete - creating a delete statement on a table. • Execute - create an execute statement to execute a procedure.
Column Name	(Not in **Execute** operations - Display only) Displays the name of the column in which the operation is to be performed.
Parameter Name	(**Execute** operations only- Display only) Displays the names of all input and output parameters used by the stored procedure.
Parameter Type	(**Execute** operations only- Display only) Displays the type associated with each input and output parameter used by the stored procedure.
Data Type	Displays what type of data is contained within the column or parameter. **Note:** This type helps determine what type of variable to use.
Variable Name	Displays the name of the variable currently mapped to this column or parameter. For information about remapping variables, see Remapping variables to columns in a database operation file on page 304.
Variable Field	Displays the name of the field associated with the variable if the variable selected in the **Variable Name** field is a complex variable.

Table continues…

Name	Description
Order By	(**Query** operations only) Displays order selections for the results to be returned. For more information about this field, see Determining the order for the query return results on page 305.
... button	Changes a data source name. If the data source does not match the database schema, there will be runtime errors.
Add	Adds data objects to the database operation file. On clicking, the system displays a dialog box with a list of data objects that you can select from, which comes from the same data source defined in the operation.
Remove	Removes the selected column.

Database operation file editor: Predicate tab

Use the **Predicate** tab to set the conditions that determine which records are operated on. For example, you want the database operation to query the database and return all records for customers who are delinquent in their payments. You can use the **Predicate** tab to set the conditions and instruct the database which records to return.

Note:

The **Predicate** tab appears and is used only with Delete, Query, and Update operations.

Compound conditions can be created within the Predicate tab. These conditions are used for nesting two conditions (Simple, Join, or Compound) that are joined by a Boolean operator. In addition, compound conditions can be nested to build complex Where clauses. For more information, see Setting compound condition for a database operation on page 311.

Database operation file editor: SQL Query tab field descriptions

The **SQL Query** tab shows the query code to be sent to the SQL server when this database operation file is executed. This is helpful to verify that the database operation does what you expect it to do. Be sure that you are familiar with SQL query commands and syntax. By testing the SQL query before actually running the application, you can find problems in the logic earlier in the design process.

By plugging in different values for variables (through simple or compound logic defined on the **Predicate** tab, the values are displayed in the **Criteria Input** field on the **SQL Query** tab), and upon executing, you can test the query and see the results.

In general, if there is an error when executing a Database operation, a message box is displayed with the error message.

Note:

By right-clicking in the grid, a **Copy** option is available for copying selected rows into Notepad.

Following are the examples of supported SQL database operations. With the combination of these supported operations, the database data can be easily manipulated without the need for an external tool.

- Example SQL Select query on page 308
- Example SQL Insert query on page 308
- Example SQL Delete query on page 309
- Example SQL Update query on page 309
- Example stored procedures SQL query on page 309

Name	Description
SQL Query	Query code to be sent to the SQL server when the database operation file is executed.
Execute	To test the SQL query. The system returns the query results in the grid below the button. The result grid displays columns that match the select statement.
Criteria Input	The values by plugging in different values for variables (through simple or compound logic defined on the **Predicate** tab.
<Bottom grid>	The query results.

Example SQL queries

Example SQL Select query

For example:

```
SELECT Customer.Cust_ID, Customer.First_Name, Customer.Last_Name
FROM dbo.Customer
WHERE ( Customer.Cust_ID = $[CustomerSQL:Cust_ID} )
```

Criteria Input Example:

Cust_ID: `1`

Click **Execute** and the results grid indicates the First and Last Name for customer with the ID of 1.

Example SQL Insert query

For example:

```
FROM dbo.Customer
( Cust_ID, First_Name, Last_Name )
VALUES( $[AddCustomerSQL:Cust_ID], $[AddCustomerSQL:First_Name], $
[AddCustomerSQL:Last_Name} )
```

Criteria Input Example:

Cust_ID: `3`

First_Name: `Jackie`

Last_Name: `Snow`

Click **Execute** and the results grid indicates the number of rows added (1, in this example).

Example SQL Delete query

For example:

```
DELETE FROM dbo.Customer
WHERE ( Customer.Cust_ID = ${CustomerSQL:Cust_ID} )
```

Criteria Input Example:

Cust_ID: `2`

Click **Execute** and the results grid indicates the number of rows deleted (1, in this example).

Example SQL Update query

For example:

```
UPDATE dbo.Customer
SET First_Name = ${UpdateCustomerSQL:First_Name}
WHERE ( Customer, Cust_ID = ${CustomerSQL:Cust_ID} )
```

Criteria Input Example:

First_Name: `Jackie`

Cust_ID: `3`

Click **Execute** and the results grid indicates the number of rows updated (1, in this example).

Example stored procedures SQL query

Example 1:

```
{ ${SPSQL:_RETURN_VALUE} = call dbo.getCustomerById;
1 ( ${SPSQL:_Param1) }
```

Criteria Input Example 1:

@Param1: `1`

Click **Execute** and the results grid indicates the result of the query.

Example 2:

{ call DDTEST,GETCUSTOMER(${ExecuteOracleDB:PARAM1) ${ExecuteOracleDB:PARAM2})

Criteria Input Example 2:

PARAM1: 2

Output Parameters:

Tom

Click **Execute** and the results grid indicates the result of the query.

Setting single condition for a database operation

About this task

A **Simple** condition sets a single condition that affects the operation.

For example, you want the database operation to return the names of customers who opened accounts within the past 30 days. You have already set up the database operation file to return the names of customers, but you want to use a **Simple** condition to return information for only those customers who have opened accounts in the past 30 days.

In a case like this, you would set up the condition to look at the column containing the date the account was opened, and return only those records for which the date is 30 days or less from the current date.

To use two conditions for a single database operation, use the **Compound** condition setting. For more information, see Setting compound condition for a database operation on page 311.

Procedure

1. From the palette, under **Condition**, select **Simple** and drag it into the workspace.
2. From the palette, under **Operand**, select **Column** and drag it into the workspace.
3. In the workspace, select the **Column** operand that you dragged, and then, in the Avaya Properties view, **Name** field, select the column to use in testing the condition.
4. From the palette, under **Operator**, select **Comparison** and drag it into the workspace.
5. In the workspace, select the **Comparison** operator that you dragged, and then, in the Avaya Properties view, **Operator** field, select the operator to use in making the comparison between the column value and the variable or constant value.
6. From the palette, under **Operand**, select one of the following operands and drag it into the workspace:
   - **Variable**: To compare the column value with a variable value, use this option.
   - **Value**: To compare the column value with a constant or absolute value, use this option.
7. In the Avaya Properties view, perform one of the following actions:
   - If you selected **Variable** as the second **Operand**, select from the **Name** drop-down list the variable to use for the comparison.

     If the variable is a complex variable, you must also select a field from the **Field** drop-down list.
   - If you selected **Value** as the second **Operand**, enter in the **Value** field the exact value to be used for the comparison.

Setting compound condition for a database operation

About this task

A **Compound** condition sets two conditions that both affect the operation.

For example, you want the database operation to return the names of customers who opened accounts within the past 30 days. You have already set up the database operation file to return the names of customers, but you want to use a **Compound** condition to return information for only those customers who have opened accounts in the past 30 days and who have an outstanding balance of more than $500.

In a case like this, you would set up a **Compound** condition with two **Simple** conditions: The first Simple condition looks at the column containing the date the account was opened, and return only those records for which the date is 30 days or less from the current date. The second Simple condition looks at the current balance column and returns only those customers from the first set of returns that have an outstanding balance of more than $500. The final returns from this operation would contain the names of only those customers who meet both conditions.

Note:

Each **Compound** condition permits you to set only two conditions. However, to set more than two conditions, you can nest **Simple** and **Compound** conditions within other **Compound** conditions to attain the number and combination of conditions you need.

If you want to use only a single condition for a database operation, use the **Simple** condition setting. For more information, see Setting single condition for a database operation on page 310.

Procedure

1. From the palette, under **Condition**, select **Compound** and drag it into the workspace.
2. Use the procedure in Setting single condition for a database operation on page 310 to add two **Simple** conditions to the **Compound** condition.

Example

The following figure shows an example of a compound condition.

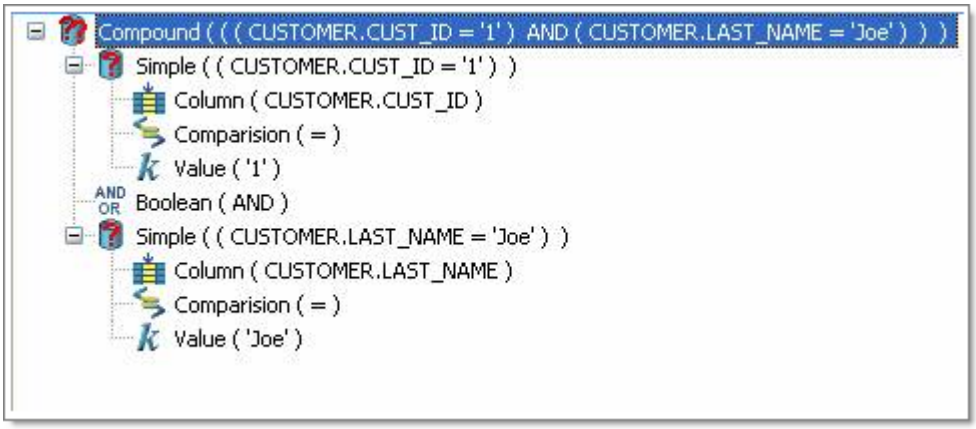


Figure 1: Example of a Compound Condition

Employing a database operation file in a call flow and message flow

About this task

After the database operation file has been created and defined, you can use it in your call flows and message flows. You can use database operation files only within a Data node.

Procedure

1. Drag a Data node into the Call Flow or Message Flow Editor main workspace.
2. To open the Data node editor, double-click the Data node.
3. From the Data node editor palette, select the **Database** item and drag it into the editor workspace.
4. In the Avaya Properties view, **Name** field, select the database operation file to use.

   Alternately, from the Avaya OD Navigator view, select the database operation and drag it into the callflow or message flow.

Database connector BLOB support

Orchestration Designer supports writing and retrieving binary data from the GUI in DBOP editor. You can handle BLOB data from the GUI.

Writing BLOB data

In order to write BLOB data, such as a file, to a database field, the database field needs to be of a Blob type (or Image type in case of SQL Server). You will need to write a Java code for setting either a File or InputStream object to an Orchestration Designer variable that binds to the db operation field.

Example

```
IVariable variable = mySession.getVariable("InsertTestBlob");
IVariableField field = variable.getComplexVariable().getField("MyImage");
String projectPath = mySession.getAbsoluteProjectFilePath();
File file = new File(projectPath + "/test.wav");
try {
      field.setValue(new FileInputStream(file));
      } catch (FileNotFoundException e) {
                  e.printStackTrace();
      }
```

Using a database query to retrieve BLOB data

To retrieve BLOB data, the db query operation is executed. The variable contains a base64 encoded string, which you have to decode into a byte array.

Example:

```
IVariable variable = mySession.getVariable("GetTestBlob");
                IVariableField field =
variable.getComplexVariable().getField("MyImage");
                String projectPath = mySession.getAbsoluteProjectFilePath();
                try {
                        FileOutputStream fos = new FileOutputStream(projectPath + "/
copytest.wav");                                 byte[] bytes =
Base64.decode(field.getStringValue());
                        fos.write(bytes);
                        fos.close();
                } catch (Exception e) {
                        e.printStackTrace();
                }
```

Retrieving a WAV file from a database without writing a Java code

About this task

You can add .wav files to the database and retrieve it without having to write a Java code.

Procedure

Assign a URL/file path and name to the variable that maps to the **add/update** operation.

Note:

Assign the file path relative to the project root.

Because the "query" operation automatically saves the BLOB into a file in the project temp folder, you just have to use the **Text Variable** and "filename" format for the play back.

Changing the behavior of the query for retrieving BLOB data

You can also change the query behavior by coding using the **setCreateTempFileForBlob(boolean flag, String fileType)** method inside the `dbop` Java class.

Example:

setCreateTempFileForBlob(false, null) - In this case, you will get back the bytes in Base64.

setCreateTempFileForBlob(boolean true, "gif") - In this case, the file created in the temp directory will have "gif" as the extension.

Chapter 20: Events

Events

In the context of Orchestration Designer, the term **event** refers to an abnormal or unexpected situation which occurs during application run time. These can include error situations, for example, when the application tries to retrieve a speech or message resource that is unavailable. Events can also occur when users do not respond as expected, either with responses for which the system cannot find a match or by not responding at all. In a speech application, an event can occur when callers do not respond as expected. For example, no response or response for which the system cannot find a match.

In a message application, an event can occur when the incoming message does not find an appropriate message application to handover the inbound text message.

The W3C VoiceXML Specification and Orchestration Designer both include a number of built-in event types that any VoiceXML or textXML-compliant server can recognize and handle. Orchestration Designer makes it easy for you to create your own custom event types to use in your applications.

In general, when an Orchestration Designer speech application or message application encounters a situation to which it does not know how to respond, or which meets predefined conditions, then Orchestration Designerthrows an event. The Avaya Application Simulator then searches to locate the appropriate *event handler* to handle, or "catch," the event. When Orchestration Designer locates the appropriate event handler, it responds according to the instructions provided in the event handler.

Event handlers and scope

Event handlers can be added at several different levels within the speech or message application:

- In the **AppRoot node**: Event handlers added to the AppRoot node act as global event handlers. These event handlers are active and ready to handle events that occur anywhere, at any level, in the application.

  See About the AppRoot node on page 132.

- In any other node: Event handlers added at the top level of any node other than the AppRoot node act as form-level event handlers. These event handlers are active and ready to handle events that occur anywhere within the node in which they are added.

- In an item within a node: Event handlers attached to node items within a node act as field-level event handlers. These event handlers are active and ready to handle events that occur only within the item to which they are attached.

Event handlers at a lower level take priority over event handlers at a higher level. For example, you have an event handler for No Match events at the field level, and you have another event handler for No Match items in the AppRoot node. If a No Match event occurs at the field level, the event handler at the field level takes precedence over the event handler in the AppRoot node.

Types of event handlers

Orchestration Designer offers two basic types of events:

- *Built-in events* are events for which Orchestration Designer has incorporated event handlers. The built-in events include the No Input, No Match, error.semantic, error.badfetch, and Internal Error events. These events, except the Internal Error event, are defined as “default catch elements” by the W3C VoiceXML Specification.

  For more information about these event handlers, see Built-in event handlers on page 315.

  Note:

  The No Input and No Match events are available for both speech and message applications. However, the error.semantic, error.badfetch, and Internal Error events are specific only to message applications.

- *Custom events* are events that you define and create handlers for within your Orchestration Designer speech applications.

  For more information about creating and handling custom events, see Custom events on page 316.

Built-in event handlers

The W3C VoiceXML 2.0 Specification specifies a number of events for which VoiceXML 2.0-compliant applications and browsers are expected to provide implicit event handlers. Orchestration Designer complies with this standard, by providing the following built-in event handlers:

Note:

For details on each event handler, click the appropriate link.

- No Input on page 769
- No Match on page 770
- On Disconnect on page 775 (exit)

If you do not use these event handlers anywhere, Orchestration Designer still employs them on a global basis to handle these types of events with a built-in default response. Use these node items when you want to override the default response or customize the event handling for these types.

In addition, Orchestration Designer has built-in event handlers for maxspeechtimeout and connect.disconnect events, as required by the VoiceXML 2.0 recommendation.

For more information about using built-in events, see Event handling process in applications on page 317.

Custom events

Custom events are events that you define and create handlers for within your Orchestration Designer speech or message applications. These events can be assigned any name. For example, “myevent.thishappened”. The W3C VoiceXML 2.0 Specification defines various event and error events.

For example, you create an event handler to handle a situation in which the account balance for a user is below a certain amount. In this case, there can be a node of the call flow or message flow in which the account balance is retrieved from a database, and the balance is assigned to a variable. You can use a Data node to compare the balance with a set amount, and if the balance is less than that amount, the node operation can then throw a custom event called **BalanceLow**.

You can then use a **Catch** event in the AppRoot to handle this event when it is thrown from anywhere in the application. Several nodes can throw this particular event. You can then set up the **Catch** event to transfer the caller to a live agent to talk about the account balance and offer overdraft protection services.

Note:

When catching errors, only catch individual specific errors and do not use a “catch all” such as “error.*” or “.”. Catching an error and taking inappropriate actions can lead to unpredictable results.

Use the Event Types Editor to create custom event types. For more information, see Creating a custom event on page 318.

After the event type is created, you can use it in your applications by throwing the event when the situation or condition is met and then catching it elsewhere in the application. For more information about this practice, see Event handling process in applications on page 317.

Try Catch event handling

Try/Catch elements in the **Data** node will *try* to execute items under the **Try** item. To handle exceptions, add **Catch** items (under the parent Try).

By default, **Catch** <*> will catch *all* exceptions. **Catch** can also be used to catch specific exception types for handling specific errors differently. Comma separated exception types can also be specified for catching different exception types, but handling them in the same way.

List of common exceptions are provided in the Properties view. Currently this list contains:

- SQLException
- IOException
- SCERuntimeException

A new variable caller **ddLastException** automatically captures the exception caught by a Catch item (includes the following data members: errorcode, message, object, stacktrace, and type).

Following is an example of the Try/Catch mechanism.

Event handling process in applications

As described in general terms in Events on page 314, the basic process for events in applications involves the following steps:

1. The application encounters an unexpected condition or error, known as an **Event**.
2. The Avaya Application Simulator identifies the unexpected condition or error and throws an event that matches it, as closely as possible.
3. Starting at the level where the event was thrown, the application works its way up from field to form to global level until it finds an appropriate event handler to catch the event. If no event handler is defined, the application uses one of the implicit (built-in) event handlers.
4. The event handler instructs the application how to handle the event.

Orchestration Designer implements this basic process in different ways, depending on what type of event is thrown and whether there is an event handler set to catch the event. Generally, Orchestration Designer has two ways of handling events, depending on whether the event is a built-in or a custom event.

Built-in event handlers for default events

The Avaya Application Simulator (AAS), which is used to run Orchestration Designer applications, has built-in event handlers for the “default catch items” as defined and required by the VoiceXML version 2.0 recommendation. This means that, for the default catch items, you are not absolutely required to use event handlers, because the AAS can handle them automatically.

These built-in event handlers are very generic in their responses and may not respond to a given event the way you want. Therefore, identify where and how to use the built-in Orchestration Designer event handlers for these events:

- Use built-in event handlers in the AppRoot node to catch any occurrences of those events for which you do not have event handlers at lower levels. For more information about using the AppRoot node for global settings, see About the AppRoot node on page 132.
- Use built-in event handlers at the node (form) or node item (field) level within a node, to catch and respond in a specific way to those events that are thrown at those levels.

For example, you have a generic No Match event handler that says “I’m sorry, I do not understand you.” This can be used for a global response, but for a particular Prompt and Collect node for speech application or Collect node for message application, you want a more specific response such as “I’m sorry, that is not one of the options. For checking account information, please say or press ’one’; for savings account information, please say or press ’two’.” In this case, add a **No Match** event handler to your Prompt and Collect node or Collect node.

Note that it is not necessary, with the default catch items, to use a **Throw** item to throw the event. This functionality is already programmed into the AAS, and the system knows when to throw these events. All you need do is provide the event handler so the system knows how to respond when the event is thrown.

Creating a custom event

About this task

Use the Event Types editor to define and modify custom events and their implementation within a project. This editor is available only from within the Call Flow and Message Flow Editor.

Also see Custom events on page 316.

Procedure

1. Perform one of the following actions:
   - Select the Call Flow or Message Flow Editor as the active workspace and click the **Event Types Editor** icon  on the main toolbar.
   - Open the Show Projects view of the Avaya OD Navigator view and double-click `project.events` file under the `flow` folder. For more information, see Avaya OD Navigator view toolbar options on page 38.

The system displays the Event Types editor.

2. In the Palette pane for the Event Types editor, click **Event Type**.
3. Click in the Event Types editor workspace.

   The system adds the event to the workspace.
4. Click the event in the workspace.
5. In the Properties view, in the **Name** field, type a name for the event.

   For information about how to use custom events, see Using and handling custom events on page 319.

Using and handling custom events

About this task

Using custom event types is different from using built-in event types in two important respects:

- You must create custom event types before you can use them.
- To create custom event types, see Creating a custom event on page 318.
- You must specifically use a **Throw** item to throw a custom event and a **Catch** event to handle it.
- There are three basic ways to throw a custom event. Perform one of the following actions:
  - Use a **Link** item in a node and then use the **Throw** item, attached to the **Link** item, to throw the event. For more information, see Link on page 762.
  - Use the **Throw** item within a **Catch** item. For more information, see Catch on page 693.
  - Use a **Throw** item as the only item in a Form node, and then use a condition within a Data node to branch to that Form node.

    For more information about the Form and Data nodes, see Application items (Basic nodes) on page 142.

    After you have set up a way to throw the custom event, you must make sure you have a **Catch** item, within the scope of the **Throw** item, to handle it.

Chapter 21: Web Services

Web services

The term **Web services** can be defined as a standardized way of integrating Web-based applications. These **Web services** use technologies such as XML (Extensible Markup Language), SOAP (Simple Object Access Protocol), WSDL (Web services Description Language), and UDDI (Universal Description, Discovery, and Integration). Generally, **Web services** use these technologies as follows:

- XML to tag the data
- SOAP to transfer the data
- WSDL to describe, in standard terms, the services available
- UDDI to list what services are available

For example, a Web service can be created to:

- Look up and return the current value of stocks
- Look up and return the weather forecast for various cities
- Book flight or car rental reservations

In short, **Web services**, like other forms of software, can be created to perform a wide range and variety of functions.

With Orchestration Designer, Web services can be employed within speech, call control, and message flow projects. A Web Service Operation wizard is used to create and define a Web service operation file. This file can be viewed or edited later (for example, to view settings and remap variables) using the Web Service Operation File Editor. You can use the Web service operation file within a speech, call control, or message application.

You can create three types of Web service operation files:

- **Web Service Operation File**: To create Web service operation files based on Axis 1.4. The Axis 1.4 Web service connector is shipped with Orchestration Designer 7.0.1 and above.
- **Web Service Operation File (Axis2)**: To create a Web service operation files based on Axis 2. Axis2 Web service connector is available with Orchestration Designer 7.0.1 and above. Avaya recommends that you use the Axis2 connector for development.

Important:

When selecting a Web service, do not select a service that uses overloading of operations. Overloading of operations is not supported in the SOAP 1.2 standard and is known to cause problems.

Note:

Axis ships with the *Tcpmon* tool for monitoring HTTP traffic. Avaya recommends that you use the Tcpmon tool to debug Web services. Documentation on how to use Tcpmon can be found at: http://ws.apache.org/axis/java/user-guide.html#AppendixUsingTheAxisTCPMonitorTcpmon

- **Web Service Operation File (REST)**: To create Web service operation files based on REST. The **Web Services (REST)** pluggable data connector is available with Orchestration Designer 7.0 and above.

Note:

You can create and use a REST Web service operation file only in a speech project and a message flow project.

Web service operation management (Axis)

Creating a Web service operation file

Procedure

1. On the **File** menu, click **New** > **Other**.

   The system displays the New wizard.

2. Double-click **Avaya OD Development**.
3. Click **Web Service Operation File**, and then click **Next**.

   The system displays the New Web Service Operation wizard.

4. On the Specify General Parameters page, specify the Web service operations parameters, and then click **Next**.
5. On the Map Input Parameters page, map the project variables to the input parameters, and then click **Next**.
6. On the Map Output Parameters page, map the project variables to the output parameters.
7. Click **Finish**.

Result

After defining a Web service using the New Web Service Operation wizard, the system creates the Web service operation file and stores it in the `<ProjectName>\connectivity\wsoperations` directory. Orchestration Designer assigns the new Web service operation file a suffix of `.wsop`.

New Web Service Operation wizard: Specify General Parameters page field descriptions

Name	Description
Available Projects	The project for which you want to create the Web service operation file.
File Name	Name of the Web service being created.
Open file for editing	Check box to automatically open the Web Service Operation File Editor after creating the file with the wizard.
WSDL URL	The URL of the WSDL file for the Web service in standard HTTP or HTTPS format. All Web services have an associated WSDL file that describes what services the Web service provides. Orchestration Designer uses this WSDL information to populate the **Operation** field list. **Note:** Orchestration Designer supports only WSDL files that contain unique method signatures. **Note:** Orchestration Designer does not support operation overloading, or the use of methods with the same name but different parameters. **Tip:** If a Web service has been previously created, Orchestration Designer remembers the URL. For future Web service creations, Orchestration Designer automatically populates this field with the URL. Orchestration Designer keeps track of the last 20 Web services URLs used in operations.

Table continues…

Name	Description
Authentication	How Orchestration Designer should authenticate with the Web service host. Options are: • **No Authentication** • **Basic** – Transmits the username/password pair in an unencrypted form from browser to server in the HTTP header. • **Digest** – Transmits the username/password pair encrypted from the browser to server in the HTTP header.
User Name	A username for authentication. The **User Name** field is available only when you select **Basic** or **Digest** in the **Authentication** field.
Password	(Optional) A password for authentication. The **Password** field is available only when you select **Basic** or **Digest** in the **Authentication** field.
Load operations from WSDL file	To load the Web services operations from the WSDL file.
Timeout	Sets the maximum time, in seconds, for fetching and parsing the WSDL file.
Use Document Wrapping	Check box to wrap multiple input and output parameters in a class.
Allow WSDL Imports	Check box to allow the WSDL file to reference other files by using the WSDL import statement.
Operation	The service to be used in this operation. Be sure that you know the following information: • What services are offered by the Web service • What form the services take
Package for types	The package name for any Java beans that are generated as a result of defining a Web service operation. The default value is “connectivity.ws.beans”.

New Web Service Operation wizard: Map Input Parameters page and Map Output Parameters page field descriptions

Use the **Map Input Parameters** page and **Map Output Parameters** page to map the input and output parameters expected by the Web service to the values of variables in Orchestration Designer. For example, for a Web service that provides a weather report, the expected input can be the name of a city; the expected output can be the weather report string (mapped to the appropriate variable).

Note:

If you do not want Orchestration Designer to automatically create the variables to map to the parameters, you must create custom variables *before* attempting to map any parameters. For more information, see Creating a variable on page 231.

Otherwise, variables can be remapped later, using the Web Service Operation File Editor. For more information, see Editing a Web service operation file on page 330.

Name	Description
Parameter Name	Name of the input/output parameter defined in the Web service.
Parameter Type	Data type that the input/output parameter expects from Orchestration Designer. Make sure that the variable assigned to each parameter is of the correct type.
Variable Name	The variable to map to the parameter. If a complex variable, also select a **Variable Field**. If a variable is selected in this field, the Auto Create option *cannot* also be selected.
Variable Field	To map to the parameter required if the variable selected in the **Variable Name** field is a complex variable.
Auto Create	Check box to have Orchestration Designer automatically create and name the variable mapped to the parameter, select this option. If selected, Orchestration Designer clears the **Variable Name** and v values, if previously set.

Table continues…

Name	Description
Use Java Obj	Indicates whether the content of the variable should be treated as a Java object to prevent Orchestration Designer from mapping it. Use this option when the structure of the data to be sent to the Web service is more complex than what can be contained within a normal variable. When using this option, subsequently, the following steps must be taken: • Create the Java object and store it in the variable, if sending the object to the Web service. • Write the required Java code by overriding an existing method, such as the following method: **requestBegin (SCESession mySession)** • Retrieve the Java object from the variable, if receiving the object from the Web service. For more information, see Example on when to set the Use Java Obj option on page 325. For more information about the Orchestration Designer API and available methods, see the Orchestration Designer *Programmer Reference* guide in the online Help system.

Example on when to set the “Use Java Obj” option

Use this option when the structure of the data to be sent to the Web service is more complex than what can be contained within a normal variable. For example, you use a Web service that requires an object with the following structure:

```
Person
 Name
  First
  Last
 Age
 Gender
```

Orchestration Designer can handle only one level of nested variables using complex variables. That means, in this example, Orchestration Designer can send only the **Person**, **Name**, **Age**, and **Gender** values. Orchestration Designer cannot, however, return the second level, that is, the **First** and **Last** values.

By selecting this option when creating or editing a Web service operation file, at run-time, the **Person** object is treated as a single Java object, including all nested values. You can use custom Java code then to send all the values for the object that are required.

When using this option, you are responsible for creating the Java object and storing it in the variable, when sending the object to the Web service. To do this, you must write the required Java code. Usually, this is done by overriding an existing method, such as the

requestBegin(SCESession mySession) method. For more information about the Orchestration Designer API and the methods you can use for this, see the Orchestration Designer *Programmer Reference* guide in the online Help system.

Creating an Axis2 Web service operation file

Procedure

1. On the **File** menu, click **New** > **Other**.

   The system displays the New wizard.

2. Double-click **Avaya OD Development**.
3. Click **Web Service Operation File (Axis2)**, and then click `Next`.

   The system displays the New Axis2 Web Service Operation wizard.

4. On the Specify General Parameters page, specify the Web service operations parameters, and then click **Next**.
5. On the Select Method page, specify the service and method to invoke, and then click **Next**.
6. On the Map Input Parameters page, map the project variables to the input parameters, and then click **Next**.
7. On the Map Output Parameters page, map the project variables to the output parameters, and then click **Next**.
8. Click **Finish**.

   After defining a Web service using the New Axis2 Web service Operation wizard, Orchestration Designer creates the database operation file and stores it in the `<ProjectName>\connectivity\wsoperations` directory. Orchestration Designer assigns the new database operation file a suffix of `.wsop2`.

New Axis2 Web Service Operation wizard: Specify General Parameters page field descriptions

Name	Description
Available Projects	Select the project for which you want to create the Web service operation file.

Table continues…

Name	Description
WSDL URL	The URL of the WSDL file for the Web service in standard HTTP or HTTPS format. All Web services have an associated WSDL file that describes what services the Web service provides. Orchestration Designer uses this WSDL information to populate the web service information on the next page of the wizard. **Note:** Orchestration Designer supports only WSDL files that contain unique method signatures. **Tip:** If a Web service has been previously created, Orchestration Designer remembers the URL. For future Web service creations, Orchestration Designer automatically populates this field with the URL. Orchestration Designer keeps track of the last 20 Web services URLs used in operations.
Browse	If the WSDL file is stored on the local computer, click Browse to navigate to the WSDL file.
Authentication	If you select the Authentication check box, the system uses the user name and password entries to authenticate with the Web service host while fetching the WSDL file. The authentication protocol is automatically selected. Orchestration Designer supports: • **Basic**: Transmits the username/password pair in an unencrypted form from browser to server in the HTTP header. • **Digest**: Transmits the username/password pair encrypted from the browser to server in the HTTP header.
User Name/Password	The User Name and Password fields are enabled only if you select the Authentication check box. Enter a username and optionally a password for authentication.
WSOP File Name	Name of the Web service being created.

Table continues…

Name	Description
Java Output Package	Specifies the package name for any Java beans that are generated as a result of defining a Web service operation. The default value is **connectivity.ws.beans**. **Note:** For Axis2, you cannot use the same package name for multiple web service operations.
Load WSDL file	Click this button to load the Web services operations from the WSDL file.

New Axis2 Web Service Operation wizard: Select Method page field descriptions

Name	Description
Service	The service and method to invoke.
Show input parameters unwrapped	If selected, the input parameters to the methods are displayed without wrapping the java class.
Show output parameters unwrapped	If selected, the output parameters from the methods are displayed without wrapping the java class.

New Axis2 Web Service Operation wizard: Map Input Parameters page and Map Output Parameters page field descriptions

Use the **Map Input Parameters** page and **Map Output Parameters** page to map the input and output parameters expected by the Web service to the values of variables in Orchestration Designer. For example, for a Web service that provides a weather report, the expected input can be the name of a city; the expected output can be the weather report string (mapped to the appropriate variable).

Note:

If you do not want Orchestration Designer to automatically create the variables to map to the parameters, you must create custom variables *before* attempting to map any parameters. For more information, see Creating a variable on page 231.

Otherwise, variables can be remapped later, using the Web Service Operation File Editor. For more information, see Editing a Web service operation file on page 330.

Name	Description
Parameter Name	Name of the input/output parameter defined in the Web service.

Table continues…

Name	Description
Parameter Type	Data type that the input/output parameter expects from Orchestration Designer. Make sure that the variable assigned to each parameter is of the correct type.
Variable Name	The variable to map to the parameter. If a complex variable, also select a **Variable Field**. If a variable is selected in this field, the Auto Create option *cannot* also be selected.
Variable Field	Field to map to the parameter required if the variable selected in the **Variable Name** field is a complex variable.
Auto Create	Check box to have Orchestration Designer automatically create and name the variable mapped to the parameter. If selected, Orchestration Designer clears the **Variable Name** and **Variable Field** values, if previously set.
Use Java Obj	Indicates whether the content of the variable should be treated as a Java object to prevent Orchestration Designer from mapping it. Use this check box when the structure of the data to be sent to the Web service is more complex than what can be contained within a normal variable. When using this option, subsequently, the following steps must be taken: • Create the Java object and storing it in the variable, if sending the object to the Web service. • Write the required Java code by overriding an existing method, such as the following method: **requestBegin (SCESession mySession)** • Retrieve the Java object from the variable, if receiving the object from the Web service. For more information about the Orchestration Designer API and available methods, see the Orchestration Designer *Programmer Reference* guide in the online Help system.

Editing a Web service operation file

About this task

You can use the Web Service Operation Editor to:

- View information about the Web service operation file.
- Remap variables to parameters.
- Assign parameters to use Java objects.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the `*.wsop` or `*.wsop2` file that you want to edit.

   The system opens the Web service operation file in the Web Service Operation Editor.

2. Modify the information.

Web Service Operation Editor field descriptions

Name	Description
Web Service Information	
Service	Displays the name of the Web service as defined in the WSDL file.
Operation	Displays which service, or operation, within the Web service this operation uses. The name of this operation is as defined in the WSDL file.
Style	Displays one of the following values: • **rpc** • **document** • **wrapped** See the W3C Web Services Description Language (WSDL) 1.1 specification. **Note:** This field is not available for the Axis2 connector.

Table continues…

Name	Description
Use	Displays one of the following values: • **literal** • **encoded** See the W3C Web Services Description Language (WSDL) 1.1 specification. **Note:** This field is not available for the Axis2 connector.
Beans Package	Displays the name of Java beans package that Orchestration Designer uses to implement the Web service.
Name Space	Displays the URL to the location where the Web service resides. **Note:** This field is not available for the Axis2 connector.
Endpoint	Displays the endpoint used to invoke the Web service. The required endpoint for the Web service is available within the WSDL file for the Web service. **Note:** When deploying the application, this endpoint can be changed to another endpoint, in Export Orchestration Designer Project wizard: Configure Web Application Descriptor page on page 430. This change can be useful, for example, if you have been using a pared-down version of the Web service while developing and testing your application, you want to use a different, official version of the Web service with the deployed application.
Timeout	Sets the HTTP socket timeout, in seconds, to be used when executing the Web service operation.

Table continues…

Name	Description
WSDL URL	The URL of the WSDL file for the Web service in standard HTTP or HTTPS format. All Web services has an associated WSDL file that describes what services the Web service provides. Orchestration Designer uses this WSDL information to populate the **Operation** field list. **Note:** Orchestration Designer supports only WSDL files that contain unique method signatures. **Tip:** If a Web service has been previously created, Orchestration Designer remembers the URL. For future Web service creations, Orchestration Designer automatically populates this field with the URL. Orchestration Designer keeps track of the last 20 Web services URLs used in operations.
Input Parameters	
Input Parameters	Displays the parameters that were configured when the Web service was created. You can edit these parameters. Using these fields, variables can be remapped after a Web service operation file is initially created. This is useful when custom variables are created after creating a Web service operation file. See New Web Service Operation wizard: Map Input Parameters page and Map Output Parameters page field descriptions on page 323 for more information.
Output Parameters	
Output Parameters	Displays the parameters that were configured when the Web service was created. You can edit these parameters. Using these fields, variables can be remapped after a Web service operation file is initially created. This is useful for when custom variables are created after creating a Web service operation file. See New Web Service Operation wizard: Map Input Parameters page and Map Output Parameters page field descriptions on page 323 for more information.
Authentication	

Table continues…

Name	Description
Authentication	Indicates how Orchestration Designer should authenticate with the Web service host. Displays the authentication configured when the Web service was created. You can edit the authentication. The options are: • **Basic**: Transmits the username/password pair in an unencrypted form from browser to server in the HTTP header. • **Digest**: Transmits the username/password pair encrypted from the browser to server in the HTTP header. **Note:** When generating the security token to include in the SOAP message for the Web service request, you can either send the password as a hash (#), or in the clear. When interfacing to .Net Web services over a secure link, it is beneficial not to hash the password because WSS4J and .Net use different hash schemes.
HTTP Headers	
Name	Identifies the name of the header as expected or returned by the web server.
Variable/Field	Lets you select the variable (and field if its a complex variable) that contains the data that will be sent in the HTTP request, or the variable that will receive the header value that is returned.
Direction	Lets you specify if the header is sent (Out), received (In) or sent and received (InOut).
Sign SOAP Message **Note:** The Web service Operation File Editor displays the Sign SOAP Message only for the Axis2 WSOPs. You can use this section to add a soap header with a signature of the soap message. To calculate the signature a key is required.	
Enable SOAP Message Signing	If selected, a soap header with a signature of the soap message is added to the soap message.
Alias	Alias of the key to use for signing in the keystore.
Keystore Path	Fully qualified path to the keystore.
Keystore Password	Password for the keystore.

Table continues…

Name	Description
Provider	The provider of the keystore. For example, SUN.

Web service operation management (REST)

Creating a REST Web service operation file

About this task

You can create a REST Web service operation file only in a speech project and a message flow project.

Procedure

1. On the **File** menu, click **New** > **Other**.
2. In the New wizard, double-click **Avaya OD Development**.
3. Click **Web Service Operation File (REST)**, and then click **Next**.
4. In the New REST Web Service Operation wizard, on the Web Service Name page, specify the information to create a REST Web service operation file.
5. Click **Finish**.

Result

The system stores the REST Web service operation file in the `<project name>\connectivity\wsoperations` directory. The system assigns a .rest extension to the REST Web service operation file.

New REST Web Service Operation wizard: Web Service Name page field descriptions

Name	Description
Available Projects	The project for which you want to create the REST Web service operation file.
File Name	A name for the REST Web service operation file.

Configuring a REST Web service operation file

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the project directory that contains the REST Web service operation file that you want to configure.
2. Double-click **connectivity** > **wsoperations**.
3. Right-click the .rest Web service operation file that you want to configure, and then click **Open With** > **Web Service Operation Editor (REST)**.
4. In the REST Web Service Operation Editor, specify the information to configure the REST Web service operation file.

REST Web Service Operation Editor field descriptions

Name	Description
Web Service Information	
Timeout	The maximum time, in seconds, that the system must wait to authenticate and connect to the Web server that hosts the REST Web service.
HTTP Method	The HTTP method to call the REST Web service. The options are: • **GET**: To retrieve data from the Web server. • **POST**: To create data on the Web server. • **PUT**: To update data on the Web server. • **DELETE**: To delete data from the Web server. **Note:** • Ensure that the Web server that hosts the REST Web service supports the HTTP method that you select. • When you create a resource on a web service using the **PUT** or **POST** method, the system might display an error message: `201 Error: Created`. This is a bug in the underlying Axis2 framework wherein the 201 event is treated as an error. You can safely disregard the message since the 201 status code indicates the resource was successfully created.

Table continues…

Name	Description
REST URL	The REST Web service URL. The **REST URL** field shows the URL that you type in the **Server/Endpoint URL** field along with input parameters that you specify in the **Input Parameters** area. **Note:** Ensure that the format of the URL matches the format that the REST Web service uses.
Server/Endpoint URL	The URL of the Web server that hosts the REST Web service.
URL Format	The format of the REST Web service URL. The options are: • **Query**: If the REST Web service uses the query format. • **Path**: If the REST Web service uses the path format. **Note:** Ensure that the format of the URL matches the format that the REST Web service uses.
Input Parameters: The parameters that you want to send as input to the Web server that hosts the REST Web service. For example, you want to fetch the weather of New York city from a weather forecast Web site. The input parameters are weather and New York city.	
Add	Adds an input parameter.
Delete	Deletes an input parameter.
REST Parameter	The REST parameter, as configured on the Web server that hosts the REST Web service, that you want to query. For example, you want to query the weather parameter that is configured on the Web server. **Note:** Ensure that the REST parameter name that you specify matches the parameter name configured on the Web server that hosts the REST Web service.

Table continues…

Name	Description
Variable	The Orchestration Designer variable for which you want to get the value from the Web server that hosts the REST Web service. For example, you want to get the weather of New York city. Select the Orchestration Designer variable that contains the value New York.
Field	If you select a complex variable in the **Variable** field, then in the **Field** field, select the complex variable field for which you want to get the value from the Web server.
URL Parameter Type	The format in which the input parameters appear in the **REST URL** field. Use the options in the **URL Parameter Type** field to format individual parameters in the REST URL. **Note:** Ensure that the format of the REST URL matches the format that the REST Web service uses. The options are: • **WSOP Defined**: To format the parameter in the format that you select in **URL Format**. • **Query**: To format the parameter in query format in the REST URL. • **Path**: To format the parameter in path format in the REST URL.

Table continues…

Name	Description
Message Body	The data, in XML form, that you want to send to the web service for PUT or POST requests. **Note:** This option is available only if you select **PUT** or **POST** HTTP method to call the REST Web service. **GET** and **DELETE** requests do not use a message body. The option are: • **Enable**: A check box to enable or disable the usage of message body in a web service call. If selected, the system sends the content of the message body to the web service. If not selected, the system saves the content of the message body in the Web Service Operation editor, but does not send the contents to the web service. • **Auto Complete**: Option to automatically inserts a corresponding closing XML tag for a tag you type in the message body. For example, if you type `<body>` and press `Enter`, the system automatically inserts a closing `</body>` tag. For a **POST** request, you can insert Orchestration Designer variables in the message body using the **Insert Variable** context menu option. You can specify the variables as *_{variablename}_*. This inserts the variable contexts into the message body at runtime. **Note:** You can also type in the *_{variablename}_* manually.
Output Parameters: The parameters that you receive as output from the Web server that hosts the REST Web service. For example, for a Web service that provides a weather report, the expected output can be the weather report string (mapped to an appropriate Orchestration Designer variable). You can use java XPath notation to extract fields out of the `XML` or `JSON` data returned by the Web service. The REST connector supports complex JSON or xml that is returned and stored directly into a complex variable. It also supports arrays of data. Ensure that the field names in the data returned exactly match the Orchestration Designer variable field names.	

Table continues…

Name	Description
Add	Adds an output parameter. **Note:** REST Web service returns data in a single block. Hence, you can add only one output parameter.
Delete	Deletes an output parameter.
Variable	The Orchestration Designer variable in which you want to store the data returned by the REST Web service.
Field	If you select a complex variable in the **Variable** field, then in the **Field** field, select the complex variable field in which you want to store the data returned by the REST Web service.

Table continues…

Name	Description
Output Format	The format in which you want to store the data returned by the REST Web service. Orchestration Designer converts and stores the data in the format that you select in the **Output Format** field. The options are: • **Server Default**: Stores the data in the format in which the data is returned by the REST Web service. **Note:** The default value is **Server Default**. • **XML String**: Stores the data returned by the REST Web service in the XML string format. • **JSON String**: Stores the data returned by the REST Web service in the JSON string format. If you select the **Use Java Object** check box, the options are: • **Server Default**: Stores the data in the format in which the data is returned by the REST Web service. **Note:** The default value is **Server Default**. • **XML**: Stores the data returned by the REST Web service in an XML Java object. • **JSON**: Stores the data returned by the REST Web service in a JSON Java object. • **String**: Stores the data returned by the REST Web service in the Java string format.
Use Java Object	The check box to store the data returned by the REST Web service in an XML Java object or a JSON Java object. Specify a format for the object in the **Output Format** field. Store the data that the REST Web service returns in an XML Java object or a JSON Java object when the structure of the data is complex and cannot be contained within an Orchestration Designer complex variable.

Table continues…

Name	Description
Custom Content Type	If the Web server that hosts the REST Web service uses nonstandard content type and returns data in JSON format, then you must specify the nonstandard content type in the **Custom Content Type** field so that the **Web Services (REST)** pluggable data connector can read and process the data. The standard content types that the **Web Services (REST)** pluggable data connector recognizes are: application/json, text/javascript, text/json, application/json/badgerfish. You need not specify these content types.
Authorization	
Authentication	Options to determine how Orchestration Designer authenticates with the Web server that hosts the REST Web service. The options are: • **No Authentication**: No authentication is required. • **Basic/Digest**: The system sends the user name and password pair in a Base64 encoded and unencrypted format in the HTTP header to the Web server.
User Name	A user name to authenticate with the Web server that hosts the REST Web service. **Note:** The **User Name** field is available only if you select **Basic/Digest** in the **Authentication** field.
Password	A password to authenticate with the Web server that hosts the REST Web service. **Note:** The **Password** field is available only if you select **Basic/Digest** in the **Authentication** field.

Table continues…

Name	Description
Preemptive Authorization	The check box to send the authentication parameters with the HTTP or HTTPS request to the Web server without waiting for a request from the Web server for the authentication parameters. Preemptive authorization reduces the time required for the authentication process. **Note:** The **Preemptive Authorization** check box is available only if you select **Basic/Digest** in the **Authentication** field.
Security	Security options to authenticate with the Web server that hosts the REST Web service. The options are: • **No Security**: If you do not want to use a secure connection to authenticate with Web server that hosts the REST Web service. **Note:** The **No Security** option is unavailable if you specify a secure website in the **Server/ Endpoint URL** field. • **SSL**: If you want to use the Secure Sockets Layer (SSL) security protocol to authenticate with Web server that hosts the REST Web service. • **TLS**: If you want to use the Transport Layer Security (TLS) protocol to authenticate with Web server that hosts the REST Web service. **Note:** If you select the **SSL** or **TLS** option, ensure that the Web server that hosts the REST Web service supports secure connection.
Port	The SSL or TLS port number. **Note:** The **Port** field is available only if you select **SSL** or **TLS** in the **Security** field.
Use Default Port	The check box to use the default ports for SSL or TLS. **Note:** The default port is 443 for SSL and TLS.
Headers	

Table continues…

Name	Description
Add	Adds an HTTP header.
Delete	Deletes the HTTP header that you select in the **HTTP Headers** area.
Name	The name of the HTTP header.
Variable	The Orchestration Designer variable that you want to use to pass values in the HTTP header. If you select **In** in the **Direction** field, then in the **Variable** field, select the variable that must store the header value that the Web server returns. If you select **Out** in the **Direction** field, then in the **Variable** field, select the variable that contains the value that you want to send in the HTTP header when sending an HTTP or HTTPS request to the Web server that hosts the REST Web service. If you select **InOut** in the **Direction** field, then in the **Variable** field, select the variable that must store and return the header value.
Field	If you select a complex variable in the **Variable** field, then in the **Field** field, select the complex variable field that you want to use to pass values in the HTTP header.
Direction	Options to specify if the header is sent, received, or sent and received. The options are: • **In**: If Orchestration Designer receives the header from the Web server that hosts the REST Web service. • **Out**: If Orchestration Designer sends the header along with the HTTP or HTTPS request to the Web server that hosts the REST Web service. • **InOut**: If Orchestration Designer sends and receives the header along with the HTTP or HTTPS request.

Table 2: Web Service Output Parameters

Name	Description
All	Stores the returned body content into an Orchestration Designer variable as a string.

Employing a Web service operation file in a call flow or a message flow

About this task

After you create a Web service operation file in a speech application or a message application, you can use the Web service operation file in the call flow or the message flow. You can use the Web service operation files only within a **Data** node.

Procedure

1. Drag a **Data** node into the call flow editor or message flow editor main workspace.
2. Double-click the **Data** node.
3. Depending on the Web service type that you want to use, drag the **Web Service**, **Web Service (Axis2)**, or **Web Service (REST)** item from the Palette pane of the data node editor into the editor workspace.
4. In the Properties view, in the **Name** field, select the Web service operation file that you want to use.

   Alternatively, in the Avaya OD Navigator view, select the Web service operation and drag it into the call flow editor or message flow editor.

Regenerating a Web service client code

About this task

When Orchestration Designer interacts with a Web service, it uses client code generated by Axis. This code is called the Web service client. This code is first generated upon creating a `.wsop` Web service operations file by using the Web service Operations File Wizard.

Note:

You can regenerate the Web service client code only for the **Web Services (Axis 1.4)** pluggable data connector.

Procedure

1. In the Navigator view or the Avaya OD Navigator view, double-click ***<project name>*** > **connectivity** > **wsoperations**.
2. Right-click the `*.wsop` file, and then click **Properties**.

   The system displays the **Properties for *<project name>*** dialog box.
3. In the left pane, click **Generate Client**.
4. In the **Generate Client** pane, click **Generate**.

Orchestration Designer connects to the Web service and requests the WSDL using the WSDL URL and regenerates the Web service Client based on the WSDL document returned by the Web service.

Web service headers

When the Web service request is sent to the server, two types of headers can be included with the request; headers included in the HTTP header section or SOAP headers. Both are configured in this section. Each header must be given a unique name and the value for the header is taken from an Orchestration Designer variable.

The direction of the header can be set to IN, OUT or both IN and OUT. If the direction is set to IN, then the value of the header is taken from the Web service response and stored in the selected Orchestration Designer variable. If the direction is set to OUT, then the value of the Orchestration Designer variable is added to a header sent with the Web service request.

Configuring web services to use SHA-2 certificate hashing

Procedure

1. On the **File** menu, click **New** > **Other**

   The system displays the **New** wizard.
2. Double-click **Avaya OD Development**.
3. Click **Web Service Operation File**, and then click **Next**.

   The system displays the New Web Service Operation wizard.
4. On the Specify General Parameters page, select one of the **Authentication** types.
5. On the Specify General Parameters page, specify the other Web service operations parameters, and then click **Next**.
6. Select the **Send Password in the Clear** check box to send the password as a clear text password without using the SHA1 encryption.
7. On the Map Input Parameters page, map the project variables to the input parameters, and then click **Next**.
8. On the Map Output Parameters page, map the project variables to the output parameters.
9. Click **Finish**.
10. Open the Java file corresponding to the WSOP file.

    The java file for a WSOP file is located at WEB-INF/src/connectivity.ws.operations and has the same name as the WSOP file.

The system opens the file in the Web Service Operation Editor.

11. In the java file, copy the following code after the line "`//}}END:CLASS:CONSTRUCTOR":`

```
try {
java.security.MessageDigest md =
java.security.MessageDigest.getInstance("SHA-256");
byte[] digest = md.digest(this.password.getBytes());
password = org.apache.ws.security.util.Base64.encode(digest);
}catch (NoSuchAlgorithmException e) {
e.printStackTrace();
}
```

12. Save and close the file.

Chapter 22: SMS

SMS files

You can create and use an SMS file in an SMS channel message application to send an automated SMS message response to an inbound SMS message. You can also use an SMS file to send a proactive automated outbound SMS message. For example, send an outbound SMS message informing a customer about the account balance.

You can specify the SMS message that you want to send in an SMS file. Create a prompt file and add the **SMS** item to the prompt file. Assign the SMS file to the **SMS** item. You can then add the **Prompt** item to the **Announce** node from where you want to send the SMS message and assign the prompt file to the **Prompt** item.

When the system executes the **Announce** node, the system sends the SMS message to the text browser that is located on Avaya Experience Portal Media Processing Platform (MPP) for further processing and delivery.

Note:

If you want to send a simple SMS message from a single prompt of an SMS channel message application, then you can use the **Text** item or **Text Variable** item instead of creating an SMS file.

You can also create and use SMS files in a speech, message, or data application that is created for a channel other than the SMS channel to send outbound SMS messages, also called notifications.

For example, a speech application receives a call from a customer requesting the phone number of a car rental service. You can use an SMS file in the speech application to send an automated SMS message notification that provides the phone number of the car rental service to the customer.

You can receive an inbound SMS message only in an SMS channel message application.

Note:

You must enable the **Notification Connector (Email, SMS)** pluggable data connector (PDC) to send an SMS message notification from a speech, message, or data application that is created for a channel other than the SMS channel.

After you enable the **Notification Connector (Email, SMS)** PDC, the **Send SMS** and **Send Email** items are available in the Palette pane of the **Data** node.

You can specify the SMS message that you want to send in an SMS file. You can then add the **Send SMS** item to the **Data** node from where you want to send the SMS message notification and assign the SMS file to the **Send SMS** item.

When the system executes the **Data** node, the system sends the SMS message notification to the platform Web service for further processing and delivery.

You can also use the **Send SMS** item in the SMS channel message application when you want to use the:

- Platform Web service to further process and deliver the SMS message.
- Features of the **Send SMS** item. For example, set the priority to send the outbound SMS message.

SMS file management

Creating an SMS file

About this task

Before you build a speech, message, or data application, create the SMS files that you want to use in the speech, message, or data application.

Note:

You can create and use an SMS file in a speech application or a message application that is created for a channel other than the SMS channel only after you enable the **Notification Connector (Email, SMS)** pluggable data connector.

Procedure

1. On the **File** menu, click **New** > **Other**.
2. In the New wizard, double-click **Avaya OD Development**.
3. Click **SMS File**, and then click **Next**.
4. In the New SMS wizard, on the Create an SMS page, specify the project name, the language, and a name for the SMS file.
5. Click **Finish**.

Result

The system displays the SMS Message editor.

Next steps

You can type the SMS message body in the SMS Message editor.

New SMS wizard: Create an SMS page field descriptions

Note:

You can create and use an SMS file in a speech, message, or data application that is created for a channel other than the SMS channel only after you enable the **Notification Connector (Email, SMS) pluggable data connector**.

Name	Description
Available Projects	The speech project, the message flow project, or the data flow project for which you want to create an SMS file. You must select a project.
Available Languages	The language for the SMS file. You must select a language for the SMS file.
SMS file name	A unique and relevant name for the SMS file.

Editing an SMS file

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project directory or the message flow project directory that contains the SMS file that you want to edit.
2. Double-click ***<language name>*** > **sms**.
3. Right-click the .sms file that you want to edit, and then click **Open With** > **SMS Editor**.
4. In the SMS Message editor, modify the SMS message body.

   You can also insert variables in the SMS message body text.

Inserting a variable in an SMS message body

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project directory or the message flow project directory that contains the SMS file in which you want to insert a variable.
2. Double-click ***<language name>*** > **sms**.
3. Right-click the .sms file in which you want to insert a variable, and then click **Open With** > **SMS Editor**.
4. In the SMS Message editor, move the pointer to the location where you want to insert a variable.

5. On the **SMS** menu, click **Insert Variable**.
6. In the Insert Variable dialog box, select the variable that you want to insert in the SMS message body.
7. Click **Insert**.

Tip:

You can also drag a variable from the **project.variables** directory to the SMS Message editor to insert the variable in the SMS message body. The **project.variables** directory is available under the ***<project name>*** > **flow** directory in the Navigator view or in the Avaya OD Navigator view. Be sure that you open the Speech perspective for a speech project and the Message perspective for a message flow project.

Insert Variable dialog box field descriptions

Name	Description
Variable	The variable that you want to insert in the message body. If you select a complex variable in the **Variable** field, then you must select a complex variable field in the **Field** field.
Field	The complex variable field that you want to insert in the message body. The **Field** field is available only if you select a complex variable in the **Variable** field.
Insert	Option to insert the variable in the message body.

Chapter 23: Email

Email files

You can create and use an email file in an email channel message application to send an automated email message response to an inbound email message. You can also use an email file to send a proactive automated outbound email message. For example, send an outbound email message informing a customer about the account statement.

You can specify the email message that you want to send in an email file. Create a prompt file and add the **Email** item to the prompt file. Assign the email file to the **Email** item. You can then add the **Prompt** item to the **Announce** node from where you want to send the email message and assign the prompt file to the **Prompt** item.

When the system executes the **Announce** node, the system sends the email message to the text browser that is located on Avaya Experience Portal Media Processing Platform (MPP) for further processing and delivery.

Note:

If you want to send an email message only from a single prompt of an email channel message application, and if the email message is small and does not require attachment files and HTML formatting, then you can use the **Text** item or **Text Variable** item instead of creating an email file.

You can also create and use email files in a speech application, or a message application, or a data that is created for a channel other than the email channel to send outbound email messages, also called notifications.

For example, a speech application receives a call from a customer requesting the price list of your products. You can use an email file in the speech application to send an automated email message notification that provides the price list of your products to the customer.

You can receive an inbound email message only in an email channel message application.

Note:

You must enable the **Notification Connector (Email, SMS)** pluggable data connector (PDC) to send an email message notification from a speech, message, or data application that is created for a channel other than the email channel.

After you enable the **Notification Connector (Email, SMS)** PDC, the **Send SMS** and **Send Email** items are available in the Palette pane of the **Data** node.

You can specify the email message that you want to send in an email file. You can then add the **Send Email** item to the **Data** node from where you want to send the email message notification and assign the email file to the **Send Email** item.

When the system executes the **Data** node, the system sends the email message notification to the platform Web service for further processing and delivery.

You can also use the **Send Email** item in the email channel message application when you want to use the platform Web service to further process and deliver the email message.

Email file management

Creating an email file

About this task

Before you build a speech, message, or data application, create the email files that you want to use in the speech, message, or data application.

Note:

You can create and use an email file in the speech, message, or data application that is created for a channel other than the email channel only after you enable the **Notification Connector (Email, SMS) pluggable data connector**.

Procedure

1. On the **File** menu, click **New** > **Other**.
2. In the New wizard, double-click **Avaya OD Development**.
3. Click **Email File**, and then click **Next**.
4. In the New Email wizard, on the Create an Email page, specify the information to create an email file.
5. Click **Finish**.

Result

The system displays the Email Message editor.

Next steps

You can type the email message body in the Email Message editor.

New Email wizard: Create an Email page field descriptions

Note:

You can create and use an email file in the speech, message, or data application that is created for a channel other than the email channel only after you enable the **Notification Connector (Email, SMS) pluggable data connector**.

Name	Description
Available Projects	The speech project, message flow project, or data flow project for which you want to create an email file. You must select a project.
Available Languages	The language for the email file. You must select a language for the email file.
Email file name	A unique and relevant name for the email file.
Email file	The following are the options: • **New** to create a new email file. • **Existing** to use an existing email file.
File name	The location of the existing email file that you want to use. This field is available only if you click **Existing**.

Editing an email file

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project directory, the message flow project directory, or the data flow project directory that contains the email file that you want to edit.
2. Double-click ***<language name>*** > **email**.
3. Right-click the .eml file that you want to edit, and then click **Open With** > **Email Editor**.
4. In the Email Message editor, in the **Message Body** area, type the email message body.

   Note:

   - You can also insert variables in the message body. The variables are inserted in the *_{variable}_* format and value of variable is added into the body of the email at runtime.
   - If you want to display the literal contents in curly braces in the email body, then you must add the content in the format same as for variable and a backslash prior to the format. For example, _*{some text}*_.

For example, to send the email with a message as “You last reset your password on _*{mydate}*_”, you must define a variable *mydate*, assign value to it, and type “You last reset your password on _*{mydate}*_” in the message body. If you do not add a backslash before _{ }_, the system interprets the value within the curly braces as a variable and might display an error message if a variable of that name is not defined in the application.

5. In the **Message Details** area, click **Attach File** to attach a file to the email message.
6. In the **Add Attachment** dialog box, perform one of the following actions:
   - To attach a file that resides on your computer, perform the following actions:
     a. Click **Local**, and then click **OK**.
     b. In the **Open** dialog box, browse to the file that you want to attach, and then click **Open**.
   - To attach a file that resides on another computer or server, perform the following actions:
     a. Click **External**, and then click **OK**.
     b. In the **External Attachment** dialog box, in the **External Attachment Location** field, type the URL path where the file resides, and then click **OK**.

   The system displays the attached file in the **Message Details** area.

Note:

You can click that corresponds to the attached file to remove the file from the email message.

Next steps

You can also insert variables in the email message body text.

Inserting a variable in an email message body

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project directory or the message flow project directory that contains the email file in which you want to insert a variable.
2. Double-click ***<language name>*** > **email**.
3. Right-click the .eml file in which you want to insert a variable, and then click **Open With** > **Email Editor**.
4. In the Email Message editor, move the pointer to the location where you want to insert a variable.
5. On the **Email** menu, click **Insert Variable**.

6. In the Insert Variable dialog box, select the variable that you want to insert in the email message body.
7. Click **Insert**.

   The system inserts a variable in the *_{variable name}_* format. Value of the variable is added into the body of the email at runtime.

Tip:

You can also drag a variable from the **project.variables** directory to the Email Message editor to insert the variable in the email message body. The **project.variables** directory is available under the ***<project name>*** > **flow** directory in the Navigator view or in the Avaya OD Navigator view. Be sure that you open the Speech perspective for a speech project and the Message perspective for a message flow project.

Note:

If you want to display the literal contents in curly braces in the email body, then you must add the content in the format same as for variable and a backslash prior to the format. For example, _*{some text}*_.

For example, to send the email with a message as “You last reset your password on *_{mydate}_*”, you must define a variable *mydate*, assign value to it, and type “You last reset your password on _*{mydate}*_” in the message body. If you do not add a backslash before _{ }_, the system interprets the value within the curly braces as a variable and might display an error message if a variable of that name is not defined in the application.

To send the email with a message as “You last reset your password on Wed 7/6/2011 11:46 AM”, type “You last reset your password on” and insert the *_{mydate}_* variable in the message body.

Insert Variable dialog box field descriptions

Name	Description
Variable	The variable that you want to insert in the message body. If you select a complex variable in the **Variable** field, then you must select a complex variable field in the **Field** field.
Field	The complex variable field that you want to insert in the message body. The **Field** field is available only if you select a complex variable in the **Variable** field.
Insert	Option to insert the variable in the message body.

Previewing an email file

About this task

Note:

Be sure that you save the email file before previewing the email file.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project directory or the message flow project directory that contains the email file that you want to preview.
2. Double-click ***<language name>*** > **email**.
3. Right-click the .eml file that you want to preview, and then click **Open With** > **Email Editor**.
4. Click in the Email Message editor.
5. On the **Email** menu, click **Preview Email**.

   The system displays the email message body in an HTML format in the Email Preview view.

Chapter 24: Conversations Management

Conversations

Avaya Experience Portal is enhanced to provide a repository for storing conversations. A conversation is defined as a dialog between the application and a user. As part of the conversation data, an application can store multiple name and value pairs. Each conversation is identified by a unique ID. The platform provides an ID as part of launching the Orchestration Designer application.

Note:

You can use the conversations feature for Voice, Email, SMS, data and HTML based projects.

Conversation management is automatic. You cannot explicitly save or restore the context.

Whenever the system starts a conversation or stores the conversation for the first time, the conversation has a lease. The conversations are temporary in nature and the system removes the conversations from the repository once the lease expires. Each time the system updates the conversation, the system renews lease of the conversation for the lease time specified as part of the update. You can define the default lease time when creating the Orchestration Designer application as one of the configurable application variables (CAV) .

Conversation ID

The conversation id must be unique in each of the conversation store. The platform creates the conversation ID and passes the ID to the application.

avayaConversationLease

The configurable application variable *avayaConversationLease*, is used by the Orchestration Designer PDC to control the conversation lease timeout. The default value of *avayaConversationLease* is 86400 seconds. You can set the default value for the conversation lease according to the need of the application.

Conversation Pluggable Data Connector (PDC)

The PDC exposes the Experience Portal conversation services to the application developer.

When an application starts a conversation using the *Start Conversation* palette item, the conversation PDC records the current session id in the conversation and uses that value to set the exitParentId, on application exit, until the conversation ends. To end a conversation the application uses the *End Conversation* palette item. Once the conversations ends, the system destroys the conversation data and sets the exitPreferredPath to 1. A commit conversation writes the conversation data to the conversation store immediately, and ignores any changes made to the conversation after the commit.

While a conversation is in progress, each time the application runs, the application variables selected as part of the conversation are set to the last saved values after normal variable initialization and CAV initialization. When the application exits, the variables defined as part of the conversation are saved.

The conversation PDC supports simple variables, complex variables, and complex variable fields. A variable can also be a collection of values.

When you are transitioning from HTML to speech, you must call a *commit conversation* before the launch outbound call. As the commit conversation writes the data to the conversation store immediately, the data is available when the call starts.

Cross Channel Conversations

In addition to maintaining context across interactions within in a single channel (SMS or Email), you can use conversations to maintain context across multiple channels. In this context, cross channel means the user switches from one channel to another and does not use both the channels concurrently. Orchestration Designer does not support concurrent channel access to conversations.

Conversation alias

In many cases, where the user switches from one channel to another, the Orchestration Designer runtime passes the conversation id to the next channel for the application. For example, an HTML5 application launching an outbound call. In other cases, the application adds an alias to the conversation. For example, an HTML5 application sending out an SMS notification. The alias is the “from” address for the new channel. If an application developer adds an alias, the Orchestration Designer runtime automatically checks for a conversation and loads it if it exists.

The following table shows the possible channel transitions and the actions required to make the conversation available to the new channel:

Table 3: Channel Transitions

	->HTML5	->SMS	->Email	->Speech
HTML5–>	N/A	B	B	*A
SMS->	A	B	B	*A
Email->	A	B	B	*A
Speech->	A	B	B	B

A: The Orchestration Designer runtime automatically adds the conversation id as a query string parameter (_ODConversationId) and then looks for this parameter when the application starts.

B: The application developer must add an alias to the conversation. The alias is the “from” address of the client. For SMS channel, the “from” is a mobile number. For Email channel, the “from” is an email address.

When an email or SMS application starts, if a conversation is not found for the conversation id generated by the Avaya Experience Portal, the Orchestration Designer runtime uses the “from” as an alias to look up the conversation.

*A: The Orchestration Designer runtime assumes that the **LaunchOutboundCall (LaunchVXML)** PDC is used to launch the speech application. For example, if the application sends an SMS to the client that says ,“To continue in speech, please call 555-1212”, then the application needs to add an alias that is the phone number the client will call from. If the speech application is on a landline, and sends out an SMS to a mobile phone, then to pass the conversation to the SMS application, the speech application needs to add the mobile phone number as an alias. If **LaunchCCXML** is used, the conversation id is sent as a parameter to the **CCXML** application and the **CCXML** application must pass this conversation id to the speech dialog using the query string as (_ODConversationId).

If the default conversation id fails but the alias matches a conversation, then the default conversation id is also added as an alias. This reduces the lookup time of the context for multi turn applications.

Note:

You do not have to use conversations when switching between channels. Conversations are only needed if you want to pass context information between the channels.

Chapter 25: Multichannel notifications

Multichannel notifications

Use the **Send SMS** item to send an SMS message, also called a notification, from a speech, message, or data application that is created for a channel other than the SMS channel.

Use the **Send Email** item to send an email message, also called a notification, from a speech, message, or data application that is created for a channel other than the email channel.

For example, a speech application receives a call from a customer requesting the account statement of the checking account. You can use the **Send Email** item in the speech application to send an email message notification that provides the account statement to the customer.

Note:

The **Send SMS** and **Send Email** items are available in the Palette pane of the **Data** node only after you enable the **Notification Connector (Email, SMS)** pluggable data connector (PDC).

When the system executes the **Data** node that contains the **Send SMS** or **Send Email** item, the system sends the SMS message or the email message to the platform Web service for further processing and delivery.

You can receive an inbound SMS message only in an SMS channel message application and an inbound email message only in an email channel message application.

You can also use the **Send SMS** item in the SMS channel message application when you want to use the:

- Platform Web service to further process and deliver the SMS message.
- Features of the **Send SMS** item. For example, set the priority to send the outbound SMS message.

You can also use the **Send Email** item in the email channel message application when you want to use the platform Web service to further process and deliver the email message.

Example of a multichannel notification

Before you begin

Ensure that you enable the **Notification Connector (Email, SMS)** pluggable data connector.

About this task

You receive an call from a customer requesting the phone number of a car rental service. You can use the **Send SMS** item in the speech application to send an automated SMS message notification that provides the phone number of the car rental service to the customer.

Procedure

1. Open the speech application from which you want to send the SMS message notification.
2. Create an SMS file and in the SMS file, specify the SMS message that you want to send.
3. Open the Data node from where you want to send the SMS message notification.
4. From the Palette pane, drag the **Send SMS** item to the data node editor.
5. In the Properties view of the **Send SMS** item, in the **SMS Constant** field, select the SMS file that you created in Step 2, and then set the other properties.

Chapter 26: Launch outbound call from a speech, or a message, or a data, or a web application

Outbound call launch from an Orchestration Designer application

Use the **LaunchVXMLCall** item to launch an automated outbound VoiceXML call from an Orchestration Designer application.

For example, a message application is unable to interpret an inbound message because the required grammars are not configured in the message application. For such an event, you can use the **LaunchVXMLCall** item in the message application to make an automated outbound call to the customer.

Similarly, use the **LaunchCCXMLCall** item to launch an automated outbound CCXML call from a speech or a message or a web application.

Note:

The **LaunchVXMLCall** and **LaunchCCXMLCall** items are available in the Palette pane of the **Data** node only after you enable the **AAEP Outbound Call** pluggable data connector (PDC).

Avaya recommends that you use either the **LaunchVXMLCall** or the **LaunchCCXMLCall** item within a **Data** node.

Orchestration Designer initiates the outbound call on Avaya Experience Portal Media Processing Platform (MPP) by launching the VoiceXML or CCXML application that is configured on Experience Portal Manager (EPM).

Example to launch an automated outbound call from an Orchestration Designer application

Before you begin

The **LaunchVXMLCall** and **LaunchCCXMLCall** items are available in the Palette pane of the **Data** node only after you enable the **AAEP Outbound Call** pluggable data connector (PDC).

Procedure

1. Open the Orchestration Designer application project from which you want to launch an outbound call.
2. Double-click **flow** > **main.flow**.
3. From the Palette pane, drag the **Data** node to the project flow from where you want to launch the outbound call.
4. Open the **Data** node.
5. From the Palette pane of the data node editor, drag one of the following items:
    a. To launch a VoiceXML call, drag the **LaunchVXMLCall** item to the data node editor and set the properties of the **LaunchVXMLCall** item.
    b. To launch a CCXML call, drag the **LaunchCCXMLCall** item to the data node editor and set the properties of the **LaunchCCXMLCall** item.
6. Drag and configure other nodes and items as needed in the project flow.

Result

When the system executes the **LaunchVXMLCall** or the **LaunchCCXMLCall** item, the system initiates the outbound VoiceXML or CCXML call.

Chapter 27: Inbound SMS or email message handoff to Avaya Aura® Contact Center for agent assistance

Forward an inbound SMS or email message to Avaya Aura® Contact Center

You can enable the **Send AACC Connector** pluggable data connector (PDC) in an SMS channel message flow project or an email channel message flow project to forward an inbound SMS or email message to Avaya Aura® Contact Center for agent assistance. After you enable the **Send AACC Connector** PDC, you can design the Orchestration Designer message application in such a way that the message application forwards the inbound message to Avaya Aura® Contact Center if the message application is unable to interpret the inbound message due to various reasons. For example, the inbound message did not match the grammars that are configured in the message flow project.

When you enable the **Send AACC Connector** PDC, you must also specify:

- The email address that is configured on Avaya Aura® Contact Center for assisted care.
- The email address from where the Orchestration Designer message application must forward the inbound message to Avaya Aura® Contact Center.

After you enable the **Send AACC Connector** PDC, the **Send to AACC** item is available in the Palette pane of a data node editor. You can create an AACC message file and specify the additional information, parameters, and message response that you want to send to the Avaya Aura® Contact Center agent. You can then assign the AACC message file to the **Send to AACC** item in the **Data** node from where you want to forward the inbound message to Avaya Aura® Contact Center.

For example, you have a **Collect** node in the message flow that collects the inbound message. You can set a **Data** node as the next node to execute after the **Collect** node. You can drag the **Send to AACC** item in the data node editor and assign an AACC message file to the **Send to AACC** item. You can configure the **Collect** node in such a way that if the message application is unable to interpret the inbound message and generates a `nomatch` event, the message application executes the **Data** node. The message application then forwards the inbound message

along with the information contained in the AACC message file to Avaya Aura® Contact Center for agent assistance.

The message application forwards the inbound message in the form of an email message to Avaya Aura® Contact Center. The email message contains the additional information and the parameters that you specify in the AACC message file. The email message also contains a **Reply to customer** link. When the Avaya Aura® Contact Center agent clicks the **Reply to customer** link, a Web page is displayed on the computer screen of the Avaya Aura® Contact Center agent. The Web page contains the message response that you specify in the AACC message file. The Avaya Aura® Contact Center agent can either use the message response to respond to the customer or overwrite the message response to send a custom response to the customer.

AACC message file management

AACC message file

You can create an AACC message file only in a message flow project.

Note:

You can create an AACC message file only after you enable the **Send AACC Connector** pluggable data connector in a message flow project.

Assign an AACC message file to the **Send to AACC** item in a **Data** node if you want to send additional information along with the inbound SMS or email message to Avaya Aura® Contact Center for agent assistance.

In the AACC message file, you can specify:

- Additional information that you want to send to Avaya Aura® Contact Center.
- Parameters, such as the arrival time and message headers of the inbound message, that you want to send to Avaya Aura® Contact Center. You must specify parameters that maintain the context of the message session.
- Whether you want to send the attachments contained in the inbound email message to Avaya Aura® Contact Center.
- A message response to the inbound message. The message response appears as a template on the computer screen of the Avaya Aura® Contact Center agent. The Avaya Aura® Contact Center agent can either use the message response to respond to the customer or overwrite the message response to send a custom response to the customer.

Creating an AACC message file

Before you begin

You can create an AACC message file only after you enable the **Send AACC Connector** pluggable data connector in a message flow project.

About this task

Note:

You can create an AACC message file only in a message flow project.

Procedure

1. On the **File** menu, click **New** > **Other**.
2. In the New wizard, double-click **Avaya OD Development**.
3. Click **AACC Message**, and then click **Next**.
4. In the New AACC message wizard, on the Create a new AACC message page, specify the information to create an AACC message file.
5. Click **Finish**.

New AACC Message wizard: Create a new AACC message page field descriptions

Note:

You can create an AACC message file only in a message flow project.

You can create an AACC message file only after you enable the **Send AACC Connector** pluggable data connector in a message flow project.

Name	Description
Available Projects	The message flow project for which you want to create an AACC message file.
Name	A name for the AACC message file.

Configuring an AACC message file

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the message flow project directory that contains the AACC message file that you want to configure.
2. Double-click **connectivity** > **aacc**.

3. Right-click the .aacc file that you want to configure, and then click **Open With** > **AACC Editor**.
4. In the AACC Message editor, specify the information to configure the AACC message file.

AACC Message editor field descriptions

Name	Description
Message to AACC Agent Details: The message application sends the information that you specify in the **Message to AACC Agent Details** area along with the inbound SMS or email message to Avaya Aura® Contact Center in the form of an email message.	
Subject	The variable that contains the subject line that you want to send to Avaya Aura® Contact Center along with the inbound message for agent assistance. **Note:** The message application sends the subject line along with the inbound message to Avaya Aura® Contact Center in the form of an email message.
Display Sub-Heading	The variable that contains the subheading that you want to send to Avaya Aura® Contact Center along with the inbound message for agent assistance. **Note:** The message application sends the subheading along with the inbound message to Avaya Aura® Contact Center in the form of an email message.
Parameters	The variables that you want to send to Avaya Aura® Contact Center along with the inbound message for agent assistance. You must specify variables that maintain the context of the message session. For example, arrival time and message headers of the inbound message. **Note:** The message application sends the variable values along with the inbound message to Avaya Aura® Contact Center in the form of an email message. For more information, see Specifying a parameter to pass to Avaya Aura Contact Center on page 369.

Table continues…

Name	Description
Include Attachments	The check box to send the attachments contained in the inbound email message to Avaya Aura® Contact Center for agent assistance.
Reply to Customer: The outbound message response details that the Avaya Aura® Contact Center agent can use when responding to the customer. When the message application forwards the inbound message in the form of an email message to Avaya Aura® Contact Center, the email message also contains a **Reply to customer** link. When the Avaya Aura® Contact Center agent clicks the **Reply to customer** link, a Web page is displayed on the computer screen of the Avaya Aura® Contact Center agent. The details that you specify in the **Reply to Customer** area are populated on the Web page from where the Avaya Aura® Contact Center agent responds to the customer. The Avaya Aura® Contact Center agent can either use these details to respond to the customer or overwrite these details to send a custom response to the customer.	
From	The variable that contains the phone number or the email address of the customer who sends the SMS message or email message. Avaya Aura® Contact Center uses this phone number or email address to send an outbound message response to the customer. The phone number or the email address appears in the To field on the Web page that is displayed on the computer screen of the Avaya Aura® Contact Center agent.
To	The variable that contains the phone number or the email address that is configured on Avaya Aura® Contact Center from where the SMS message or email message response is sent to the customer. The phone number or the email address appears in the From field on the Web page that is displayed on the computer screen of the Avaya Aura® Contact Center agent.
Subject	The variable that contains the subject line for the email message response to be sent to the customer from Avaya Aura® Contact Center.
Message	The variable that contains the message body for the SMS or email message response to be sent to the customer from Avaya Aura® Contact Center. **Note:** Ensure that the message body is in the text format.

Specifying a parameter to pass to Avaya Aura® Contact Center

About this task

You must specify variables in an AACC message file to maintain the context of the message session. The message application sends the values of the variables that you specify along with the inbound SMS or email message in the form of an email message to Avaya Aura® Contact Center for agent assistance.

For example, you can specify the variables that contain the arrival time and message headers of the inbound message.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the project folder that contains the AACC message file in which you want to specify the variable that you want to pass to Avaya Aura® Contact Center.
2. Double-click **connectivity** > **aacc**.
3. Right-click the .aacc file in which you want to specify the variable that you want to pass to Avaya Aura® Contact Center, and then click **Open With** > **AACC Editor**.
4. In the AACC Message editor, in the **Parameters** area, click **Add**.
5. In the Variable dialog box, in the **Description** field, type a description for the variable that you want to pass to Avaya Aura® Contact Center.
6. In the **Variable** field, click  .
7. In the Select Variable dialog box, select the simple variable or the complex variable field, the value of which you want to pass to Avaya Aura® Contact Center.
8. Click **OK**.
9. In the Variable dialog box, click **OK**.

Changing the parameter passed to Avaya Aura® Contact Center

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the project folder that contains the AACC message file in which you want to change the variable that is passed to Avaya Aura® Contact Center.
2. Double-click **connectivity** > **aacc**.
3. Right-click the .aacc file in which you want to change the variable that is passed to Avaya Aura® Contact Center, and then click **Open With** > **AACC Editor**.
4. In the AACC Message editor, in the **Parameters** area, select the variable that you want to change, and then click **Edit**.

5. In the Variable dialog box, in the **Description** field, type a description for the variable that you want to pass to Avaya Aura® Contact Center instead of the current variable.
6. In the **Variable** field, click  .
7. In the Select Variable dialog box, select the simple variable or the complex variable field, the value of which you want to pass to Avaya Aura® Contact Center instead of the current variable.
8. Click **OK**.
9. In the Variable dialog box, click **OK**.

Removing the parameter passed to Avaya Aura® Contact Center

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the project folder that contains the AACC message file from which you want to remove the variable that is passed to Avaya Aura® Contact Center.
2. Double-click **connectivity** > **aacc**.
3. Right-click the .aacc file from which you want to remove the variable that is passed to Avaya Aura® Contact Center, and then click **Open With** > **AACC Editor**.
4. In the AACC Message editor, in the **Parameters** area, select the variable that you do not want to send to Avaya Aura® Contact Center, and then click **Delete**.

Chapter 28: AACC treatments

AACC treatments

Avaya Aura® Contact Center might transfer calls to an Orchestration Designer speech application in various scenarios. For example, Avaya Aura® Contact Center receives a call which requires:

- The caller to validate secure data first such as the pin number of the caller.
- The caller data, such as the name and the date of birth of the caller, to be collected first from the caller so that the Avaya Aura® Contact Center agent can process the call further.
- A prompt to be played to the caller that informs the caller about the business hours.

In a SIP-enabled Avaya Aura® Contact Center, Avaya Aura® Contact Center can transfer such calls to an Orchestration Designer speech application that is managed on Avaya Experience Portal.

Depending on the treatment type that Avaya Aura® Contact Center requests, the Orchestration Designer speech application processes the call. For example, the speech application plays a prompt to the caller, collects the required information from the caller and passes the information along with the call to Avaya Aura® Contact Center for agent assistance, thereby, reducing the time required for assisted care.

Avaya Aura® Contact Center uses the **`GIVE IVR`** command to transfer an inbound call to an Orchestration Designer speech application that is managed on Avaya Experience Portal. The **`GIVE IVR`** command determines the AACC treatment type to be applied to the call, the parameters to be passed to the Orchestration Designer speech application, and the variables in the Avaya Aura® Contact Center script that must store the data that the Orchestration Designer speech application returns.

Avaya Aura® Contact Center uses a script to pass the parameters to the Orchestration Designer speech application. For example, parameters that contain information about the prompt to play and the number of digits that the speech application should collect from the caller. You can design the speech application to use either the parameters passed by the Avaya Aura® Contact Center script or the values that you set in the speech application.

To receive the call that Avaya Aura® Contact Center transfers, you must enable the **AACC Treatments** pluggable data connector (PDC) in the Orchestration Designer speech project.

Note:

The **AACC Treatments** PDC is available only in speech projects.

After you enable the **AACC Treatments** PDC, you can either use the default AACC treatments or create custom AACC treatments.

Depending on the treatment type that Avaya Aura® Contact Center requests in the Avaya Aura® Contact Center script, Orchestration Designer determines the treatment type to apply to the call. AACC treatments determine:

- Whether the speech application must return data to Avaya Aura® Contact Center.
- The data that the speech application must return to Avaya Aura® Contact Center. For example, caller input and pin validation result.

You can design a speech application to process requests for multiple treatment types or you can create separate speech applications to process requests for individual treatment type. For example, play a prompt to the caller and collect digits from the caller and pass the digits to Avaya Aura® Contact Center.

After you enable the **AACC Treatments** PDC in an Orchestration Designer speech project, Orchestration Designer adds a ccxml.jsp file to the project directory.

Note:

You must specify the HTTP path of the ccxml.jsp page as the destination URI on the VPMS server when you host the Orchestration Designer speech application as a managed application on the VPMS server.

Avaya Aura® Contact Center transfers the inbound call to Avaya Experience Portal along with the parameters that are configured in the Avaya Aura® Contact Center script. Avaya Experience Portal passes the parameters received from the Avaya Aura® Contact Center script to the ccxml.jsp page along with the details of the Avaya Aura® Contact Center SIP endpoint from which Avaya Experience Portal received the call. The ccxml.jsp page invokes the speech application that is requested in the Avaya Aura® Contact Center script. When the speech application terminates, Orchestration Designer returns the control of the call to the ccxml.jsp page. The ccxml.jsp page packages the return data into a SIP message and returns the call along with the data to the SIP endpoint from where the connection was received.

After you enable the **AACC Treatments** PDC in an Orchestration Designer speech project, the system adds two complex variables to the speech project: **aacc_data** and **aacc_context**.

The **aacc_data** variable contains various fields that store the parameters that the system receives from the Avaya Aura® Contact Center script. The **aacc_data** variable also contains the **returnvalue** variable field which returns the data that the treatment type requests to Avaya Aura® Contact Center. You must design the speech application to assign the data requested by an AACC treatment type to the **returnvalue** variable field.

For **Context Creation** treatment type, the speech application uses the **aacc_context** variable. Unlike other AACC treatments that return only the data requested by the treatment type to Avaya Aura® Contact Center, you can add any number of variable fields to the **aacc_context** variable depending on the data that you want to send to Avaya Aura® Contact Center. The **aacc_context** variable sends the data in name and value pairs in XML format to Avaya Aura® Contact Center. The **aacc_context** variable also contains a **cad** field. The speech application sets the call

attached data (CAD) in the string format in the **cad** field and returns the CAD data to Avaya Aura® Contact Center.

You must design the speech application to assign the data that you want to send to Avaya Aura® Contact Center to the variable fields that you create under the **aacc_context** variable. You can also set the CAD data that you want to send to Avaya Aura® Contact Center in the **cad** field. If you do not want to send data to Avaya Aura® Contact Center, then do not assign values to these fields.

Default AACC treatment types

The following are the default AACC treatment types:

Default AACC treatment	Description
Play Only	Plays a prompt to the caller and does not expect any return data from the speech application.
Digit Collection	Collects a series of digits from the caller and sends the digits to Avaya Aura® Contact Center.
PIN Validation	Validates the pin number that the caller provides and sends the pin validation result to Avaya Aura® Contact Center.
Context Creation	Passes the data that you want to send to Avaya Aura® Contact Center in name and value pairs to Avaya Aura® Contact Center.

AACC treatment management

Creating an AACC treatment

Before you begin

You can create an AACC treatment in a speech application only after you enable the **AACC Treatments** pluggable data connector.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, click the project for which you want to create an AACC treatment.
2. On the **Project** menu, click **Properties**.
3. In the Properties for <project name> dialog box, in the left navigation pane, click **Orchestration Designer**.

4. Click the **Pluggable Connectors** tab.
5. In the **Category** field, click **Contact Center**.
6. Click **AACC Treatments**.
7. In the **AACC Treatments Configuration** area, click **Add**.
8. In the AACC Treatments dialog box, specify the information to create an AACC treatment.
9. Click **OK**.
10. In the Properties for <project name> dialog box, click **OK**.

AACC Treatments dialog box field descriptions

Name	Description
Treatment Name	Unique name for the AACC treatment.
Treatment Type	The AACC treatment type as configured in Avaya Aura® Contact Center. **Note:** Treatment types are case sensitive. Ensure that the name of the treatment type is the same as that configured in Avaya Aura® Contact Center.
SIP Message Type	The SIP message type as configured on Avaya Aura® Contact Center for the treatment type. The system passes the SIP message type in the SIP header when sending the return data to Avaya Aura® Contact Center. For example, **application/ vnd.nortelnetworks.digits**

Table continues…

Name	Description
SIP Message Preamble	The SIP message preamble as configured on Avaya Aura® Contact Center for the treatment type. SIP message preamble determines the format of the data that an Orchestration Designer speech application sends to Avaya Aura® Contact Center. For example, p=Digit-Collection\r\ny=Digits\r \nd=Digits%3D1234 Where: • p: Represents the name of the treatment type. • y: Represents the type of the data that is returned to Avaya Aura® Contact Center. For example, digit, string. • d: Represents the return data. The system appends the data requested by the treatment type after %3D. • \r: Represents carriage return. • \n: Represents newline feed. • 1234: Represents the data returned by the speech application. The system appends the data requested by the treatment type to the SIP message preamble and passes the SIP message preamble in the SIP header to Avaya Aura® Contact Center.
Return Expected	Option to specify whether the AACC treatment must return data to Avaya Aura® Contact Center. The following are the options: • **true**: If the AACC treatment must return data to Avaya Aura® Contact Center. • **false**: If the AACC treatment must not return data to Avaya Aura® Contact Center.

Editing an AACC treatment

About this task

Note:

You cannot edit the default AACC treatments. The default AACC treatments are prefixed with an asterisk (*) sign.

You cannot edit the name of a custom AACC treatment.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, click the project in which you want to edit a custom AACC treatment.
2. On the **Project** menu, click **Properties**.
3. In the Properties for <project name> dialog box, in the left navigation pane, click **Orchestration Designer**.
4. Click the **Pluggable Connectors** tab.
5. In the **Category** field, click **Contact Center**.
6. Click **AACC Treatments**.
7. In the **AACC Treatments Configuration** area, click the custom AACC treatment that you want to edit, and then click **Edit**.
8. In the AACC Treatments dialog box, edit the AACC treatment.

Deleting an AACC treatment

About this task

Note:

You cannot delete the default AACC treatments. The default AACC treatments are prefixed with an asterisk (*) sign.

If you delete a custom AACC treatment and if Avaya Aura® Contact Center transfers a call with a request for the AACC treatment that you deleted, the system shows run-time errors.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, click the project from which you want to delete a custom AACC treatment.
2. On the **Project** menu, click **Properties**.
3. In the Properties for <project name> dialog box, in the left navigation pane, click **Orchestration Designer**.
4. Click the **Pluggable Connectors** tab.
5. In the **Category** field, click **Contact Center**.
6. Click **AACC Treatments**.
7. In the **AACC Treatments Configuration** area, click the custom AACC treatment that you want to delete, and then click **Delete**.
8. In the Delete AACC Treatment dialog box, click **Yes**.

Example on assigning a return value to return the data to Avaya Aura® Contact Center

About this task

The following is an example on how to assign a value to the **returnvalue** variable field for the **Digit Collection** treatment type.

The speech application must collect the account number of the caller and pass the account number to Avaya Aura® Contact Center.

Note:

The following example uses the **Digit Collection** treatment type to collect the account number of the caller.

Procedure

1. In the call flow editor, drag a **Prompt and Collect** node.
2. In the Prompt and Collect node editor, click the **Input** item.
3. In the Property view of the **Input** item, set the **Name** property to `AccountNumber`, and then set the other properties.

   Note:

   The system creates a variable with the name **AccountNumber**.
4. Configure the other items in the **Prompt and Collect** node as required.

   Note:

   Use built-in grammars of type digits in the **Grammar** item.
5. In the call flow editor, drag a **Data** node after the **Prompt and Collect** node, and then double-click the **Data** node.
6. In the data node editor, drag an **Assign** operation.
7. In the Properties view of the **Assign** operation, map the **value** variable field of the **AccountNumber** variable to the **returnvalue** variable field of the **aacc_data** variable.

   Note:

   Similarly, for the **Context Creation** treatment type, you can map the variables that collect the data that you want to send to Avaya Aura® Contact Center to the variable fields that you create under the **aacc_context** variable to send the data to Avaya Aura® Contact Center.

Chapter 29: Call transfer to Avaya Aura® Contact Center

Call transfer to Avaya Aura® Contact Center by using the Open Interface Web Services

Transferring a call to Avaya Aura® Contact Center using the Open Interface Web Services

You might need to transfer a call to Avaya Aura® Contact Center in various scenarios. For example,

- The caller selects the option to talk to an Avaya Aura® Contact Center agent.
- You design the speech application in such a way that when the spoken or DTMF input provided by the caller does not match the grammars in the speech application, the speech application transfers the call to Avaya Aura® Contact Center for agent assistance.
- Caller data, such as name and date of birth of the caller, needs to be collected first from the caller so that the Avaya Aura® Contact Center agent can process the call further.

Use the **AACC AML Blind Transfer**, **AACC AML Bridged Transfer**, and **AACC AML Consultation Transfer** nodes in an Orchestration Designer speech application that is managed on Avaya Experience Portal to transfer a call from the Orchestration Designer speech application to an AML-based Avaya Aura® Contact Center. The Orchestration Designer speech application uses the Avaya Aura® Contact Center Open Interface Web Services to transfer the call.

You can also pass the call-associated user-to-user information (UUI), UCID, contact-intrinsic data, and call attached data (CAD) from the speech application to Avaya Aura® Contact Center.

When you add the **AACC AML Blind Transfer**, **AACC AML Bridged Transfer**, or **AACC AML Consultation Transfer** node to the call flow, the system creates a default structure for the node and adds the following items to the node:

- **Blind Transfer**, **Bridged Transfer**, or **Consultation Transfer** item depending on the node
- **Prompt** item
- **Next** item

In the **Blind Transfer**, **Bridged Transfer**, or **Consultation Transfer** item, you can specify the UUI data and UCID of the call that you want to pass to Avaya Aura® Contact Center.

Note:

UCID consumes 10 bytes of UUI data. Avaya Aura® Communication Manager can pass up to 96 bytes of hex- encoded UUI data.

AML-based solutions do not support SIP UUI data.

By default, the **Blind Transfer**, **Bridged Transfer**, and **Consultation Transfer** items use the **AML** transfer protocol and **Shared UUI** transfer mode to transfer the call and pass the call-associated data to Avaya Aura® Contact Center by using the Open Interface Web Services.

Avaya Aura® Contact Center Open Interface Web Services are a series of licensed SOAP-based open interfaces available to applications that are based on Service-Oriented Architecture (SOA). An Orchestration Designer speech application uses the Open Interface Web Services to reserve a Landing Pad on the target Avaya Aura® Contact Center and transfer the call along with the call-associated data to Avaya Aura® Contact Center. Avaya Aura® Contact Center Open Interface Web Services are supported in TDM-based and SIP-enabled contact center.

The Orchestration Designer speech application managed on Avaya Experience Portal processes an inbound call and collects the call-associated UUI, UCID, contact intrinsic data, and call attached data (CAD).

To transfer the call and pass the call-associated data to Avaya Aura® Contact Center by using the Open Interface Web Services, you must enable the **AACC Landingpad Web Service (Axis 1.4)** pluggable data connector.

After you enable the **AACC Landingpad Web Service (Axis 1.4)** pluggable data connector, Orchestration Designer adds the `aacclandingpad.jar` and `AACCLandingpadClient.properties` files under the `<project name>/WEB-INF/lib` directory.

In the `AACCLandingpadClient.properties` file, you:

- Must specify the Avaya Aura® Contact Center destination Controlled Directory Number (CDN) where you want to transfer the call.
- Must specify the URL of the Avaya Aura® Contact Center Open Interface Web Services.
- Can specify the telephone number that you want to use for simulation.
- Can configure the contact-intrinsic data and CAD data that you want to pass to Avaya Aura® Contact Center.

Note:

Landing Pads support up to 4K bytes of call attached data (CAD).

You can also use the CAD data in custom applications that are created by using Communication Control Toolkit (CCT). CCT is a toolkit to create custom applications to interact with Avaya Aura® Contact Center. CCT application is a client/server application that integrates a phone on a desktop with client/server-based applications.

When the speech application executes the AACC AML transfer node, the speech application connects to the Contact Center Manager Server (CCMS) Open Interface Web Services and sends a request for a Landing Pad reservation number. The speech application then provides the destination CDN that you specified in the `AACCLandingpadClient.properties` file, contact ID number, and call-associated data to the Open Interface Web Services. CCMS then reserves the Landing Pad and sends the reservation number to the speech application. The speech application uses the reservation number to transfer the call from Avaya Experience Portal to Avaya Aura® Contact Center.

For more information, see the *Avaya Aura® Contact Center and Avaya Voice Portal Integration* guide.

Example procedure for blind transfer of an inbound call to Avaya Aura® Contact Center by using the Open Interface Web Services

Before you begin

You must enable the **AACC Landingpad Web Service (Axis 1.4)** pluggable data connector to transfer a call to Avaya Aura® Contact Center by using the Avaya Aura® Contact Center Open Interface Web Services.

After you enable the **AACC Landingpad Web Service (Axis 1.4)** pluggable data connector, Orchestration Designer adds the `aacclandingpad.jar` and `AACCLandingpadClient.properties` files under the `<project name>/WEB-INF/lib` directory.

Procedure

1. Open the `AACCLandingpadClient.properties` file. Perform the following actions:
    a. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project in which you want to configure the `AACCLandingpadClient.properties` file.
    b. Double-click **WEB-INF** > **lib**.
    c. Right-click the `AACCLandingpadClient.properties` file, and then click **Open With** > **Properties File Editor**
2. Configure the `AACCLandingpadClient.properties` file. Perform the following actions:
    a. Specify the Avaya Aura® Contact Center destination Controlled Directory Number (CDN) where you want to transfer the call and the URL of the Open Interface Web Services.
    b. Specify the contact-intrinsic data and call attached data (CAD) that you want to pass to Avaya Aura® Contact Center along with the call.
    c. Specify the telephone number that you want to use for simulation.
3. Drag the **AACC AML Blind Transfer** node to the call flow from where you want to transfer the call to Avaya Aura® Contact Center.

The system creates a default node structure for the **AACC AML Blind Transfer** node, which comprises of a **Blind Transfer** item, **Prompt** item, and **Next** item.

4. Configure the **AACC AML Blind Transfer** node. Perform the following actions:
   a. Set the **Blind Transfer** item properties.
   b. **(Optional)** In the **Prompt** item, set the prompt that you want to play to the caller while the system transfers the call.
   c. In the **Next** item, specify the next node in the call flow that you want to execute.

   Note:

   You can also add other items to the node and configure the node as you want.
5. After you complete the call flow, export and deploy the speech application.

Configuring the AACCLandingpadClient.properties file to transfer an inbound call to Avaya Aura® Contact Center by using the Open Interface Web Services

About this task

Note:

The `AACCLandingpadClient.properties` file is available in a speech project only after you enable the **AACC Landingpad Web Service (Axis 1.4)** pluggable data connector.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project in which you want to configure the `AACCLandingpadClient.properties` file.
2. Double-click **WEB-INF** > **lib**.
3. Right-click the `AACCLandingpadClient.properties` file, and then click **Open With** > **Properties File Editor**.
4. In the **DestinationCDN** property, specify the constant Avaya Aura® Contact Center Controlled Directory Number (CDN) or the variable that contains the CDN where you want to transfer the call.

   If you specify a variable, enclose the variable name within {}.

   If you specify a complex variable, then specify the variable and variable field in the `{variable name:variable field name}` format.
5. In the **URL** property, specify the URL of the Avaya Aura® Contact Center Open Interface Web Services.
6. Specify the contact-intrinsic data. Perform the following actions:

Note:

You can pass up to five key name and value pairs of contact-intrinsic data to Avaya Aura® Contact Center.

Be sure that you delete the unused key name and value pairs.

a. In the **key<*number*>** property, type the key name.

Note:

Key names must be unique. The maximum length of key name is 25 characters.

b. In the **value<*number*>** property, specify the constant contact-intrinsic data or the variable that contains the contact-intrinsic data that you want to pass to Avaya Aura® Contact Center.

If you specify a variable, enclose the variable name within {}. For example, `{PreferredAgent}`.

If you specify a complex variable, then specify the variable and variable field in the `{variable name:variable field name}` format. For example, `{CCData:PreferredAgent}`.

Note:

The maximum length of a value is 80 characters.

7. In the **CAD** property, specify the constant call attached data (CAD) data or the variable that contains the CAD data that you want to pass to Avaya Aura® Contact Center.

   If you specify a variable, enclose the variable name within {}. For example, `{PhoneNumber}`.

   If you specify a complex variable, then specify the variable and variable field in the `{variable name:variable field name}` format. For example, `{CustomerName:FirstName}`.

   Note:

   Landing Pads support up to 4K bytes of CAD data.

8. Save and close the `AACCLandingpadClient.properties` file.

Call transfer to a SIP-enabled Avaya Aura® Contact Center

Transferring a call to a SIP- enabled Avaya Aura® Contact Center

You might need to transfer a call to Avaya Aura® Contact Center in various scenarios. For example,

- The caller selects the option to talk to an Avaya Aura® Contact Center agent.
- You design the speech application in such a way that when the spoken or DTMF input provided by the caller does not match the grammars in the speech application, the speech application transfers the call to Avaya Aura® Contact Center for agent assistance.
- Caller data, such as name and date of birth of the caller, needs to be collected first from the caller so that the Avaya Aura® Contact Center agent can process the call further.

Use the **AACC Blind Transfer**, **AACC Bridged Transfer**, and **AACC Consultation Transfer** nodes in an Orchestration Designer speech application to transfer a call from the Orchestration Designer speech application that is managed on Avaya Experience Portal to a SIP-enabled Avaya Aura® Contact Center for assisted care. You can also pass the call-associated user-to-user information (UUI), UCID, contact-intrinsic data, and call attached data (CAD) from the Orchestration Designer speech application to Avaya Aura® Contact Center.

When you add the **AACC Blind Transfer**, **AACC Bridged Transfer**, or **AACC Consultation Transfer** node to the call flow, the system creates a default structure for the node and adds the following items to the node:

- **Blind Transfer**, **Bridged Transfer**, or **Consultation Transfer** item depending on the node.
- **Prompt** item.
- Two **External Property** items that represent the name and value pair of the SIP header. If you want to add more SIP headers, you can add more **External Property** items. You can pass the contact-intrinsic data in the SIP headers. If you want to pass more contact-intrinsic data, you can use the P-Intrinsics SIP header in the **External Property** items. You can also pass CAD data by using the P-Intrinsics SIP header. P-Intrinsics can pass up to 1024 bytes of hex-encoded data.
- **Next** item.

In the **Blind Transfer**, **Bridged Transfer**, or **Consultation Transfer** item, you:

- Must specify the Avaya Aura® Contact Center destination Controlled Directory Number (CDN) where the speech application must transfer the call.
- Can specify the UUI data and UCID of the call that you want to pass to Avaya Aura® Contact Center along with the call.

Note:

UCID consumes 10 bytes of UUI data. Avaya Aura®Communication Manager can pass up to 96 bytes of hex- encoded UUI data.

By default, the **Blind Transfer**, **Bridged Transfer**, and **Consultation Transfer** items within the **AACC Blind Transfer**, **AACC Bridged Transfer**, and **AACC Consultation Transfer** nodes use the **SIP** transfer protocol and **Shared UUI** transfer mode to transfer the call and pass the call-associated data to Avaya Aura® Contact Center.

The Orchestration Designer speech application managed on Avaya Experience Portal processes the inbound call and collects the call-associated UUI and contact-intrinsic data.

When the speech application executes the AACC transfer node, the speech application connects to Avaya Aura® Contact Center Manager Server (CCMS) and provides the destination CDN, contact ID number, call-associated UUI, UCID, and contact-intrinsic data to CCMS. The speech application then transfers the call from Avaya Experience Portal to Avaya Aura® Contact Center.

For more information, see *Avaya Aura® Contact Center and Avaya Voice Portal Integration* guide.

Example procedure for blind transfer of a call to a SIP-enabled Avaya Aura® Contact Center

Procedure

1. Drag the **AACC Blind Transfer** node to the call flow from where you want to transfer the call to Avaya Aura® Contact Center.

   The system creates a default node structure for the **AACC Blind Transfer** node, which comprises of a **Blind Transfer** item, **Prompt** item, two **External Property** items, and **Next** item.

2. Configure the **AACC Blind Transfer** node. Perform the following actions:

   a. Set the **Blind Transfer** item properties.

   b. **(Optional)** In the **Prompt** item, set the prompt that you want to play to the caller while the system transfers the call.

   c. In the **External Property** items, set the SIP header name and value.

      To add more SIP headers, copy the two default **External Property** items or drag two **External Property** items from the Palette pane, and set the SIP header name and value.

      You can pass the contact-intrinsic data in the SIP headers.

      If you want to pass more contact-intrinsic data, you can use the P-Intrinsics SIP header in the **External Property** items. You can also pass CAD data by using the P-Intrinsics SIP header. P-Intrinsics can pass up to 1024 bytes of hex-encoded data.

   d. In the **Next** item, specify the next node in the call flow that you want to execute.

3. After you complete the call flow, export and deploy the speech application.

Chapter 30: Application testing by simulation

Testing applications by simulation

One of the strongest features of Orchestration Designer is the capability for testing and debugging your speech, message, data, and web application projects. In Orchestration Designer, you can simulate and test virtually every aspect of your speech, message, data, and web application project, including its response to error conditions.

The simulation and debugging tools in Orchestration Designer make it possible for you to test virtually every part of your speech, message, data, and web application project before you deploy it to a live IVR or messaging system. This means that you can spend less time having to test and debug your speech, message, data, and web applications on the target system.

What can be tested by simulation

In Orchestration Designer, you can simulate the following features and functions of a call flow:

- The calling number (ANI)
- The called number (DNIS)
- DTMF inputs
- ASR inputs, either using a microphone or by typing in a response
- Problems with input recognition, such as No Match and No Input conditions
- Caller hang ups during the call
- Call transfers and all possible transfer results
- Passing variable values from one module to another, when you are testing individual modules
- Confidence level of ASR recognition
- Interaction Center (IC) and Application Enablement Sevices (AES) connectors, using a connector simulator and connector scripting

Limitations of testing by simulation

You cannot test the following functions by simulation in Orchestration Designer:

- Databases - Orchestration Designer does not provide a way to simulate database operations, so if you want to test those, you must test with a real database. You can, however, use a pared down version of a database for development and testing and then change to a run-time version just before you deploy the application.
- Web services - Orchestration Designer does not provide a way to simulate Web service operations, so if you want to test those, you must test with a real Web service. You can, however, use a local or a pared down version of a Web service for development and testing and then change to a run-time version just before you deploy the application.
- When using MRCP for ASR in the simulator, make sure you select the right grammar compatibility depending on the speech server you are using. Select your project and go to Project -> properties -> Orchestration Designer-> Speech and select the needed grammar compatibility.
- When MRCP ASR is enabled, you cannot provide text input for speech recognition.
- MRCP V2 is not supported in this release.
- When using MRCP for ASR, spoken input should be provided to the simulator using a microphone only after you see the message “Waiting for ASR input” in the simulator.
- In simulation, since there is only one input (microphone) and one output (speakers) for multiple call appearances, dialogs must be “serialized” to use the resources. Once a dialog starts a form, it must complete before the next form with a call appearance may use the resources. This generally provides a reasonable simulation of an application, however, in a CCXML application you have multiple call appearances and if one has input, a subsequent call appearance playing an informational message will be blocked until the input is completed and exists. This can occur in the ConferencePredefined application when a conference attendee hangs up. Since the incoming call has a dialog running, listening for an add or drop command, the message “XXX left the conference” will not be heard until the input on the incoming call completes. This does not happen on the platform.
- There is no support for G729 encoding in the simulator.

Debugging features in Orchestration Designer

Orchestration Designer provides the following features and tools designed to help you debug applications:

- Highlighting during simulation. While the application project is in a simulation mode, the Orchestration Designer highlights each node as the Avaya Application Simulator (AAS) processes the nodes. This feature makes it easy to track progress through the application as the simulation progresses.

- Input tab, AAS progress display. During application simulation, the AAS presents a step-by-step readout of what is happening as the simulation progresses. The AAS displays this progress in a pane of the **Input** tab. Even after the simulation ends, you can scroll back through this output to analyze what happened during simulation.
- Debug tracing to output in Console view or trace log file. If debug tracing is enabled, Orchestration Designer directs the output of the debug tracing to two destinations: the Console view display and the trace log file. This output consists primarily of a transcript of the VoiceXML and textXML output that is generated while the application is being run. For more information, see Trace on page 846.
- Application tracking node with debug options. Orchestration Designer provides a special node, the application Tracking node, which was designed to aid in debugging applications. This node makes use of two palette options—the **Trace** item and the **Report** item—to help with application tracking and debugging. For more information, see Report on page 815 and Trace on page 846.
- Scripting of inputs and responses. For those situations where you do not want or are not able to use the various built-in mechanisms for simulating caller responses or responses for message applications, you can create an XML script to tell the application how the caller responds during simulation. For more information, see Creating scripts for testing on page 587.
- Highlighting of Operation items that use a local variable. When you click a local variable within a Data Node Editor, the system highlights all the Operation items that use the local variable within the data node. You can use the Operation items that the system highlights to track and debug the logic that uses the local variable within the data node.

  Note:

  You can click the root item in the Data Node Editor to clear the selection.

Using element level debugger in the simulation environment

With the increasing complexity of speech applications that you can design using Orchestration Designer, it becomes imperative to identify and correct errors when designing the application. In order to build an error free application or reduce the number of errors, Orchestration Designer provides the element level debugging feature for debugging call flows for speech applications and message flows for message applications in the simulation environment.

The element level debugger is available from the **Debug Application** button on the application simulator.

This debugger is integrated into the standard Eclipse debugging framework, and uses the same views and framework as Java debugging or any Eclipse debugger.

Note:

- You must run the debugger in the debug perspective. For information about opening the debug perspective, see Setting the Debug perspective on page 390.
- You must clear the **Don't run Tomcat in debug mode** check box if you want to debug the breakpoints set in the Java code. For more information, see Debugging the breakpoints set in the Java code on page 390.

Instead of setting breakpoints in Java code, you can now focus on the application design and debug it by setting breakpoints on the Flow or Data nodes, without touching the Java code.

For information about setting breakpoints in the application call flow or message flow, see Call flow editor and message flow editor on page 129.

You can also enable or disable the breakpoints in the same view as Java breakpoints.

When the debug process hits a breakpoint:

- The associated node and node item is highlighted in the call flow or message flow
- Display which breakpoint is hit
- Display process status
- Displays the current simple, complex and collection variables. You can also modify these variables.
- Displays the execution path of the node that is being executed.
- User can resume the debug process, and step into, step return, and step over a Sub Flow.
- The node with an unhandled run-time error is highlighted in red.

Debug perspective views

The Debug perspective shows the following views when you debug an application:

Name	Description
Debug	Shows the runtime steps in the debug process.
Servers	Shows a list of all the servers and projects that are associated with that server. You can use the Servers view to determine the current status and state of the server and the projects added to the server from the workbench. Do not use the Servers view for the debug process.
Variables	Shows all the variables in the application. You can monitor or modify the values of the appropriate variables to validate the application.

Table continues…

Name	Description
Breakpoints	Shows a list of all breakpoints set in the application. This list includes the breakpoints set on the node items and the Java code. You can clear the check box associated with each breakpoint to temporarily disable it. You can also disable the breakpoints by right-clicking the node item and select Disable Breakpoint. You can delete one or all the breakpoints by selecting the breakpoints and clicking the  or  button.
Execution path	Shows a list of already executed nodes and the node that is currently being executed, with their locations in the call flow.

Setting the Debug perspective

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the **Preferences** dialog box.
2. In the left navigation pane, double-click **Run/Debug**.
3. Click **Perspectives**.
4. In the right pane, in the **Mode/Perspectives** area, in the **Debug** field, click **Debug**.
5. Click **Apply**, and then click **OK**.

Debugging the breakpoints set in the Java code

About this task

You can configure the setting to debug the breakpoints set in the Java code. Instead of setting breakpoints in Java code, you can now focus on the application design and debug it by setting breakpoints on the Flow or Data nodes, without touching the Java code.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Tomcat**.
3. Click **JVM Settings**.

4. Clear the **Don't run Tomcat in debug mode** check box.
5. Click **Apply**, and then click **OK**.

Project flow breakpoint management

Adding a breakpoint to a project flow

Procedure

1. Open the application in which you want to add a breakpoint.
2. In the project flow editor, perform one of the following actions:
   - To add a breakpoint to a call, or message, or web flow node, right-click the node item to which you want to add a breakpoint.
   - To add a breakpoint within a data node, double-click the data node to open the data node editor, right-click the node entry where you want to add a breakpoint.
3. Click **Add Breakpoint**.

Disabling a breakpoint in a project flow

About this task

You can temporarily disable breakpoints in a project flow.

Procedure

1. Open the application in which you want to disable a breakpoint.
2. In the project flow editor, perform one of the following actions:
   - To disable a breakpoint in a call flow, or message flow, or web flow node, right-click the node item in which you want to disable a breakpoint.
   - To disable a breakpoint within a data node, double-click the data node to open the data node editor, right-click the node entry where you want to disable a breakpoint.
3. Click **Disable Breakpoint**.

Removing a breakpoint from a project flow

Procedure

1. Open the application from which you want to remove a breakpoint.

2. In the project flow editor, perform one of the following actions:
   - To remove a breakpoint from a call flow, or message flow, or web flow node, right-click the node item from which you want to remove a breakpoint.
   - To remove a breakpoint from a data node, double-click the data node to open the data node editor, right-click the node entry from where you want to remove a breakpoint.
3. Click **Remove Breakpoint**.

Enabling tracing for simulation

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Avaya** > **Application Simulation Orchestration Designer Simulation**.
3. Select the **Enable Orchestration Designer logging of tracing output** check box.
4. Select the **Enable Application logging of tracing output** check box.

   These and other Avaya Application Simulator preferences are described in Configuring Application Simulation preferences on page 519.
5. Click **Apply**, and then click **OK**.

   Tip:

   When you enable the **Enable Orchestration Designer logging of tracing output** option, Orchestration Designer shows the VoiceXML or textXML output in the Console view and writes the output to a trace log file. For more information, see Console view on page 45. This file is located at *applicationName*`/data/log/trace.log-`*yyyy-mm-dd*`.log`, where *applicationName* represents the top level application directory, and *yyyy-mm-dd* represents simulation runs generated today. To view this log file, locate the file in the Navigator pane and double-click the file name.

Debugging an application

Procedure

1. Open the project that you want to debug.
2. In the application call flow, or message flow, or web flow editor, set the breakpoints.

   For more information, see Adding a breakpoint to a project flow on page 391.

Note:

To temporarily disable a breakpoint, see Disabling a breakpoint in a project flow on page 391. To remove a breakpoint, see Removing a breakpoint from a project flow on page 391.

3. After the breakpoints are set, in the application simulator, click **Debug Application**.

   The debug process starts. The debug process suspends temporarily when the debugger hits a breakpoint.

4. Monitor the state of the application, values of the variables, breakpoints and their execution path using the **Variables**, **Breakpoint**, and the Execution path views in the Debug perspective.

   Note:

   The breakpoints that are currently being executed are highlighted in the call flow or message flow or web flow. Any unhandled errors are represented by highlighting that particular node in red.

5. Modify the values of the variables in the Variables view in the Debug perspective.
6. Resume the debug process by clicking **Run** > **Resume** or by clicking  .

   You can also use the step into, step over or step return the Sub Flow to move into or move out of a Sub Flow.

7. Repeat Step 4 through Step 6 for all the breakpoints.

Setting Simulation Store base path

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.

2. In the left navigation pane, double-click **Avaya Aura** > **Application Simulation** > **Orchestration Designer Simulation** > **Simulation Store** .
3. In the Simulation Store window, enter the directory path in the **Simulation Store Base Path** field.
4. Click **OK**.

Specifying a telephone number to simulate call transfer to Avaya Aura® Contact Center

About this task

To simulate a speech application that uses the **AACC AML Blind Transfer** node, **AACC AML Bridged Transfer** node, or **AACC AML Consultation Transfer** node to transfer calls to Avaya Aura® Contact Center, you must specify a telephone number that you want to use for simulation in the `AACCLandingpadClient.properties` file.

During simulation, the speech application does not invoke the Avaya Aura® Contact Center Open Interface Web Services. Instead, the speech application uses the telephone number that you specify to simulate the call transfer.

Note:

The `AACCLandingpadClient.properties` file is available in a speech project only after you enable the **AACC Landingpad Web Service (Axis 1.4)** pluggable data connector.

Before deploying the speech application, you must delete the telephone number that you specify for simulation or comment out the **Simulation** property.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, double-click the speech project in which you want to configure the `AACCLandingpadClient.properties` file.
2. Double-click **WEB-INF** > **lib**.
3. Right-click the `AACCLandingpadClient.properties` file, and then click **Open With** > **Properties File Editor**.
4. In the **Simulation** property, type the telephone number that you want to use to simulate the call transfer.
5. Save and close the `AACCLandingpadClient.properties` file.

Media Resource Control Protocol support

Media Resource Control Protocol (MRCP) is designed for network-based solutions where ASR and TTS servers are configured to work together with VoiceXML interpreters, media gateways, and application servers using a VoIP connection. MRCP uses Real Time Protocol (RTP) as the transport mechanism for audio information captured from callers as well as audio played to callers either from recordings or conversion of Text-to-speech. Real Time Streaming Protocol (RTSP) is used as the foundation of the control between the media gateway and speech recognition and TTS servers. In addition to the standard RTSP, MRCP can handle barge-in events for ASR, manage rate and volume of TTS output, and support various speech recognition and TTS management issues.

The Avaya Application Simulator supports the MRCP protocol to allow you to simulate your applications against actual third-party speech systems which are installed on a remote server. For example, a shared speech server in a lab that is used by multiple Orchestration Designer installations and Avaya Experience Portal.

You can test the application in the simulator against the same third-party speech systems that the application uses in the deployment. For example, if you are using Nuance or Loquendo, you can test the application using the MRCP feature. This allows for more thorough testing including additional language support, SSML, n-best results, and more accurate speech recognition.

For information about configuring the Media Resource Control Protocol feature, see Configuring MRCP settings on page 521.

While testing an application by simulation, you can:

- Specify input parameters.
- Simulate a variety of error conditions.
- Test for DTMF and spoken responses.
- Interact with databases, actually or by simulation.
- Invoke Web services.

For more information, see Application testing by simulation on page 386.

Simulation management

Using the Avaya Application Simulator to simulate applications

About this task

During simulation, you can test a wide variety of run-time and event scenarios.

Note:

Avaya recommends that you open the appropriate perspective for the project that you want to simulate. For example, to simulate a speech application project, open the Speech perspective.

Use the Console view to view the simulation event logs.

Procedure

1. Generate the project that you want to simulate. Perform the following actions:
   a. In the Navigator view or in the Avaya OD Navigator view, click the project that you want to generate.
   b. On the **Project** menu, click **Generate** > **Generate Project**.

Note:

If the system encounters an error while generating the project, the system displays an alert message. You must cancel the rest of the operation, rectify the errors, and then regenerate the project. Use the Problems view to view the errors.

2. On the **Tomcat** menu, click **Start Tomcat**.

   Note:

   If you do not start Tomcat, the system automatically starts Tomcat when you simulate the application.

   If you make changes to your application while Tomcat is running, you must save the application before you simulate the application again. Each time you save the application, Tomcat automatically updates the context to stay in sync with the latest version of the application. Be aware, however, that if your application uses modules, Tomcat does not automatically update module contexts.

3. If you make changes to an application that uses modules, perform one of the following actions to update the module context:

   - Restart Tomcat before you simulate the application again.

     This option takes more time, but is more reliable.

   - To manually reload the context for the application, in the Navigator view or in the Avaya OD Navigator view, right-click the <*project name*> directory, then click **Tomcat project** > **Reload this context**.

     If the system shows an error, consult the Tomcat documentation to verify that you have set up the Tomcat Manager application properly.

4. On the **Window** menu, click **Show view** > **Application Simulator**.
5. In the Application Simulator view, on the Application tab, in the **Available Projects** area, click the application project that you want to simulate.
6. In the **Simulation Profile** field, click the simulation profile that you want to use to simulate the application.
7. Click **Run Application**.

Result

- Avaya Application Simulator (AAS) verifies if Tomcat is running. If Tomcat is not running, the system automatically starts Tomcat. AAS then runs the application simulation.
- If you simulate a speech application, AAS displays the **Call::uid_<*uid number*>** tab. You can simulate run-time event scenarios for the speech application on the **Call::uid_<*uid number*>** tab.
- If you simulate a call control application, AAS displays the **Call:tel:<*telephone number*>:uid_<*uid number*>** tab. You can simulate call failure events, call progress events, and call classification scenarios for the call control application on the **Call:tel:<*telephone number*>:uid_<*uid number*>** tab.

- If you simulate a message application or an application that contains a multichannel notification, AAS displays the Messages tab. The Messages tab displays the outbound messages and notifications that the application sends during application simulation.

Next steps

To:

- Simulate DTMF input, see Simulating DTMF inputs for a speech application on page 401.
- Simulate spoken input with a microphone, see Simulating spoken inputs with a microphone for a speech application on page 402.
- Simulate spoken input without a microphone, see Simulating spoken inputs without a microphone for a speech application on page 403.
- Send an SMS message to AAS to simulate an SMS channel message flow project, see Sending an SMS message to AAS to simulate an SMS channel message application on page 405.
- Send an email message to AAS to simulate an email channel message flow project, see Sending an email message to AAS to simulate an email channel message application on page 406.
- Simulate call failure events, call progress events, and call classification scenarios for a call control application, see Simulating run-time events and scenarios for a call control application on page 404.

Application Simulator view: Application tab option descriptions

Name	Description
Available Projects	The project that you want to simulate. You can select only one project at a time.
Input Parameters	The input expected by a module that is used by the speech application that you simulate. The format of the input parameter must match the format that the speech application expects. **Note:** Avaya Application Simulator (AAS) activates the **Input Parameters** area if the module that a speech application uses expects input from another module or speech application.

Table continues…

Name	Description
Speech Project for Call Control	The speech project that you want to simulate along with the call control project. **Note:** The **Speech Project for Call Control** area is available only after you select a call control project in the **Available Projects** area.
Simulation Profile	The simulation profile that you want to use to simulate the project.
Run Application	Starts the simulation. **Note:** The **Run Application** option is available only after you select a project in the **Available Projects** area. For web applications, it launches the system web browser by default. You can change the web browser using **Windows** > **Preferences**.
Debug Application	Debugs the application. **Note:** The **Debug Application** option is available only after you select a speech project or a message flow or web flow project in the **Available Projects** area. Be sure that you set the **Debug** perspective to run the debugger. For more information, see Setting the Debug perspective on page 390 and Using element level debugger in the simulation environment on page 388.

Table continues…

Name	Description
End Application	Ends the simulation before the simulation is complete. **Note:** The **End Application** option is available only when a project is in the simulation mode. In speech application, the **End Application** option is different from the **Hang Up** option that is available on the **Call::uid_<*uid number*>** tab. When you click **End Application**, the system stops the application wherever it is. The **Hang Up** option simulates a scenario in which the caller gets disconnected before reaching the end of the call flow either by hanging up or by some other means. For more information, see Application Simulator view: Call::uid_<uid number> tab option descriptions on page 406.
Do Not Run As Module	Check box to simulate an Orchestration Designer speech application that is a module, as a standalone application. If you clear the **Do Not Run As Module** check box, the system generates a default VXML page to invoke the Orchestration Designer application as a subdialog. **Note:** The **Do Not Run As Module** check box is available only after you select a speech project in the **Available Projects** area.
SMS Simulation	The inbound SMS message that you send to the application by using the **Enter SMS Message** field and the outbound SMS message response sent by the application. **Note:** The **SMS Simulation** area is available only after you select an SMS channel message flow project in the **Available Projects** area.

Table continues…

Name	Description
Enter SMS Message	The SMS message body contained in the simulation profile that you select in the **Simulation Profile** field. You can also type any other SMS message in the **Enter SMS Message** field if you want to override the SMS message contained in the simulation profile. AAS simulates the SMS message contained in the **Enter SMS Message** field as an inbound SMS message. **Note:** The **Enter SMS Message** field is available only after you select an SMS channel message flow project in the **Available Projects** area.
Email Simulation	The inbound email message that you send to the application by using the **Enter Email Message** field and the outbound email message response sent by the application. **Note:** The **Email Simulation** area is available only after you select an email channel message flow project in the **Available Projects** area.
Enter Email Message	The email message body contained in the simulation profile that you select in the **Simulation Profile** field. You can also type any other email message in the **Enter Email Message** field if you want to override the email message contained in the simulation profile. AAS simulates the email message contained in the **Enter Email Message** field as an inbound email message. **Note:** The **Enter Email Message** field is available only after you select an email channel message flow project in the **Available Projects** area.

Table continues…

Name	Description
Send	Option to send the email message or SMS message contained in the **Enter Email Message** or **Enter SMS Message** fields as an inbound message for simulation. **Note:** The **Send** option is available only after you simulate an email channel message flow project or an SMS channel message flow project. For more information, see Using the Avaya Application Simulator to simulate applications on page 395.

Simulating run-time scenarios

During simulation of an application with the Avaya Application Simulator (AAS), you can test a wide variety of run-time scenarios. These scenarios include:

- Caller inputs and telephony system responses on page 401 - Caller inputs include DTMF and spoken responses to prompts. Telephony system responses include primarily transfer errors.
- Responses from scripts on page 414 - You can use automated scripts to simulate two types of responses:
  - Response scripts simulate verbal responses, message inputs, and telephony system responses to various conditions.
  - IC and Application Enablement Services (AES) connector functions simulate IC system and AES system responses to various types of requests and error conditions.
- Other inputs on page 414 - Other inputs include inputs from other modules, database operations, and Web service operations.

Caller inputs and telephony system responses

Usually, a speech application works by prompting a caller for inputs and then interpreting and responding to those inputs. Inputs can be in the form of DTMF (touchtone) key presses or spoken replies. In Orchestration Designer, you can simulate those responses when you test applications.

Simulating DTMF inputs for a speech application

About this task

You can provide DTMF inputs to simulate run-time and event scenarios for a speech application or a call control application that uses a speech application.

To indicate that Avaya Application Simulator (AAS) is waiting for a DTMF input, the **Waiting DTMF** option turns green.

Note:

You can also simulate other events such as No Input, No Match, Record End, Hang Up, and Xfer Status (Transfer status). For more information, see Application Simulator view: Call::uid_<uid number> tab option descriptions on page 406.

Use the Console view to view the simulation event logs.

Procedure

1. Perform one of the following actions:
   - Simulate the speech application.
   - Simulate the call control application that uses a speech application, and select **live_voice** as the call classification scenario.

   For more information, see Using the Avaya Application Simulator to simulate applications on page 395 and Simulating run-time events and scenarios for a call control application on page 404.
2. When the **Waiting DTMF** option turns green on the **Call::uid_*<uid number>*** tab, click the digits on the DTMF keypad that you want to send as a DTMF input for simulation.

   When you click the digits on the DTMF keypad, AAS simulates the corresponding DTMF tone and displays the digits in the field below the DTMF keypad.
3. Click **Send Digits**.

   Note:

   Depending on your monitor size and resolution, increase the size of the Application Simulator view to view the **Send Digits** button.

Simulating spoken inputs with a microphone for a speech application

Before you begin

Before you use a microphone to simulate ASR input, you must install and configure Microsoft Speech API 6.0.

For more information, see the *Getting started with Orchestration Designer* guide.

Note:

Microsoft Windows 7 already have Microsoft Speech API 6.0 installed.

If Microsoft SAPI Speech is already installed on Windows 7, verify that the **Speech Recognition** and **Text-to-Speech** tabs are available in **Control Panel** > **Speech Recognition** > **Text-to- Speech**.

About this task

When you simulate a speech application or a call control application that uses a speech application, then depending on the call flow, Avaya Application Simulator (AAS) prompts for a response. To indicate that AAS is waiting for a spoken response, the **Waiting ASR** option turns green.

You can also simulate barge-in by using AAS. By default, the barge-in feature of AAS is disabled. You can enable the barge-in feature by selecting the **Enable Speech Bargein detection** check box. For more information, see Configuring Orchestration Designer Simulation preferences on page 520.

Note:

Use a headset when running AAS with barge-in enabled because a microphone of a standard computer can easily pick up sounds from the computer speakers.

You can also simulate other events such as No Input, No Match, Record End, Hang Up, and Xfer Status (Transfer status). For more information, see Application Simulator view: Call::uid_<uid number> tab option descriptions on page 406.

Use the Console view to view the simulation event logs.

Procedure

1. Perform one of the following actions:
   - Simulate the speech application.
   - Simulate the call control application that uses a speech application, and select **live_voice** as the call classification scenario.

   For more information, see Using the Avaya Application Simulator to simulate applications on page 395 and Simulating run-time events and scenarios for a call control application on page 404.
2. When the **Waiting ASR** option turns green on the **Call::uid_<*uid number*>** tab, with a microphone attached to your computer and turned on, speak the response.

Result

If AAS recognizes the spoken input, Orchestration Designer automatically moves on to the next node of the call flow. If AAS does not recognize the spoken input, the application throws a No Match event and AAS responds with the appropriate No Match prompt.

Simulating spoken inputs without a microphone for a speech application

About this task

You can simulate ASR input without using a microphone. You can type the ASR input that you want to simulate instead of using the microphone.

You can provide ASR input for simulation only when you simulate a speech application or a call control application that uses a speech application.

During simulation, Avaya Application Simulator (AAS) prompts for a response. To indicate that AAS is waiting for a spoken response, the **Waiting ASR** option turns green.

Note:

AAS does not evaluate or validate the ASR response that you type. AAS processes whatever ASR response that you type as valid. Hence, it is better to simulate an application by using a microphone.

You can also simulate other events such as No Input, No Match, Record End, Hang Up, and Xfer Status (Transfer status). For more information, see Application Simulator view: Call::uid_<uid number> tab option descriptions on page 406.

Use the Console view to view the simulation event logs.

Procedure

1. Perform one of the following actions:
   - Simulate the speech application.
   - Simulate the call control application that uses a speech application, and select **live_voice** as the call classification scenario.

   For more information, see Using the Avaya Application Simulator to simulate applications on page 395 and Simulating run-time events and scenarios for a call control application on page 404.
2. When the **Waiting ASR** option turns green on the **Call::uid_<*uid number*>** tab, in the **Recognition** field, type the ASR input that you want to simulate.
3. To specify a confidence level for the ASR server response, in the **Confidence** field, type a number between `0.0` and `1.0`.

   Where the number 0.0 indicates minimum confidence and the number 1.0 indicates maximum confidence.

   Note:

   The **Recognition** and **Confidence** fields are unavailable if you enable the MRCP feature. If you enable the MRCP feature, you must provide spoken input by using a microphone.
4. Click **Send ASR**.

Simulating run-time events and scenarios for a call control application

About this task

Note:

Use the Console view the view the simulation event logs.

Procedure

1. Simulate the call control application.

   For more information, see Using the Avaya Application Simulator to simulate applications on page 395.
2. On the **Call:tel:<*telephone number*>:uid_<*uid number*>** tab, perform one of the following actions:
   - To simulate a call connection failure event, in the drop-down list that is located above **Send Call Progress**, click the call connection failure event that you want to simulate.

     The system sends a connection.failed event to the application.

- To simulate a call classification scenario, click the call classification scenario that you want to simulate in the drop-down list that is located above the **Progress Info** field, and then click **Answer**.

  When you click **Answer**, the system sends a connection.signal event followed by a connection.connected event to the application. The connection.signal event contains the event$.info.callprogress variable. The event$.info.callprogress variable contains the results of the call classification scenario that you select.

- To simulate DTMF and spoken input, click **live_voice** in the drop-down list that is located above the **Progress Info** field, and then click **Answer**.

  AAS displays the option to simulate DTMF and spoken inputs depending on the call flow.

  Note:

  The default value is **live_voice**.

  For more information, see Simulating DTMF inputs for a speech application on page 401, Simulating spoken inputs with a microphone for a speech application on page 402, and Simulating spoken inputs without a microphone for a speech application on page 403.

- To simulate a call connection progress event before a call is connected, perform the following actions:

  a. Type the call connection progress information in the **Progress Info** field, and then click **Send Call Progress**.

     The system sends a connection.progressing event to the application. The connection.progressing event contains the call progress information that you type in the **Progress Info** field.

  b. Simulate a call classification scenario or a call failure event.

Sending an SMS message to AAS to simulate an SMS channel message application

Procedure

1. Simulate the SMS channel message application.

   For more information, see Using the Avaya Application Simulator to simulate applications on page 395.

   The **Enter SMS Message** field shows the SMS message contained in the simulation profile that you select in the **Simulation Profile** field.

2. To override the SMS message contained in the simulation profile, type the SMS message in the **Enter SMS Message** field.

3. Click **Send**.

The **SMS Simulation** area shows the inbound SMS message that you sent by using the **Enter SMS Message** field and the outbound SMS message response sent by the application.

4. Click the Messages tab to view the outbound SMS messages and email notifications that the application sends during application simulation.

Sending an email message to AAS to simulate an email channel message application

About this task

Note:

The current release of Orchestration Designer does not support attachments in inbound email messages.

Procedure

1. Simulate the email channel message application.

   For more information, see Using the Avaya Application Simulator to simulate applications on page 395.

   The **Enter Email Message** field shows the email message contained in the simulation profile that you select in the **Simulation Profile** field.

2. To override the email message contained in the simulation profile, type the email message in the **Enter Email Message** field.
3. Click **Send**.

   The **Email Simulation** area shows the inbound email message that you sent by using the **Enter Email Message** field and the outbound email message response sent by the application.

4. Click the Messages tab to view the outbound email messages and SMS notifications that the application sends during application simulation.

Application Simulator view: Call::uid_<uid number> tab option descriptions

The **Call::uid_<*uid number*>** tab is available only when you simulate a speech application or when you select **live_voice** as the call classification scenario when you simulate a call control application that uses a speech application.

You can simulate a variety of run-time and event scenarios during application simulation.

You can simulate events to determine how a speech application responds when the unexpected happens. For instance, if a caller does not respond when prompted, the system throws a No Input event. You can use event handlers to instruct the system how to respond to such events.

Note:

Use the Console view to view the simulation event logs.

Name	Description
DTMF keypad	Number keys to provide DTMF input for simulation. **Note:** The DTMF keypad is available only after the **Waiting DTMF** option turns green.
Send Digits	Option to send the DTMF input that you specify by using the DTMF keypad for simulation. **Note:** The **Send Digits** option is available only after the **Waiting DTMF** option turns green.
Call Active	Green indicates that the call is active. Red indicates that the call is processed, disconnected, or hung up.
Waiting ASR	Green indicates that Avaya Application Simulator (AAS) is waiting for an ASR input.
Waiting DTMF	Green indicates that AAS is waiting for a DTMF input.
Input Result drop-down list	
No Input	Option to instruct AAS to interpret a period of silence as a No Input event. AAS then processes the response to the prompt as if the caller did not respond and throws a No Input event. **Note:** AAS does not use timeout settings to determine when a caller has not responded to a prompt.
No Match	Option to simulate the ASR input that you type in the **Recognition** field as a No Match event, in which the response by the caller does not match any expected response. AAS then throws a No Match event. Use this option because the ASR engine that Orchestration Designer uses for simulation is capable of recognizing input, but it is not as robust as many third-party ASR engines.

Table continues…

Name	Description
Send ASR	Option to send the ASR input that you type in the **Recognition** field for simulation. **Note:** The **Send ASR** option is available only after the application that you simulate prompts you for an ASR input.
Record End	Option to simulate a scenario in which the system should terminate a recording based either on an extended silence or on a DTMF key press. AAS simulates a scenario as if the system terminates the recording, and then proceeds according to the application call flow. Use this option because AAS cannot detect extended silence at the end of a recording and cannot automatically use record timeout settings to terminate the recording. **Note:** The **Record End** option is available only if AAS encounters a **Record** node while simulating the application. AAS does not record the entire call. AAS only records the spoken inputs that you provide.
Hang up	Option to simulate a scenario in which the caller hangs up before reaching the end of the call flow. AAS then throws a caller disconnect event.
Recognition	The spoken input that you want to send for simulation without using a microphone. **Note:** The **Recognition** field is unavailable if you enable the MRCP feature. If you enable the MRCP feature, you must provide spoken input by using a microphone. For more information, see Simulating spoken inputs with a microphone for a speech application on page 402 and Configuring MRCP settings on page 521.

Table continues…

Name	Description
Confidence	The confidence level for the spoken input that you type in the **Recognition** field. Valid value is any value between 0.0 and 1.0. Where the number 0.0 indicates minimum confidence and the number 1.0 indicates maximum confidence. **Note:** The **Confidence** field is unavailable if you enable the MRCP feature. If you enable the MRCP feature, you must provide spoken input by using a microphone. For more information, see Simulating spoken inputs with a microphone for a speech application on page 402 and Configuring MRCP settings on page 521.
Xfer Status (Transfer status)	Options to simulate the following possible transfer results: • **No Answer**: Simulates a scenario in which there is no answer from the destination number. • **Busy**: Simulates a scenario in which the destination number is busy. • **Disconnect**: Simulates a scenario in which the transferred call is disconnected on the far end, usually as a result of the destination party hanging up. • **Maxtime**: Simulates a scenario in which the transferred call reaches the maximum allowable time for a transferred call and is terminated by the system. **Note:** The **Xfer Status (Transfer status)** field is available only when AAS encounters a **Bridged Transfer** node while simulating the application.

Application Simulator view: Call:tel:<telephone number>:uid_<uid number> tab option descriptions

Use the **Call:tel:<*telephone number*>:uid_<*uid number*>** tab to simulate run-time event scenarios for a call control application.

The **Call:tel:<*telephone number*>:uid_<*uid number*>** tab is available only if you simulate a call control application.

Note:

Use the Console view to view the simulation event logs.

Name	Description
DTMF keypad	Number keys to provide DTMF input for simulation. **Note:** The DTMF keypad is available only when: • In the drop-down list that is located above the **Progress Info** field, you click **live_voice**, and then click **Answer**. • The **Waiting DTMF** option turns green.
Send Digits	Option to send the DTMF input that you specify by using the DTMF keypad, for simulation. **Note:** The **Send Digits** option is available only when: • In the drop-down list that is located above the **Progress Info** field, you click **live_voice**, and then click **Answer**. • The **Waiting DTMF** option turns green.
Call Active	Green indicates that the call is active. Red indicates that the call is processed, disconnected, or hung up.
Waiting ASR	Green indicates that AAS is waiting for an ASR input.
Waiting DTMF	Green indicates that AAS is waiting for a DTMF input.
Drop-down list that is located above **Send Call Progress**: Contains various call failure events for outbound calls.	
Call Fail Result	Option to simulate a scenario when the system is unable to connect the call.
Ring No Answer	Option to simulate a scenario when the dialed party does not answer the call.
Busy	Option to simulate a scenario when the dialed number is busy.
No Resource	Option to simulate a scenario when the system does not find the necessary resources to connect the call.
Network Busy	Option to simulate a scenario when the network is busy.
No Route	Option to simulate a scenario when the system is unable to route the call.

Table continues…

Name	Description
Unreachable	Option to simulate a scenario when the dialed number is unreachable.
Answer	Option to answer a call by using the various call classification scenarios. When you select a call classification scenario in the drop-down list that is located above the **Progress Info** field and click **Answer**, the system sends a connection.signal event followed by a connection.connected event to the application. The connection.signal event contains the event $.info.callprogress variable. The event $.info.callprogress variable contains the results of the call classification scenario that you select.
Drop-down list that is located above the **Progress Info** field: Contains various call classification scenarios that you can simulate after the call is connected.	
busy_tone	Option to simulate a scenario when the system receives a busy tone from the dialed number. This is commonly referred to as the "slow busy" tone. No further classification is sent for the call. **Note:** The **busy_tone** call classification applies to outbound calls only.
Reorder	Option to simulate a scenario when the system encounters a switch error, such as all circuits are busy. This is commonly referred to as the "fast busy" tone. No further classification is sent for the call. **Note:** The **reorder** call classification applies to outbound calls only.
sit_tone	Option to simulate a scenario when the system receives a special information tone from the dialed number which indicates that the call could not be completed. This is a three frequency tone that is often followed by a spoken message. No further classification is sent for the call. **Note:** The **sit_tone** call classification applies to outbound calls only.

Table continues…

Name	Description
live-voice	Option to answer a call with a spoken or DTMF input. **Note:** The **live-voice** call classification applies to outbound calls only.
recorded_msg	Option to simulate a scenario when the system detects an answering machine on the dialed number and a recorded message starts. The **recorded_msg** call classification is always followed by either a **msg_end** or a **timeout** call classification. **Note:** The **recorded_msg** call classification applies to outbound calls only.
msg_end	Option to simulate a scenario when the system detects the end of a recorded message such as that played by an answering machine. After the recorded message ends, the Orchestration Designer application plays a prompt depending on the call flow. No further classification is sent for the call. **Note:** The **msg_end** call classification applies to outbound calls only.
fax_answer_tone	Option to simulate a scenario when the system detects a fax machine on the dialed number. No further classification is sent for the call. **Note:** The **fax_answer_tone** call classification applies to outbound calls only.

Table continues…

Name	Description
timeout	Option to simulate a scenario when the call classification algorithm fails to classify the call before the timeout is reached. By default, the timeout value is 20 seconds or 20000 milliseconds. To override the default timeout value for outbound calls, open the call control project. In the call_classification_timeout parameter set the timeout value. The call_classification_timeout parameter is available in the **hints** attribute of the **<createcall>** tag. AAS stops the call classification analysis when the timeout is reached. No further classification is sent for the call. **Note:** The **timeout** call classification applies to inbound as well as outbound calls.
error	Option to simulate a scenario when an internal error occurs during the classification analysis and the system is unable to classify the call. No further classification is sent for the call. **Note:** The **error** call classification applies to inbound as well as outbound calls.
Progress Info	The call progress information that you want to simulate.
Send Call Progress	Option to send the call progress information that you type in the **Progress Info** field for simulation.

Application Simulator view: Messages tab field descriptions

The Messages tab shows the outbound messages and notifications that an application sends during application simulation.

Name	Description
Type	Icon representing the type of the channel that the system uses to send the message or notification.
From	The phone number of the sender of the SMS message or the email address of the sender of the email message as appropriate.
To	The phone number of the receiver of the SMS message or the email address of the receiver of the email message as appropriate.

Table continues…

Name	Description
Subject	The subject line of the email message.
Attachments	The URL of the attachment sent along with the outbound email message.
Message Details	The details of the message that you select in the left pane.

Responses from scripts

In Orchestration Designer, you can use scripts to automate certain kinds of inputs when testing applications by simulation. When using scripts to automate simulations, you can use scripts for two basic purposes:

- To automatically simulate caller responses. For more information, see Using scripts to simulate caller responses on page 587.
- To simulate IC and AES connector functions and responses. For more information, see Using scripts to simulate IC and AES connectors on page 592.

For more information about the creation and use of scripts for simulation, see Creating scripts for testing on page 587.

Other inputs

With Orchestration Designer, you can create speech applications or message applications that use a variety of inputs other than caller responses. Among these are inputs from other modules, database queries and other operations, and use of Web services.

Simulation of module inputs

When you use modules as part of a speech application project, you need to provide inputs to that module or receive outputs from that module. When a module expects to receive input from another application or module, the AAS indicates this by activating the **Input Parameters** display pane of the **Call** tab, when you select the module from the **Available Projects** list.

Simulation of database operation inputs

Orchestration Designer does not provide a simulation version of a database operation file that you can use during application simulation. Instead, you can use the actual run-time database operation file that uses the run-time database to simulate the application. If you do not want to use the actual run-time database, then you can create a pared-down development version of the actual run-time database. You can use the pared-down database in the database operation file, use the database operation file in the application, and then simulate the application.

Before deploying the application, you can change your application to use the run-time version of the database.

Simulation of Web service operation inputs

Orchestration Designer does not provide a simulation version of a Web service operation file to simulate an application. Instead, you can use the actual run-time Web service operation file that uses the run-time Web service to simulate the application. If you do not want to use the actual run-time Web service or if you do not want to use a Web service outside your development environment, then you can create a pared-down development version of the actual run-time Web service. You can use the pared-down Web service in the Web service operation file, use the Web service operation file in the application, and then simulate the application.

Before deploying the application, you can change your application to use the run-time version of the Web service.

Viewing the simulation report logs

About this task

Note:

If you are using an Avaya Experience Portal system, you can view an automatically generated Application Report of the log data. See “Application Report” in your Avaya Experience Portal documentation.

Tip:

The name of the log file for the current day is always `report.log`. Each time you run a simulation, Orchestration Designer appends the log data to the end of this file. After the end of each day on which data has been written to the log file, Orchestration Designer renames the log file to: `report.log`.*yyyy-mm-dd*, where *yyyy-mm-dd* represents the date on which the log file was compiled.

- Locate the appropriate log file in the <*ProjectName*>/data/log directory and double-click it.

Chapter 31: Application deployment

Process for deploying an Orchestration Designer application

Orchestration Designer speech, or call control, or message control, or data, or web applications can be deployed to an application server that the IVR system invokes, or in the case of speech, message, data, or web applications, as a reusable module to be used in another Orchestration Designer speech or message or web project.

Deploying an Orchestration Designer speech application to an IVR system, or call control application to a Avaya Experience Portal system is a two-stage process.

1. The first stage involves exporting the application and all its related files to a compressed package, similar to a ZIP file. Depending on where the application will be deployed, this package is a WAR (Web ARchive) file or an EAR (Enterprise ARchive) file. The WAR or EAR file is delivered to the application server where the deployed application will run.

   See Exporting an Orchestration Designer project on page 419.

   Note:

   Avaya recommends that you stop Tomcat before exporting an application. (If you do not, a message dialog box may appear, that can be ignored; it will not adversely affect the export.)

2. The second stage involves unpacking and installing the WAR or EAR file on the application server. When this is done and the IVR system is configured to run the application, the application is successfully deployed.

   See Prerequisite files on the application server on page 434.

Orchestration Designer also supports another type of deployment: speech, or message, or data, or web projects that can be used as reusable modules in other speech, or message, or data, or web applications.

See Deploying a speech, or a message flow, or a web project as an Orchestration Designer reusable module on page 431.

Run-time support file export

Run-time support files

Use the Export Application Runtime Support Files wizard to export the run-time support files and runtimeconfig files. Using this wizard, Orchestration Designer creates a `runtimeSupport<application server name>.zip` file that contains the JAR files and other configuration files required by the Orchestration Designer runtime and associated data connectors.

Use the Export Application Runtime Support Files wizard to export Web applications such as the icconnector and aesconnector to a user-defined location and also include additional libraries or files in the `runtimeSupport<application server name>.zip` file .

WebLogic does not work with some of the Axis2 run-time support files in the system classpath. Hence, if you export the run-time support files of a project for the WebLogic application server, Orchestration Designer creates a `runtimeSupportWeblogic.war` file in addition to the `runtimeSupportWeblogic.zip` file. Orchestration Designer packages all the Axis2 related run-time support files in the `runtimeSupportWeblogic.war` file. When you deploy the `runtimeSupportWeblogic.war` file to the WebLogic application server, the `runtimeSupportWeblogic.war` file is deployed as a shared library for the all the applications.

After you deploy the `runtimeSupportWeblogic.zip` file and before you deploy the `runtimeconfig.war` and other application files to the WebLogic application server, you must deploy the `runtimeSupportWeblogic.war` to the WebLogic application server.

Note:

- Web Services connector libraries (Axis 1.4) is not included in the `runtimeSupport<application server name>.zip` file by default.
- You must install the runtime support files on each application server that hosts Orchestration Designer applications.
- If you have added or modifies certificates in the `trusted_weblm_certs.jks` file, you must save copy of the existing `trusted_weblm_certs.jks` file before re-deploying run-time support files to an existing deployment or updating an existing deployment to a newer version. When you extract or unzip the run-time support files, the certificate file gets over written. You can then use the copy of the `trusted_weblm_certs.jks` file

The run-time support files need to be installed only once per application server and after an Orchestration Designer version update.

Use the Runtime Configuration application to configure various settings such as the license server URL and proxy settings. Orchestration Designer applications use these configurations at run time.

Exporting the run-time support files

Procedure

1. On the **File** menu, click **Export**.

   The system displays the Export wizard.

2. Double-click **Avaya OD Development**.
3. Click **Export Runtime Support Files**, and then click **Next**.

   The system displays the Export Application Runtime Support Files wizard.

4. On the Export Runtime Support page, specify the information to export the runtime support files, and then click **Next**.
5. On the Library Selection page, select any additional libraries or Web applications to include in the runtime support package.
6. Click **Finish**.

Result

Depending on the application server that you select in the **Application Server** field, the system creates a `runtimeSupport<application server name>.zip` file in the directory that you select in the **Destination Directory** field.

If you export the run-time support files of a project for the WebLogic application server, Orchestration Designer creates a `runtimeSupportWeblogic.war` file in addition to the `runtimeSupportWeblogic.zip` file. Orchestration Designer packages all the Axis2 related run-time support files in the `runtimeSupportWeblogic.war`.

Export Application Runtime Support Files wizard: Export Runtime Support page field descriptions

Name	Description
Runtime Support	
Application Server	The application server that is supported by the run-time files. Select one of the following: • **Apache Tomcat 7.0** • **Apache Tomcat 8.0** • **Apache Tomcat 9.0** • **IBM WebSphere** • **Oracle WebLogic** • **JBoss EAP/Wildfly**

Table continues…

Name	Description
Runtime Configuration	
Export Orchestration Designer runtime configuration application	Check box to create the run-time configuration application.
Destination Directory	A path where you want to save the run-time support files and runtimeconfig files.

Export Application Runtime Support Files wizard: Library Selection page field descriptions

Name	Description
Pluggable Connector Common Libraries	
Web Services (Axis 1.4)	Check box to include Web Services connector common libraries (Axis 1.4) in the runtimeSupport<`appserver`>.zip file. **Note:** The Web Services connector libraries (Axis 1.4) are not included in the runtimeSupport<`appserver`>.zip file by default. You may need these libraries to support legacy web services.
Pluggable Connector Web Applications	
Avaya AES Connector (aesconnector.war)	Check box to export the Application Enablement Services Connector (AESC) Web applications to the destination directory.
Avaya IC Connector (icconnector.war)	Check box to export the Avaya IC Connector Web applications to the destination directory.

Orchestration Designer project export

Exporting an Orchestration Designer project

About this task

To deploy an Orchestration Designer project, you must first export the Orchestration Designer project to a Web ARchive (WAR) file or an Enterprise ARchive (EAR) file.

Note:

- A WAR file or an EAR 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 support the WAR file format. The Tomcat servlet engine, the Oracle WebLogic servlet engine, and the JBoss EAP/Wildfly servlet engine are optimized to process WAR files. IBM WebSphere and WebSphere Express servlet engines use the EAR file format.
- Before you export an Orchestration Designer project, stop Tomcat.

Procedure

1. On the **File** menu, click **Export**.
2. In the Export wizard, double-click **Avaya OD Development**.
3. Perform one of the following actions:
   - To export an Orchestration Designer call control project, click **Export Orchestration Designer Call Control Project**.
   - To export an Orchestration Designer data project, click **Export Orchestration Designer Data Project**.
   - To export an Orchestration Designer message flow project, click **Export Orchestration Designer Message Flow Project**.
   - To export an Orchestration Designer speech project, click **Export Orchestration Designer Speech Project**.
   - To export an Orchestration Designer web project, click **Export Orchestration Designer Web Project**.
4. Click **Next**.
5. In the Export Orchestration Designer Project wizard, on the Specify Export Parameters page, specify the project that you want to export and the directory where you want to export the project, and then click **Next**.
6. On the Specify Platform Details page, specify the target platform information on which you want to deploy and run the exported application, and then click **Next**.
7. On the Specify Deployment Parameters page, specify the deployment parameters, and then click **Next**.

   Select the **Allow HTML Mode** check box to enable HTML mode in speech applications.

   Note:

   Orchestration Designer does not include validation pages in the exported WAR file by default. To include validation pages in the exported WAR file, select the **Include Validation JSP pages** check box.
8. On the Configure Web Application Descriptor page, configure the application descriptors for the target platform.
9. Click **Finish**.

Result

The system packages and exports the selected project to the designated destination based on the following rules:

- If the target destination is a Tomcat application server, WebLogic application server, or JBoss EAP/Wildfly application server, Orchestration Designer generates a .war file named <project name>.war, where <project name> is the name of the application project.
- If the target destination is an IBM WebSphere application server or WebSphere Express application server, Orchestration Designer generates an .ear file named <project name>.ear, where <project name> is the name of the application project.

Next steps

Before you deploy an Orchestration Designer project, you must install the project files on the destination application server.

Export Orchestration Designer Project wizard: Specify Export Parameters page field descriptions

Name	Description
Available Projects	The Orchestration Designer project that you want to export. **Note:** You can export only one project at a time. Hence, if a speech project, or a message flow project, or a web flow project uses reusable modules, then you must export each reusable module separately.
Destination Directory	The directory path where you want to export the project.

Export Orchestration Designer Project wizard: Specify Platform Details page field descriptions

On the Specify Platform Details page, optionally configure the target platform settings for the project that you want to export.

Name	Description
Target Platform	

Table continues…

Name	Description
Platform	The run-time platform on which you want to deploy and run the Orchestration Designer application after you export the application project. The following are the options: • **Experience Portal**: Optimizes the application to run on an Avaya Experience Portal system. • **IR**: Optimizes the application to run on an Avaya IR system. **Note:** The **IR** option is available only for speech projects. • **Desktop**: Optimizes the application to run on one of the supported operating system desktops. The **Desktop** option is intended primarily for development. **Note:** The default is **Desktop**. • **Other**: Optimizes the speech application or the call control application to run on another VoiceXML 2.0-compliant IVR system. **Note:** The **Other** option is available only for call control projects and speech projects. • **MPS**: Optimizes the application to run on an MPS system. When Orchestration Designer encounters an MMF file, it generates a VXML based on the phrase definition. **Note:** If you set **Platform** to **MPS**, you must set **DTMF Compatibility** to **native**. The **MPS** option is available only for speech applications.

Table continues…

Name	Description
Servlet container	Servlet container options for the IVR system or the text processing system. The following are the options: • **Apache Tomcat 7.0**: The IVR system or the text processing system uses Apache Tomcat 7.0 application server software. • **Apache Tomcat 8.0**: The IVR system or the text processing system uses Apache Tomcat 8.0 application server software. • **Apache Tomcat 9.0**: The IVR system or the text processing system uses Apache Tomcat 9.0 application server software. • **IBM WebSphere**: The IVR system or the text processing system uses IBM WebSphere or WebSphere Express application server software. • **Oracle WebLogic**: The IVR system or the text processing system uses Oracle WebLogic application server software. • **JBoss EAP/Wildfly**: The IVR system or the text processing system uses JBoss EAP/Wildfly application server software.
Speech **Note:** The following options are available only when you export a speech project.	

Table continues…

Name	Description
Grammar compatibility	Type of the ASR server engine to target the ASR grammars. The following are the options: • **Desktop Microsoft Speech SAPI**: Grammars are optimized to work with Microsoft Speech API (SAPI). • **Google Speech**: Grammars are optimized to work with Google ASR. If static grammars are used, then you must fill the content of the grammar files with the suffix “google” in the **data/<language>grammars** folder. • **IBM**: Grammars are optimized to work with IBM WebSphere or WebSphere Express ASR engines. • **Loquendo**: Grammars are optimized to work with Loquendo ASR engine. • **Nuance 10.0 and above**: Grammars are optimized to work with Nuance 10.0 and above. • **Nuance 9.0 (Quantum)**: Grammars are optimized to work with Nuance 9.0. • **Nuance OSR**: Grammars are optimized to work with Nuance OpenSpeech Recognition products including Quantum. • **SRGS-Literals**: Grammars conform to the SRGS version 1.0 standard. **Note:** The default is **SRGS-Literals**. • **SRGS-SISR**: Grammars are generated in the SRGS format with SISR as per the standards and do not take into account any vendor specific nuances. By default, Orchestration Designer creates grammars to support all ASR engines. The grammars that Orchestration Designer uses at run time are based on the type of the ASR server engine that you select in the **Grammar compatibility** field, and ultimately the value of the runtime-asr variable in the web.xml file.

Table continues…

Name	Description
Grammar Caching	The grammar caching option to use. The following are the options: • **none**: Grammar caching is not permitted for the application that you export. • **default**: The application that you export uses the default setting of the IVR system for grammar caching.
DTMF Compatibility	The DTMF grammar option to use. The following are the options: • **local SRGS-SISR**: If you select this option, grammars are generated with SISR tags and are processed locally by the voice browser. Use this option for Avaya Voice Portal 5.0 and later. • **local SRGS-Literals**: Grammars are generated in SRGS literal format and are processed by the voice browser. Use this option for Desktop, IR, and Avaya Voice Portal 4.1 and earlier. **Note:** The default is **local SRGS-Literals**. • **native**: Generates the DTMF grammars in the native format of the ASR engine that you specify in the **Grammar Compatibility** field. If you select **native**, then on the Avaya Voice Portal platform or the Avaya Experience Portal platform, under advanced options, you must enable the application for remote DTMF processing. The ASR engine then handles the DTMF grammar processing. **Note:** Select **native** only if you select the platform as **Voice Portal**, **Experience Portal**, or **MPS** in the **Platform** field. If you select the platform as **MPS** in the **Platform** field, you must set **DTMF Compatibility** to **native**.

Table continues…

Name	Description
Enable Speech Synthesis Markup Language (SSML) generation in project prompts	Check box to enable SSML markers in the prompts. If you select this check box, when the voice browser generates the TTS content, it also generates the SSML data. **Note:** If you select this check box, Avaya Application Simulator (AAS) generates the SSML data, even in the simulation mode. The Microsoft TTS engine that Orchestration Designer uses, however, is not capable of generating or using SSML data. Hence, you cannot test SSML data during simulation.

Export Orchestration Designer Project wizard: Specify Deployment Parameters page field descriptions

Name	Description
Actions	
Regenerate application	Check box to regenerate the project that you want to export as part of the export process packaging. The **Regenerate application** check box is not selected by default. However, if you retarget grammars, for example, if you change the grammar compatibility, the **Regenerate application** check box is automatically selected so that the grammars are generated properly within the application.
Rebuild application	Check box to rebuild the project that you want to export as part of the export process packaging. The **Rebuild application** check box is not selected by default. However, if you retarget grammars, for example, if you change the grammar compatibility, the **Rebuild application** check box is automatically selected so that the grammars are built properly within the application.
Packaging	

Table continues…

Name	Description
Deploy project class files as • **JAR file** • **class files**	Options to determine how Orchestration Designer should deploy the class files for the project. The following are the options: • **JAR file**: Packages the .class files of the project into a single, self-contained Java ARchive (JAR) file. **Note:** The default is **JAR file**. • **class files**: Deploys the .class files of the project as separate files within the WAR file. For run time, it does not make any difference which option you select. The **JAR file** option, however, has the following advantages: • JAR files keep the run-time environment cleaner. • A single JAR file is easier to replace than to replace a whole batch of .class files.
Include source code	Check box to include the source code of the project in the .war or .ear file. Including the source code in the .war or .ear file helps you to study, use, modify, and debug the source code.
Include project meta files	Check box to include the project meta files in the .war or .ear file. Project meta files are files such as .prompt, .gram, and .flow files. Including the project meta files in the .war or .ear file makes it easier to check in the files into a source control system, email the files, or deploy the project with all files.
Include Validation JSP pages	Check box to include the validation JSP and other supporting JSP pages. These JSP pages can be loaded in the web browser by the application administrator to verify if the application has been correctly deployed. The JSP pages are considered while deploying tools and can be leveraged by potential security threats. Therefore, remove the JSP pages after the deployment is validated. To remove the JSP pages, delete the jsp folder from the application directory.

Table continues…

Name	Description
Allow HTML Mode	Check box to allow the application to be launched and viewed in the web browser that has access to the application server. It is a convenient way to execute and verify the application flow and logic without making a call to the platform. The html mode is designed for validating the application, and can be leveraged by potential security threats. Therefore, disable the html mode after the application is validated. To disable the html mode, go to the web.xml deployment descriptor, and set the parameter `htmlmode` to false.
Include extra files and folders	Check box to include the extra files and folders in the .war or .ear file. The system adds the files and maintains the folder structure in the exported .war or .ear file. The system copies the resources into the root of the .war or .ear file and maintains the directory structure. **Note:** Be sure that you add the files and folders to the `//data/` directory. The system then automatically includes the files and folders in the packaged .war or .ear file. The `//data/` directory is available within a project directory in the Package Explorer view in the Java perspective.
Exclude JDBC resources from the context file	Check box to exclude the JDBC resources from the context file when exporting the application project to a .war or an .ear file. Select this check box when a global database definition is defined in the Tomcat configuration files that the Orchestration Designer application uses. **Note:** The **Exclude JDBC resources from the context file** check box is available only after you select **Apache Tomcat 7.0**, **Apache Tomcat 8.0**, or **Apache Tomcat 9.0** in the **Servlet container** field on the Specify Platform Details page of the Export Orchestration Designer Project wizard.

Table continues…

Name	Description
Exclude languages	Check box to exclude language files that are used in the speech, message flow, or web project from the .war or .ear file. **Note:** The **Exclude languages** check box is unavailable if the speech, message flow, or web project uses only one language. The **Exclude languages** check box is unavailable if you export a call control project.
Exclude phrase audio files	Check box to exclude the audio files that are used in the speech project from the .war or .ear file. You can select this check box to deploy a speech project update without updating the audio files on the server. Thereby, generating a smaller .war or .ear file. **Note:** The **Exclude phrase audio files** check box is available only if you export a speech project.
Reusable Modules **Note:** This setting is available only if you export a speech project or a message flow or a web project.	
Deploy project to the Orchestration Designer reusable modules directory	Check box to deploy the speech, message flow, or web project as a reusable module. If you select the **Deploy project to the Orchestration Designer reusable modules directory** check box, the system deploys the .war or the .ear file to the reusable modules directory. For more information about this option, see Deploying a speech, or a message flow, or a web project as an Orchestration Designer reusable module on page 431.
Tracing **Note:** This setting is available only if you export a speech, or a message flow, or a web project.	

Table continues…

Name	Description
Reset tracing in the deployed application to the default (disabled) settings	By default, tracing is disabled in a deployed application. Clear this check box to enable tracing and to enable the localapptrace parameter in the ddrt.properties file. **Note:** If you deploy an application as a reusable module and enable tracing, the system logs the trace and report information of the module in the log file of the parent application of the module. The system does not log the trace and report information of the module in the log file of the module.
Build File	
Save build file after exporting the project	Select the **Save build file after exporting the project** check box to save the build file when you are exporting an Orchestration Designer application. When you select this check box, the ant build file gets saved as `build.xml` in your project directory. Also, a `buildFiles` folder that stores the files to package with the `war` or `ear` is created. If you export the project from Orchestration Designer next time without selecting the **Save build file after exporting the project** check box, then the `build.xml` file and the `buildFiles` folder is deleted.

Export Orchestration Designer Project wizard: Configure Web Application Descriptor page

You can use the Configure Web Application Descriptor page of the Export Orchestration Designer Project wizard to review and edit all previously defined export settings.

The following are the two tabs:

- Application tab: Provides a review and editable interface for the application settings that affect the way the project is exported. You can modify most of the settings that are displayed in the Parameters area in one of the previous pages of the Export Orchestration Designer Project wizard. You can also modify these settings directly on the Application tab of the Configure Web Application Descriptor page.

  For example, the runtime-Platform parameter takes the value of the platform that you specify in the **Platform** field on the Specify Platform Details page of the Export Orchestration Designer Project wizard. The settings and options that you can modify on the Application tab of the Configure Web Application Descriptor page are the same as those on the Web

Descriptor tab of the Properties for <project name> dialog box. For more information, see Application tab field descriptions on page 511.

- Servlets tab: Provides a review and editable interface for the various parameters and settings of the deployment Java servlets. In almost all cases, changes to these settings are not needed. The settings and options that you can modify on the Servlets tab of the Configure Web Application Descriptor page are the same as those on the Web Descriptor tab of the Properties for <project name> dialog box. For more information, see Servlets tab on page 512.

  Caution:

  Do not modify these settings. Modifying these settings can adversely affect your application.

Orchestration Designer reusable module

Note:

This section applies only to speech applications and message applications.

When creating a large and complex speech application or message application, it is often a good idea to identify parts of the application that can be reused by other parts of the application. You can create reusable parts as modules.

For example, in a speech application for banking, a module collects the account number of a caller and then writes the account number to a variable. You can use this module in many different parts of the main application, including the parts devoted to checking accounts, savings accounts, mortgage accounts, and so on.

You can also dynamically switch between different versions of the same module without redeploying the application. This ensures uninterrupted service on the application server. For more information, see Deploying a new version of a module in an application on page 461.

Deploying a speech, or a message flow, or a web project as an Orchestration Designer reusable module

About this task

Note:

You can deploy only speech, or message flow, or web projects as Orchestration Designer reusable modules.

Procedure

1. On the **File** menu, click **Export**.
2. In the Export wizard, double-click **Avaya OD Development**.

3. Perform one of the following actions:
   - To deploy an Orchestration Designer speech project as a reusable module, click **Export Orchestration Designer Speech Project**.
   - To deploy an Orchestration Designer message flow project as a reusable module, click **Export Orchestration Designer Message Flow Project**.
   - To deploy an Orchestration Designer web flow project as a reusable module, click **Export Orchestration Designer Web Project**.
4. Click **Next**.

   The system displays the Export Orchestration Designer Project wizard.
5. Follow the pages on the wizard and configure the settings.

   For more information, see Exporting an Orchestration Designer project on page 419.
6. On the Specify Deployment Parameters page, select the **Deploy project to the Orchestration Designer reusable modules directory** check box.

   For more information, see Export Orchestration Designer Project wizard: Specify Deployment Parameters page field descriptions on page 426.

   Note:

   Orchestration Designer displays a warning with notification that after the module is deployed to the reusable module directory, there will be both the reusable module *and* the project, each with the same name in the project workspace. To avoid potential conflicts between the project and the reusable module, close the project and move the project out of the workspace. You can move the project to the workspace when you want to modify and redeploy the project.

   Even if you retain the speech, or the message flow, or the web project in the workspace, the reusable module that is available in the Palette pane of the call flow editor, or the message flow editor, or the web flow editor and the speech project, or the message flow project, or the web project that is available in the Avaya OD Navigator view are different.
7. In the Deploy Project message box, click **Yes**.
8. Follow the pages on the wizard and configure the settings.

   For more information, see Exporting an Orchestration Designer project on page 419.
9. Click **Finish**.

Result

Orchestration Designer deploys a copy of the speech, or the message flow, or the web project to the reusable modules directory.

Next steps

You can import a reusable module only in an Orchestration Designer project that is created for a channel that is same as that of the reusable module. For example, you can import a reusable speech application module only in a speech project.

When you import the reusable module in an Orchestration Designer project, the system adds a module item for the new module in the Palette pane of the call flow editor or the message flow editor or the web flow editor.

Note:

The Export wizard also creates a copy of the WAR file at the location that you specify as the destination directory on the Export Orchestration Designer Project wizard. You must deploy this WAR file to the application server for use with the parent application. If you do not deploy this WAR file to the application server along with any other application project WAR files, the parent application will not be able to find or use it.

Note:

If you store custom object (objects that you define) in session properties and want to pass them between two modules (Web applications), the code for the objects must be in `<tomcat_home>/common/lib` so that the class loader works properly. If that code is on `<appname>/WEB-INF/lib`, then an object instantiated in project A cannot be accessed on project B. By moving the implementation of the class to the common lib and in turn the common class loader, the implementation can be shared across Web applications.

Exporting multiple applications into a single EAR file

About this task

To deploy an Orchestration Designer project for Websphere application server, you must first export the Orchestration Designer project to a Enterprise Archive (EAR) file.

Note:

- An EAR file is a compressed set of files, similar to a ZIP file. IBM WebSphere and WebSphere Express Administrative Console deploy applications packaged in EAR file format.
- Before you export an Orchestration Designer project, stop Tomcat.

Procedure

1. Select a primary Orchestration Designer project that uses or depends on other Orchestration Designer applications as modules.

   Ensure that the module applications are available in the same workspace.

2. On the **File** menu, click **Export**.
3. In the Export wizard, double-click **Avaya OD Development**.
4. Perform one of the following actions:
   - To export an Orchestration Designer call control project, click **Export Orchestration Designer Call Control Project**.
   - To export an Orchestration Designer message flow project, click **Export Orchestration Designer Message Flow Project**.

- To export an Orchestration Designer speech project, click **Export Orchestration Designer Speech Project**.

5. Click **Next**.
6. In the Export Orchestration Designer Project wizard, on the Specify Export Parameters page, specify the project that you want to export and the directory where you want to export the project, and then click **Next**.
7. On the Specify Platform Details page, specify **IBM Websphere**, and click **Next**.
8. On the Specify Deployment Parameters page, specify the deployment parameters, and then click **Next**.
9. On the Configure Web Application Descriptor page, configure the application descriptors for the target platform, and click **Next**.

   The Configure Deployment Package page, displays a list of applications including the primary and module applications.
10. **(Optional)** To add other modules in the EAR package:
    a. On the Configure Deployment Package page, click **Add**.
    b. In the Select a Project dialog box, select a module application and click **OK**.
    c. Repeat step 10 (b) for the number of applications you want to add as modules.

    Note:

    Ensure that you do not add an application that is already deployed on the application server. WebSphere does not support more than one instances of the same application installed, either in an EAR file or with other application in another EAR file. If any of the application is already deployed, you must remove the application from the Configure Deployment Package page during the export process.
11. Click **Finish**.

Result

The system packages and exports the selected projects to the designated destination based on the following rule:

- If the target destination is an IBM WebSphere application server or WebSphere Express application server, Orchestration Designer generates an `.EAR` file named `<project name>.ear`, where *<project name>* is the name of the primary project.

Prerequisite files on the application server

Before you can deploy the Orchestration Designer speech application on the application server, you must export the application project.

Application server requirements

Project files must be installed on a server that meets the following requirements:

- Operating system requirements on page 435
- Software requirements on page 435

Operating system requirements

Orchestration Designer applications can run within the following operating systems:

- Windows Server 2008 R2 32/64 bit
- Windows Server 2012
- Windows Server 2016 64 bit
- Windows 7
- Windows 10
- RedHat Linux ES 6.x
- RedHat Linux ES 7.x 64 bit

Software requirements

Orchestration Designer applications must run with one of the following servlet containers:

- Apache Tomcat 7.x, Tomcat 8.0.x, or Tomcat 9.0.x
- IBM WebSphere Application Server 8.5 or 9
- Oracle WebLogic 10.3.6, 11g, or 12c
- JBoss EAP/Wildfly

Note:

If you are using Tomcat for your deployment server, the Tomcat version for your development environment must match the Tomcat version on your deployment application server.

Configuring the application server to use a proxy server

Procedure

1. Perform one of the following actions to configure the proxy:
   - If you are using an HTTP proxy server, set up the **http.proxyHost**, **http.proxyPort**, and **http.nonProxyHosts** for the Java Virtual Machine.
   - If you are using an HTTPS proxy server, set up the **https.proxyHost**, **https.proxyPort**, and **https.nonProxyHosts** for the Java Virtual Machine.
2. Open the Orchestration Designer Admin Console and configure the HTTP and HTTPS proxy parameters for the application server. For more information, see Orchestration

Designer Admin (ddadmin) Web application configuration on page 441 and Configuring connectivity settings on page 442.

Deploying the run-time support files

About this task

For each application server where Orchestration Designer applications are run, run-time support files must be deployed.

WebLogic does not work with some of the Axis2 run-time support files in the system classpath. Hence, if you export the run-time support files of a project for the WebLogic application server, Orchestration Designer creates a `runtimeSupportWeblogic.war` file in addition to the `runtimeSupportWeblogic.zip` file. Orchestration Designer packages all the Axis2 related run-time support files in the `runtimeSupportWeblogic.war` file. When you deploy the `runtimeSupportWeblogic.war` file to the WebLogic application server, the `runtimeSupportWeblogic.war` file is deployed as a shared library for the all the applications.

After you deploy the `runtimeSupportWeblogic.zip` file and before you deploy the `runtimeconfig.war` and other application files, you must deploy the `runtimeSupportWeblogic.war` file to the WebLogic application server.

For more information, see Deploying the runtimeSupportWeblogic.war file to the WebLogic application server on page 440.

Warning:

If you are updating a previous deployment of the run-time support files on an existing application server, you must delete all previous instances of the run-time support files before running Orchestration Designer applications against the new run-time support files.

Procedure

1. Generate the run-time support files using the Export Runtime Support Files wizard.

   The system exports the run-time support files and the runtimeconfig files to the `runtimeSupport<application server name>.zip` file.

   For more details, see Exporting the run-time support files on page 418.

2. Locate the appropriate run-time support file for your application server.

   The following is the list of the run-time support files:

   - `runtimeSupportTomcat7.zip`
   - `runtimeSupportTomcat8.zip`
   - `runtimeSupportTomcat9.zip`
   - `runtimeSupportWebsphere.zip`
   - `runtimeSupportWeblogic.zip` and `runtimeSupportWeblogic.war`
   - `runtimeSupportJBossEAP.zip` or `runtimeSupportJBossWildfly.zip`

3. Extract or unzip the run-time support files to one of the following locations:
   - For an IBM WebSphere or WebSphere Express run-time environment:

     `WebSphereHome\AppServer\`

     where *WebSphereHome* is the directory in which you installed the IBM WebSphere servlet engine software.
   - For an Apache Tomcat run-time environment:
     - Tomcat 7: Extract the `runtimeSupportTomcat7.zip` files to the Tomcat root directory `TomcatHome\`.
     - Tomcat 8: Extract the `runtimeSupportTomcat8.zip` files to the Tomcat root directory `TomcatHome\`.
     - Tomcat 9: Extract the `runtimeSupportTomcat9.zip` files to the Tomcat root directory `TomcatHome\`.

     where *TomcatHome* is the directory in which you installed the Tomcat servlet engine software.
   - For a WebLogic run-time environment:

     `bea\user_projects\domains\domainname\`
   - For a JBoss EAP/Wildfly run-time environment:

     `JBossHome\modules\com\avaya\od\main`

     where *JBossHome* is the directory in which you installed the JBoss EAP/Wildfly servlet engine software.

List of files included in the run-time support files

Filename	Description
scertcommon-versionNumber.jar	The shared Orchestration Designer run-time classes. **versionNumber** represents the version of Orchestration Designer.
VPAppLogClient.log4j.properties	The logging configuration file for the Avaya Experience Portal logging client.
VPAppLogClient_<versionNumber>.jar	Used for writing logs to Avaya Experience Portal. **versionNumber** represents the version of Avaya Experience Portal.
VPAppLogClientWS_<versionNumber>.jar	Used by VPAppLogClient to interact with the Axis2 library.
weblm.jar	The client library for license management.
trustedcert.properties	The properties file used by weblm.jar for locating and accessing the trust store (trusted_weblm_certs.jks), which contains the WebLM certificate.

Table continues…

Filename	Description
trusted_weblm_certs.jks	The trust store that contains certificates for HTTPS communications with WebLM. Can also contain other user or application-added certificates.
tsapi.pro	Used for configuring the AES component.
log4j-1.2.15.jar	The classes for log4j logging activities.
log4j.properties	Used for configuring the logs.
activation-1.1.jar	Used by Axis2 and REST for Web service connectivity.
axiom-api-1.2.13.jar	Used by Axis2 and REST for Web service connectivity.
axiom-dom-1.2.13.jar	Used by Axis2 and REST for Web service connectivity.
axiom-impl-1.2.13.jar	Used by Axis2 and REST for Web service connectivity.
axis2-adb-1.6.2.jar	Used by Axis2 and REST for Web service connectivity.
axis2-adb-codegen-1.6.2.jar	Used by Axis2 for Web service connectivity.
axis2-codegen-1.6.2.jar	Used by Axis2 for Web service connectivity.
axis2-json-1.6.2.jar	Used by REST for Web service connectivity.
axis2-kernel-1.6.2.jar	Used by Axis2 and REST for Web service connectivity.
axis2-saaj-1.6.2.jar	Used by Axis2 for Web service connectivity.
axis2-transport-http-1.6.2.jar	Used by Axis2 for Web service connectivity.
axis2-transport-local-1.6.2.jar	Used by Axis2 for Web service connectivity.
commons-codec-1.3.jar	Used by Axis2 for Web service connectivity.
commons-httpclient-3.1.jar	Used by Axis2 for Web service connectivity.
commons-logging-1.1.1.jar	Used by Axis2 for Web service connectivity.
geronimo-stax-api_1.0_spec-1.0.1.jar	Used by Axis2 and REST for Web service connectivity.
httpcore-4.0.jar	Used by Axis2 for Web service connectivity.
jettison-1.3.1.jar	Used by REST for Web service connectivity.
json.jar	Used by REST for Web service connectivity.
mail-1.4.jar	Used by Axis2 for Web service connectivity.
neethi-3.0.2.jar	Used by Axis2 and REST for Web service connectivity.
woden-api-1.0M9.jar	Used by Axis2 for Web service connectivity.
woden-impl-dom-1.0M9.jar	Used by Axis2 for Web service connectivity.
wsdl4j-1.6.2.jar	Used by Axis2 for Web service connectivity.
wss4j-1.5.8.jar	Used by Axis2 and REST for Web service connectivity.
wstx-asl-3.2.9.jar	Used by Axis2 for Web service connectivity.
XmlSchema-1.4.7.jar	Used by Axis2 for Web service connectivity.
xmlsec-1.4.3.jar	Used by Axis2 for Web service connectivity.
xalan-2.7.0.jar	Used by Axis2 for Web service connectivity.

Note:

If you use Orchestration Designer 7.0 with Voice Portal 5.0 or earlier and you want to use the Application Logging feature in Orchestration Designer 7.0, or if the Orchestration Designer applications are using the old Web Service connector, you must install the Axis 1.4 libraries.

The additional libraries are:

Filename	Description
axis.jar	Used by Axis for Web service connectivity.
commons-discovery.jar	Used by Axis for Web service connectivity.
jaxrpc.jar	Used by Axis for Web service connectivity.
opensaml-1.1.jar	Used by Axis for Web service connectivity.
saaj.jar	Used by Axis for Web service connectivity.

Performing additional WebSphere configuration

About this task

For WebSphere 7.0 and later, the Axis2 library files provided by Orchestration Designer conflicts with the Axis2 library files from WebSphere installation. Therefore, you must configure a separate class loader for Orchestration Designer applications to use Axis2 library effectively.

Perform the following steps for the additional WebSphere configuration:

Procedure

1. Open the WebSphere Administrative Console.
2. On the **Environment** menu, click **Shared libraries** and create a shared library.
3. Copy the following entries into the **Classpath** field:

```
${WAS_INSTALL_ROOT}/lib/ext/axiom-api-1.2.13.jar
${WAS_INSTALL_ROOT}/lib/ext/axiom-dom-1.2.13.jar
${WAS_INSTALL_ROOT}/lib/ext/axiom-impl-1.2.13.jar
${WAS_INSTALL_ROOT}/lib/ext/axis2-adb-1.6.2.jar
${WAS_INSTALL_ROOT}/lib/ext/axis2-adb-codegen-1.6.2.jar
${WAS_INSTALL_ROOT}/lib/ext/axis2-json-1.6.2.jar
${WAS_INSTALL_ROOT}/lib/ext/axis2-kernel-1.6.2.jar
${WAS_INSTALL_ROOT}/lib/ext/axis2-saaj-1.6.2.jar
${WAS_INSTALL_ROOT}/lib/ext/axis2-transport-http-1.6.2.jar
${WAS_INSTALL_ROOT}/lib/ext/axis2-transport-local-1.6.2.jar
${WAS_INSTALL_ROOT}/lib/ext/neethi-3.0.2.jar
${WAS_INSTALL_ROOT}/lib/ext/woden-api-1.0M9.jar
${WAS_INSTALL_ROOT}/lib/ext/woden-impl-dom-1.0M9.jar
${WAS_INSTALL_ROOT}/lib/ext/woden-tool-1.0M9.jar
${WAS_INSTALL_ROOT}/lib/ext/XmlSchema-1.4.7.jar
${WAS_INSTALL_ROOT}/lib/ext/VPWebServiceClient-<version>.jar
${WAS_INSTALL_ROOT}/lib/ext/VPAppLogClientWS_<version>.jar
```

   You must modify the *version* for `VPWebServiceClient -<version>.jar` according to the file deployed in the `lib` or `ext` directory.

4. Click **Servers** > **Server Types** > **WebSphere application servers** to navigate to the server configuration page.

5. From the server list, select the server that you want to associate with the shared library.
6. Click **Java and Process Management** in **Server Infrastructure** section, and click the **Class Loader** link.
7. Create a new class loader, and set the class loader order to **Classes loaded with local class loader first (parent last)**. Click **OK**.
8. Select the newly create class loader.
9. On the Configuration page, click the **Shared library references** link.
10. On the Configuration page, ensure that the shared library you created is selected.
11. Click **OK** to save the configuration.
12. Make sure you verify all the configuration has been saved, and restart the application server.

Deploying the runtimeSupportWeblogic.war file to the WebLogic application server

About this task

WebLogic does not work with some of the Axis2 run-time support files in the system classpath. Hence, if you export the run-time support files of a project for the WebLogic application server, Orchestration Designer creates a `runtimeSupportWeblogic.war` file in addition to the `runtimeSupportWeblogic.zip` file. Orchestration Designer packages all the Axis2 related run-time support files in the `runtimeSupportWeblogic.war` file. When you deploy the `runtimeSupportWeblogic.war` file to the WebLogic application server, the `runtimeSupportWeblogic.war` file is deployed as a shared library for the all the applications.

After you deploy the `runtimeSupportWeblogic.zip` file and before you deploy the `runtimeconfig.war` and other application files to the WebLogic application server, you must deploy the `runtimeSupportWeblogic.war` file to the WebLogic application server.

Procedure

1. Copy the `runtimeSupportWeblogic.war` file to a temporary directory on the WebLogic machine, separate from the actual application directory where all other applications are located.
2. On the left hand navigation menu, select **Deployments**.
3. Click **Install**.
4. Navigate to the `runtimeSupportWeblogic.war` file.
5. Accept the default settings and move to the next screen.
6. Click **Finish**.

Prerequisite configurations for running Orchestration Designer applications on an application server

Before an Orchestration Designer application can be run on an application server, the application server where the Orchestration Designer application will be run must be prepared.

Orchestration Designer Admin (ddadmin) Web application configuration

In previous releases of Orchestration Designer, application server configuration items were configured in an application's `web.xml` file. With Orchestration Designer 4.0 and above, common application server configuration are centrally configured within a Web-based configuration module —the Orchestration Designer Admin Console.

Note:

It is advisable not to keep Java security-manager active when DDAdmin is running.

After configuring DDAdmin and restarting Tomcat, Orchestration Designer loads the file `ddconfig.xml` based on the classpath as a java resource. If Orchestration Designer fails to do so, it looks for the `ddconfig.xml` file in the same directory as the `log4j.properties` file. Incase that also fails, Orchestration Designer creates the `ddconfig.xml` file in the directory where the class "com.avaya.sce.runtimecommon.SCESession" is present.

Accessing the Orchestration Designer Admin Console

About this task

In previous releases of Orchestration Designer, application server configuration items were configured in an application's `web.xml` file. With Orchestration Designer 4.0 and above, common application server configuration are centrally configured with a Web-based configuration module—the Orchestration Designer Admin Console.

The Orchestration Designer Admin Console is often referred to as **ddadmin**.

Procedure

1. With Eclipse open, start Tomcat.
2. Within a Web Browser, access the Orchestration Designer Admin Console at the following URL, typically: `http://localhost:8080/runtimeconfig`

   When first accessing this URL, the Login page is displayed.
3. Enter the default username and password pair: `ddadmin/ddadmin`.

   When you log in to the Admin console for the first time, Orchestration Designer forces you to change the default "ddadmin" password. Also, Orchestration Designer enforces to provide stronger passwords.

Note:

If the Login page does not display, or the default username/password does not work, contact Avaya Orchestration Designer support.

You must configure the **Preference** settings of Orchestration Designer for the application server that hosts your deployed applications.

Configuring license server

About this task

A valid license is required to run Orchestration Designer applications on Avaya IR, Avaya Voice Portal, Avaya Experience Portal, Avaya MPS, and other Voice XML platforms.

Tip:

Avaya recommends that you install the Orchestration Designer license on the existing WebLM license server that is installed with Voice Portal, Avaya Experience Portal, Avaya IR, or other Avaya products that use the WebLM license server.

If you run the Orchestration Designer applications on MPS, then you must install the WebLM license server and configure the license information of Orchestration Designer. You must install a separate WebLM license server because the WebLM license server is not installed during the MPS installation.

For more information about installing and configuring the WebLM license server, see the *Getting Started with Orchestration Designer* guide.

Procedure

1. Open the Orchestration Designer Admin Console.

   For more information, see Accessing the Orchestration Designer Admin Console on page 441.

2. In the left pane, click **License Server**.
3. In the **License URL** field, type the URL of the WebLM license server host.

   The same URL is used for all applications on the application server. For example, http://myhost:8080/

4. In the **License Check Timeout** field, type the time in seconds that the system must wait for a response from the WebLM license server while attempting to connect to the WebLM license server.

   The default value is zero seconds. Zero indicates that there is no timeout.

Configuring connectivity settings

About this task

Connectivity settings are required to be configured for the application server. Set both the non-secure and secure proxy HTTP settings, as well as non-proxy hosts.

For specific information related to IBM Websphere, Oracle WebLogic, and JBoss EAP/Wildfly proxy server settings, see the Orchestration Designer release notes.

Procedure

1. Open the Orchestration Designer Admin Console.

   For more information, see Accessing the Orchestration Designer Admin Console on page 441.

2. In the left pane, click **Connectivity Settings**.
3. On the Connectivity Settings page, configure the proxy settings. .

   For more information, see Orchestration Designer Admin Console: Connectivity Settings field descriptions on page 443

Orchestration Designer Admin Console: Connectivity Settings field descriptions

Field	Description
Use Proxy	Flag to indicate that the non-secure proxy settings should be used.
Proxy Host	Full HTTP path to, or URL of, the proxy server.
Proxy Port	Port that Orchestration Designer can use to access the proxy server.
Non Proxy Hosts	Names of hosts that do not need to be accessed through the proxy.
Use Http for Https	Option to quickly populate the “secure” proxy settings with same values indicated in the non-secure proxy settings.
Secure Use Proxy	Flag to indicate that the secure proxy settings should be used.
Secure Proxy Host	Full HTTPS path to, or URL of, the proxy server.
Secure Proxy Port	Port that Orchestration Designer can use to access the HTTPS proxy server.
Secure Non Proxy Hosts	Names of secure hosts that do not need to be accessed through the proxy.
Secure Fetch Port	Set to indicate the platform to use this port instead of the default 443 for https requests to fetch VXML and dependent resources from the application server.

Certificate management in a run-time environment

Orchestration Designer(OD) enables you to configure and manage certificates in a run-time environment. You must install the appropriate certificates to simplify the use of:

- Web services over HTTPS
- Mutual authentication for CCXML httpio applications
- Other SSL applications

Orchestration Designer Security Certificate Validity and Expiration

Orchestration Designer runtime checks once everyday for certificates that are nearing expiry, or that have expired, and issues an alarm accordingly. In the OD `runtimeconfig`, the user has an option to set the **Number of days to issue warnings and alarms before certificate expires**. You can set the **Number of days to issue warnings and alarms before certificate expires** to a value between 30 and 60 days. The default value is 60.

The `runtimeconfig` displays a warning in red if any certificates are nearing expiration. The default certificate expiration period for self-signed certificates is 1186 days or approximately 39 months.

Orchestration Designer Security Domain Validation

You can enable or disable extended hostname validation in certificates. This validation will be performed for https with mutual authentication. The validation is performed to verify whether the certificate has the name of the host you are accessing. For example, if you are accessing `https://abc.acme.com/foobar`, then the subject or subject alternative name needs to have “abc.acme.com” in the subject or subject alternative name.

Configuring a certificate in a run-time environment

Procedure

1. Open the Orchestration Designer Admin Console.

   For more information, see Accessing the Orchestration Designer Admin Console on page 441.
2. In the left pane, click **Certificates**.
3. On the Certificates page, click **Use other**.
4. Click **Change**.
5. On the Change Keystore page, perform the following actions:
   a. In the **Keystore Path** field, enter the path of the keystore.
   b. In the **Password** field, type a password to access the keystore.
   c. In the **Confirm** field, re-enter the password.
   d. Click **Validate** to validate the keystore location and the password.
   e. Click **Save** to save the changes.

      The system displays the Certificates page.
6. On the Certificates page, click **Use other**.
7. Click **Save**.

Adding a certificate in a run-time environment

Procedure

1. Open the Orchestration Designer Admin Console.

   For more information, see Accessing the Orchestration Designer Admin Console on page 441.

2. In the left pane, click **Certificates**.
3. On the **Certificates** page, click **Add**.
4. On the **Add Certificate** page, perform the following actions:
   a. In the **Name** field, type the name of the certificate.
   b. In the **Enter Certificate Path** field, enter the path of the certificate. You can also click **Browse** to locate the directory in which the certificate is located.
   c. Click **Continue**.

   The system displays the **Certificates** page with the newly added certificate listed under the **Name** column.
5. Click **Save** to update the keystore file with the newly added certificate and save the changes.

Fetching a certificate from a server in a run-time environment

About this task

Use the **Fetch Certificate** page to fetch a certificate from the specified URL and add it to the keystore.

Procedure

1. Open the Orchestration Designer Admin Console.

   For more information, see Accessing the Orchestration Designer Admin Console on page 441.
2. In the left pane, click **Certificates**.
3. On the **Certificates** page, click **Fetch**.
4. On the **Fetch Certificate** page, perform the following actions:
   a. In the **Name** field, type a name for the certificate.
   b. In the **Location** field, type the URL of the server from where you want to fetch the certificate.
   c. Click **Continue**.

      The system displays the **Certificates** page with the newly added certificate listed under the **Name** column.
5. Click **Save** to update the keystore file with the newly added certificate.

Generating a self-signed certificate in a run-time environment

Procedure

1. Open the Orchestration Designer Admin Console.

   For more information, see Accessing the Orchestration Designer Admin Console on page 441.
2. In the left pane, click **Certificates**.

3. On the **Certificates** page, click **Generate**.
4. On the **Generate Self-Signed Certificate** page, perform the following actions:
   a. In the **Alias** field, enter the alias name for the certificate you are generating.
   b. In the **Common Name** field, enter a common name for the certificate. You can enter the name of a person or product as a common name.
   c. In the **Organization Unit** field, enter the name of an organization unit. You can enter the name of a division or department within a company.
   d. In the **Organization** field, enter the name of the organization.
   e. In the **City** field, enter the name of the city.
   f. In the **State** field, enter the name of the state.
   g. In the **Country** field, enter the name of the country.
   h. In the **Subject Alternative Names (SAN)** field, enter the subject alternative names seperated by “,” or “;”. For example, IP:1.1.1.1;DNS:pom.avaya.com. SAN is an optional field. Specify value in this field only if you are using the extended host name validation.
   i. In the **Number of days Certificate is Valid** field, enter the duration in days for which the certificate will be valid.
   j. Click **Continue**.

      The system displays the **Certificates** page with the newly generated certificate listed under the **Name** column.
5. Click **Save** to update the keystore file with the newly generated certificate.

   Note:

   The validity of the certificate is 1189 days (39 months).

Deleting a certificate in a run-time environment

Procedure

1. Open the Orchestration Designer Admin Console.

   For more information, see Accessing the Orchestration Designer Admin Console on page 441.
2. In the left pane, click **Certificates**.
3. On the Certificates page, in the lower pane, select the check box corresponding to the certificate that you want to delete, and then click **Delete**.

   The system deletes the certificate from the keystore.

Configuring AES settings

About this task

If the Orchestration Designer application uses the AES connector, the application server must be configured to communicate with the TServer/AES.

After a server has been added, the entry is added to the AES server table, and links are added to configure channel extension mappings or to add a failover server. When clicking **map**, a new page is displayed to configure TServer extension mappings.

Note:

If you are mapping extensions across two or more AESs, ensure the extensions do not overlap.

Channels can be mapped one at a time, or a sequence of channels can be mapped using a “1-n” reference. For Interactive Response, channels must map to IR setup. For Avaya Experience Portal, channel mappings are arbitrary. (For this release, ignore Observe On Setup option.)

Tip:

A delay can be incurred when the AESs are attempting to establish connections to TServers, which can slow down the Tomcat launch as well. When connectors are not installed, initialization is quicker. If this is problematic, use “localhost” in the `tsapi.pro` file instead.

See About AES connectors on page 610.

Procedure

1. Open the Orchestration Designer Admin Console.

   For more information, see Accessing the Orchestration Designer Admin Console on page 441.

2. In the left pane, click **AES**.
3. On the AES page, configure the AES settings to add a TServer/AES server.

   For more information, see Orchestration Designer Admin Console: AES Connector page field descriptions on page 447.

Next steps

You will need to restart the AES for changes to take affect. You will also need to modify `tsapi.pro` that is included with your runtime support files before connecting to a TServer/AES. Copy or move the `tsapi.pro` file from `bea\user_projects\domains\domainname\ lib` to `WeblogicHome\common\lib`. You can then edit `tsapi.pro` file in this directory as well.

Orchestration Designer Admin Console: AES Connector page field descriptions

Name	Description
General Settings	
Timeout	Time in milliseconds allowed for the application to wait to retrieve call information from the AES connector at the start of a call. Tuned this parameter according to the performance of the AES. If a call arrives on AES before than on EP platform, the timeout value is set greater than the default of 2 seconds.

Table continues…

Trace Verbosity	Amount of debug output. Possible values are: • 0–off • 3–full
AES Settings	
Name	A unique server name to identify the TServer. Once added, the table list provides a link to configure channel-extension mapping, as well as a failover server (in the event that the primary server goes down). Channel-extension mapping is the same as the primary for the failover server, so these configurations are not required also for the failover server. **Note:** Server name must be unique and the TServer/AES and failover names cannot contain '*'. TServer/AES names cannot be duplicated (even Failover name cannot be the same as TServer/AES).
Service Name	Identifies the service provider in the following format: vendor#switch#type#server For example: AVAYA#S8300DD#CSTA#MLDDaes
User Name	Username to connect to this tserver/AES.
Password	Unencrypted password to connect to this tserver/aes.
Confirm Password	Confirm password must match password.

Connecting AES connector to AES

About this task

AES Connector cannot communicate with AES if you set AES Link to “Encrypted”.

Procedure

1. Change the Service Name from AVAYA#S8720DD#CSTA#aes05 to AVAYA#S8720DD#CSTA-S#aes05.
2. Add the following text to the `tsapi.pro` file on all of your application servers that run AES enabled Orchestration Designer applications:

   ```
   trustStoreLocation=C:/Program Files/Apache Software Foundation/
   Tomcat <version number>/webapps/aesconnector/WEB-INF/lib/
   avayaprca.jks
   ```

```
trustStorePassword=password
```

Note:

The **Service Name** varies depending on the customer. The **trustStoreLocation** value can be different depending on the location of `avayaprca.jks` file on your computer. However, the trustStorePassword value must be **password**.

For AES 7 and later, the `avayaprca.jks` file provided by Avaya does not work. Customers must provide their own certificates and configure AES 7 according to the related documentation.

Configuring IR Channel Map

About this task

The IR Channel Map configuration page is where virtual channel mapping for IR systems is defined. IR systems can overlap their physical channels. This mapping changes a physical channel to a virtual channel so that two or more IR systems can co-exist without overlap.

This configuration is necessary for both IC and AES connectors.

Tip:

Leave one system straight 1:1 mapping, change the rest.

Procedure

1. Open the Orchestration Designer Admin Console.

   For more information, see Accessing the Orchestration Designer Admin Console on page 441.

2. In the left pane, click **IR Channel Map**.
3. On the IR Channel Mapping page, in the **CallTag** field, type a call tag to identify an Avaya IR system, channel number, and timestamp.

   The format must be as follows:

   For IR 4.0: *<"machine_name"-"channel_num""timestamp">*

   For IR 3.0 SP3: *<"machine_name"-"channel_num"-"timestamp">*

   For IR 3.0 SP2 and earlier: *<"machine_name" "channel_num" "timestamp">*

4. Click **Add**.

   Call tags are specific for each IR system and are passed into the application. Use only the first part of the call tag, that is, computer name, to identify the IR system.

Result

The system displays the call tag in **CallTag**, and a **VChannel Map** link in **Channel Map**.

Next steps

Click the **VChannel Map** link to enter the physical channel and the virtual channel for this mapping. All IR systems have the same physical channel. Be sure that virtual channels do not overlap.

Configuring IC settings

About this task

If the Orchestration Designer application uses the Interaction Center (IC) connector, the application server must be configured to make the connection with either the VOX server or the VRUSM/HTTPVox in the IC system.

For more information, and the procedure to configure the application server to use the IC connector, see About the IC connector on page 622.

There are three links from the Orchestration Designer Admin application for configuring IC settings.

Procedure

1. Open the Orchestration Designer Admin Console.

   For more information, see Accessing the Orchestration Designer Admin Console on page 441.

2. In the left pane, perform one of the following actions:
   - Click **IC Common**. Perform the following action:
     - In the **Trace Verbosity** field, configure the trace verbosity. That is, amount of debug output. Possible values are 0 for off or 3 for full.
   - Click **IC VOX** to configure the IC VOX connection.

     For more information, see Orchestration Designer Admin Console: IC VOX field descriptions on page 450.
   - Click **IC VRUSM/HTTPVOX**.

     For more information, see Orchestration Designer Admin Console: IC VRUSM/HTTPVOX field descriptions on page 451.

Orchestration Designer Admin Console: IC VOX field descriptions

Name	Description
IC VOX Settings	
Name	Unique name to identify the IC
VOX Address	Address of VOX Server. Only applicable if ICC initiate the connection to VOX
Port	TCP/IP port
Timeout	

Table continues…

Name	Description
Timeout Time	Time in ms to wait for a VOX response. Default is 4000
IC Channel Mapping	
IC Channel Mapping	Used to inform ICC which VOX to target the call to. This mapping is only necessary in client mode and if you have more than one VOX connection. Enter the channels, and the ICC match the channel to the correct VOX and send the call to it.

Orchestration Designer Admin Console: IC VRUSM/HTTPVOX field descriptions

If using the VOX, configure the VOX connection. The VRUSM does not need a channel mapping.

Name	Description
General Settings	
Enable IC VRUSM/HTTPVox Connector	Clear this if you do not want the VRUSM connector to run; otherwise select it.
Use HTTPVox	Use this option to indicate communication to the HTTPVox. If this option is not selected, ICC communicates with the VRUSM.
Timeout	Time in ms to wait for a VRUSM response. Default is 4000.
Thread pool size	Number of active threads in pool handling application requests to the HTTP server(s).
Ping Timeout	Time in ms to ping HTTP server(s).
Show Pings in log	Log pings.
Attributes Description	
Trace Verbosity	Amount of debug output. Possible values are: • 0 - off • 3 - full
IC VRUSM Settings	
Name	Unique name to identify the IC.
Address	Address of VRUSM (HTTP) Server.
Port	TCP/IP port.

Table continues…

Name	Description
Secure Connection (SSL)	When selected, ICC uses SSL to communicate to the VRUSM/HTTPVox using a secure connection. **Note:** To use SSL, you must provide your own certificate and deploy it either to the Orchestration Designer Trust Store - trusted_weblm_certs.jks, or to your own external trust store. You can configure the external trust store from the Orchestration Designer configuration servlet on the **Certificates** link. Ensure that IC has a similar configuration. For more information, see the IC documentation.

User management

The default configured user is **ddadmin**. Additional users can be added for managing or configuring the settings in the Orchestration Designer Admin Console by adding them to this page.

Adding a user

Procedure

1. Open the Orchestration Designer Admin Console.

   For more information, see Accessing the Orchestration Designer Admin Console on page 441.

2. In the left pane, click **Users**.

   The system displays the **Users** page.

3. In the **User Name** field, type a user name for the user.
4. In the **Password** field, type a password for the user.
5. In the **Confirm Password** field, re-type the password.
6. Click **Add**.

Changing the password of a user

Procedure

1. Open the Orchestration Designer Admin Console.

   For more information, see Accessing the Orchestration Designer Admin Console on page 441.

2. In the left pane, click **Users**.

   The system displays the Users page.

3. In the Change Password column, click **change** corresponding to the username for which you want to change the password.

4. In the **Old Password** field, type the old password of the user.
5. In the **New Password** field, type a new password for the user.
6. In the **Retype New Password** field, re-type the new password.

Deleting a user

Procedure

1. Open the Orchestration Designer Admin Console.

   For more information, see Accessing the Orchestration Designer Admin Console on page 441.
2. In the left pane, click **Users**.

   The system displays the Users page.
3. Select the check box corresponding to the user that you want to delete, and then click **Delete**.

   A confirmation message appears.
4. Click **OK** to delete the user.

Configuring an IVR system to use a speech application

The procedures to configure a particular IVR system to use a speech application depends on the IVR system itself. The Avaya Experience Portal system, for example, requires you to administer and configure the application in the Avaya Experience Portal Management System.

For more information and the procedures required for your system, see the “Installing Avaya IVR Designer on your PC” section of *Avaya IVR Designer* document.

Running an Orchestration Designer application under SSL control

About this task

To deploy your Orchestration Designer applications to work under SSL control, configure your application to run under SSL control.

Procedure

1. Configure your application to run under SSL control. Perform the following actions:
   a. Use `https://<MyServer>/<MyDDApp>:8443:/<StartURL>`
   b. Add a security constraint in the Orchestration Designer application `web.xml` file so that either http:// or https:// can be used. Tomcat forces SSL when it is run.

2. In addition, perform the following actions:
   - If you are using VoicePortal 3.0 and Nuance OSR, you must modify your Orchestration Designer applications to use dynamic grammars if Nuance OSR has an issue with static or external grammars.
   - If the parent Orchestration Designer application is under SSL control, you must also put all of its child applications under SSL control as well. This includes modules, AESCs, and ICs.

Project file deployment

Deploying the project files on a Tomcat application server

Procedure

1. Copy the WAR file to the `webapps` directory.

   If you installed Tomcat to the default directory, the path for this directory is one of the following locations:

   - `C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps\`
   - `C:\Program Files\Apache Software Foundation\Tomcat 8.0\webapps\`
   - `C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\`

2. Start or restart the Tomcat server engine.

   When Tomcat is started, be sure that the application is available and functional.

   Note:

   Before deploying speech applications, be sure that you know how to configure database connection pooling (DBCP) to the Web server. The following link provides useful information about the values supplied, but which need to be tuned for each installation: http://jakarta.apache.org/commons/dbcp/configuration.html

Deploying the project files on a WebSphere application server

About this task

To deploy the project files on an IBM WebSphere or WebSphere Express application server, use the WebSphere Administrative Console as follows:

Note:

- The following procedure is intended as a general guideline. Be sure that you are familiar with the WebSphere Administrative Console, how to use it, when to stop and restart the software, and so on. For detailed information about using the IBM WebSphere Administrative Console, see the IBM WebSphere documentation.
- WebSphere does not support more than one instances of the same application installed, either in an EAR file or with other application in another EAR file. If any of the application is already deployed, you must remove the application from the Configure Deployment Package page during the export process.

Procedure

1. Open the WebSphere Administrative Console.
2. On the **Applications** menu, click **Install New Application**.
3. Use the browse window to locate the `ProjectName.ear` file you copied to the server.
4. If your application uses a JDBC data source, define the data source:
   - For the JDBC data source name, enter the name of your JDBC data source.
   - For the JNDI name, enter `jdbc/`, followed by the name of your JDBC data source.

     For example, if the name of your JDBC data source is MyJDBC:
   - For the JDBC data source, enter `MyJDBC`
   - For the JNDI name, enter `jdbc/MyJDBC`

     Note:

     If the JNDI name is not properly defined, your Orchestration Designer application will probably fail to access the data source properly.

   Note:

   To determine the name of your JDBC data source, see Data source management on page 295.
5. For the rest of the options, accept the defaults.
6. After the application is deployed, save your changes and start the application.

Managing project files on a WebLogic application server

Project file deployment on a WebLogic application server

Use the WebLogic Administrative Console to first set up JDBC data sources, and then deploy the project files on an Oracle WebLogic application server.

Note:

The following topics are intended as a general guideline. Be sure that you are familiar with the WebLogic Administrative Console, how to use it, when to stop and restart the software, and so on. For detailed information about using the WebLogic Administrative Console, see the *Oracle WebLogic* documentation.

See the following sections for more information:

- Setting up a JDBC data source on page 456
- Deploying the runtimeSupportWeblogic.war file to the WebLogic application server on page 440
- Deploying the project files on a WebLogic application server on page 457

Setting up a JDBC data source

Procedure

1. Click **Lock & Edit** to edit the configuration.
2. On the left navigation menu, click **Services** > **JDBC** > **Data Sources**.
3. Click **New** to create new data source.
4. Fill out the following data source details:
   - **Name**: Name the data source a logical name.
   - **JNDI Name**: jdbc/NAME_OF_DATASOURCE (the current tip works fine to determine name)
   - **Database Type**: Choose your database from list.
   - **Driver**: Choose the correct driver for your database.

   Move to the next screen.
5. Accept the default settings and move to the next screen.
6. Fill out the following information for the connection properties:
   - **Database name**: The name of your database.
   - **Hostname**: The name of the computer hosting the database.
   - **Post**: The port the database is listening on.
   - **Database username**: Username for the database.
   - **Password**: Password for the username.

   Move to the next screen.
7. Click **Test** to test the database.
8. Click **Finish**.
9. Click **Activate Changes** to save all changes.

Deploying the project files on a WebLogic application server

Procedure

1. Copy the `<project>.war` file to a temporary directory on the WebLogic machine, separate from the actual application directory where all other applications are located.
2. On the left hand navigation menu, select **Deployments**.
3. Click **Install**.
4. Navigate to your application.
5. Accept the default settings and move to the next screen.
6. Click **Finish**.

Deploying the project files on a JBoss EAP / Wildfly application server

About this task

The following procedure is intended as a general guideline. Be sure that you are familiar with the JBoss EAP / Wildfly Administrative Console, how to use it, when to stop and restart the software, and so on. For detailed information about using the JBoss Administrative Console, see the JBoss EAP / Wildfly documentation.

Procedure

1. Copy the `runtimeSupportJBossEAP.zip` or `runtimeSupportJBossWildfly.zip` file to a temporary directory on the computer where the JBoss EAP / Wildfly application server software is installed.

   Note:

   Do not copy the `runtimeSupportJBossEAP.zip` or `runtimeSupportJBossWildfly.zip` file to the actual application directory where all other applications are located.

2. Extract the `runtimeSupportJBossEAP.zip` or `runtimeSupportJBossWildfly.zip` file to the root directory where the JBoss EAP / Wildfly application server software is installed.

   The files are extracted to the `<JBossHome>/modules/com/avaya/od/main` directory. Where, `<JBossHome>` is the root directory where JBoss EAP / Wildfly application server software is installed.

3. Copy all the .war files exported by Orchestration Designer including the `runtimeconfig.war`, `<projectname>.war`, `aesconnector.war`, and `icconnector.war` files to the `<JBossHome>/standalone/deployments` directory.
4. If the Orchestration Designer project uses database operations, copy the JDBC driver file into the `<JBossHome>/standalone/deployments` directory.

5. Configure the JDBC data source by using the JBoss EAP / Wildfly Administrative Console.

   The JDBC name must be in the java:/jdbc/<*Orchestration Designer data source name*> format.

6. Restart JBoss EAP / Wildfly to access the runtimeconfig and the Orchestration Designer application.

IC and AESC installation

Installing IC on the production system

Procedure

1. Manually deploy ICConnector to your production system if you have an application referencing the ICC.

   The necessary files can be found in:

   - `eclipse\plugins\com.avaya.sce.ic.ui\data`
   - `icconnector.war`
   - `icconnector.ear`

2. Perform one of the following actions:

   - For Tomcat:
     - Copy the `icconnector.war` to your webapps.

       When you restart Tomcat, it will expand and deploy for you.
   - For Websphere:

     Use the `icconnector.ear` file and deploy the .ear file by using the Websphere Administrative Console.
   - For WebLogic:

     Use the `icconnector.ear` (preferred) or `.war` file and deploy it by using the WebLogic Administrative Console.
   - For JBoss EAP / Wildfly:

     Use the `icconnector.war` file and deploy the .war file by using the JBoss Administrative Console.

Next steps

After ICC is deployed, configure it using the Orchestration Designer Runtime Admin Configuration. See Orchestration Designer Admin (ddadmin) Web application configuration on page 441.

Installing AESC on the production system

Procedure

1. Manually deploy AES Connector to your production system if you have an application referencing the AESC.

   The necessary files can be found in:

   - `eclipse\plugins\com.avaya.sce.aes.ui\data`
   - `aesconnector.war`
   - `aesconnector.ear`

2. Perform one of the following actions:

   - For Tomcat:
     - Copy the `aesconnector.war` to your webapps.

       When you restart Tomcat, it will expand and deploy for you.
   - For Websphere:
     - Use the `aesconnector.ear` file and deploy the .ear file by using the Websphere Administrative Console.
   - For WebLogic:
     - Use the `aesconnector.ear` (preferred) or `.war` file and deploy it by using the WebLogic Administrative Console.
   - For WebLogic 12c:
     a. On the Weblogic 12c application server create a directory with the name `aesconnector.war`. That is, `/odapps/aesconnector.war`.
     b. Extract the `aesconnector.war` file in the `/odapps/aesconnector.war` directory.
     c. Deploy AES Connector. You must select the `aesconnector.war` directory while deploying the AES Connector.
     d. Copy the `tsapi.pro` file from the `lib` directory to the domain directory,

        That is, copy `tsapi.pro` from `/opt/Oracle/Middleware/user-projects/domains/OD/lib` to `/opt/Orcle/Middleware/user-projects/domains/OD`.
   - For JBoss EAP or Wildfly:
     - Use the `aesconnector.war` file and deploy the .war file by using the JBoss Administrative Console.

Next steps

After the AESC is deployed, configure it using the Orchestration Designer Runtime Admin Configuration. See Orchestration Designer Admin (ddadmin) Web application configuration on page 441.

Hot deployment: Update applications without bringing the system down

Avaya recommends that you design and create highly available Web applications, or Web applications that can be deployed or updated without bring the system down. This is referred to as “hot deployments.” Typically, this is achieved using hardware or software solutions to achieve application updates utilizing multiple servers and a load balancer.

Hot deployments essentially use HTTP routing (load balancer or IP sprayer) to route requests away from the server that will be upgraded. With sticky sessions (or session affinity), all active sessions on the server are allowed to finish (or “drain”) while other servers take the load of the server that is being taken down. When all sessions have been drained, the application on the server can be stopped and the update applied.

In such a case, the server does not actually has to stop unless common libraries are updated. This means that other applications on the server can continue to run uninterrupted.

After updating the application, it can be restarted and then traffic can be gradually routed to it to make sure that it is running properly. This approach also allows updates to be backed out, if there are problems with the new application, by routing requests back to the servers hosting the older application version.

Hot deployment approaches for developing Web application updates while maintaining application availability is widely documented on the Web for deployment to enterprise-level application servers.

The following are the references. For more references, perform a search on the Internet.

- High availability Tomcat:

  http://www.javaworld.com/javaworld/jw-12-2004/jw-1220-tomcat.html

- Maintain continuous application availability while updating WebSphere Application Server enterprise applications:

  http://www-128.ibm.com/developerworks/websphere/techjournal/0412_vansickel/0412_vansickel.html

- Understanding WebLogic integration high availability:

  http://download.oracle.com/docs/cd/E13214_01/wli/docs81/deploy/highav.html

- High availability for JBoss EAP:

  https://developers.redhat.com/products/eap/overview/

- High availability for JBoss Wildfly:

  http://wildfly.org/

Alternatively, to deploy and undeploy applications without restarting the application server, you can specify a location other than the default location for the log files and temporary directories. For more information, see Deploy and undeploy an application without restarting the application server on page 462.

Deploying a new version of a module in an application

About this task

You can deploy a new version of a module that is used in an already deployed application. You need not redeploy the application that invokes the module.

For example, if you have a deployed application named MyApplication which contains a module named MyModule, you can deploy a new version of MyModule as MyModule_V2, and then update the runtimeconfig to point to MyModule_V2. You need not modify or redeploy MyApplication.

You can deploy only different versions of a module having the exact inputs and outputs. For example, if you have a main application that calls module X that contains two parameters and you dynamically switch to module Y that contains three parameters, the system shows a run-time error.

You can dynamically switch between different versions of the same module without redeploying the application. This ensures uninterrupted service on the application server.

Procedure

1. Use the runtimeconfig Web application and perform the following actions:
   a. Open the Orchestration Designer Admin Console.

      For more information, see Accessing the Orchestration Designer Admin Console on page 441.

   b. In the left pane, click **Application Configuration**.

   c. In the **Application Name** field, type the name of the deployed application.

      Ensure that all the modules of the application are deployed. The Web application examines the application and its modules, and then creates a list of entry points and modules used by the application.

   d. Under **Module Map**, click the module map of the application to view the modules and their entry points.

   e. On the **Module Map** page, edit the entry point of the module.

      At run time, the system consults the runtimeconfig to determine if there is an override in the run-time configuration of the module that the application invokes.

2. Implement the **`updateWebAppEntryPoint`** method.

The default implementation of this method returns null. In this case, the system uses the default value. If the **`updateWebAppEntryPoint`** method returns a value other than null, the system does not consult the runtimeconfig. This method provides greater flexibility than runtimeconfig Web application, but requires modifications to be done in the application that invokes the module.

Deploy and undeploy an application without restarting the application server

When you retain the log files and temporary directories in the default location and redeploy a new version of the WAR file, the application server cannot completely delete the previous WAR file directory. This is because there can be open file handles to the log files.

Hence, to deploy and undeploy applications without restarting the application server, you should specify a location, other than the default location, for saving the application log files and temporary directories.

Specifying a location for the log files and temporary directories

To deploy and undeploy applications without restarting the application server, you must specify a location other than the default location for saving the application log files and temporary directories, or ensure that the default directory is accessible. For more information, see Deploy and undeploy an application without restarting the application server on page 462.

Note:

You must specify a location, other than the default location, for saving the application log files and temporary directories of the packed WAR files.

The following table lists the default location of the application log files and temporary directories depending on the application server on which you deploy the packed WAR file:

Application server	Default location of the application log files and temporary directories
Tomcat 7.0, Tomcat 8.0, or Tomcat 9.0	`<TOMCAT_HOME>/webapp/<appname>`

Table continues…

Application server	Default location of the application log files and temporary directories
WebLogic 11g or WebLogic 12c	`C:/<appname>` on Windows-based computers and `/<appname>` on Linux-based computers. **Note:** By default, WebLogic saves the application log files and temporary directories in the `<appname>` directory in the root directory of the operating system.
IBM Websphere 8.0 or IBM Webshphere 9.0	`<Websphere_HOME>/AppServer/profiles/AppSrv01/installedApps/<cellname>/<appname_Speech_Application.ear>/<appname>.war`
JBoss EAP / Wildfly	`<JBoss_HOME>/standalone/tmp/vfs/temp<UID>/<appname>.war/`

To deploy packed WAR files, the system requires a directory to write application logs and save the recording files. You can set a directory for the log files and temporary directories of packed as well as unpacked WAR files by using the -Davaya.dd.tempdir system configuration property or the runtimeconfig Web application configuration.

The -Davaya.dd.tempdir configuration overrides the runtimeconfig Web application configuration to ensure backward compatibility for applications that are built by using the earlier versions of Orchestration Designer. The -Davaya.dd.tempdir configuration sets a temporary directory for all applications on an application server. The runtimeconfig Web application configuration sets a location to save individual application log files and temporary directories of a single deployed application.

When you specify a location, other than the default location, for the log files and temporary directories by using the -Davaya.dd.tempdir configuration or the runtimeconfig Web application configuration and record a message, the system returns the URL for the recorded file in a file:// protocol format. When replaying the recording in a prompt, the platform on which you execute the voice browser cannot access this URL because the URL is not in an HTTP protocol format. Hence, to replay the recording, you must write a Java code to move the file from the directory location to a Web context and provide a reference to the file by using the HTTP protocol.

Note:

Before you change the location of the application log files and temporary directories, ensure that the application is deployed.

Specifying a location for the application log files and temporary directories using the -Davaya.dd.tempdir configuration

About this task

Note:

Before you change the location of the application log files and temporary directories, ensure that the application is deployed.

Procedure

In the avaya.dd.temp.dir system property, set a directory in which you want to save the log files and temporary directories. For each application, the system creates a subdirectory within the directory. For example, -Davaya.dd.tempdir=c:\temp.

Note:

If the avaya.dd.tempdir system property is set to not writable, then recording, tracing, and local reporting are not enabled.

Specifying a location for the application log files and temporary directories in the runtimeconfig Web application configuration

About this task

Note:

Before you change the location of the application log files and temporary directories, ensure that the application is deployed.

Procedure

1. Open the Orchestration Designer Admin Console.

   For more information, see Accessing the Orchestration Designer Admin Console on page 441.
2. In the left pane, click **Application Configuration**.
3. In the **Application Name** field, type the name of the deployed application for which you want to specify the log file and temporary directory location.
4. Click **Add**.
5. Under **Application Information**, click **Application Information**.
6. In the **Temporary Directory** field, specify a location other than the default location for the log files and temporary directories.
7. Click **Update**.

Configuring certificate based authentication

About this task

You can specify the type of authentication and configure certificate based on the authentication.

Procedure

1. Open the Orchestration Designer Admin Console.

   For more information, see Accessing the Orchestration Designer Admin Console on page 441.

2. In the left pane, click **Application Configuration**.

   If the deployed application has not already been added, enter the name of the application in the **Application Name** field, and click **Add**.

3. Click **Authentication Information**.

4. On the Authentication Information page, in the **Type** field, select the authentication method.

   The options are:

   - **Password**: To authenticate the application by password.
   - **Certificate**: To authenticate the application by certificate.
   - **Password and Certificate**: To authenticate the application by a two-factor authentication, password and certificate. If one factor fails, the application cannot log in to the EPM.

5. Do the following for **Password** authentication type:

   a. Select **Password** in the **Type** field.

      No other options are required.

   b. Click **update**.

6. Do the following for **Certificate** and **Password and Certificate** authentication types:

   a. Click **Choose File** to choose any file of `*.jks` or `*.keystore` type in the local file system.

   b. Click **upload** to upload the keystore file to the application server.

      The keystore file is saved in the default directory with all other configuration files, `$CATALINA_HOME/lib` for Tomcat. The **Keystore file** field is automatically updated with the keystore file name.

   c. In the **Type** field, select **Certificate** or **Password and Certificate**.

   d. **(Optional)** In the **Keystore directory** field, change the upload directory.

      A new value is applied for the next uploading. It does not affect the keystore file that you have uploaded earlier.

e. In the **Keystore password** field, specify a password for the keystore file.

   If your password is incorrect, these authentication types are not enabled.

f. Click **update**.

Java code to move the recorded file from the directory location to a Web context

Use the following Java code to move the file from the directory location to a Web context:

```
public static void copyFile(File source, File dest) throws IOException {
byte[] data = new byte[BUFFER_SIZE];
int count = 0;
FileInputStream sourceStream = new FileInputStream(source);
FileOutputStream destStream = new FileOutputStream(dest);
try {
  do {
   count = sourceStream.read(data);
   if (count != -1) {
    destStream.write(data, 0, count);
   }
  } while (count != -1);

} finally {
  // close streams, ignore any exceptions closing streams
  if (sourceStream != null) {
   try {
    sourceStream.close();
   } catch (IOException e) {
   }
  }
  if (destStream != null) {
   try {
    destStream.close();
   } catch (IOException e) {
   }
 }
}
}
```

Application execution environment

At run time, the IVR system starts the speech application on the application server. To call the application and start it running, you must provide your IVR system with the URL to the start page for the application. This URL should follow the format:

http://*myAppServer.myCompany.com:port/appName/* Start

where:

- **myAppServer.myCompany.com** is the fully qualified domain name or IP address of your application server.

- **port** is the port used to access the application. The default for this varies according to the application server:

Application servers	Default port
Tomcat	8080
IBM WebSphere	9080
IBM WebSphere Express	7080
Oracle WebLogic	7001
JBoss EAP / Wildfly	8080

- **appName** is the file name of the application you want to run.

During application execution, the speech application directs the call flow and issues requests for additional resources.

The application server for the IVR system, in turn:

- Controls execution of the generated code and subsequent generation of the VoiceXML pages.
- Uses the servlet engine to serve the VoiceXML pages to the appropriate component within the IVR system.
- If appropriate, uses SOAP and WSDL to access and use Web services.
- If set up to do so, the IVR system accepts application data, generates trace and report data, and writes the data to the appropriate files on the IVR system or application server.

The deployment process also provides tools that allow you to check various aspects of the application in the run-time environment. See Application deployment verification on page 467.

Application deployment verification

If your application does not behave as expected in the deployment environment, Orchestration Designer offers a number of utilities to help you debug and figure out what is going on. You can access these utilities with a Web browser.

Note:

These utilities are generated during the application deployment process. These utilities are only available to applications that you have actually deployed, not in the Orchestration Designer application development environment.

Accessing the application deployment utilities

The procedure to access the application deployment utilities is slightly different, depending on whether you deployed your application to a Tomcat, IBM WebSphere, WebSphere Express, Oracle WebLogic, or JBoss EAP/Wildfly application server.

Accessing utilities on a Tomcat application server

Procedure

1. Open a Web browser window.
2. In the Address field, enter the following URL: `http://domain.name:8080/`

   where *domain.name* is the fully qualified domain name of the Tomcat server.

   Tip:

   As a shortcut, or if you do not want to use the Tomcat Manager, you can enter the following URL in this step: `http://domain.name:8080/appName`, where *appName* is the name of the application you want to check on.
3. Click **Tomcat Manager**.
4. Click the name of the application you want to check on.

   The Web browser displays the index.html page for the selected application.

Accessing utilities on an application server

Procedure

1. Open a Web browser window.
2. In the **Address** field, enter one of the following URLs:
   - For JBoss EAP/Wildfly: `http://domain.name:8080/appName/index.html`
   - For WebSphere: `http://domain.name:9080/appName/index.html`
   - For WebSphere Express: `http://domain.name:7080/appName/index.html`
   - For WebLogic: `http://domain.name:7001/appName`

   where:

   - *domain.name* is the fully qualified domain name of the application server.
   - *appName* is the name of the application you want to check on.

     The Web browser displays the index.jsp page for the designated application.

   After the index.jsp page is open, there are four options, described in the following sections:

   - Validating the application deployment on page 469
   - Viewing the application in HTML mode on page 470
   - Monitoring application performance on page 471
   - Viewing the application trace and report logs on page 472

Validating the application deployment

About this task

The utility for validating application deployments is designed to help ensure that the application has been properly deployed on the application server. While it is not guaranteed to catch all deployment errors, it usually catches the vast majority of them.

Procedure

1. Access the index.html page.

   To access this page, see Accessing the application deployment utilities on page 467.

2. Click **Validate** (in the bullet **Validate the application dependencies**).

   The Web browser displays the validation page for the selected application.

3. Inspect the validation page for any sign of a problem with the installation and deployment of the application on the application server. For more information, see Validation page description on page 469.

   Tip:

   In general, green text indicates that a setting is properly configured. Yellow text indicates that an optional setting is not as expected, but in most cases, it does not indicate a condition that would prevent the application from running properly. Red indicates that a required setting is not right and, therefore, that the application probably will not function as expected.

Validation page description

The **Validation** page is divided into three basic areas:

- **Application details**: Provides information about:
  - Orchestration Designer version, framework runtime version (scert.jar), framework runtime common version (scertcommon.jar), Axis version, Axis2 version, starting language, platform, ASR, SSML usage, and WebLM URL.
  - License status for platform, including whether runtime, AES and IC connectors are enabled, or not.
  - HTTP and HTTPS Proxy settings
- **General Dependency Area Details**: Provides information for the following dependency areas:
  - **Libraries**: Displays the status of logging, licensing, and encoding run-time support library files that must be installed for the application to run properly.
  - **Web Services**: Displays the status of client Web services that must be installed for the application to run properly.

- **Database**: Displays the status of database connectivity files that must be installed for the application to run properly, including:
  - Jar files used by Tomcat for connection pooling
  - Installed JDBC driver files (application server dependent)
  - Testing option to validate data sources added in Orchestration Designer
- **Interaction Center**: Displays the status of the Interaction Center connector which is relevant only if an application will be using the IC connector.
- **Application Enablement Services (AES)**: Displays the status of the AES connector which is relevant only if an application will be using the AES connector.

- **Required dependencies**: Displays information about any other modules that the application requires in order to run properly.

  Any modules that an application depends on must reside on the same application server. If the module is missing, the validation utility flags it as an error. If the version is different or the application and module are set to use different versions of the run-time support files, the validation utility flags a potential error.

  Note:

  If you are storing “custom object” (objects that you define) in session properties and want to pass them between to modules (Web applications), the code for the objects must be in the `<tomcat_home>/common/lib` so that the class loader works properly. If that code is on `<appname>/WEB-INF/lib` then an object instantiated in project A cannot be accessed on project B. By moving the implementation of the class to the common lib and in turn the common class loader, the implementation can be shared across Web applications.

See Prerequisite configurations for running Orchestration Designer applications on an application server on page 441.

Viewing the application in HTML mode

About this task

The utility to view an application in HTML mode is provided as a debugging tool. This utility allows you to step through an application that has been deployed, checking it at each step to make sure that prompts and actions are working as expected. It does not require that you make a call to the system, but rather translates the application’s activity into an HTML mode that allows you to interact with the application in a simulation mode through a Web browser.

Procedure

1. Access the index.html page.

   To access this page, see Accessing the application deployment utilities on page 467.

2. Click **View** (in the bullet **View the application in HTML mode**).

The Web browser starts the application. The displayed page is split into two parts:

- The root page for the selected application.

  The same root page information is displayed on every page of the application.

- A **Start application** settings section that allows you to preset and simulate various options that an application can expect to use during its run.

  For example, if your application is designed to make use of the DNIS, you must supply the DNIS you want to simulate in the **DNIS** field.

  At the bottom of each page are the following buttons.

Button	Function/Comments
Continue	Moves the application to the next page.
Show Doc VXML	Displays, in a separate browser window, the generated VoiceXML code for the current page.
Show Root VXML	Displays, in a separate browser window, the generated VoiceXML code for the root page.

Note:

The **Show ... VXML** buttons are intended primarily as debugging aids. Clicking these buttons may not work as expected in every case, because they require the Web browser to send another request for the VXML contents. This request can cause session variables to change and alter the test flow of the application.

Note:

In a Firefox Web browser, only the system outputs (prompts) are displayed. To see the VoiceXML code, use Internet Explorer.

Monitoring application performance

About this task

The performance monitoring utility allows you to view and analyze a variety of statistical data associated with system performance while running the application.

Procedure

1. Access the index.html page.

   To access this page, see Accessing the application deployment utilities on page 467.

2. Click **Performance Monitoring** (in the bullet **Performance Monitoring for the application**).

   The Web browser displays the performance monitoring page for the selected application.

Performance Monitoring page description

The **Performance Monitoring** page is divided into three basic areas:

- **Application details**: Displays the name and version of the application.
- **Performance statistics**: Displays statistics related to the application's performance in five columns:
  - **Counter**: Names the operation or method in the application for which statistical data is being displayed on that row.
  - **Average (ms)**: Displays the average amount of response time for all calls to the designated operation or method, in milliseconds.
  - **Min (ms)**: Displays the fastest response time among all calls to the designated operation or method, in milliseconds. A negative one (**-1**) in this column indicates that no performance data has been captured.
  - **Max (ms)**: Displays the slowest response time among all calls to the designated operation or method, in milliseconds. The text, if any, displayed in square brackets [ ] is the name of the operation or method.
  - **Total Samples**: Displays the total number of times the designated operation or method was called during the current sampling period.
- **Command buttons**: At the bottom of the page are three command buttons:
  - **Enable/Disable**: Toggles between **Enable** and **Disable**, according to its current state. When performance monitoring is disabled, this button is labeled Enable, and Disable when enabled.

    Before data can be collected for analysis, performance monitoring must be enabled for the application. To turn off performance monitoring for the application, click **Disable**.
  - **Refresh**: Use this button when you have two browser windows open. For example, when using one browser window to view the application in HTML mode and a second window to display the performance statistics. Using this button refreshes the data after running several tests of the application.
  - **Reset**: This button clears all statistical data in the performance monitoring data file and resets all values to zero (for times) or null (for method or operation names).

    You can use this button to flush the results of previous tests and start over.

Viewing the application trace and report logs

About this task

The application trace and report logs option displays a variety of statistical data associated with system performance while running the application.

This utility allows the viewing of three common log files:

- report.log: Contains data related to the use of report items in the call flow.
- savereport.log: For Avaya Experience Portal applications only, this log contains report data that has been queued to the VPMS, but for some reason the system has not been able to send the data as scheduled.
- trace.log: Contains the VXML generated when running the application, as well as any trace item data that is generated within the call flow.

Procedure

1. Access the index.html page.

   To access this page, see Accessing the application deployment utilities on page 467.

2. Click **Application Trace and Report Logs**.

   For information about enabling and disabling these logs, see Trace on page 846 and Enabling or disabling tracing in a run-time environment on page 473.

Enabling or disabling tracing in a run-time environment

Procedure

1. Deploy and configure the speech application on the application server.
2. Edit the `ddrt.properties` file in the `/data` directory. A `ddrt.properties` file is available in the project `/data` directory that controls logging parameters.

   In the development environment, these parameters can be controlled through Avaya Application Simulator Preferences settings. In a deployed system, by editing the `ddrt.properties` file, the next run session picks up the settings. For information about the `ddrt.properties` file configuration settings, see Configurable parameters for the ddrt.properties file on page 473.

Configurable parameters for the ddrt.properties file

The following table lists the configuration settings of the `ddrt.properties` file:

Parameter	Description
localddtrace	Enables the Orchestration Designer built-in tracing. Valid values are **enabled** or **disabled**. **Note:** Avaya recommends that you disable the localddtrace parameter on the production system when using private variables. For more information about private variables, see Field on page 733, Property on page 799, and Simple Variable on page 832.
localapptrace	Enables application tracing. Valid values are **enabled** or **disabled**.
showxml	Displays the generated XML output. Valid values are **enabled** or **disabled**.
showxmlroot	Displays the generated XML output for the AppRoot in the trace.log file. Valid values are **once**, **always**, **never**. The default value is **once**.
localreportlog	Enables the local report.log file on platforms such as Avaya Experience Portal. Valid values are **enabled** or **disabled**.
skeletonreporting	Enables application skeleton reporting. This is reporting that tracks the path a caller takes through an application. Valid values are **enabled** or **disabled**.

Viewing the application report logs at run time

About this task

- For IR systems, locate the appropriate log file on the application server and view it with your text viewer or word processor of choice.

  To locate the file, start with the directory in which you have the Tomcat server software installed. From there, use the following path to the log file:

  ```
  TomcatHome/webapps/applicationName/data/log
  ```

  where *TomcatHome* is the directory in which you installed the Tomcat software and *applicationName* is the name of your application.
- For Avaya Experience Portal systems, from the main menu, select **Reports** >**Application**. For more information about this report tool, see *Application Report* in your Avaya Experience Portal documentation library.

Example application deployment

About this task

The following example provides a step by step description of deploying a simple Orchestration Designer application with Apache Tomcat Server on Windows server.

To deploy the application on a Avaya Experience Portal (VP) system, refer the following Avaya Experience Portal documentation:

- *ConvertToSingleServer.pdf* (Chapter 7: Adding the Avaya Experience Portal Test Application and Chapter 8: Running the Sample Application)

To deploy the application on a Interactive Response (IR) system, refer the following Interactive Response (IR) document:

- *ApplicationDevelopment.pdf*

Procedure

1. Configure Apache Tomcat 7, Tomcat 8, or Tomcat 9. Perform the following actions:
   a. Download and install the Apache Tomcat software to your Windows server.
   b. Run the installation using the default settings.
2. Create an application. Write a simple “Hello World” application in the Eclipse development environment that:
   - Starts up
   - Says “Hello World” (Use the “Announce” node in the call flow.)
   - Completes and Hangs up
3. Test your application.

   Test the application in the simulator in the Eclipse IDE. You should hear your announcement “Hello world”.

   In case the application is not working, refer to the VXML log in the simulator.
4. Export your application. Perform the following actions:
   a. After testing the application, stop the Tomcat server.
   b. Click **File** > **Export** to export your project.

      The system displays the Export wizard.
   c. Double-click **Avaya OD Development**.
   d. Click **Export Orchestration Designer Speech Project**, and then click **Next**.

      The system displays the Export Orchestration Designer Project wizard.
   e. Follow the pages on the wizard and configure the settings. For more information, see Exporting an Orchestration Designer project on page 419.
   f. On the Specify Platform Details page, depending on your base platform in the **Platform** field, click **Experience Portal** or **IR**. The default value is **Desktop**. Click **Next**.
   g. Follow the pages on the wizard and configure the settings.

      Note:

      On the Configure Web Application Descriptor page, retain the default settings.

h. Click **Finish**.

5. Export the application run-time support files. Perform the following actions:
   a. On the **File** menu, click **Export**.

      The system displays the Export wizard.

   b. Double-click **Avaya OD Development**.
   c. Click **Export Runtime Support Files**, and then click **Next**.

      The system displays the Export Application Runtime Support Files wizard.

   d. Follow the pages on the wizard and configure the settings. For more information, see Exporting the run-time support files on page 418.

      At your chosen destination directory, the following three files are created:

      - `HelloWorld.war`
      - `Runtimeconfig.war`
      - `runtimesupportTomcat<version number>.zip`

      The application is now exported.

Note:

Installing runtime support files and the runtime configuration web application only has to be done during the initial application server setup or after installing Orchestration Designer updates or service packs. If the Application server already has been configured for Orchestration Designer applications then steps 5-7 can be skipped.

6. Set up the application server. Perform the following actions:
   a. Unzip the `runtimeSupportTomcat<version number>.zip` into Tomcat's common directory.
   b. Copy the `runtimeconfig.war` file to the `webapps` folder. For example, `C:\Program Files\ApacheSoftware Foundation\Tomcat 5.5\webapps`
   c. Restart the Apache Tomcat Service on the Application Service.
   d. Launch the runtimeconfig Web page using your server IP address.

      For example, `http://<server_IP_address>:8080/runtimeconfig`

      User Id: `ddadmin`

      Password: `ddadmin`

   e. Under **License Server**, provide the license URL of the WebLM license server and the time in seconds that the system must wait for a response from the WebLM license server while attempting to connect to the WebLM license server.
   f. Under **Connectivity Settings**, set the same proxy settings as in your Eclipse IDE. For more information, see Configuring connectivity settings on page 442 and Considerations for enabling an HTTP or HTTPS proxy connection on page 538.

Set the proxy settings only when the application server machine is on a different side of the firewall from the license server, or any other network resources (that is, web services) that the applications need to access.

7. Deploy the application. Perform the following actions:
   a. Copy the application .war file (`HelloWorld.war`) to the `webapps` folder of the application server.
   b. Restart the Apache Tomcat server.

Chapter 32: Troubleshooting

Troubleshooting

Problem description

When using AACC Handoff in simulation a long message results in:

java.lang.NullPointerException

java.util.StringTokenizer.<init>(StringTokenizer.java:182)

java.util.StringTokenizer.<init>(StringTokenizer.java:204)

com.avaya.sce.pdc.aacc.AACCReply.requestHandler(AACCReply.java:123)

com.avaya.sce.pdc.aacc.AACCReply.doGet(AACCReply.java:49)

javax.servlet.http.HttpServlet.service(HttpServlet.java:621)

javax.servlet.http.HttpServlet.service(HttpServlet.java:722)

Solution

You must limit the size of the messages when testing AACCHandoff in simulation, as long messages are because of the query string limits that the Eclipse Browser widget imposes.

Problem description

When running certain applications (E.g. ConferencePredefined sample application), if the system enables Dialog Designer->Avaya Application Simulator->Enable display of the Avaya Voice Browser output in a console window, the application can behave irregularly.

Solution

Disable Dialog Designer->Avaya Application Simulator->Enable display of the Avaya Voice Browser output, if you encounter the application behaving irregularly.

Problem description

If the system finds any leading \r\n in a document before the ?,xml… the parse fails.

Solution

The work around in the JSP page is to have:

<?xml version=”1.0” encoding=”UTF-8”?>

<!DOCTYPE ccxml PUBLIC “http://www.w3.org/TR/ccxml” “ccxml.dtd”>

<ccxml xmlns=“http://www.w3.org/2002/09/ccxml” version=”1.0”>

for the first 3 lines before <%import...

Problem description

When making a call to a line on the same switch and the call is in the busy state, you might get back a “failed” instead of a “busy” state indication.

Solution

You must not check specifically for the “busy” state.

Problem description

An OD application fails to run if the application is compiled with a Java version that is newer than the one on the Application Server.

Solution

Select the Java Compiler compliance level that matches what is installed on the Application Server located at Window->Preferences->Java->Compiler in eclipse.

In the email editor, if the email extends beyond the view, Eclipse creates a scroll bar for both the email message body and the view.

Solution

You must manually change the view scroll bar for operations such as find/replace asthat operation does not auto control the view scroll bar.

Problem description

Encountering the error “Runtime error” when using the online help.

Solution

You must change your Internet Explorer settings to work correctly with Sun’s JavaDoc utility. While in your IE browser, go to Preferences, Tools, Internet Options. Select the Advanced tab. Ensure that you check the “Disable Script Debugging (Internet Explorer)” and “Disable Script Debugging (other)”. Click OK. Close all IE windows and Eclipse and then restart Eclipse.

Problem description

Running the same application twice in a row without restarting tomcat generates exception errors.

Solution

To work around this issue, enable “Automatically restart Tomcat when running an application.”

Problem description

For Email/SMS simulation, the system does not display multibyte characters correctly.

Solution

Go to Eclipse Preferences>General>Workspace and select Other: UTF-8 as the Text File Encoding.

Problem description

Experiencing slowness in simulation or unexpected delays.

Solution

Change the garbage collection settings by adding either:

-XX:+UseConcMarkSweepGC –XX:+UseParNewGC or –Xincgc to your eclipse startup.

Example: Create a shortcut to eclipse and in the properties target:

C:/SageRel-4.1\eclipse\eclipse.exe –vmargs –XX:+UseConcMarkSweepGC –XX:+UseParNewGC

Add 2 entries in the Dialog Designer->Speech->Simulators table, one for vox->ic and one for ic->vox.

Problem description

The console window might display warning messages about invalid log4j configuration when starting Tomcat from Eclipse.

Solution

Add a log4j.properties file to the Tomcat classpath (that is, to <tomcat home>/common/classes). The properties file must contain these lines:

log4j.rootLogger=info, stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d{dd/MM/yyyy HH:mm:ss} %5p - %m%n

Problem description

When running on WebLogic 10.3.1 or above, the app losses control of the http session it starts initially. The http session can go idle, eventually times out, and the applicaton server removes it. The time-out interval depends on what you set in the deployment descriptor either by console or editing the weblogic.xml. If you are seeing this symptom caused by WebLogic starting to have the cookie-http-only parameter set to true by default, you must change the value of the parameter to false.

Solution

You must deploy a weblogic.xml with the following content in the application's WEB-INF directory. If you already have the weblogic.xml, you must incorporate the following content. However, if it is too much work, you can choose not to do anything. We believe that the accumulation of http sessions (before they get removed by the system) must not be harmful since each http session used by OD is really light weight - we don't use it to store information.

Problem description

If the application executes the number of database operations that exceeds the number of the connection pool specified for the data source, the trace log says javax.servlet.ServletException: Connection not available, Timed out waiting for 180001. The database operation hangs for a while before the system throws an error.

Solution

The configuration must change as follows:

Add the following node to the web.xml of the application

<resource-ref>

<!— replace DS with the real data source name in OD ->

<res-ref-name>jdbc/DS</res-ref-name>

<res-type>javax.sql.DataSource</res-type>

<res-auth>Container</res-auth>

<res-sharing-scope>Unshareable</res-sharing-scope>

</resource-ref>

Export the app for deployment; ensure that this is done after the web.xml is changed; otherwise, you have to redeploy the app.

Problem description

If you have trouble deploying the Tsapi.pro file or WAS install scripts on a Linux machine.

Solution

Try using the utility dos2unix on the files.

Problem description

Changes to AES, including TSAPI 6.3.1 and higher, have resulted in changes to the reporting of a call in a busy state after a CTI Dial operation or CTI Consultation Call. Some busy calls may be reported as failed instead of busy. Your application may get affected if it specifically checks calls for a busy return state instead of just checking for a failed call.

Solution

1. Login to AES using the customer or craft account.
2. Run the following command:

   ```
   psql -U avaya -d mvap -p 5430 -c "insert into reg_dword
   (hostname,name,value) values
   ('localhost','EnableGuessFailingDevice','1');"
   ```

3. Verify whether the value is changed by running the following command:

   ```
   psql -U avaya -d mvap -p 5430 -c "select * from reg_dword order by
   name"
   ```

4. After adding the value to the database, restart the TSAPI service from the **Maintenance** > **Service Controller** page on OAM.

Troubleshooting tips

Eclipse configuration settings

You must make changes in Eclipse configuration settings when you have many variables. Symptoms of an issue related to these settings are that Eclipse hangs and the .log file shows lots of these exceptions: “**org.eclipse.swt.SWTError: No more handles**”. The recommended configuration changes to the eclipse.ini file are:

-vmargs

-Xms256m

-Xmx512m

-XX:MaxPermSize=512m

-XX:+UseConcMarkSweepGC

-XX:+CMSClassUnloadingEnabled

-Dfile.encoding=UTF-8

httpioprocessor CCXML

When using the httpioprocessor CCXML sample application, you must ensure that a copy of the commons-httpclient-3.1.jar is in the Tomcat lib directory. This information is not provided by default with Tomcat 7.

For new classification

If the incoming call classification is “LIVE VOICE” then you must go to the params tab in the simulation profile and select a new call classification and then select “live voice”

About AES

When mapping extensions across 2 or more AES’s, the extensions cannot overlap.

CTI (AES) Connector

The CTI (AES) Connector has a setting in the web.xml called providerstartupdelay (defaults to 4000 ms) that permits you to adjust a maximum delay value (in ms) to permit the provider to come in service. The connection is lost and retried, if the provider does not come into service before the timer expires..

You can now disable the application from gathering call information automatically by changing the flag sage.cti.autoinvoke.callinfo to false in your web descriptor. If you make this change, then you must use a CallInfo item in your data node and must set the property: Initial Call to true before doing any CTI (AES) Call operations.

The system configures the CTI (AES) Connector to startup when the application server is started. You can disable this functionality if you do not have the CTI (AES) Connector configured or if you do not want it to run by editing the web.xml file found in your CTIConnector/WEB-INF directory on your appserver. Look for the value: <load-on-startup>1</load-on-startup> and remove or comment out this tag.

Note:

Setting it to 0 does not turn it off. You must manually start the CTIConnector, if this flag is removed or commented out.

Passing data using the UUI

When passing data using the UUI, the maximum number of bytes you can use to write is 96. You can use a complex variable as the data for UUI. The properties permit entry of a variable/field in the dial operation.

Connectors

You must not use the "Refresh" button on your HTML browser when an application relies on a connector (CTI, AES, IC, or more). Because of the behavior of some HTML browsers, refresh action causes the connectors to initialize a second time for the session which the system does not allow. You must close the HTML browser and then startup a new session.

Databases

When using Windows 7, to create an ODBC datasource to access Microsoft Access databases, which some of our sample applications (such as LexingtonLend) use, you must run odbcad32.exe directly from the C:\windows\sysWOW64 directory instead of running the ODBC admin from Control Panel.

Design Constraints

The Orchestration Designer framework has constraints on the number of iterations through call flow nodes that do not generate VXML. Those nodes are: Data, Servlet, and Tracking. OD uses a servlet request forward to move between consecutive data nodes that do not generate VXML. This is faster and more efficient than redirecting the HTTP request. However, if you want to iterate more than 100 times on consecutive data nodes, you might run out of stack space. If you want to iterate more than 100 times, you must move your iteration into Java code. If the data is pure operation on a variable collection, you can use the new Loop Collection item in the data node to loop through the collection.

Deployment

To deploy packed war files (vs expanded directory) on WebLogic, you need a directory for writing application logs and recording files. By default, OD creates a directory with the application name in the OS's root ("C:/" for windows and "/" for Linux), and the log files and recording files are written to the data/logs and data/temp directories respectively. Alternatively, you can set a specific directory by using the system property avaya.dd.temp.dir. The system creates a subdirectory for each application. For example:

-Davaya.dd.tempdir=c:\temp or some other value.

In either case (default or tempdir), OD assumes that it can create directories and files in that location. If the system disables the directory, then you must create a subdirectory for each OD web application deployed with sub directories temp and data/log. When recording, temp recording files are placed in temp and if tracing is enabled the trace.log and report.log are in data/log. If you have a deployed application named Simple, under c:\temp you can see: c:\temp\Simple \temp and \data\log.

If the avaya.dd.tempdir is not writable then recording does not work, nor the tracing and local reporting can. The avaya.dd.tempdir can also be used with non-packed war files as well, thus providing a mechanism to move the temp and log files out of the web application to a separate area. Lastly when altering the location of the temporary directories,, the URL returned for the recorded file will be in the file:// protocol format and is NOT directly usable in a prompt to replay the recording since the file URL is not accessible on the platform where the voice browser is executing. To replay the message you must write java code to move the file from the temp location to a web context and reference the file there using the http protocol.

Garbage collection performance

To optimize garbage collection performance on the Application Server, you must add the following:

For windows:

Create a system level environment variable:

export JAVA_OPTS = “-server -Xmx1024m -XX:MaxNewSize=30m -XX:+UseParNewGC -XX:+UseConcMarkSweepGC-XX:CMSInitiatingOccupancyFraction=60 –XX:ThreadStackSize=512”

For Linux:

when the system starts Tomcat as a service, add the flowing to the beginning of your /etc/init.d/tomcat file:

JAVA_OPTS="-server -Xmx1024M -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:ThreadStackSize=512"

ulimit -n 8192

Note:

The Xmx1024M parameter must be set to somewhere around half the size of the RAM configured on the box.

Restart the machine after setting the JAVA_OPTS, and verify that the new setting are in effect by doing an: ps -ef | grep java.

ECMA Script Expressions

1. Single quotes that are part of the value of an ECMAScript expression generates a semantic error in the Voice Browser if not formatted with an escape character (\). For instance:

<var name="Output" expr="'John's return value'"/> incorrect expression

<var name="Output" expr="'John\'s return value'"/> correct expression

For Orchestration Designer to correctly generate the VXML output you must add another escape character when building the expression in your call flow, otherwise Java interprets it as an escape character itself.

Within your call flow you can create the expression as follows:

John\\'s return value

This ensures the escape character is generated in the VXML output. Other characters to watch out for include ('gt;' for '>', '<' for '<', and '&' for '&').

Email

1. If sending an attachment that contains a multibyte name in an outgoing email, the email client may not decode the name causing it to show the encoded name, although the contents of the attachment will be intact.
2. When using the new Email (and SMS) features in simulation remember to click Send after Run Application. Once the application is running you can send multiple messages.
3. Variable substitution in the email body use _{**variable**}_ to indicate the contents are to be replaced with the variable value. If you want to use _{ in the body, you need to escape the leading underscore with the backslash. For example, the following email body contents in design: “using a backslash _ {will treat this as literal}_.”. At runtime, the content of the message will be “using a backslash _{*will treat this as literal*}_.”

Grammars

For external grammar deployment, the application web.xml must be edited to change localhost to the actual ip address of the application server. This can be done before or after deployment. As an alternative, you can use relative urls for external grammars, or store the host and port in a OD

variable and reference the OD variable in the url %DDVAR_myextgrammarsloc%/Myapp/somedir/agrammar.grxml MR 5312.

1. The quality of the Japanese and Chinese ASR provided by Microsoft (used in OD simulation) for built in grammars is not very high. When testing built-in grammars using Japanese or Chinese, it is recommended to use the “Send ASR” feature.
2. The following illustrates how the runtime can be used to generate large external grammars. The example is simple, but can be extended to build the grammar from an external data source. The reason to do this is for performance at runtime.

```
import com.avaya.sce.runtime.DynamicGrammar;
import com.avaya.sce.runtime.GrammarItem;
import com.avaya.sce.runtime.GrammarRule;
import com.avaya.sce.runtimecommon.SCESession;
public class MyGrammar extends DynamicGrammar {
public MyGrammar() {   }
public void buildGrammar(SCESession mySession) {
GrammarRule rule;
rule = new GrammarRule();
rule.add(new GrammarItem("dog", "DOGTAG"));
rule.add(new GrammarItem("cat", "CATTAG"));
add(rule);
rule = new GrammarRule();
 rule.add(new GrammarItem("blue", "BLUETAG"));
rule.add(new GrammarItem("red", "REDTAG"));
add(rule);
}
public static void main(String[] args) {
try {
MyGrammarg = new MyGrammar();
g.buildGrammar(null);
String result = g.toString("en-us",
SCERT.ASR_TYPES_MAP.get(SCERT.ASR_DISPLAY_LOGUENDO),
SCERT.VOICE_MODE);
XmlWriter writer = new XmlWriter();
Document doc = XML.getDocumentFromXMLString(result);
writer.write(doc, System.out);
}catch (Exception e) {
```

```
e.printStackTrace();
}
}
```

IC

1. To use SSL, you must provide your own certificate and either deploy to the Orchestration Designer Trust Store: trusted_weblm_certs.jks or your own external trust store which can be configured from the Orchestration Designer configuration servlet on the Certificates link. Similar configuration is necessary on the IC side. Please consult IC documentation for those details.
2. You cannot run an ICC enabled module in the simulator standalone as it will not invoke the ICC newcall operation. You must invoke it using a parent application that is also IC enabled.
3. If you implement a prompt to play while transferring a call to IC, the transfer can sometimes occur faster than the prompt is played. The flush prompt feature should be used for this scenario.
4. Some variables require a type. Specifically, variables used in a web operation might have to be converted to a specific type (int, long, etc). Fields on the VDU variable and VDU cache variable do not perform this conversion as they deal strictly with Strings. If you need to use a VDU field within your web operation that requires a type to be set, you should copy the VDU (or VDU cache) field into another local variable that does perform the conversion.
5. If you are running an IC-enabled application in the simulator, you must check to make sure there is a VOX or VRUSM/HTTPVox simulator configured in Window.

NDMs

When using NDM version 5.2.0 or later, there are some code changes required as follows:

Once the NDM war file is expanded in your app server, navigate to the ndm-core directory:

Open dialogs/application/ndm folder

Change MainMenu.jsp - vxml version to 2.1

Change menu_standalone_subdialog.jsp - vxml version to 2.1

Change rootSubdialog.jsp - vxml version to 2.1Open dialogs/framework/ndm/osdm.jsp

Change 175 to <%=appName%>/Controller/logdatasetter

Change 165 to <%=appName%>/Controller/logdatagetter

Open WEB-INF/WEB.xml

Change vxml version to 2.1.

If you are using Tomcat 7.0, you will need to make a change to either the conf/context.xml file, which would be a global change or you can make the change to the individual context file for your app which is found in conf/Catalina/localhost/<appname>.xml

Add the option useHttpOnly=”false” to your context to allow Nuance code to pass the JSESSIONID cookie which it requires <Context useHttpOnly="false"….

Simulation

1. When simulating using the VRUSM simulator, it is recommended that you use the OD Runtime configuration servlet and set the ping timeout = 0 secs. This will effectively disable pings which aren't necessary for simulation. A future release/service pack will allow you to do this automatically using the simulation page in OD.
2. Microsoft Speech doesn’t support SSML. To use the simulator to test applications using SSML, you must use the MRCP client and the target speech server.

SSL

When using SSL and Scansoft/Nuance ASR, grammars should be built as dynamic, even if they don’t need to be, so they will be processed, in-line. The ASR engine has a problem processing regular grammars if they exceed 4 elements.

SMS

1. When using the new SMS (and Email) features in simulation remember to click Send after Run Application. Once the application is running you can send multiple messages.
2. Variable substitution in the SMS (and email) body use _{*variable*}_ to indicate the contents are to be replaced with the variable value. If you want to use _{ in the body, you need to escape the leading underscore with the backslash. For example, the following email body contents in design: “using a backslash _ {will treat this as literal}_.”. At runtime, the content of the message will be “using a backslash _{*will treat this as literal*}_.”

Tomcat

1. Tomcat folders in the classpath should not contain duplicate files or any unnecessary files as this may cause a conflict.
2. For runtimeconfig to support multi-byte entries, you need to set the following JVM value if running Tomcat:

   -Dfile.encoding=UTF-8

   If running tomcat standalone, add this value to your catalina.bat (or catalina.sh) file before the set JAVA_OPTS call that already exists:

   set JAVA_OPTS="-Dfile.encoding=UTF-8"

   If you are invoking tomcat from your Eclipse IDE, add the following to your eclipse.ini and restart Eclipse

   -Dfile.encoding=UTF-8
3. After deploying an application using Tomcat, if you click on the application to validate the application use caution when clicking on your browser's back arrow (button). In some cases this could result in your application being undeployed. Specifically this can happen when going backwards from the .index page (the one with validate and view options) to the Tomcat Manager application.
4. Tomcat 6.0.24 and newer added a new memory monitor, JreMemoryLeakPreventionListener, that may erroneously warn about memory leaks caused by threads started--but not stopped--by a web application. This is the expected behavior for Orchestration Designer applications caused by threads that are started by shared runtime (scertcommon.jar). These threads are shared by all Orchestration Designer applications and are not stopped intentionally.

5. Tomcat will reload the context of your speech application when it detects a change. This will occur pretty much anytime you change your project and save it while Tomcat creates a new current one for your servlet and reloads the dependencies. OD does not have a way to indicate to Tomcat that a sub module is a dependency of its parent application. Therefore, when the parent's context is loaded into the new class loader there is a conflict with the class loader that was used to load the sub module. The solution is twofold:
   - If your context file points to your speech application, you can simply reload the context of the sub module right in OD using the Tomcat context option "Reload this context."
   - If you have deployed your sub module into Tomcat and Tomcat is running it out of the webapps directory, the simplest solution is to restart Tomcat or you can use the Tomcat Manager application. Refer to the Tomcat documentation for more information on how to do this using the Tomcat manager.
6. a. The console window may display warning messages about invalid log4j configuration when starting Tomcat from Eclipse. To fix this add a log4j.properties file to the Tomcat classpath (i.e. to <tomcat home>/common/classes). The properties file should contain these lines:

   log4j.rootLogger=info, stdout

   log4j.appender.stdout=org.apache.log4j.ConsoleAppender

   log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

   log4j.appender.stdout.layout.ConversionPattern=%d{dd/MM/yyyy HH:mm:ss} %5p - %m %n

TTS

Using symbols in TTS (i.e. &, <, >, etc.) in a prompt may result in runtime errors. These symbols generate invalid VXML code. **Workaround**: spell out the word. For example: '&'"and", '<' "less than", etc

Validate Menu Option

Running the validate menu option on any project causes a Warning to show up in the Problems view. The Resource is web.xml and the Location is line 8 which is: -><! DOCTYPE web-app PUBLIC"-//Sun Microsystems, Inc//DTD Web Application 2.3//EN" http://java.sun.com/dtd/web-app 2 3.dtd> There is currently no work-around to get rid of thiswarning once it occurs.

WebLogic Application Server

When running on WebLogic 10.3.1 or above, the app losses control of the http session it starts initially. The http session would go idle, eventually times out and get removed by the application server. The time-out interval depends on what you set in the deployment descriptor either through the console or editing the weblogic.xml. If you are seeing this symptom, it is caused by WebLogic starting to have the cookie-http-only parameter set to true by default. You need to change the value of the parameter to false. To do that, you would need to deploy a weblogic.xml with the following content in the application's WEB-INF directory. If you already have the weblogic.xml, you would need to incorporate the following content. However if you think it's too much work, you can choose not to do anything. We believe that the accumulation of http sessions (before they get removed by the system) should not be harmful since each http session used by OD is really light weight - we don't use it to store information.

weblogic.xml:

<!DOCTYPE weblogic-web-app PUBLIC "-//BEA Systems, Inc.//DTD Web Application 8.1//EN"

"weblogic810-web-jar.dtd">

<weblogic-web-app>

<session-descriptor>

<cookie-http-only>false</cookie-http-only>

</session-descriptor>

</weblogic-web-app>

Web Services

1. After generating a new web service operation, you may see warnings in the java code generated by Axis. These warnings are not harmful and you do not have to take any action to correct them.
2. The graphical Web Services tool is intended for use with simpler web services. I.e. Web services that return base java types (string, int). If your Web service returns an object that has nested objects, at a minimum, you will need to select “Use Java Obj” on the Web Service wizard. This will instruct the runtime code to take the returned value and place it into the Orchestration Designer variable as an Object. You will need to write some Java code to examine your returned Object and store values into Orchestration Designer variables. This is necessary as Orchestration Designer only supports a simple variable system of either a variable or a grouping of variables.

WebSphere Application Server

1. If you have trouble deploying the Tsapi.pro file or WAS install scripts on a Linux machine, try using the utility dos2unix on the files.
2. Some database drivers work fine in Orchestration Designer when designing the speech application, but do not work in some versions of Websphere. In that case, replace the driver in Websphere with an updated or different version depending on your database type and configure the settings appropriately referring to the WebSphere documentation.

Chapter 33: Project properties

Orchestration Designer project properties management

Orchestration Designer project properties

Many project properties are established at the time an Orchestration Designer speech project is created. Other properties you are prompted to examine and possibly change when you deploy the project for use.

However, there can be times when you want or need to change various project properties; for instance, to add a database source to the project or to add a language.

The Orchestration Designer properties pane has the following tabs, depending one whether the properties are for a speech application project or call control application project:

- General tab: View and edit a variety of project properties.
- Speech tab: *For speech application projects only*. Set ASR and TTS related options.
- Languages tab: *For speech application projects only.* Configure project language settings.
- Pluggable Connectors tab: Define the connectors enabled to support your Orchestration Designer project.
- Web Descriptor tab: View and edit various application parameters that govern how the application operates during development and at run time.

Note:

For CCXML projects, the project properties pane consists of only the following tabs:

- General tab
- Pluggable Connectors tab
- Web Descriptor tab

Setting Orchestration Designer project general properties

Procedure

1. In the Navigator view or the Avaya OD Navigator view, click the project for which you want to set the general properties.

2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left navigation pane, click **Orchestration Designer**.
4. Click the **General** tab.
5. Set the project general and meta information.
6. Click **OK**.

Orchestration Designer project properties: General tab field descriptions

Use the **General** tab of the Orchestration Designer properties pane to:

- View general and project meta information.
- Edit selected project information.

Name	Description
Project	
Name	(Display only) This field is the name of the project as assigned when the project was created.
Version	The number to designate this version of the application. By default, Orchestration Designer assigns a value of 1.0.0 to this field.
Runtime Platform	The type of IVR platform the application is targets to run on. The options are: • Avaya Experience Portal: Deployment optimizes the application to run on an Avaya Experience Portal system. • IR: Deployment optimizes the application to run on an Avaya IR system. • Desktop: (Default) Deployment optimizes the application to run on one of the supported operating system desktops. This option is intended primarily for development. • Other: Deployment optimizes the application to run on other systems that support the W3C VoiceXML 2.0 or 2.1 Recommendation. • MPS: Deployment optimizes the application to run on MPS system.

Table continues…

Name	Description
VXML Compatibility	The VXML version compatible with the selected runtime platform. By default, Orchestration Designer sets the projects to VXML 2.1 compatibility.
Enable DTMF tone variables in prompts	Check box to enable DTMF generation that copies DTMF audio tones into the project. It will enable “DTMF Tone” when using Phrase Variables in the Prompt Editor.
Allow HTML Mode	Check box to allow HTML mode in speech applications.
Icon Properties	
Small	Displays the *small* icon that Orchestration Designer uses to represent the speech project in the Call Flow Editor palette (when exporting it as an Orchestration Designer reusable module). See Deploying a speech, or a message flow, or a web project as an Orchestration Designer reusable module on page 431. For information about selecting an icon for a speech project, see Setting an icon for a reusable module on page 496. **Note:** By default, Orchestration Designer includes two small graphic files: `defaultapplicationsm.gif` and `defaultmodulesm.gif`. To use a different icon, create and import a unique file. To do this, See Importing an icon for the reusable modules on page 495.

Table continues…

Name	Description
Large	Displays the *large* icon that Orchestration Designer uses to represent the speech project in the Call Flow Editor palette (when exporting it as an Orchestration Designer reusable module). See Deploying a speech, or a message flow, or a web project as an Orchestration Designer reusable module on page 431. For information about selecting an icon for a speech project, see Setting an icon for a reusable module on page 496. **Note:** By default, Orchestration Designer includes two large graphic files: `defaultapplicationlg.gif` and `defaultmodulelg.gif`. To use a different icon, create and import a unique file. To do this, see Importing an icon for the reusable modules on page 495.
Meta Information Properties	
Vendor	If a vendor name was entered during project creation, this field displays that name. To change it, enter a new vendor name.
Category	If a category descriptor was entered during project creation, this field displays that descriptor. To change it, enter a new category descriptor. **Tip:** If planning to use this project as a reusable module, what is entered in this field determines where in the Call Flow Editor palette that this module will appear. If left blank, by default, the module appears in a group labeled “Modules.” If something else is entered, the module is grouped with other modules that have the same category label. This is useful when building a large and complex application that uses several modules that can be grouped together in the Call Flow Editor palette.

Table continues…

Name	Description
Description	If a project description was entered during project creation, this field displays that description. To change it, enter the new project description. **Tip:** If the project is exported as an Orchestration Designer reusable module, this description appears as a pop-up message when the cursor hovers over the name of the module in the Call Flow Editor palette.

Orchestration Designer project properties: JS/CSS/Theme tab field descriptions

You can use **JS/CSS/Theme** to customize your web application.

Name	Description
Use JQuery Based FrameWork	Select this checkbox if you want to provide your own `JavaScript` framework. By default, this chekbox is checked, indicating the application is using JQuery based framework. If you uncheck this checkbox, then Orchestration Designer removes all the JQuery references from the application.
JQuery Theme	
Theme Name	Enter the name of the customized theme that you created using JQuery Theme Roller. Ensure this name is same as the name of one of the `CSS` files from the theme package.
Theme Swatch	Enter the swatch that you want to use.
Use custom templates instead of system templates to generate jsp files	Select this checkbox to switch to a set of custom templates that the code generator uses to create the runtime `JSP` files. The custom templates – `AppRoot_template.jsp`, `Form_template.jsp`, `Menu_template.jsp`, and `Return_template.jsp` are added to the `custom` folder of the project.

Button	Description
Add File	You can use **Add File** button to browse and add a `CSS` or `JavaScript` file from any directory to the list. Once you add the `CSS` or `JavaScript` file, Orchestration Designer copies it into its respective folder (`CSS` or `JS`) in the project, and creates links in the application pages.
Add Link	You can use this button to add file references to the application pages. The link can be a URL or a resource existing in the project.
Delete	You can use **Delete** button to remove references from the application pages. However, it does not remove the files from project folders.
Navigation buttons: **Back Button Label ID** **Next Button Label ID**	By default, Orchestration Designer labels the two navigation buttons, at the bottom of each application page, as “Back” and “Next”. You can change the labels or make them language specific by using text items from a `Textset` file.

Importing an icon for the reusable modules

About this task

You can import a graphic file to use as an icon for reusable modules.

Procedure

1. In the Navigator view, right-click the **icons** directory in the project where you want to import the graphic.
2. On the pop-up context menu, click **Import**.

   Orchestration Designer displays the **Select** page of the **Import** wizard.
3. In the list of import sources, click **File system** and then click **Next**.

   Orchestration Designer displays the **File system** page of the **Import** wizard.
4. Perform one of the following actions:
   - Enter the path to the directory that contains the graphics file you want to import.
   - Use the **Browse** button to locate and select the directory that contains the file you want to import.

     Orchestration Designer displays in the left pane the directory you selected and in the right pane a list of all files in that directory. Each file is preceded by a check box.
5. Select the file you want to import.

Note:

Make sure that the file or files you select meet the size and format requirements for Orchestration Designer icon files. Files must be GIF format, 16 x 16 pixels for small icons or 32 x 32 for large icons. You can select multiple files to import at one time.

6. Verify that the **Into folder** field shows the path to the icons directory for your project.

   If not, use the **Browse** button to locate and select the appropriate directory.

7. Click **Finish**.

Setting an icon for a reusable module

About this task

You can set an icon for a speech project that you want to export as a reusable module. After setting the icon, when you export the speech project as a reusable module to the reusable modules directory, the system adds the reusable module in the form of a module item under **Modules** in the **Palette** pane of the Call Flow Editor. The reusable module is represented by the icon that you set.

If you change the icon after exporting a speech project as a reusable module, you must re-export the speech project as a reusable module to use the new icon.

Procedure

1. In the Navigator view or the Avaya OD Navigator view, click the project in which you want to set an icon for the reusable module.
2. On the **Project** menu, click **Properties**.

   The system displays the **Properties for** <*project name*> dialog box.

3. In the left navigation pane, click **Orchestration Designer**.
4. Click the **General** tab.
5. Expand **Icons**.
6. Select a small icon for the reusable module, perform the following actions:
   a. Click **Browse** next to **Small**.

      The system displays the **Contents of** “icons” dialog box.

   b. Select the graphic file of the small icon. The graphic selected must be a GIF file, exactly 16 pixels by 16 pixels in size.

   c. Click **OK**.

      If graphic file of a wrong size is selected, Orchestration Designer displays an **Invalid icon** message and retains the currently selected icon. Otherwise, Orchestration Designer uses the icon that you select to represent the reusable modules in the Palette pane.

Note:

By default, Orchestration Designer includes two small graphic files: **defaultapplicationsm.gif** and **defaultmodulesm.gif**.

7. Select a large icon for the reusable module, perform the following actions:
   a. Select a graphic file of the large icon. The graphic selected must be a GIF file, exactly 32 pixels by 32 pixels in size.
   b. Click **Browse** next to **Large**.

      The system displays the **Contents of** “icons” dialog box.
   c. Click **OK**.

      If wrong size graphic file is selected, Orchestration Designer displays an **Invalid icon** message and retains the currently selected icon. Otherwise, Orchestration Designer uses the icon that you select to represent the reusable modules in the Palette pane.

Note:

By default, Orchestration Designer includes two large graphic files: **defaultapplicationlg.gif** and **defaultmodulelg.gif**.

Note:

In the current release, Orchestration Designer uses only small icons to represent the reusable modules.

Setting Orchestration Designer project speech properties

About this task

Note:

This tab is displayed only for speech application project properties.

Procedure

1. In the Navigator view or the Avaya OD Navigator view, click the project for which you want to set the speech properties.
2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left navigation pane, click **Orchestration Designer**.
4. Click the **Speech** tab.
5. Set the project speech properties.
6. Click **OK**.

Configuring the default timeout for prompts in a single speech project

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, click the speech project in which you want to set the default timeout for the prompts.
2. On the **Project** menu, click **Properties**.
3. In the Properties for <project name> dialog box, in the left navigation pane, click **Orchestration Designer**.
4. Click the **Speech** tab.
5. In the **Default Prompt Timeout Setting** area, perform one of the following actions:
   - Click **Constant Timeout** and specify the timeout in seconds or milliseconds in the fields that correspond to **Constant Timeout**.
   - Click **Variable Timeout in msec** and specify the variable that contains the timeout in milliseconds in the fields that correspond to **Variable Timeout in msec**.

     The variable must contain only integer values.
6. Click **OK**.
7. In the **Would you like to replace timeout settings in existing prompts with these new defaults** dialog box, perform one of the following actions:
   - Click **Yes** to apply the default timeout to the existing as well as the new prompts that you create in the speech project.
   - Click **No** to apply the default timeout only to the new prompts that you create in the speech project.

Orchestration Designer project properties: Speech tab field descriptions

Note:

This tab is displayed only for speech application project properties.

Use the **Speech** tab to:

- Set options for speech recognition.
- Enable SSML for Text-to-Speech.
- Configure the default timeout settings for prompts in the speech project.

Name	Description
Speech Recognition	
Grammar Compatibility	The type of ASR server engine to target ASR grammars. The options are: • **Desktop Microsoft Speech SAPI**: Grammars are optimized to work with Microsoft Speech API (SAPI). • **IBM**: Grammars are optimized to work with IBM WebSphere or WebSphere Express ASR engines. • **Google Speech**: Grammars are optimized to work with Google ASR. If static grammars are used, you must fill the content of the grammar files with the suffix "google" in the **data/<language>/grammars** folder. • **Loquendo**: Grammars are optimized to work with Loquendo ASR engine. • **Nuance 10.0 and above**: Grammars are optimized to work with Nuance 10.0 and above. • **Nuance 9.0 (Quantum)**: Grammars are optimized to work with Nuance 9.0. • **Nuance OSR**: Grammars are optimized to work with Nuance OpenSpeech Recognition products (including Quantum). • **SRGS-Literals**: (Default) Grammars conform to the SRGS Version 1.0 standard. • **SRGS-SISR**: Grammars are generated in the SRGS format with SISR per the standards and do not take into account any vendor specific nuances. **Note:** Nuance 9.0 (Quantum) and Nuance OSR use the same format. Orchestration Designer, by default, creates grammars to support all ASR engines. The grammars used at run-time is based on what is selected in this field, and ultimately the value of the following variable in the `web.xml` file: *runtime-asr*.

Table continues…

Name	Description
Grammar Caching	The grammar caching option to use. The options are: • **none**: No grammar caching is permitted for this application. • **default**: The application uses the IVR system's default setting for grammar caching.
DTMF Compatibility	DTMF grammar option to use. The options are: • **native**: Generates the DTMF grammars in the ASR engine's Native format as specified in the **Grammar Compatibility**. This also requires that on the Avaya Experience Portal platform you enable the application for remote DTMF processing (under advanced options). The ASR engine then handles DTMF grammar processing. You can use this option for Avaya Experience Portal and MPS platforms only. • **local SRGS-Literals**: This is the default option. Grammars are generated in SRGS literal format and are processed by the Voice Browser. You can use this option for Desktop, IR, and Avaya Experience Portal 4.1 and earlier. • **local SRGS-SISR**: If you select this option, grammars are generated with SISR tags and processed locally by the Voice Browser. You can use this option from Avaya Experience Portal.
Text-to-Speech	
Enable Speech Synthesis Markup Language (SSML) generation in project prompts	Check box to enable SSML markers to work in prompts. To disable SSML markers, clear this check box. If selected, when the Avaya Application Simulator is generating the TTS content, it also generates the SSML data to help with the rendering of that content (even in simulation mode during testing). The Microsoft TTS engine used by Orchestration Designer, however, is not capable of generating or using SSML data, so it is not apparent during testing that it is being generated.
Default Prompt Timeout Setting	

Table continues…

Name	Description
Constant Timeout	The option to specify a default timeout for the prompts in the speech project. In the field that corresponds to **Constant Timeout**, specify the default timeout value for the prompts. In the drop-down list that corresponds to **Constant Timeout**, specify the unit of time, either in **seconds** or **milliseconds**, for the default prompt timeout.
Variable Timeout in msec	The option to specify the variable that contains the default timeout value that you want to use for the prompts in the speech project. In the drop-down list that corresponds to **Variable Timeout in msec**, specify the variable that contains the default timeout value that you want to use for the prompts. If you select a complex variable in the first drop-down list, then in the second drop-down list, select the complex variable field that contains the default timeout value that you want to use for the prompts. **Note:** Ensure that the variable that you select contains only integer values. The system processes the timeout value contained in the variable as milliseconds.

Viewing the channel type of a message flow project

About this task

Note:

The Channel tab is available only for message flow projects.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, click the message flow project for which you view the channel type.
2. On the **Project** menu, click **Properties**.
3. In the **Properties for <*project name*>** dialog box, in the left navigation pane, click **Orchestration Designer**.
4. Click the Channel tab.

   The system displays the channel type of the message flow project in the **Channel Name** field.

Setting Orchestration Designer project language and localization bundles

About this task

Note:

This tab is displayed only for speech application project properties.

Procedure

1. In the Navigator view or the Avaya OD Navigator view, click the project for which you want to set the language and localization bundles.
2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left navigation pane, click **Orchestration Designer**.
4. Click the **Languages** tab.
5. Set the project language and localization bundles.
6. Click **OK**.

Orchestration Designer project properties: Languages tab field descriptions

Note:

This tab is displayed only for speech application project properties.

Use the **Languages** tab to:

- View information about languages and localization bundles installed in Orchestration Designer.
- Add, edit, or delete languages. See About project languages on page 277.
- Select a starting language for the application.
- Install or uninstall localization bundles. See About localization bundles on page 289.
- Install standard phrases. See Installing the standard phrasesets of a localization bundle on page 293.

Name	Description
Project Languages	
Add	To add a project language. For more information, see Adding a project language on page 278.

Table continues…

Name	Description
Edit	To edit a project language. For more information, see Editing a project language on page 281.
Delete	To delete a project language. For more information, see Deleting a project language on page 284.
Starting Language	To change the default or starting language of a project. For more information, see Changing the project default language on page 283.
Localization Bundles	
Install	To install a localization bundle. For more information, see Installing a localization bundle on page 291.
Uninstall	To uninstall a localization bundle. For more information, see Uninstalling a localization bundle on page 291

Enabling Orchestration Designer project pluggable data connectors

About this task

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, click the project for which you want to enable the pluggable data connector.
2. On the **Project** menu, click **Properties**.
3. In the **Properties for** <*project name*> dialog box, in the left navigation pane, click Orchestration Designer.
4. Click the **Pluggable Connectors** tab.
5. In the **Category** field, click the category of the pluggable data connector that you want to enable.

   The lower pane displays the pluggable data connectors that are grouped under the category.
6. Select the check box that corresponds to the pluggable data connector that you want to enable.
7. Click **OK**.

Orchestration Designer project properties: Pluggable Connectors tab field descriptions

The **Pluggable Connectors** tab shows the pluggable data connectors (PDCs) that you can enable for the Orchestration Designer projects.

The following table shows the available PDCs:

Name	
All	Shows all the PDCs that are available in the project.
Connectors	
Generate URL to HTML Application	An outbound application may need to send an email or SMS that contains a URL for the message receiver. The message receiver can click this URL to launch the HTML application. Use the HTML application URL generator PDC to generate a URL to an Orchestration Designer HTML application that is configured on Experience Portal.

Table continues…

Name	
Notification Connector (Email, SMS)	The check box to enable the **Notification Connector (Email, SMS)** PDC to send multichannel SMS message and email message notifications. After you enable the **Notification Connector (Email, SMS)** PDC, the **Send SMS** and **Send Email** items are available in the **Palette** pane of the data node editor. For more information, see Send SMS on page 825 and Send Email on page 822. Use the **Send SMS** and **Send Email** items to send SMS and email message notifications from an application that is created for a channel different from the channel through which you want to send the message notification. For example, send an SMS message notification from a speech application or from an email channel message application. For more information, see Multichannel notifications on page 360. The **Notification Connector (Email, SMS)** PDC sends the SMS message and email message notification to the platform Web service for further processing and delivery. You can also use the **Send SMS** item in an SMS channel message application and the **Send Email** item in an email channel message application. For a speech application, use one or both of the following options to add AVP/AEP Configurable Application Variables PDC to the speech or message project: • **Add cdrstorage to project**: Adds the cdrstorage complex variable to the project. The cdrstorage variable contains the number of characters, stored in the call data record (CDR), that are present in the subjects of an email or in the body of an SMS. • **Add notificationappname to project**: Adds the notificationappname variable to the project. Use this variable to set the application name where you want to send the delivery notifications for the message. If you do not added the notificationappname variable to project or the value of this variable is blank, then the application that originates the delivery notification acts as the destination for the notification event. The **Notification Connector (Email, SMS)** PDC is available in speech projects, message flow projects, and data projects. However, you can define the cdrstorage and notificationappname variables only for speech projects.
Contact Center	

Table continues…

Name	
Avaya IC Connector	For Interaction Center. If the IC connector is selected, also designate whether the application if for VOX or VRUSM. Select either: • Enable IC Connector and Components for VOX • Enable IC Connector and Components for VRUSM (required for SIP) When you first enable IC, Orchestration Designer installs the IC connector software in Tomcat. After that, to get the IC connector to work for the first time, you must restart the Tomcat server engine. For more information about the IC connector, see IC connector on page 622. The **Avaya IC Connector** PDC is available only for speech projects.
Send AACC Connector	
Send AACC Connector	The check box to enable the **Send AACC Connector** PDC to forward an inbound SMS or email message to Avaya Aura® Contact Center for agent assistance. After you enable the **Send AACC Connector** PDC, the **Send to AACC** item is available in the Palette pane of a Data node editor. For more information, see Forward an inbound SMS or email message to Avaya Aura Contact Center on page 364. **Note:** The **Send AACC Connector** PDC is available only in an SMS channel message flow project or an email channel message flow project. Set the following fields in the **Send AACC Connector Configuration** area: • **To**: The email address that is configured for assisted care on Avaya Aura® Contact Center. The message application forwards the inbound SMS or email message to the email address that you specify in the **To** field. • **From**: The email address from where the Orchestration Designer message application must forward the inbound SMS or email message to Avaya Aura® Contact Center.

Table continues…

Name	
AACC Treatments	The check box to enable the **AACC Treatments** PDC and receive the calls that are transferred by Avaya Aura® Contact Center in an Orchestration Designer speech application that is managed on Avaya Experience Portal. **Note:** The **AACC Treatments** PDC is available only in speech projects. After you enable the **AACC Treatments** PDC, you can use the default AACC treatments or create custom AACC treatments to process the call depending on the treatment type that is requested in the Avaya Aura® Contact Center script. After you enable the **AACC Treatments** PDC in an Orchestration Designer speech project, the system creates two variables for the speech project: **aacc_data** and **aacc_context**. **Note:** Enable the **AACC Treatments** PDC in a speech application project only if the speech application is specifically created to handle the calls transferred by Avaya Aura® Contact Center. When you enable the **AACC Treatments** PDC in a speech application, Avaya Experience Portal invokes the ccxml.jsp file, which in turn invokes the speech application. Do not directly invoke a speech application that has the **AACC Treatments** PDC enabled. For more information, see AACC treatments on page 371 and System variable fields and properties on page 563.
Data Access	
Alarm Offset	When you add the **Alarm Offset** PDC to a project, it creates a new Configurable Application Variable *avayaAlarmOffset*. You can set values from 0 through 9 for this variable. A value “0” does not change the behavior of alarms on Avaya Experience Portal. So, a value “0” is equivalent to not enabling the **Alarm Offset** PDC. If *avayaAlarmOffset* is assigned values from 1 through 9, then the alarms on Avaya Experience Portal are offset by this value for a warning, error, and fatal alarm level . For example, if there are 3 alarms on Avaya Experience Portal, QAPP_00001, QAPP_00002 and QAPP_0003, then setting *avayaAlarmOffset* to “3” offsets these alarms to QAPP_00031, QAPP_00032, and QAPP_00033 respectively.
Database	

Table continues…

Name	
Phone Information	Using **Phone Information**, you can check whether a phone number is a mobile number. When you enable the **Phone Information** PDC, it provides a new operation **Is Mobile** that takes “from” and “to” phone numbers as input. **Is Mobile** returns true if the “to” number is a mobile number, else it returns false.
Publish Realtime Oceanalytics Event	Using Orchestration Designer, an application developer can send real time events to Avaya Analytics™. To send events, you must enable the **Publish Realtime Oceanalytics Event** PDC for an application.
AVP/AEP Configurable Application Variables	The check box to enable the Avaya Experience Portal Configurable Application Variables PDC. For more information, see Configurable Complex Variable on page 854. The **AVP/AEP Configurable Application Variables** PDC is available for speech projects, message flow, and data flow projects.
Web Services (Axis 1.4)	The check box to enable the **Web Services (Axis 1.4)** PDC. The **Web Services (Axis 1.4)** PDC is available for speech, message, and data flow projects. For more information, see Web services on page 320.
Web Services (Axis2)	The check box to enable the **Web Services (Axis 2)** PDC. The **Web Services (Axis 1.4)** PDC is available for speech, message and data flow projects. For more information, see Creating an Axis2 Web service operation file on page 326.
Web Services (REST)	The check box to enable the **Web Services (REST)** PDC. After you enable the **Web Services (REST)** PDC, the **Web Service (REST)** item is available in the Palette pane of the **Data** node editor. You can use the **Web Service (REST)** item to invoke a REST Web service operation. **Note:** The **Web Services (REST)** PDC is available for speech , message, and data flow projects.. If you disable the **Web Services (REST)** PDC in an Orchestration Designer project that uses a REST Web service operation, Orchestration Designer shows run-time errors. Orchestration Designer automatically enables the **Web Services (REST)** PDC in a speech project or a message flow project when you create a REST Web service operation file in the speech project or the message flow project. For more information, see Creating a REST Web service operation file on page 334 and Web Service (REST) on page 857.

Table continues…

Name	
Conversations Management	The **Conversations Management** PDC supports simple variables, complex variables, and complex variable fields. A variable can also be a collection of values. The **Conversations Management** PDC is available for Email, SMS, Voice, data, and HTML5–based applications. The variable is saved and restored as a text, string value. The conversation PDC does not support Orchestration Designer variables that contain an object. A conversation can span modules. Each time the module is entered, once or multiple times within a single session, the module is initialized with the saved values that is defined as part of the conversation. On module exit, the defined conversation values are saved. Select the Conversation Management check box to export or import conversation file. The options are: • **Export convox file**: In the **Destination for Convox File** field, click **Browse** to export the `.convox` file. The `.convox` file consists of export information from a project with a `conversation.conversation` file, which has conversation and project variables. Click **Export Convox File** to export the convox file. • **Import convox file**: In the **Convox file for Import** field, click **Browse** to import the `.convox` file. Importing the `.convox` file creates a complex or simple project variable, and adds a reference to the variables in the new project `conversation.conversation` file. Importing the files add fields to pre-existing or new complex variables, if required. Click **Import Convox File** to import the convox file. • **Import conversation file from another project**: In the **Conversation File for Import** field, click **Browse** to import conversation file directly from another project. Ensure that the convox file is in the same workspace. • **No import or export required**: Click this field when you do not require export or import of the convox file.
Telephony	

Table continues…

Name	
AAEP Outbound Call	The check box to enable the **AAEP Outbound Call** PDC to launch an outbound call from a speech application or a message application. After you enable the **AAEP Outbound Call** PDC, the **LaunchVXMLCall** and **LaunchCCXMLCall** items are available in the Palette pane of the data node editor. For more information, see LaunchVXMLCall on page 758 and LaunchCCXMLCall on page 753. Use the **LaunchVXMLCall** item to launch an automated outbound VoiceXML call from a speech application or a message application. Use the **LaunchCCXMLCall** item to launch an automated outbound CCXML call from a speech application or a message application. The **AAEP Outbound Call** PDC is available in speech projects, message flow projects, and data flow projects.
Avaya AES Connector	The check box to enable the Avaya AES PDC for telephony server integration. For more information, see AES connectors on page 610 The **Avaya AES Connector** PDC is available only for speech projects.
Text Processing	
Language ID	The check box to enable the **Language ID** PDC to accept a text string and the language hint and return the ISO-639-1 language code for the text. Language hint is an optional input that is an ISO-639-1 language code that can hint to the expected language of the input text. That is, if you expect English input, specify **en**. If you expect Spanish, specify **es**. The default hint is **en**. After you enable the Language ID PDC, the language ID item is available in the Palette pane of the data node editor. For more information, see LanguageID on page 750. **Note:** The **Language ID** PDC is available only in an SMS channel message flow project or an email channel message flow project.
Workflow	

Table continues…

Name	
Avaya Breeze® platform/ED Integration (Workflows and Context Store)	Orchestration Designer integrates seamlessly with Avaya Breeze® platform and allows an Orchestration Designer application to interact with the Avaya Engagement Designer workflows and pass the collected data in several ways. The workflow PDC facilitates this integration of Orchestration Designer and Engagement Designer. The workflow PDC supports the operations related to workflows and Context Store.

Configuring Orchestration Designer project development, runtime, and Java servlets settings

About this task

Use the Web Descriptor tab to configure various settings that the application needs to run properly on the target platform. The Web Descriptor tab has the following two tabs:

- Application tab: Provides options related to the way the project is to operate in both the development and the run-time environment.
- Servlets tab: Provides options related to naming and behavior of the Java servlets that Orchestration Designer generates.

Procedure

1. In the Navigator view or in the Avaya OD Navigator view, click the project for which you want to set the project development, runtime, and Java servlet settings.
2. On the **Project** menu, click **Properties**.
3. In the Properties for <project name> dialog box, in the left navigation pane, click **Orchestration Designer**.
4. Click the Web Descriptor tab.
5. Click the Application tab and configure the development and run-time environment settings.
6. Click the Servlets tab and configure the settings for Java servlets.
7. Click **OK**.

Application tab field descriptions

Name	Description
General	

Table continues…

Name	Description
Use container default	Check box to use the servlet engine settings, that is, Tomcat, Oracle WebLogic, IBM WebSphere, or JBoss EAP/Wildfly settings, for the session timeout value. If you want to specify your own session timeout value for the application, clear the **Use container default** check box and specify the session timeout value in the **Session timeout** field.
Session timeout	The number of minutes a session can remain inactive on an IVR system or on a text processing system before the system times out and terminates the session. Valid values for this field are integers. A value of 0 (zero) or less means that the session never times out. **Note:** The **Session timeout** field is unavailable if you select the **Use container default** check box.
Parameters	
Add	To add a context parameter. For more information, see Adding a context parameter on page 513.
Edit	To edit a context parameter. For more information, see Editing a context parameter on page 513.
Delete	To delete a context parameter. For more information, see Deleting a context parameter on page 514.

Servlets tab

On the **Servlets** tab, you can manually edit various parameters and settings for the Java servlets that deployment includes. In almost all cases, you do not need to make changes to these settings.

Caution:

Avaya recommends strongly that you do not change these settings except in rare cases, and if you are sure of what you are doing.

Context parameter management

Adding a context parameter

Procedure

1. In the Navigator view or the Avaya OD Navigator view, click the project for which you want to add a context parameter.
2. On the **Project** menu, click **Properties**.

   The system displays the **Properties for** <*project name*> dialog box.
3. In the left navigation pane, click Orchestration Designer.
4. Click the **Web Descriptor** tab.
5. Click the **Application** tab.
6. Click **Add**.
7. In the **Add context parameter** dialog box, perform the following actions:
   a. In the **Name** field, type a name for the context parameter.
   b. In the **Value** field, type a value for the context parameter.
   c. In the **Description** field, type a description for the context parameter.
   d. Click **OK**.

Editing a context parameter

Procedure

1. In the Navigator view or the Avaya OD Navigator view, click the project in which you want to edit a context parameter.
2. On the Project menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left navigation pane, click Orchestration Designer.
4. Click the **Web Descriptor** tab.
5. Click the **Application** tab.
6. In the Parameters area, click the parameter that you want to edit, and then click **Edit**.
7. In the Edit context parameter dialog box, perform the following actions:

   Important:

   Do not edit the name of the context parameter.
   a. In the **Value** field, edit the value for the context parameter.
   b. In the **Description** field, edit the description for the context parameter.

c. Click **OK**.

Deleting a context parameter

Procedure

1. In the Navigator view or the Avaya OD Navigator view, click the project from which you want to delete a context parameter.
2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left navigation pane, click Orchestration Designer.
4. Click the **Web Descriptor** tab.
5. Click the **Application** tab.
6. In the Parameters area, click the parameter that you want to delete, and then click **Delete**.

Context parameters for an Orchestration Designer project

Parameter	Description
sage.startlanguage	The **Value** column displays the language that is set as the starting language for the application. Avaya recommends you do not change the starting language. The default setting for this field is the project language that was selected and set when the project was first created.

Table continues…

Parameter	Description
runtime-ASR	If the application uses an ASR server, enter the type of ASR server engine that the application is to use. This also dictates the type of grammars to be used. Valid values are: • **Desktop Microsoft Speech SAPI** • **Loquendo** • **SRGS-Literals**: (Default) Deployment optimizes the application grammars for standard SRGS usage. This means that any ASR server that supports SRGS grammars should be able to work with this application. • **SRGS-SISR**: Grammars are generated in the SRGS format with SISR per the standards and do not take into account any vendor specific nuances. • **Nuance 9.0 (Quantum)**: Deployment optimizes the application grammars to work with Nuance 9.0 ASR servers. • **Nuance 10.0 and above**: Deployment optimizes the application grammars to work with Nuance 10.0 and above ASR servers. • **Nuance OSR**: Deployment optimizes the application grammar to work with Nuance OSR ASR servers (includes Quantum support). • **Google Speech**: Framework runtime will assign URLs that point to the grammar files for Google ASR or directive to execute code to generate dynamic grammar for Google ASR during VXML generation. • **IBM**: Deployment optimizes the application ASR to work with IBM WebSphere Voice servers.
runtime-Platform	Enter in this **Value** field, the type of the IVR platform the application is targeted to run on. Valid values include: • Avaya Experience Portal: Deployment optimizes the application to run on an Avaya Experience Portal system. • **IR**: Deployment optimizes the application to run on an Avaya IR system. • **Desktop**: (Default) Deployment optimizes the application to run on one of the supported operating system desktops. This option is intended primarily for development. • **Other**: Deployment optimizes the application to run on other systems that support the W3C VoiceXML 2.0 Recommendation. • **MPS**: Deployment optimizes the application to run on the MPS system.

Table continues…

Parameter	Description
grammar-caching	Enter in this **Value** field, the setting that determines how the application handles grammar caching. Valid values include: • **none**: No grammar caching is permitted for this application. • **default**: The application uses the IVR system default setting for grammar caching.
runtime-SSML	Enter in this **Value** field one of the following: • **true**: Enables SSML generation for run time. • **false**: (Default) Disables SSML generation for run time.
sage.ic.throwexceptions	Allows exceptions to be disabled when generated in an IC operation so they can be analyzed in the data node. Possible values are: • **true**: (Default) “IC throw” run-time exception is thrown in case of error. • **false**: No exception is thrown.
runtime-VXML-Compatibility	This specifies the VXML compatibility of the project. Possible values are: • **2.1**: Allows VXML 2.1 extensions. • **2.0**: Uses VXML 2.0 only.
runtime-DTMF-Native	Allows DTMF generation in the ASR native format. Possible values are: • **true**: DTMF is generated in ASR native format. • **false**: DTMF is not generated in ASR native format. In conjunction with this setting, configure the application for remote DTMF processing. (Effectively, DTMF grammars are sent to the ASR engine instead of being process locally by the Avaya Voice Browser).
sage.autoinvoke.callinfo	Indicates whether information should be gathered automatically or not when the application starts running (through Auto Invoke CallInfo). Possible values are: • **true**: (Default) information is gathered automatically. • **false**: If set to false, you must use the info palette item in your data node before doing any other operations. You should set the property Initial Call to true when using this the first time to synch up with the new call.

Table continues…

Parameter	Description
module.<*module name*>.entry point	The first time you edit the module entry point, add the **module.<*module name*>.entry point** parameter. In the **Value** field, specify the new URL where the external module resides. You can use this parameter to edit the entry point of an Orchestration Designer reusable module that you have imported in the Orchestration Designer speech project. For more information about importing an external module, see Importing a VXML subdialog on page 70.

Tomcat properties management

Tomcat properties

Verify the **Tomcat** property settings carefully, especially after updating to a new version of Orchestration Designer. If you are having trouble updating applications when you update Orchestration Designer, check these settings.

The **Tomcat** properties pane is created by a third-party plug-in. This pane is integrated with Orchestration Designer. When you create a project, Orchestration Designer automatically and properly configures the project's Tomcat properties. Orchestration Designer does not, however, continue to check or update these properties as you continue to work on and develop the project.

This section is provided primarily as reference, against the eventuality that these settings somehow get changed or corrupted and the project stops running correctly in the simulator.

Caution:

To prevent "breaking" your project and causing it to no longer run correctly in the simulator, Avaya recommends that you do not change any of the settings in this pane, unless the settings have been inadvertently changed by some other means. If you find it necessary to change them, you should change them only to match the settings in Tomcat properties field descriptions on page 518.

The **Tomcat** properties pane is where **project properties** are available for how Tomcat is set to work with Orchestration Designer projects. There is one tab, the General tab, used to:

- Verify that the project is a Tomcat project.
- Make general settings for how Tomcat is to work with the Orchestration Designer project.

Setting Tomcat properties

About this task

Caution:

To prevent “breaking” your project and causing it to no longer run correctly in the simulator, Avaya recommends that you do not change any of the settings on this tab, unless the settings have been inadvertently changed by some other means. If you find it necessary to change them, you should change them only to match the settings in Tomcat properties field descriptions on page 518.

Procedure

1. In the Navigator view or the Avaya OD Navigator view, click the project for which you want to set the Tomcat properties.
2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left navigation pane, click **Tomcat**.
4. On the **General** tab, set the Tomcat properties.
5. Click **Apply**, and then click **OK**.

Tomcat properties field descriptions

Name	Description
Is a Tomcat Project	Check box to ensure that all Orchestration Designer projects to work properly. Be sure that this check box is selected.
Context name	Displays and consists of only a slash (/) followed by the project name. For example, if your project name is **MyProject**, this field displays: **/MyProject**
Can update server .xml file	Check box to indicate that Tomcat can update the server .xml file. Be sure that this check box is selected.
Mark this context as reloadable	Check box to ensure that the context is reloadable. Be sure that this check box is selected.
Redirect context logger to Eclipse console	Check box to redirect the context logger to the Eclipse console. Be sure that this check box is cleared.
Extra information	Optionally, use this field to enter notes or comments regarding the use of Tomcat with this project.
Subdirectory to use as web application root (optional)	Do not use this field. Its only content should be a slash (/).

Chapter 34: Preference settings configuration

Preference settings

Orchestration Designer includes a number of preference panels with settings that you can configure uniquely for your Orchestration Designer development environment. If you do not adjust these settings, Orchestration Designer uses the pre-set default settings.

For more information about other Eclipse preferences, see the Eclipse documentation:

http://www.eclipse.org/documentation/

Simulation preferences configuration

Simulation preferences include all preferences related to application simulation in Orchestration Designer.

Configuring Application Simulation preferences

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the **Preferences** dialog box.

2. In the left navigation pane, double-click **Avaya**.
3. Click **Application Simulation**.
4. In the **Application Simulation** pane, configure the application simulation preferences. For more information, see Application Simulation preferences field descriptions on page 520.
5. Click **Apply**, and then click **OK**.

   Additional Application Simulation preferences are split into two areas: settings related to Orchestration Designer simulations, such as AVB settings and VOX/VRUSM simulators, and simulation profiles. See the following sections:

   - Configuring Orchestration Designer Simulation preferences on page 520

Application Simulation preferences field descriptions

Name	Description
Automatically restart Tomcat when running an application.	Though not recommended, when selected, this option restarts Tomcat before making a simulation call.
Tomcat port	Indicates the port that Tomcat will listen on. 8080 is the default. Update this field if you have configured Tomcat to use a different port with Orchestration Designer.
Default Web Browser	For web applications, change the browser you want the Simulator to use to launch the application. For example, you can set "`C:\Program Files (x86)\Mozilla Firefox\firefox.exe`" to change the browser from system web browser to Mozilla Firefox.

Configuring Orchestration Designer Simulation preferences

About this task

The Orchestration Designer Simulation preferences control the way the Avaya Application Simulator (AAS) functions with Orchestration Designer applications during simulations. Usually, these settings do not need to be changed.

Procedure

1. On the **Window** menu, click **Preferences**.
2. In the **Preferences** dialog box, in the left navigation pane, double-click **Avaya** > **Application Simulation**.
3. Click **Orchestration Designer Simulation**.
4. In the **Orchestration Designer Simulation** pane, in the **Simulation Properties** area, configure the Orchestration Designer simulation preferences.
5. Click **Apply**, and then click **OK**.

Configuring MRCP settings

About this task

MRCP settings control the speech server that you want to use for simulation. You can choose either a TTS server or an ASR server or both. If you do not select any server, then by default, the Avaya Application Simulator uses Microsoft SAPI Speech for TTS and ASR functions.

Note:

Currently, the MRCP feature supports only Nuance and Loquendo speech servers. Currently, Orchestration Designer supports only MRCP version 1, and hence, MRCP version 1 is selected by default.

Procedure

1. On the **Window** menu, click **Preferences**.
2. In the **Preferences** dialog box, in the left navigation pane, double-click **Avaya** > **Application Simulation**.
3. Click **Orchestration Designer Simulation**.
4. In the **Orchestration Designer Simulation** pane, in the **MRCP Settings** area, perform the following actions:
   a. Click the **ASR** tab to specify the connection settings of the ASR server that you want to use for simulation.
   b. Click the **TTS** tab to specify the connection settings of the TTS server that you want to use for simulation.
5. Click **Apply**, and then click **OK**.

Orchestration Designer Simulation preferences field descriptions

Name	Description
Simulation Properties	
Enable logging of tracing output	When selected, enables framework tracing, which logs framework debug tracing to the trace.log file and to the console view.
Enable Application logging of tracing output	When selected, logs any of the user-defined tracing statements in the callflow to the trace.log file and to the console view.
Enable display of generated XML	When selected, logs the XML output that is generated by the application to the trace.log file and to the console view.

Table continues…

Name	Description
Enable display of AppRoot XML	Option to specify how often the system should display the XML output generated for the AppRoot in the trace.log file. The options are **Once**, **Always**, and **Never**. The default is **Once**.
Enable Application Framework reporting	When selected, logs “breadcrumb” items to the report log (or for VoicePortal to the application report). **Note:** This option is not currently supported in Orchestration Designer.
Enable display of the Avaya Voice Browser output in a console window	When selected, opens a MSDOS console window which shows all of the low level output of the Voice Browser.
Automatically hangup Avaya Voice Browser when call ends	When selected, a simulation Avaya Voice Browser process is terminated when a simulated call ends (disconnect or application exit).
Use external configuration file	Though not recommended, when selected, allows further tuning of the personal Voice Browser.
Show AutoVon keys	When selected, adds four more DTMF buttons next to the dial keypad for sending A, B, C, and D AutoVon tones for input.
Enable Speech Bargein detection	Avaya Application Simulator does barge-in detection when this is selected. This means when the browser or ASR engine detects speech input, it barges in (interrupts) the prompt and tries to match the speech with an item in the active grammars.
MRCP Settings	
ASR	Check box to use an ASR server with the MRCP feature.
Network Address	The IP address of the ASR server.
Base Port	The port number of the ASR server.
Engine Type	The engine type of the ASR server. Currently, MRCP feature supports only Nuance and Loquendo speech servers.
MRCP Version 1	The default MRCP version number. Currently, Orchestration Designer supports only MRCP version 1.
TTS	
Enable MRCP TTS	Check box to use a TTS server with the MRCP feature.
Network Address	The IP address of the TTS server.

Table continues…

Name	Description
Base Port	The port number of the TTS server.
Engine Type	The engine type of the TTS server. Currently, MRCP feature supports only Nuance and Loquendo speech servers.
MRCP Version 1	The default MRCP version number. Currently, Orchestration Designer supports only MRCP version 1.

Note:

For the Dialog Designer 5.x release and for Orchestration Designer 7.0.1 release, the Media Preview view opens automatically when previewing a media page or when you run an application with media content with the simulator.

Configuring AVB settings

About this task

Avaya Voice Browser settings control how the voice browser handles fetches, the maximum number of documents that can be loaded at a given time, and tracing and application logging settings for various components.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Avaya** > **Application Simulation** > **Orchestration Designer Simulation**.
3. Click **AVB Settings**.
4. In the AVB Settings pane, configure the AVB settings.
5. Click **Apply**, and then click **OK**.

AVB Settings field descriptions

Name	Description
VoiceXML Properties	

Table continues…

Name	Description
Fetch Timeout	The default timeout to use when fetching resources from the application server. If this time elapses with no response from the application server, the AVB returns a Bad Fetch error.
Voice Browser Properties	
Maximum Documents	The maximum number of documents that can be loaded simultaneously at any given moment. This limitation is to prevent a recurrence to run unbound.
Tracing and Logging	Configure tracing and application logging settings for the different component areas of the Avaya Voice Browser. Settings can be configured independently for all the following component areas: • Client • INET • Interpreter • Java Script Interface • Object • Platform • Prompt • Recognition • Telephony
Tracing	The greater the degree of tracing, the larger and more detailed the trace output is. A setting of Finest for some of the fields can generate huge amounts of trace logging output. Hence, Avaya recommends that you use only the degree of tracing required to obtain the data you need. Try setting the level to Fine first and increase it only if you do not get the data you need.

Table continues…

Name	Description
Application Logging	Determines what messages will be logged by the application. On a live system, these logs are sent back to the platform. During simulation these logs show up in the VXML Log tab. The log levels can be: • None: All application log messages are ignored. • Fatal: Only fatal level messages are displayed. • Error: Fatal and error level messages are displayed. • **Warning:** Fatal, error, and warning level messages are displayed. • Info: All messages are displayed.

Configuring VOX/VRUSM simulators

About this task

The VOX/VRUSM Preferences panel provides options to configure a number of VOX servers or VRUSM simulators. A VOX/VRUSM can operate in one of the two modes:

- Initiate the connection to the IC connector
- Allow the IC connector to connect to it

The IC Connector on the local computer automatically collects these settings. If you for example configure one VOX simulator to initiate a connection to the IC Connector on port 3000, then when you start the local Tomcat server up its IC Connector will automatically come up and listen for a VOX connection on port 3000. These settings will be in effect on the local computers ICC next time that Tomcat is started.

Note that by setting the host, it is also possible to connect to a real VOX server while your application is run in the Avaya Applications Simulator.

Note:

Only for the VOX which has host set to localhost, an instance of the VOX simulator will start. These settings are in effect on the local computer’s IC connector the next time you start Tomcat.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.

2. In the left navigation pane, double-click **Avaya** > **Application Simulation** > **Orchestration Designer Simulation**.

3. Click **IC Simulators**.
4. In the IC Simulators pane, click the **VOX** tab.
5. Click any VOX server settings to edit them. For Mode, a drop down option is offered, to select either **IC Connector** > **VOX Simulator** , or **VOX Simulator** > **IC Connector**.

   By setting the host, it is also possible to connect to a real VOX server while an application is run in the Avaya Applications Simulator.

   Note:

   Similarly, you can configure a VRUSM/HTTPVOX simulator.

Adding a VOX simulator

Procedure

1. On the Window menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. Double-click **Avaya** > **Application Simulation** > **Orchestration Designer Simulation**.
3. Click **IC Simulators**.
4. In the IC Simulators pane, click the **VOX** tab.
5. Click **Add**.

   A VOX server instance is added with default settings.
6. Click again to add another, though a warning message also appears stating that “All VOXs must have a unique name.”
7. Click the name of the second VOX server added to edit the name.

   Note:

   Similarly, you can add a VRUSM/HTTPVOX server instance.

   Note:

   To delete a VOX simulator, click a field of an existing VOX server instance, then click **Delete**. The VOX server instance is deleted. Similarly, you can delete a VRUSM/HTTPVOX instance.

VOX/VRUSM Simulators preference settings field descriptions

Name	Description
VOX Properties	

Table continues…

Name	Description
Name	Name of the VOX simulator system.
Host	Host name of the VOX simulator system, or localhost if on the same computer as Orchestration Designer.
Port	Port on which it communicates with the IC connector (by default, 3000)
Extensions	Extensions to be called.
Mode	Select either of the following: • VOX Simulator > IC Connector • IC Connector > VOX Simulator
VRUSM Properties	
Name	Name of the VRUSM simulator system.
Host	Host name of the VRUSM simulator system, or localhost if on the same computer as Orchestration Designer.
Port	Port on which it communicates with the IC connector (by default, 3000)

Simulation profile management

The Simulation Preferences area allows you to define simulation profiles. Profiles include calling parameters, Orchestration Designer run-time session parameters, Orchestration Designer scripts, and administrative variable definitions.

Adding and configuring a simulation profile

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the **Preferences** dialog box.

2. In the left navigation pane, double-click **Avaya** > **Application Simulation**.
3. Click **Simulation Profiles**.
4. In the **Simulation Profiles** pane, click **New**.

   The system displays the **New Profile** dialog box.

5. In the **Enter the new profile name** field, type a name for the simulation profile, and then click **OK**.

   The system displays the **Simulation profile <simulation profile name>** dialog box.

6. In the **Simulation profile <simulation profile name>** dialog box, perform the following actions:
   a. On the Call tab, specify the communication numbers that you want to use for the simulation. For more information, see Simulation Profile: Call tab field descriptions on page 530.
   b. On the OD Runtime Session tab, configure the parameters to simulate different settings or configurations in VXML and CCXML applications. For more information, see Simulation Profile: OD Runtime Session tab field description on page 531.
   c. On the OD Scripts tab, configure the settings to enable and import connector and application simulation file scripts. For more information, see Simulation Profile: OD Scripts tab field description on page 534.
   d. On the Message tab, specify the simulation profile details to simulate SMS or email messages for message applications. For more information, see Simulation Profile: Message tab field descriptions on page 536.
   e. On the Variable Administration tab, set the administrative variable definitions. For more information, see Simulation Profile: Variable Administration tab field description on page 538.
   f. Click **OK**.
7. In the **Preferences** dialog box, click **OK**.

Editing a simulation profile

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Avaya** > **Application Simulation**.
3. Click **Simulation Profiles**.
4. In the Simulation Profiles pane, in the **Profiles Available** field, click the simulation profile that you want to edit, and then click **Edit**.

   The system displays the Simulation profile <simulation profile name> dialog box.
5. In the Simulation profile <simulation profile name> dialog box, perform the following actions:
   a. On the Call tab, edit the communication numbers that you want to use for the simulation. For more information, see Simulation Profile: Call tab field descriptions on page 530.
   b. On the OD Runtime Session tab, edit the parameters to simulate different settings or configurations in VXML and CCXML applications. For more information, see Simulation Profile: OD Runtime Session tab field description on page 531.
   c. On the OD Scripts tab, edit the settings to enable and import connector and application simulation file scripts. For more information, see Simulation Profile: OD Scripts tab field description on page 534.

d. On the Variable Administration tab, edit the administrative variable definitions. For more information, see Simulation Profile: Variable Administration tab field description on page 538.

e. Click **OK**.

6. In the Preferences dialog box, click **OK**.

Importing a simulation profile

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the **Preferences** dialog box.

2. In the left navigation pane, double-click **Avaya** > **Application Simulation**.
3. Click **Simulation Profiles**.
4. In the **Simulation Profiles** pane, click **Import**.

   The system displays the **Import Avaya Simulation Preferences** dialog box.

5. Select the simulation profile XML file to import, and then click **Open**.
6. In the **Preferences** dialog box, click **Apply**, and then click **OK**.

   To view the expected format, create a dummy profile, and then export it. Edit the profile as desired, and import the profile again.

Exporting a simulation profile

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.

2. In the left navigation pane, double-click **Avaya** > **Application Simulation**.
3. Click **Simulation Profiles**.
4. In the Simulation Profiles pane, in the **Profiles Available** field, click the simulation profile that you want to export.
5. Click **Export** to export the simulation profile to an XML file.

   The system displays the Export Avaya Simulation Preferences dialog box.

6. Select the location where you want to export the simulation profile, and then click **Save**.
7. In the Preferences dialog box, click **Apply**, and then click **OK**.

Removing a simulation profile

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the **Preferences** dialog box.

2. In the left navigation pane, double-click **Avaya** > **Application Simulation**.
3. Click **Simulation Profiles**.
4. In the **Simulation Profiles** pane, in the **Profiles Available** field, click the simulation profile that you want to remove, and then click **Remove**.

Simulation Profile: Call tab field descriptions

In the Call tab, specify the communication numbers to use in the simulation.

Name	Description
Calling Number	Simulates ANI. Enter the number that represents the originating telephone number in this field. Use numbers only. **Note:** If you use characters other than numbers, the Avaya Application Simulator passes them along exactly as you enter them in this field. However, in a live system with a deployed application, only numbers are passed to the application. For this reason, Avaya recommends that you use only numbers in this field during simulations.
Called Number	Simulates DNIS. Enter the number that represents the telephone number the caller must dial to get to this application. Use numbers only. **Note:** If you use characters other than numbers, the Avaya Application Simulator passes them along exactly as you enter them in this field. However, in a live system with a deployed application, only numbers are passed to the application. For this reason, Avaya recommends that you use only numbers in this field during simulations.

Table continues…

Name	Description
Converse On Data	Lets you simulate data that you can receive from the switch. **Note:** The application must be specifically designed to collect converse on values. For setting up converse on data, on the actual runtime platform requires configuration on both the switch and vectors as well as the Interactive Response or Avaya Experience Portal. For more information about working with converse on data, • For Avaya Experience Portal, see the "Symptoms of common MPP problems" section of *Troubleshooting Avaya Experience Portal Guide*. • For Interactive Response, see the "Miscellaneous instructions" section of *Application Development Guide* and "ASAI versus the converse vector step" section of *Specialized applications Guide*.

Simulation Profile: OD Runtime Session tab field description

The OD RunTime Session tab allows you to configure parameters to simulate different settings or configurations in VXML and CCXML applications. It is like you are launching the application through a VXML or CCXML Web service on the Avaya Experience Portal.

Name	Description
Call Control Protocol Parameters	Allows protocol specific parameters to be set, and passed when simulating your application. Entered parameters must be the following format: **session.connection[<protocol_name>].*** Enter a protocol name and version. Then enter a parameter name and value, and click **Add Param**. In a real deployment, these values can otherwise be set automatically by the supporting platform (for example, Avaya Experience Portal platform).

Table continues…

Name	Description
Call Control Session Parameters	Allows session-based parameters to be set, for a VXML session to simulate launching the CCXML Web services. Entered parameters must be the following format: **session.*** Enter a protocol name and version. Then enter a parameter name and value, and click **Add Param**. In a real deployment, these values can otherwise be set automatically by the supporting platform (for example, the Avaya Experience Portal platform).

Table continues…

Name	Description
AAI/UUI Options	Allows Avaya Experience Portal specific UUI/AAI parameters to be set. There are two modes: • Shared: Means that the format of the configured value is fixed, and as specified by Avaya Experience Portal. When you set this parameter to Shared, the system fetches the UUI/AAI ID and value pairs and stores them in the **shareduui** variable. For more information, see the **shareduui** variable in System variable fields and properties on page 563. • Service Provider: Means that the value is uninterrupted. If the mode is shared, the following considerations apply: • VXML: session.connection.aai is unpacked into the shareduui variable; session.avaya.uui.mode stored in session:sharedmode; if there is a UCID in the AAI, it is decoded into session.avaya.ucid and session:sessionlabel is set to the UCID. • CCXML: connection.aai is unpacked into connection.avaya.uui.shared; mode is in connection.avaya.uui.mode; if there is a UCID in the AAI, it is decoded into connection.avaya.ucid An example of the AAI is: **12, 01DR12345678; FA, 0001001C46E70028** where 12 and FA is the application ID, and the remaining characters in the string are the values, respectively. For more information, see Transferring UUI data in a SIP configuration on page 630. To set the Avaya Experience Portal specific UUI/AAI parameters for an actual running application, see the “User-to-User Interface (UUI) data passed in SIP headers” section in the Avaya Experience Portal documentation. For more information about UUI data passed in SIP headers, see the “SIP application support” section of *Administering Avaya Experience Portal Guide*.

Table continues…

Name	Description
Call Classification	These parameters allow you to simulate different types of calls at the voice platform in CCXML. The types of calls are: • live_voice: person answered • fax_calling_tone: sending tone, inbound call from a fax machine • timeout: timeout on classification expired • error The call classifications are passed to an application using the “call classification” query string variable to the Web service.
Speech Application Parameters	These parameters, when set to values in a VXML session take the parameters in the namelist. Entered parameters must be the following format: **session.*** Enter a protocol name and version. Then enter a parameter name and value, and click **Add Param**. The parameter is added to the table list, and passed to the simulated application. It allows you to test your application as though it was launched in CCXML. In a real deployment, these values can otherwise be set automatically by the supporting platform. For example, IR or the Avaya Experience Portal platform.

Simulation Profile: OD Scripts tab field description

This tab provides options to enable and import connector and application simulation file scripts. The following table lists sample scripts as examples:

Name	Description
Connector Scripts	
Enable connector simulation file processing	Check box to enable connector simulation processing for the designated file.
Simulation file name	Directory path of the connector simulation file script.
Application Scripts	
Enable Application simulation file processing	Check box to enable application simulation processing for the designated file.
Simulation file name	Directory path of the application simulation file script.

Example simulation scripts

Example script for call control and speech input

As an alternative to simulating caller responses and other inputs during simulation, create an XML script to simulate them automatically running a simulation. Following are some general command guidelines and examples of scripts to simulate caller response.

Following is a list of all the valid commands used in the Call Control and Speech Input scripts. Note that the “value” attribute for the rec asr, rec dtmf, record err, and record dtmf will vary. For a rec type asr, you can have one or more items. Using multiple items allows you to simulate n-best results.

```
<?xml version=“1.0” encoding =“UTF-8”?>
  <callscript>
  <call>
       <command name=“rec” type=“asr”>
           <item value=“dog blue”/>
       </command>
       <command name=“rec” type=“asr”>
           <item value=“dog blue” confidence=“.7”/>
       </command>
       <command name=“rec” type=“asr”>
           <item value=“dog blue”/>
              <item value=“fog blue”/>
       </command>
       <command name=“rec” type=“asr”>
           <item value=“dog blue” conficence=“.7”/>
           <item value=“fog blue” conficence=“.4”/>
       </command>
       <command name=“rec” type=“dtmf” value=“1” />
       <command name=“rec” type=“err” value=“nomatch” />
       <command name=“rec” type=“err” value=“noinput” />
       <command name=“tel” type=“hangup” delay=“10”/>
       <command name=“tel” type=“answered”/>
       <command name=“tel” type=“answered” value=“live_voice”/>
       <command name=“tel” type=“progress” value=“some reason value”/>
       <command name=“tel” type=“busy”/>
       <command name=“tel” type=“noanswer”/>
       <command name=“record” type=“max”/>
       <command name=“record” type=“done”/>
       <command name=“record” type=“err” value=“45” />
       <command name=“record” type=“dtmf” value=“1” />
       <command name=“quit” type="“ value=”“ />
       <command name=”merge“ type=”“ value=”ok“ />
       <command name=”merge“ type=”“ value=”fail“ />
  </call>
  </callscript>”
```

Example script for handling a transfer, far end hangs up after 10 seconds

```
<?xml version=1.0“ encoding =”UTF-8“?>
  <callscript>
  <call>
  </call>
  <call>
       <command name=”tel“ type=”answered“ value=”live_voice“/>
       <command name=”tel“ type=”hangup“ delay=”10“/>
  </call>
  </callscript>”
```

Example script for handling a transfer, near end hangs up after 10 seconds

```
<?xml version=1.0" encoding =”UTF-8"?>
  <callscript>
  <call>
       <command name=”tel" type=”hangup delay="10”/>
  </call>
  <call>
       <command name="tel” type="answered”/>
  </call>
  </callscript>
```

Example script for handling a transfer, hot word recognition

```
<?xml version="1.0” encoding ="UTF-8”?>
  <callscript>
  <call>
       <command name="rec” type="asr”>
           <item value="dog blue”/>
       </command>
  </call>
  <call>
       <command name="tel” type="answered”/>
  </call>
  </callscript>
```

Example script for merging two calls

```
<?xml version="1.0” encoding="UTF-8”?>
  <callscript>
     <call>
         <command name="merge” value="okay”/>
      </call>
     <call>
         <command name="tel” type="answered”/>
         <command name="merge” value="okay”/>
      </call>
  </callscript>
```

Note that any command can have a delay attribute. This attribute specifies the time taken to activate a command after the processing of the previous command. In the far end disconnect example, the hangup is sent 10 seconds after the call is answered. In the near end disconnect the hangup is sent 10 seconds after the initial incoming call. Note that the first call in the script never has an answer that is implicit. If you do answer the incoming call, an error will result.

Simulation Profile: Message tab field descriptions

Use the Message tab to create a simulation profile for an SMS channel message application or an email channel message application.

AAS simulates the SMS or email message contained in the simulation profile as an inbound message.

Note:

The current release of Orchestration Designer does not support attachments in inbound email messages.

Field	Description to create a simulation profile for an SMS channel message application	Description to create a simulation profile for an email channel message application
Message Headers		
Header Name	The name of the SMS message header.	The name of the email message header.
Header Value	The value of the SMS message header.	The value of the email message header.
Add Header	Option to add the header name and header value that you specify in the **Header Name** and **Header Value** fields, to the simulation profile.	Option to add the header name and header value that you specify in the **Header Name** and **Header Value** fields, to the simulation profile.
Delete Selected	Option to delete a header name and the header value from the simulation profile.	Option to delete a header name and the header value from the simulation profile.
To	The phone number of the receiver of the SMS message. **Note:** Use a semicolon (;) to separate multiple phone numbers. This field is mandatory.	The email address of the receiver of the email message. **Note:** Use a semicolon (;) to separate multiple email addresses. This field is mandatory.
From	The phone number of the sender of the SMS message. **Note:** This field is mandatory.	The email address of the sender of the email message. **Note:** This field is mandatory.
CC	—	The email address of the receiver to whom you want to send a copy of the email message. **Note:** Use a semicolon (;) to separate multiple email addresses.
BCC	—	The email address of the receiver to whom you want to send a blind copy of the email message. **Note:** Use a semicolon (;) to separate multiple email addresses.
Subject	—	The subject line of email message.
Body	The message body of the SMS message.	The message body of the email message.

Simulation Profile: Variable Administration tab field description

Setting	Description
Name	Name of the configurable variable.
Value	Value for the configurable variable used for simulation.
Param Name	Configurable name input. If you click the **Add Variable** button, the variable gets added to the table.
Param Value	Configurable variable value input that pairs the variable name.
Import Variables	This button launches the **project selector** dialog box. The project selector dialog box allows you to choose the project to import all its configurable variables into a table instead of typing them in one-by-one.

Orchestration Designer preferences management

Considerations for enabling an HTTP or HTTPS proxy connection

The Orchestration Designer Preferences panel includes a setting to enable an HTTP or HTTPS proxy connection.

Proxy settings are required when all of the following conditions are true:

- The system where Orchestration Designer is installed is behind a firewall.
- Access is required to resources that reside outside the firewall for your Orchestration Designer speech projects. These resources can include Web services, databases, or other outside resources.
- Access to these resources requires the use of either an HTTP or HTTPS proxy server.

When these conditions are true, proxy settings for Orchestration Designer must be configured, even if proxy settings are already configured for your Internet browser or email client. If you have a proxy server configured for your Internet browser, use the same proxy settings for Orchestration Designer. For more information, see *Admin (ddadmin) Web application configuration* in the *Avaya Orchestration Designer Developer's Guide*.

Enabling an HTTP or HTTPS proxy connection

Procedure

1. On the Eclipse user interface, click **Window > Preferences**.

   The Eclipse user interface displays the Preferences dialog box.

2. In the navigation pane, double-click **Avaya**.
3. Click **Orchestration Designer**.
4. In the Orchestration Designer pane, do one of the following:
   - In the **Proxy Settings** area, configure the HTTP proxy settings.
   - In the **HTTPS Proxy Settings** area, configure the HTTPS proxy settings.
5. Click **Apply**.
6. Click **Apply and Close**.

Configuring a run-time license server

About this task

You can configure a run-time license server for your application. If you run the applications through Application Simulator, you do not need the run-time license.

Note:

Specify a run-time license server only if Avaya Experience Portal, IR, or MPS accesses your application from the development environment.

Procedure

1. On the Eclipse user interface, click **Window > Preferences**.

   The Eclipse user interface displays the Preferences dialog box.
2. In the navigation pane, double-click **Avaya**.
3. Click **Orchestration Designer**.
4. In the **Runtime License Server** area, in the **Server URI** field, enter the URI of the run-time license server.

   The format of this URI is `http://webServerName:port`, where:

   - **webServerName** is the fully qualified host name or IP address of your WebLM license server.
   - **port** is the number of the HTTP/HTTPS port that the system uses to access the license server.

   For example, `http://licenseServer.myCompany.com:8080`.
5. In the **License Check Timeout** field, enter the time in seconds.

   The system must wait for a response from the WebLM license server for the specified time while attempting to connect to the WebLM license server.

   The default value is zero seconds. Zero indicates that there is no timeout.
6. Click **Apply**.

7. Click **Apply and Close**.

Removing the context files on closing a project

About this task

Tomcat opens the context files of all projects each time you simulate a project. Therefore, Orchestration Designer performance can degrade if you have several workspaces with a large number of projects.

Orchestration Designer recreates the context file when a project is reopened. This improves the performance by controlling the size of the workspaces.

Note:

This option does not affect the projects that are not opened in the current session. This applies only to Orchestration Designer projects that you open and close subsequently.

Procedure

1. On the Eclipse user interface, click **Window > Preferences**.

   The Eclipse user interface displays the Preferences dialog box.
2. In the navigation pane, double-click **Avaya**.
3. Click **Orchestration Designer**.
4. In the Orchestration Designer pane, in the **Context Files** area, select the **Remove context files on project close** check box.

   Orchestration Designer automatically deletes the corresponding context files when you close the Orchestration Designer projects.
5. Click **Apply**.
6. Click **Apply and Close**.

Configuring Secure Fetch Port

About this task

You can specify the port number used by the application server to use HTTPS to get and post data from form nodes, such as prompt and collect, announce, menu, record, and transfer. If you are using Tomcat, the default port is 8443.

Procedure

1. On the Eclipse user interface, click **Window > Preferences**.

   The Eclipse user interface displays the Preferences dialog box.
2. In the navigation pane, double-click **Avaya**.
3. Click **Orchestration Designer**.

4. In the **Secure Fetch** area, in the **Secure Fetch Port** field, enter the port number that the application server uses.
5. Click **Apply**.
6. Click **Apply and Close**.

Orchestration Designer preferences field descriptions

Name	Description
Proxy Settings	
Enable HTTP proxy connection	Select to enable HTTP proxy connection if you need a proxy server for Internet access.
Ignore hosts with addresses	Specify the IP address of the HTTP hosts. Orchestration Designer ignores HTTP hosts with these addresses. For multiple addresses, use a comma or semicolon as a separator .
HTTP proxy host address	Specify the full HTTP path or the URL of the proxy server.
HTTP proxy host port	Specify the port that Orchestration Designer can use to access the proxy server.
Copy HTTP settings to HTTPS	Click to copy the configured HTTP settings to HTTPS settings automatically.
HTTPS Proxy Settings	
Enable HTTPS proxy connection	Select to enable HTTPS proxy connection if you need a proxy server for Internet access. Clear the check box if you do not need a proxy server for Internet access. If cleared, Orchestration Designer disables the other options in the HTTPS Proxy Settings area.
Ignore HTTPS hosts with addresses	Specify the IP address of the HTTPS hosts. Orchestration Designer ignores HTTPS hosts with these addresses. For multiple addresses, use a comma or a semicolon as a separator .
HTTPS proxy host address	Specify the full HTTPS path or the URL of the proxy server. If you do not know this address, see the proxy server settings of your Internet browser software.

Table continues…

Name	Description
HTTPS proxy host port	Specify the port that Orchestration Designer can use to access the HTTPS proxy server. If you do not know the URI, contact the Avaya technical service representative. **Note:** These settings are required even if proxy options are set in Microsoft Internet Explorer or any other web browser.
Runtime License Server	
Server URI	Specify the URI of the run-time license server. **Note:** You must specify a run-time license server only if Avaya Experience Portal, IR, or MPS accesses your application from the development environment. If you run applications through Application Simulator, you do not need a run-time license. The format of this URI is *http://webServerName:port* where: • **webServerName** is the fully qualified host name or IP address of your WebLM license server. • **port** is the number of the HTTP/HTTPS port the system uses to access the license server. For example, `http://licenseServer.myCompany.com:8080`.
License Check Timeout	Specify the time in seconds. The system must wait for a response from the WebLM license server for the specified time while attempting to connect to the WebLM license server. The default value is zero seconds. Zero indicates that there is no timeout.
Context Files	

Table continues…

Name	Description
Remove context files on project close	Select to automatically delete corresponding context files when you close Orchestration Designer projects. Orchestration Designer recreates the context file when a project is reopened. This improves the performance by controlling the size of the workspaces. If you clear this check box, Tomcat opens the context files of all projects each time you simulate a project. Therefore, Orchestration Designer performance can degrade if you have several workspaces with a huge number of projects. **Note:** This option does not affect the projects that are not opened in the current session. This applies only to Orchestration Designer projects which you open and close subsequently.
Secure Fetch	
Secure Fetch Port	Specify the port number used by the application server. Use this only if you want to use HTTPS to get and post data from form nodes, such as prompt and collect, announce, menu, record, and transfer. If you are using Tomcat, the default port is 8443.

Orchestration Designer call control project preferences configuration

Setting the maximum number of participants for a conference

About this task

You can create a conference and then add (join) connections to that conference to create a multi-party conference call. Connections can also be unjoined. You can specify the maximum number of participants for a conference and test error conditions like “conference full”.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the **Preferences** dialog box.

2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer**.

3. Click **Call Control**.
4. In the **Call Control** pane, in the **Max Conference Participants** field, specify the maximum number of participants for a conference.

   Typically, two or three participants are specified. The default is **3**, but more than three participants can participate in the conference.

CCXML template management

You can configure CCXML templates to efficiently develop CCXML code by using reusable code parts.

To simplify repetitive CCXML application development, the designer can create reusable CCXML templates.

Creating a CCXML template

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Call Control**.
3. Click **CCXML Templates**.
4. In the Templates pane, click **New**.

   The system displays the New Template dialog box.
5. Specify the information to create a new template.
6. Click **OK**.

New Template dialog box field descriptions

Name	Description
Name	Name of the template. Enter an appropriate name which explains the purpose of the template.

Table continues…

Name	Description
Context	Context of the template, in terms of how you expect to use the template in your CCXML application development. Possible values are: • **ccxml_all**: A template that applies to all CCXML code. • **ccxml_new**: A clean slate general template, for getting started. • **ccxml_base**: A clean slate general template. • **ccxml_tag**: A template with one or more commonly used CCXML tags. • **ccxml_attribute**: A template with one or more commonly used CCXML attributes. • **ccxml_attribute_value**: A template with one or more commonly used CCXML attribute values.
Description	Additional information to describe the template and its context.
Automatically insert	Check box to indicate whether the template should be automatically inserted into your CCXML code development. Otherwise, leave ed.
Pattern	Indicate a common reusable pattern or code section for the CCXML template being defined. After completing the step, click **OK**.
Insert Variable	When building your patterns, use the Insert Variable option to quickly insert commonly used variables. Available insertion variables are: • cursor: The cursor position after editing template variables. • current date: Current date. • dollar: Dollar symbol. • encoding: Creating file encodings. • line_selection: Selected lines. • time: Current time. • user name: User name. • work_selection: Selected lines. • year: Current year.

Editing a CCXML template

Procedure

1. On the **Window** menu, click **Preferences**.

The system displays the Preferences dialog box.

2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Call Control**.
3. Click **CCXML Templates**.
4. In the Templates pane, in the Create, edit, remove templates pane, select the check box corresponding to the CCXML template that you want to edit, and then click **Edit**.

   The system displays the Edit Template dialog box.
5. Edit the template, and then click **OK**.

Importing a CCXML template

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the **Preferences** dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Call Control**.
3. Click **CCXML Templates**.
4. In the **Templates** pane, click **Import** to import a CCXML template.

   The system displays the **Import Templates** dialog box.
5. Select the CCXML template XML file that you want to import, and then click **Open**.
6. In the **Preferences** dialog box, click **Apply**, and then click **OK**.

Exporting a CCXML template

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Call Control**.
3. Click **CCXML Templates**.
4. In the Templates pane, in the Create, edit, remove templates pane, select the CCXML template that you want to export.
5. Click **Export** to export the CCXML template to an XML file.

   The system displays the Export Templates dialog box.
6. Select the location where you want to export the templates, and then click **Save**.
7. In the Preferences dialog box, click **Apply**, and then click **OK**.

Removing a CCXML template

Procedure

1. On the **Window** menu, click **Preferences**

The system displays the **Preferences** dialog box.

2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Call Control**
3. Click **CCXML Templates**.
4. In the **Templates** pane, in the **Create, edit, remove templates** pane, select the check box corresponding to the CCXML template that you want to remove, and then click **Remove**.

Certificate management

The Certificates Preferences panel provides settings to manage certificates in the simulation environment.

Note:

The default keystore for certificates is trusted_weblm_certs.jks and the password for trusted_weblm_certs.jks is “password”.

Changing the keystore for certificates

About this task

You can select a different keystore for certificates or create a new keystore file. You must enter a valid password to use a keystore. If you create a new keystore file, you must define a password to use that keystore.

Note:

The default keystore for certificates is `trusted_weblm_certs.jks` and the password for `trusted_weblm_certs.jks` is “password”.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer**.
3. Click **Certificates**.
4. In the Certificates pane, click **Change**.

   The system displays the Keystore dialog box.
5. Click **Browse** and select the required file to use for the keystore.
6. In the **Password** field, type the password to access the keystore.
7. In the **Confirm** field, re-enter the password.
8. Click **OK**.

The system displays the Certificates preferences panel.

9. Click **Apply**, and then click **OK**.

Adding a certificate

About this task

You can add a certificate to the keystore from a file.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the **Preferences** dialog box.

2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer**.
3. Click **Certificates**.
4. In the **Certificates** pane, click **Add**.

   The system displays a file manager window for navigating to the required certificate (.jks) file.

5. Select the required certificate file, click **Open**.

   The system displays the Certificates preferences panel.

6. Click **Apply**, and then click **OK**.

Fetching a certificate from a URL

About this task

You can fetch a certificate from the specified URL and add it to the keystore.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the **Preferences** dialog box.

2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer**.
3. Click **Certificates**.
4. In the **Certificates** pane, click **Fetch**.

   The system displays the **Fetch Certificate** dialog box.

5. In the **URL** field, type the URL of the server from where you want to fetch the certificate.
6. In the **Name** field, type a name for the certificate.
7. Click **OK**.

The system displays the Certificates preferences panel.

8. Click **Apply**, and then click **OK**.

Deleting a certificate

About this task

You can delete a certificate from the keystore.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer**.
3. Click **Certificates**.
4. In the Certificates pane, in the Names pane, select the certificate you want to delete.
5. Click **Delete**.
6. Click **Apply**, and then click **OK**.

JDBC driver management

The Database Preferences panel provides a place for adding and removing database JDBC drivers that are used when running Orchestration Designer applications.

Adding a JDBC driver

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the **Preferences** dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer**.
3. Click **Database**.
4. In the **Database** pane, click **Add**.

   The system displays a file manager window for navigating to a .jar JDBC driver file.
5. Select a `*.jar` file, then click **Open**.

   The system displays the Database Preferences panel.

6. Click **Apply** to save the selected driver as a driver for Orchestration Designer, and then click **OK**.

Removing a JDBC driver

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the **Preferences** dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer**.
3. Click **Database**.
4. In the **Database** pane, in the **JDBC Drivers** pane, click the JDBC driver that you want to delete.
5. Click **Remove**.
6. Click **Apply**, and then click **OK**.

Enabling tracing output for the Eclipse development environment

About this task

The Debug Tracing Preferences pane provides a setting that, if enabled (selected), collects tracing output for the Eclipse IDE development environment. Tracing output is saved to the following file:

`eclipse_home/OrchestrationDesigner/orchestrationdesigner.log`

In addition, the `orchestrationdesigner.properties` file, in the same directory, is a `log4j properties` file that controls the tracing output.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the **Preferences** dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer**.
3. Click **Debug Tracing**.
4. In the **Debug Tracing** pane, select the **Enable Trace Output for Dialog Designer** check box.
5. Click **Apply**, and then click **OK**.

Creating an encrypted password

About this task

The Password Encryption Preferences panel provides a password encryption utility for users to create encrypted passwords that can then be copied into the target application's `web.xml` file.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer**.
3. Click **Password Encryption**.
4. In the Password Encryption pane, in the **Password** field, type a password to be encrypted.
5. In the **Confirm password** field, re-enter the password.
6. Click **Encrypt**.

   An encrypted value is displayed in the **Encrypted value** field.
7. Highlight and copy (`Control+C`) this value to the appropriate place within the target application's `web.xml` file. For example:
   - For a T-Server password change, copy and replace the encrypted password string in the <param-value> by the <param-name> - ".tserver".
   - For a database password change, copy and replace the encrypted password string found at the end of the <param-value> by the <param-name> - "<dd.ds.[ds name]>".

     Note:

     The database connector allows the database password to change without requiring the application to re-deploy.
8. In the Preferences dialog box, click **OK**.

Orchestration Designer speech project preferences configuration

The Speech Preferences panel provides preferences and settings applicable to speech application editors and capabilities.

Generating a timestamp during code generation

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the **Preferences** dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer**.
3. Click **Speech**.
4. In the **Speech** pane, select the **Generate timestamp during code generation** check box to include a timestamp during code generation.
5. Click **Apply**, and then click **OK**.

Call Flow Editor preferences configuration

The Call Flow Editor Preferences panel provides control color mappings for elements used in the Call Flow Editor, warning and error settings related to call flow design activity, and backup management, including enabling and disabling of call flow design backups.

Validating a call flow automatically

About this task

You can configure the setting to automatically validate a call flow upon saving a speech application.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the **Preferences** dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Speech**
3. Click **Call Flow Editor**.
4. In the **Call Flow Editor** pane, select the **Validate call flow on save** check box.
5. Click **Apply**, and then click **OK**.

Organizing your imports after copying a Java code

About this task

This option is enabled by default. It allows slowing down of the copy and paste function. Advanced users can disable this option for better performance.

User interaction is required if Eclipse is unable to resolve the import. To organize imports manually, open the Java class (**Source** > **Organize Imports** or `Control Shift O`).

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Speech**.
3. Click **Call Flow Editor**.
4. In the Call Flow Editor pane, select the **Organize imports after copying Java code** check box.
5. Click **Apply**, and then click **OK**.

Closing an editor tab automatically when opening a new tab

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Speech**.
3. Click **Call Flow Editor**.
4. In the Call Flow Editor pane, select the **Automatically close editor tab when opening new tab** check box.
5. Click **Apply**, and then click **OK**.

Viewing node location coordinates

About this task

You can configure the setting to view the node location coordinates when viewing the call flow in the Call Flow Editor.

Procedure

1. On the **Window** menu, click **Preferences**

   The system displays the **Preferences** dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Speech**.
3. Click **Call Flow Editor**.
4. In the **Call Flow Editor** pane, select the **Display node location coordinates** check box.
5. Click **Apply**, and then click **OK**.

Color coding the elements in the call flow editor

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.

2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Speech**.
3. Click **Call Flow Editor**.
4. In the Call Flow Editor pane, in the Colors area, click the element for which you want to set a color.
5. Click the button next to **Color**.

   The system displays a color picker.
6. Click a color for the element, and then click **OK**.
7. In the Preferences dialog box, click **Apply**, and then click **OK**.

Enabling the call flow editor warning and error logging

About this task

The warning and error conditions that you enable are logged to the Problems view when working with call flows. For more information, see Problems view on page 44.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Speech**.
3. Click **Call Flow Editor**.
4. In the Call Flow Editor pane, in the Warnings and Errors area, select the check boxes corresponding to the warnings and errors that you want to enable.
5. Click **Apply**, and then click **OK**.

Backing up a call flow

About this task

By default, Call Flow Editor maintains a backup of the flow document, in the event that the main flow is corrupted. Prior to saving the contents of the main flow, the old main flow file is copied to a backup folder using a round-robin file numbering system to maintain multiple backups. The default is 2 and the maximum is 10. The backup files are numbered, for example, main.flow.1, mainflow.2, but after the maximum number of backup is reached, the numbering sequence starts over.

Procedure

1. On the **Window** menu, click **Preferences**

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Speech**.
3. Click **Call Flow Editor**
4. In the **Call Flow Editor** pane, in the **Backups** pane, select the **Enable backups** check box.

5. In the **Number of backups to keep** field, type the number of backups that you want to retain for the call flow.

   Note:

   By default, 2 backups are retained at any given time. The maximum number of backups is 10.

6. Click **Apply**, and then click **OK**.

Call Flow Editor preferences field descriptions

Name	Description
General Settings	
Validate callflow on save	Check box to validate the call flow on each save of an application.
Organize imports after copying Java code	This check box is selected by default. It allows slowing down of the copy and paste function. Advanced users can disable this option for better performance. **Note:** User interaction is required if Eclipse is unable to resolve the import. To organize imports manually, open the Java class (**Source** > **Organize Imports** or `Control Shift O`).
Automatically close editor tab when opening new tab	When this is selected, the Call Flow Editor tab is closed when opening a new tab in your Dialog Design work area. This option is purely preferential, based on a designer's work styles.
Display node location coordinates	Check box to display the node location coordinates when displaying the call flow in the Call Flow Editor.
Colors Settings	

Table continues…

Name	Description
Colors	Allows the configuration of color coding to the various elements used in call flows. By clicking on an element, then clicking on the Color button, a color picker is displayed where a color can be mapped for each element. The following elements can have colors assigned to them: • Bookmarks • Bookmark Text • Labels • Label Text • Selected Connection
Warnings and Errors Settings	
Warnings and Errors	Select from (enable, or disable) the following warning and error conditions that are desired to be logged to the Problems pane when working with call flows: • Display code generation failure errors. • Display paste errors or warnings. • Warn when renaming call flow nodes. • Warn when renaming call flow items that will trigger variables to be renamed. • Warn before deleting items that will trigger variables to be deleted.
Backups The Call Flow Editor by default maintains a backup of the flow document, in the event that the main flow is corrupted. Prior to saving the contents of the main flow, the old main flow file is copied to a backup folder using a "round-robin" file numbering system to maintain multiple backups. The backup files are numbered (for example, main.flow.1, mainflow.2, and so on) but after the maximum number of backups is reached, the numbering sequence starts over.	
Enable backups	Check box to enable backups of call flows. By default, this option is enabled.
Number of backups to keep	If backups are enabled, type the number of backups that you want to retain for each call flow. By default, 2 backups are retained at any given time. The maximum number of backups is 10.

Configuring the settings for grammar tags

About this task

You can set the background and foreground color for the tags that you apply to your grammar entries. You can also set the color for the tag values and cursor.

You can configure the setting to view the tags that are associated with the grammar entry in the grammar editor.

For more information about applying tags to your grammar entries, see About tags on page 256.

Procedure

1. On the **Window** menu, click **Preferences**

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Speech**.
3. Click **Grammar**.
4. In the Grammar pane, in the Colors area, click the option for which you want to set the color.
5. Click the button next to **Color**.

   The system displays a color picker.
6. In the color picker, click the color that you want to apply, and then click **OK**.
7. Select the **Show tags in editor** check box, if you want to view the tags that are associated with the grammar entry in the grammar editor.

   If you clear the **Show tags in editor** check box, then you can view the associated tag in the grammar editor only if you click the cell that contains the tag.
8. Click **Apply**, and then click **OK**.

Configuring languages

About this task

The Languages Preferences panel provides settings to define what languages Orchestration Designer recognizes and makes available for use in speech applications. There are three language types that can be added to Orchestration Designer for use with applications: ASR, TTS, and localization bundles.

Procedure

1. On the **Window** menu, click **Preferences**.
2. In the Preferences dialog box, in the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Speech**.

3. Click **Languages**.
4. In the **Languages** pane, in the **Automated Speech Recognition** area, configure the ASR language that you want.

   For more information, see About automated speech recognition languages on page 284.
5. In the **Text-to-Speech** area, configure the TTS language that you want.

   For more information, see About text-to-speech languages on page 287.
6. In the **Audio Localization Packages** area, configure the localization bundles that you want.

   For more information, see About localization bundles on page 289.
7. Click **Apply**, and then click **OK**.

Configuring the default settings for recording phrases

About this task

The Phrase Preferences panel provides default settings for recording phrases in Orchestration Designer.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Speech**.
3. Click **Phrase**.
4. In the Phrase pane, perform the following actions:
   a. In the Default audio extension area, click the default audio recording file format to use when saving phrases in Orchestration Designer.
   b. Perform one of the following actions:
      - Click **wav**. The .wav format is usually used on Windows-based development environments.
      - Click **au**. The .au format is more commonly used on Sun and UNIX development environments.
   c. In the Audio recorder area, perform the following actions:
      a. In the **Sampling rate** field, click the default sampling rate for audio recordings.
      b. In the **Sampling size** field, click the default sampling size for audio recordings.

Important:

Do not change the **Audio recorder** settings. Currently, Orchestration Designer works only with the default settings.

5. Click **Apply**, and then click **OK**.

Prompt preferences configuration

The Prompt Preferences panel provides configurable default settings for a number of prompt management capabilities.

Configuring the default barge-in settings for the prompts

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Speech**.
3. Click **Prompt**.
4. In the Prompt pane, perform the following actions:
   a. In the Default Barge-in area, perform one of the following actions:
      - Click **true** to enable the barge-in by default.
      - Click **false** to disable the barge-in by default.
   b. In the Default Barge Type area, perform one of the following actions:
      - Click **speech** to trigger the barge-in and stop the prompt because of a spoken response by default.
      - Click **hotword** to trigger the barge-in and stop the prompt because of a complete match of an active grammar by default.
5. Click **Apply**, and then click **OK**.

Configuring the default play order for the prompts

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Speech**.
3. Click **Prompt**.
4. In the Prompt pane, in the Default Play Order area, click the play order that you want to set as the default play order for the prompts.

   For more information, see Play order for prompt levels in speech applications on page 203.

5. Click **Apply**, and then click **OK**.

Configuring the default timeout for prompts in all Orchestration Designer speech projects

About this task

The default is 8 seconds.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Speech**.
3. Click **Prompt**.
4. In the Prompt pane, in the **Default Time Out** area, perform the following actions:
   a. In the **Timeout** field, type the number of seconds or milliseconds for the timeout.
   b. In the corresponding field, click **seconds** or **milliseconds**.
5. Click **Apply**, and then click **OK**.

Viewing the standard phrases in the prompt file editor

About this task

You can configure the setting to view and use the installed standard phrases in the prompt file editor. For more information, see Prompt file editor in a speech application on page 207 and Installing the standard phrasesets of a localization bundle on page 293.

Procedure

1. On the **Window** menu, click **Preferences**.

   The system displays the Preferences dialog box.
2. In the left navigation pane, double-click **Avaya** > **Orchestration Designer** > **Speech**.
3. Click **Prompt**.
4. In the **Prompt** pane, in the **Standard Phrases** area, click the **Show standard phrases in the combo list** check box to view and use the installed standard phrases in the prompt file editor.
5. Click **Apply**, and then click **OK**.

Prompt preferences field descriptions

Name	Description
Default Bargein	Indicates whether bargein (allowing callers to barge in, or interrupt, prompts) is enabled or not. Options are: • **true**: enabled. This is the default. • **false**: disabled
Default Barge Type	Type of action that triggers barge-in, if enabled for the prompt. Options are: • **speech**: Any spoken response triggers barge-in and stops the prompt. This is the default. • **hotword**: Only a complete match of an active grammar triggers barge-in and stops the prompt.
Default Play Order	Determines the order in which prompt levels are played when the prompt is repeated. Options include: • **standard** (default) • **first** • **random** • **sequential** See Play order for prompt levels in speech applications on page 203.
Default Time Out	Number of seconds or milliseconds that the system must wait for a response from the caller, after the prompt is finished playing, and before the system throws a No Input event. 8 seconds is the default.
Standard Phrases	Check box to show any standard phrases that have been installed to be available for use in the prompt file editor. See Prompt file editor in a speech application on page 207.

Configuring Tomcat Preferences

About this task

Tomcat preferences provide settings that determine how Orchestration Designer works with the Apache Tomcat servlet engine during simulations.

If you install Tomcat with the default settings, Tomcat preferences are already configured. Verify that the Tomcat version, home directory, and contexts directory are appropriate.

Note:

If you are running only Orchestration Designer in your development environment without running deployed applications, the `runtimeconfig` file is installed automatically. You only have to set up your production system when you are deploying and running live applications.

Important:

Do not run `runtimeconfig` on your ADE. Otherwise, connection timeout exceptions occur. To recover, stop Tomcat, stop Orchestration Designer, restart Orchestration Designer, and change your configuration in Preferences.

Procedure

1. On the Eclipse user interface, click **Window > Preferences**.

   The Eclipse user interface displays the Preferences dialog box.
2. In the navigation pane, click **Tomcat**.
3. In the Tomcat version area, click the Tomcat version that is installed on your computer.
4. In the **Tomcat home** field, click **Browse** and select the directory where Tomcat is installed.
5. In the Context declaration mode area, click **Context files**.
6. In the **Contexts directory** field, click **Browse** and select the context directory.
7. Click **Apply**.
8. Click **Apply and Close**.

Chapter 35: System variables

System variables

Every speech application project you create with Orchestration Designer has a set of complex variables that are automatically created with the project. These variables are automatically built-in to every speech project you create in Orchestration Designer. These are known as *system variables*. These mostly read-only variables can be put to a wide variety of uses in your speech application projects.

For more information about these and about variables in general, see Variables on page 226.

System variable fields and properties

The following table shows descriptions of the fields and properties for the system variables:

Variable	Associated fields	Descriptions/comments
date:This variable gets data on the current date from the system.		
	audio	Returns the current date. This field is specifically designed and optimized to be used in an audio variable. Therefore, to use this field, you must have the localization bundle and standard phrases installed. See Audio Variable on page 676. When you use this field in a prompt, you can specify additional properties that govern how the information is presented to the caller. See Audio Field properties on page 586.
	dayofmonth	Returns the number of the current day. For example, if the current date is May 23, this field returns the value **23**.
	dayofweek	Returns the name of the current day of the week, for example, **Tuesday**.
	dayinweeknum	Returns the number of the day in the week, where Sunday is 0 and Saturday is 6.
	dayofweeknum	Returns the current day of the week as an integer, for example, Sunday is **1** ... Saturday is **7**.

Table continues…

Variable	Associated fields	Descriptions/comments
	dayofyear	Returns the number of the current day in the year. For example, if the current date is May 23, in a non-leap year, this field returns the value **143**.
	month	Returns the name of the current month, for example, **May**.
	monthinyear	Returns the number of the current month in the year. For example, if the current month is May, this field returns the value **05**.
	year	Returns the current year, for example, **2005**.
ddLastException: This variable is automatically updated when a Java exception is caught using the Catch (Exception) item. The ddLastException fields will have data that is extracted from the Java exception that was caught.		
	errorcode	The error code associated with the exception, if provided. Currently only SQLExceptions provide error codes and the error codes are JDBC vendor specific.
	message	The exception message. Contains information about why the exception was thrown.
	object	The exception Java Object. This is stored so that you can access the Java Exception object.
	stacktrace	The stack trace of the exception stored in a string form.
	type	The type of exception. The type is the fully qualified name of the exception class. For example, “java.io.FileNotFoundException”, or “java.lang.NullPointerException”.
redirectinfo: This variable gets data from the session connection redirect variable. This variable is an array representing the connection redirection paths.		
	presentationinfo	Returns the value of the **pi** (presentation information) property for the **session.connection.redirect** variable for each element of the array.
	screeninginfo	Returns the value of the **si** (screening information) property for the **session.connection.redirect** variable for each element of the array.
	uri	Returns the value of the **uri** property for the **session.connection.redirect** variable for each element of the array.
session: This variable takes the values for its fields from the call session information generated at run time.		

Table continues…

Variable	Associated fields	Descriptions/comments
	aai	Returns the value of the session variable **session.connection.aai**. This variable contains application-to-application information that is passed during connection setup. **Note:** This variable is available only in speech applications.
	ani	Returns the ANI, or the number that the caller is calling from. **Note:** This variable is available only in speech applications.
	calltag	Returns a string, which contains the following elements: • Host name: Name of the Avaya IR system from which the call session data was obtained • Channel number: Number of the IR system port where the call was received • Date and time the IR system received the call For example, a typical return value for this field can be: IR 11 15 Wed Aug 3 11:51:54 2005 In IR 3 SP3, the format of the calltag is: machine_name-channel_number-sessionID **Note:** • This field is specific to Avaya IR systems. • This variable is available only in speech applications.
	channel	Returns the number of the port on which the call resides. **Note:** This variable is available only in speech applications.
	conversefirst	Contains the DTMF digit data captured from an Avaya Call Center vector program. This data captured from the platform variables is used to initiate a session with the Orchestration Designer application. **Note:** • These fields collects the DTMF digit data for the converse -on step in Avaya IR and Avaya Experience Portal and stores them in the **session.telephone.firstconversedata** and

Table continues…

Variable	Associated fields	Descriptions/comments
	conversesecond	**session.telephone.secondconversedata** session variables. • These variables are available only in speech applications.
	currentlanguage	Returns the current default language as defined by the application.
	dnis	Returns the DNIS, or the number that the caller dialed. **Note:** This variable is available only in speech applications.
	exitReason	These fields are placed into the SDR (Session Data Record) on Avaya Experience Portal and eventually loaded into Avaya IQ for reporting. It is up to you to set these values and you can change them depending on the application flow. For example, if the application catched **connection.disconnect**, they can set the **exitReason** to “caller hungup” or if the application exited normally before exiting the **exitReason** can be set to “transaction complete”. Or these variables can be used to track how far a caller proceeded through a flow before hanging up. These variables will be generated into all uses of the VXML <exit/> tag namelist attribute as: • “_avayaExitReason” • “_avayaExitInfo1” • “_avayaExitInfo2” By default the value of the **exitReason** will be set to ““ and **exitInfo1** and **exitInfo2** will be set to ””. CCXML will receive these values in the **dialog.exit event**. It is up to the CCXML application to handle these values so they eventually make it to IQ or other appropriate destination. The default VXML handlers provided by Avaya Experience Portal (formerly default.vxml) will need to be augmented and also look for the same values (as above, that is, “_avayaExit<type>” to be defined. If doing so, place them in the namelist of any VXML <exit/> tags. These values will be defined (generated) into every Orchestration Designer generated VXML page. The default handlers catch exit cases where an error occurs and no handler is defined in the application to catch and handler the error. In addition, Orchestration Designer will (try to, if
	exitInfo1	

Table continues…

Variable	Associated fields	Descriptions/comments
	exitInfo2	feasible) generate a “default” **connection.disconnect** handler if the application does not provide one. Avaya Experience Portal is responsible for documenting this so that non-Orchestration Designer applications can utilize this mechanism, and to allow custom CCXML applications to take advantage of this feature. It is unknown what if any effect this will have on OSDM’s. However, you can likely assume that if an application terminates in an OSDM no termination information will be logged. The above serves as the design for the standpoint of Orchestration Designer and as a contract between Orchestration Designer and Avaya Experience Portal.

Table continues…

Variable	Associated fields	Descriptions/comments
	exitParentId	The system creates the exitParentId variable only for message applications. Unlike speech application, which considers the entire conversation in a call as one single session, a message application considers the process of receiving an inbound SMS or email message and sending one or more outbound message responses as one session. The message application processes the other inbound messages as separate sessions even if the inbound messages are correlated to the first inbound message. Every time the message application receives an inbound message, the system assigns a different session ID to the message application. To bind the correlated messages together for reporting, message applications use the exitParentID variable. For example, a customer sends an SMS message that contains the `Start quiz 101` text to start a quiz. The text browser launches the appropriate SMS channel message application. The system assigns a session ID to the message application. The message application analyzes the inbound SMS message and sends an outbound SMS message response that contains the first question of the quiz. The message application then assigns the session ID of the current session to the exitParentID variable of the current session. The message application also stores the session ID of the current session, the ANI, the key that points to the current location in the quiz, and other quiz-related data such as the current score of the customer, and then exits. The customer sends an SMS message that contains the answer to the first question of the quiz. The text browser again launches the SMS channel message application. The system assigns a different session ID to the message application. The message application verifies if the ANI of this inbound SMS message is the same as the ANI of the first inbound SMS message and if the customer has already started the quiz. If the ANI values are the same and if the customer has already started the quiz, the message application considers these two inbound SMS messages as correlated. The message application then retrieves the current state of the message application including the location in the quiz where the key is pointing to and the session ID of the first session. The message application then assigns the session ID of the first session to the exitParentID variable of the second inbound message. The message application verifies if the answer

Table continues…

Variable	Associated fields	Descriptions/comments
		provided by the customer is correct and sends an outbound SMS message that contains the correct answer to the first question, the score of the customer, and the second question of the quiz. The message application then stores the session ID of the current session, the ANI, the key that points to the current location in the quiz, and other quiz-related data such as the current score of the customer, and then exits. In the above example, for the two sessions, the message application creates two session data records (SDR) and four call data records (CDR), one CDR for each inbound and outbound message. The message application assigns the session ID of the first session to the exitParentID variables of both the sessions. The message application similarly processes all the correlated inbound SMS messages received from the customer. Similarly, an email channel message application processes the correlated inbound email messages. The only difference is that instead of ANI, an email channel message application uses the email addresses to determine whether the inbound email messages are correlated. The system stores the exitParentID variable values in Session Data Record (SDR) on Avaya Experience Portal. During reporting, the exitParentID variable values are used to bind together the correlated SMS messages and email messages. However, if a transaction lasts for only one session and the message application processes the transaction as complete, the message application does not assign the session ID of the inbound message to the exitParentID variable of the inbound message. For example, a customer sends an SMS message enquiring about the balance in the checking account. The message application interprets the inbound SMS message and sends an outbound SMS message response to the customer providing the balance information, and then exits.
	exitCustomerId	Use this field to store an identifier for a caller. For example, you can store an account number or a user ID in this field. The reporting software can use this field to provide repeat caller analysis as well as the ability to generate reports of all the calls for a particular customer. At the end of the call, the system returns the value of this field to Avaya Experience Portal for processing and reporting.

Table continues…

Variable	Associated fields	Descriptions/comments
	exitPreferredPath	Orchestration Designer uses this variable for tracking if the caller followed the preferred paths of execution through the application. The Orchestration Designer runtime automatically calculates this value based on the setting of each node in the callflow. The default value is “true”. At the end of the call, the system returns this value to Avaya Experience Portal for processing and reporting.
	exitTopic	Use this field to track the type of activity the application was performing that resulted in the exit. For example, 'check balance' or 'inquire about product', and so on. At the end of the call, the system returns this value to Avaya Experience Portal for processing and reporting.
	lasterror	When an IC operation completes, error information is returned from the VOX in this variable. If the operation succeeded, the lastError field will be empty. In conjunction with the web parameter **sage.ic.throwexceptions**, the application can be disabled from throwing runtime exceptions for IC errors by setting this value to false. This requires that you always check for a value in the **lastError** field, but provide the flexibility of doing so within the data node instead of their error handler. Regardless of the value of **sage.ic.throwexceptions**, **session.lastError** will always be populated after an IC call.
	mediatype	Returns the type of media file as audio, video, image, or text. **Note:** This variable is available only in speech applications.
	protocolname	Returns the value of the session variable **session.connection.protocol.name**. This variable is the name of the connection protocol. **Note:** This variable is available only in speech applications.
	protocolversion	Returns the value of the session variable **session.connection.protocol.version**. This variable is the version of the connection protocol. **Note:** This variable is available only in speech applications.
	sessionid	Returns the session ID number assigned to the call.

Table continues…

Variable	Associated fields	Descriptions/comments
	sessionlabel	Contains any data you want to assign to it. This is the only field of the session variable to which you can assign a value. Assign the value using the **Value** field in the Avaya Properties view. **Note:** This field is specific to Avaya Experience Portal. It was designed to be used as an application report filter.
	sharedmode	Either **shared** or **serviceprovider**. If shared, then uui consists of fields of id and value, Example: **12, 01DR12345678; FA, 0001001C46E70028** The **uui** is then automatically unpacked so that the id value can be accessed. **Note:** This variable is available only in speech applications.
	ucid	The Universal Call Identifier (UCID) assigned to the call by the platform or the PBX.
	uui	Returns the contents of the **uui** (user-to-user information) variable field. **Note:** This variable is available only in speech applications.
	videobitrate	Video bit rate used in the call to specify the number of bits processed per second. Valid value range for this parameter is 0 - 19200. The default value is 1920. You can change this value based on the network requirement. If the value is set to 0, the platform ignores this value and uses the default value 1920 * 100 bps. **Note:** • It is observed that If the value is less than 1200, the video quality is degraded. • This variable is available only in speech applications.

Table continues…

Variable	Associated fields	Descriptions/comments
	videocodec	Video codec used in the call to enable compression and decompression for digital video. You can set this value to H263 or H263+ (or H263-1998). **Note:** This variable is available only in speech applications.
	videoenabled	Decides whether the support for video is enabled or disabled. You can set this value as True or False. **Note:** This variable is available only in speech applications.
	videofarfmtp	Provides the format specific information from the session invitation. This value depends on the far end capabilities, that is, caller's phone. The default is QCIF=2;CIF=3;MAXBR=1960. **Note:** This variable is available only in speech applications.
	videonearfmtp	Provides the format specific information from the Media Processing Platform (MPP). The default is QCIF=4.
	videoformat	Defines the size of the video frame. Orchestration Designer supports CIF (352 X 288) and QCIF (176 X 144) video formats. End-point sends the list of formats it supports in the session invitation. MPP sends back the appropriate video format at which the video is sent to the end-point. It also sends the same value to Orchestration Designer. **Note:** This variable is available only in speech applications.
	videofps	Defines the frame rate in Frame Per Second. This value depends on the end device. You can set this value to 15 or 30. **Note:** This variable is available only in speech applications.

Table continues…

Variable	Associated fields	Descriptions/comments
	videoheight	Specifies the height of the video stream in pixels. The default height is 144 for QCIF and 288 for CIF. This value is specified using the **videoformat** parameter. **Note:** This variable is available only in speech applications.
	videowidth	Specifies the width of the video stream in pixels. The default width is 176 for QCIF and 352 for CIF. This value is specified using the **videoformat** parameter. **Note:** This variable is available only in speech applications.
	vpcalledextension	Returns the number of the station on the Avaya Experience Portal system **Note:** • This field is specific to Avaya Experience Portal. Its value is obtained from the Avaya Experience Portal Media Processing Platform (MPP). • This variable is available only in speech applications.
	vpconverseondata	Contains the DTMF digit data from an Avaya Call Center vector program. This data is used to initiate a session with the Orchestration Designer application. This value is also placed in **conversfirst** and **conversesecond** fields. If you use '#' as a separator, the data is split at the '#' character and the first part is stored in **conversefirst** and the second part is stored in **conversesecond**. **Note:** • This field is specific to Avaya Experience Portal. Its value is obtained from the Avaya Experience Portal Media Processing Platform (MPP). For more information about converse-on transactions, see your Avaya Experience Portal documentation. • This variable is available only in speech applications.

Table continues…

Variable	Associated fields	Descriptions/comments
	vpcoveragereason	Returns the reason the call is being transferred to the Avaya Experience Portal system. Possible values include: • noanswer • busy **Note:** • This field is specific to Avaya Experience Portal. Its value is obtained from the Avaya Experience Portal Media Processing Platform (MPP). • This variable is available only in speech applications.
	vpcoveragetype	This field is reserved for future use. **Note:** This field is specific to Avaya Experience Portal. Its value is obtained from the Avaya Experience Portal Media Processing Platform (MPP).
	vprdnis	Returns the original number that the caller dialed, when the caller has been transferred or redirected. For example, if the caller dialed extension 1234, but the party at extension 1234 did not answer, the system can be set to automatically transfer the call to extension 9000. In this example, the **dnis** field would be set to 9000, and this field, the **vprdnis** field would be set to 1234. **Note:** • This field is specific to Avaya Experience Portal. Its value is obtained from the Avaya Experience Portal Media Processing Platform (MPP). • This variable is available only in speech applications.
	vpreporturl	Returns the URL of the Web service to which the system writes the application report data. **Note:** • This field is specific to Avaya Experience Portal. Its value is obtained from the Avaya Experience Portal Media Processing Platform (MPP). • This variable is available only in speech applications.

Table continues…

Variable	Associated fields	Descriptions/comments
shareduui: When you set the Avaya Experience Portal specific UUI/AAI parameter to shared, the system fetches the UUI/AAI ID and value pairs and stores them in the **shareduui** variable. To set Avaya Experience Portal specific UUI/AAI parameters for simulation, see Simulation Profile: OD Runtime Session tab field description on page 531. To set the Avaya Experience Portal specific UUI/AAI parameters for an actual running application, see the “User-to-User Interface (UUI) data passed in SIP headers” section in the *Avaya Experience Portal* documentation.		
	id	Contains the shareduui ID.
	value	Contains the value of the shareduui.
time: This variable gets data on the current time from the system.		
	timezone	Returns the time zone in which the system is configured, as a number relative to “Zulu” or Greenwich Mean Time. For example, if the system taking the call is configured for Eastern Daylight Savings Time, then this field returns a value of **-0400**.
	minute	Returns the current minute past the hour on the system clock. For example, if the current time is 5:32 p.m., this field returns **32**.
	audio	Returns the current time. This field is specifically designed and optimized to be used in an audio variable. Therefore, to use this field, you must have the localization bundle and standard phrases installed. For more information about using audio variables, see Audio Variable on page 676. When you use this field in a prompt, you can specify additional properties that govern how the information is presented to the caller. See Audio Field properties on page 586.
	millisecond	Returns the current millisecond of the system clock at the time the request is made.
	hour	Returns the current hour of the system clock, in a 24-hour format. For example, if the current time is 5:32 p.m., this field returns **17**.
	second	Returns the current second of the minute past the hour on the system clock. For example, if the current time is 5:32:45 p.m., this field returns **45**.
message: The **message** variable in an SMS channel message application contains fields that store the message details of the inbound SMS messages and the delivery status of the outbound SMS messages. The system creates the **message** variable only for message applications.		
	arrivaltime	Stores the arrival time of the inbound SMS message.
	bodycontenttext	Stores the body text of the inbound SMS message.
	from	Stores the phone number of the sender of the SMS message.

Table continues…

Variable	Associated fields	Descriptions/comments
	msgid	For incoming messages, the message ID is an internally generated ID. For a delivery notification of an outbound message, **msgid** is same as the ID returned from the platform web service when the email or SMS message is sent. You can use the **msgid** to match the delivery receipt with the corresponding email or SMS request.
	providerid	Stores a message id supplied by SMSC or email server.
	subject	Stores the subject line of the inbound SMS message.
	to	Stores the phone number of the receiver of the SMS message.

Table continues…

Variable	Associated fields	Descriptions/comments
	messagetype	Stores the type of the message that the SMS channel message application receives. The following are the message types: • **normal**: The SMS channel message application receives an inbound SMS message. • **notification**: The SMS channel message application receives a notification status code from Short Message Service Center (SMSC) when the Avaya Experience Portal platform submits an outbound SMS message to SMSC for delivery. • **delivery_status**: The SMS channel message application receives a delivery status code from SMSC when SMSC delivers the SMS message to the destination telephone number. **Note:** The **messagetype** variable stores the **notification** and **delivery_status** information only for those outbound SMS messages that the SMS channel message application sends in response to the inbound SMS message. Ensure that on EPMS, you enable the feature to receive notification messages and delivery status receipts for the outbound SMS messages from SMSC. For more information, see the Avaya Experience Portal documentation. **Tip:** To store the **notification** and **delivery_status** message types for the SMS notifications that are sent by using the **Send SMS** item, create a separate SMS channel message application. The telephone number that you configure on the Avaya Experience Portal platform to receive the notification and delivery status messages must be the same as that you specify in the **From Constant**, **From Variable**, or **From Variable Field** field of the **Send SMS**. Ensure that you set the **Request Receipt** property of the **Send SMS** item to **true**. An SMS channel message application considers the process of receiving an inbound SMS message and sending one or more outbound SMS message responses as one session. The SMS channel message application

Table continues…

Variable	Associated fields	Descriptions/comments
		starts a new session if a new inbound SMS message arrives, even if the inbound SMS message is correlated to the previous inbound SMS message. You can use the **exitParentId** variable to bind the correlated SMS messages together for unified reporting. You can use the **From** and **To** variable fields of the **message** variable to maintain the context of the correlated SMS messages in a database.
	status	Stores the success and error codes received from SMSC that represent the delivery status of the outbound SMS message. **Note:** If the message type is **notification**, then the status code zero '0' indicates that SMSC has submitted the SMS message for delivery. If the message type is **delivery_status**, then the status code zero '0' indicates that SMSC has successfully delivered the SMS message to the destination telephone number. Ensure that on EPMS, you enable the feature to receive notification messages and delivery status receipts for the outbound SMS messages from SMSC. For more information, see the Avaya Experience Portal documentation. **Tip:** To process the success and error codes of the SMS notifications that are sent by using the **Send SMS** item, create a separate SMS channel message application. The telephone number that you configure on the Avaya Experience Portal platform to receive the inbound SMS messages that contain the success and error codes must be the same as that you specify in the **From Constant**, **From Variable**, or **From Variable Field** field of the **Send SMS** item. Ensure that you set the **Request Receipt** property of the **Send SMS** item to **true**.
message: The **message** variable in an email channel message application contains fields that store the message details of the inbound email messages and the delivery status of the outbound email messages. The system creates the **message** variable only for message applications.		
	arrivaltime	Stores the arrival time of the inbound email message.

Table continues…

Variable	Associated fields	Descriptions/comments
	bcc	Stores the email addresses of the recipients who received a blind copy of the inbound email message.
	bodycontentoriginal	Stores the body of the inbound email message in the format in which the email is received. The email message body can be either in a text or an HTML format.
	bodycontenttext	Stores the body of the inbound email message in a text format after removing the HTML tags.
	bodytype	Stores the original format of the inbound email message. The options are: text and HTML.
	cc	Stores the email addresses of the recipients who received a copy of the inbound email message.
	from	Stores the email address of the sender of the email message.

Table continues…

Variable	Associated fields	Descriptions/comments
	messagetype	Stores the type of the message that the email channel message application receives. The following are the message types: • **normal**: The email channel message application receives an inbound email message. • **notification**: The email channel message application receives a notification status code from the email server when the Avaya Experience Portal platform submits an outbound email message to the email server for delivery. • **delivery_status**: The email channel message application receives a delivery status code from the email server when the email server delivers the email message to the destination email address. **Note:** The **messagetype** variable stores the **notification** and **delivery_status** information only for those outbound email messages that the email channel message application sends in response to the inbound email message. Ensure that on EPMS, you enable the feature to receive notification messages and delivery status receipts for the outbound email messages from the email server. For more information, see the Avaya Experience Portal documentation. An email channel message application considers the process of receiving an inbound email message and sending one or more outbound email message responses as one session. The email channel message application starts a new session if a new inbound email message arrives, even if the inbound email message is correlated to the previous inbound email message. You can use the **exitParentId** variable to bind the correlated email messages together for unified reporting. You can use the **From** and **To** variable fields of the **message** variable to maintain the context of the correlated email messages in a database.
	msgid	For incoming messages, the message ID is an internally generated ID. For a delivery notification of an outbound message, **msgid** is same as the ID returned from the platform web service when the email or SMS message is sent. You can use the **msgid** to match the delivery receipt with the corresponding email or SMS request.

Table continues…

Variable	Associated fields	Descriptions/comments
	providerid	Stores a message id supplied by SMSC or email server.
	subject	Stores the subject line of the inbound email message.
	to	Stores the email address of the receiver of the inbound email message.
messageHeaders: Stores the header names and headers values contained in the inbound SMS or email message. The system creates the **messageHeaders** variable for message applications only.		
	name	Stores the header names contained in the inbound SMS or email message.
	value	Stores the header values contained in the inbound SMS or email message.
messageAttachments: Stores the URL of the attachments contained in the inbound email message. The system creates the **messageAttachments** variable for email channel message applications only.		
aacc_context: The system creates the **aacc_context** variable in a speech project after you enable the **AACC Treatments** pluggable data connector in the speech project. The system uses the **aacc_context** variable for the **Context Creation** treatment only. You can add any number of variable fields to the **aacc_context** variable depending on the data that you want to send to Avaya Aura® Contact Center. The **aacc_context** variable returns data in name and value pairs in an XML format to Avaya Aura® Contact Center. The **aacc_context** variable also contains a **cad** field. For more information, see Enabling Orchestration Designer project pluggable data connectors on page 503, AACC treatments on page 371, and Example on assigning a return value to return the data to Avaya Aura Contact Center on page 377.		
	cad	Stores the call attached data (CAD) that the speech application collects during the call. The **cad** field can store up to 1KB of data. The system stores the CAD data in a string format and depending on the Avaya Aura® Contact Center script, returns the CAD data to Avaya Aura® Contact Center.
aacc_data: The system creates the **aacc_data** variable in a speech project after you enable the **AACC Treatments** pluggable data connector in the speech project. The **aacc_data** variable contains variable fields that store the parameters passed by Avaya Aura® Contact Center through the Avaya Aura® Contact Center script. The **aacc_data** variable also contains a **returnvalue** variable field that returns the data requested in the treatment type to Avaya Aura® Contact Center. For more information, see Enabling Orchestration Designer project pluggable data connectors on page 503, AACC treatments on page 371, and Example on assigning a return value to return the data to Avaya Aura Contact Center on page 377.		

Table continues…

Variable	Associated fields	Descriptions/comments
	apptype	Stores the type of the application to invoke as configured in the Avaya Aura® Contact Center script. The options are CCXML and VXML. **Note:** The default is VXML.
	gslid	Stores the call ID generated by Avaya Aura® Contact Center for the call. When AACC transfers the call to an Orchestration Designer speech application that is managed on Avaya Experience Portal, Avaya Experience Portal assigns the call ID generated by Avaya Aura® Contact Center to the call for unified reporting.
	im	—
	interdigittimeout	Stores the maximum time, in seconds, configured in the Avaya Aura® Contact Center script for collecting digits from the caller. **Note:** You can design the speech application to either use the interdigit timeout passed by the Avaya Aura® Contact Center script or the interdigit timeout that you set in Orchestration Designer speech application.
	numberofdigits	Stores the number of digits that the speech application should collect from the caller as configured in the Avaya Aura® Contact Center script. **Note:** You can design the speech application to either use the number of digits passed by the Avaya Aura® Contact Center script or the number of digits that you set in Orchestration Designer speech application.
	prompttoplay	Stores the prompt that the speech application should play to the caller as configured in the Avaya Aura® Contact Center script. **Note:** You can design the speech application to either play the prompt requested in the Avaya Aura® Contact Center script or play the prompt that you set in Orchestration Designer speech application.
	returns	Stores the options, as configured in the Avaya Aura® Contact Center script, that determine whether the Orchestration Designer speech application must return data to Avaya Aura® Contact Center.

Table continues…

Variable	Associated fields	Descriptions/comments
	returnvalue	Returns the data requested by the treatment type to Avaya Aura® Contact Center. You must design the speech application to assign the data requested by an AACC treatment type to the **returnvalue** variable field. The system does not use the **returnvalue** variable field for the **Context Creation** treatment type. For more information, see Example on assigning a return value to return the data to Avaya Aura Contact Center on page 377.
	sipbodymsgpreamble	Stores the SIP message preamble as determined by the treatment type that is requested by Avaya Aura® Contact Center. The **AACC Treatments** pluggable data connector sets the value for the **sipbodymsgpreamble** variable field.
	sipbodytype	Stores the SIP message type as determined by the treatment type that is requested by Avaya Aura® Contact Center. The **AACC Treatments** pluggable data connector sets the value for the **sipbodytype** variable field.
	termchar	Stores the terminating character, as configured in the Avaya Aura® Contact Center script, that terminates the digit collection before the maximum number of digits is reached. **Note:** You can design the speech application to either use the terminating character passed by the Avaya Aura® Contact Center script or use the terminating character that you set in the Orchestration Designer speech application.
	treatmenttype	Stores the treatment type to be applied to the call as requested by Avaya Aura® Contact Center in the Avaya Aura® Contact Center script.
	urltopush	—
	vxmlfrom	—
	vxmlto	—

Variable formats in localization bundles

All language localization bundles are required to support certain variable data formats. These formats, in turn, are the basis of the related language localization bundle formats.

For example, all language localization bundles must support the **date** data format:

YYYY[-] ***MM***[-] ***DD***

where *YYYY* is the four-digit year, *MM* is the two-digit month, *DD* is the two-digit day, and the hyphens are optional.

If you want to use the standard localization bundle formats in custom variables that you want to use as audio variables, you must ensure that your custom variables follow the standard formats as described in the following table. Then you can use the localization bundle formats to render the data.

For example, you want to query a database for all customer transactions that took place in the previous five days. When you query the database, you can use the **Operation** item of a Data node to convert the date field for each record to the format ***YYYYMMDD***. You can then use audio variables to convert this ***YYYYMMDD*** value into a set of audio clips to speak the date.

The following table lists and describes the standard formats that all language localization bundles must support. The **Data input format** column provides information about the formats that you can use in your own custom variables.

Note:

Localization bundles can, and often do, include additional variable formats that are not listed in this table. These formats are only the ones required of *all* localization bundles. For more information about any additional variable formats that can be part of a particular localization bundle, see the supplemental documentation that came with that bundle.

Format	Description	Data Input Format [1]
Date	Formats a string into a set of audio clips to render the string as an audio variable in a date format, as defined by the localization bundle.	***Type***: String ***Input format***: ***YYYY***[-] ***MM***[-] ***DD*** ***Examples***: `2005-05-25` or `20050525`
Time	Formats a string into a set of audio clips to render the string as an audio variable in a time format, as defined by the localization bundle.	***Type***: String ***Input format***: ***HH***[:] ***MM***[:][***SS***] ***Examples***: `03:15:45` or `03:15` or `031545`

Table continues…

Format	Description	Data Input Format [1]
Number	Formats a number-digit string into a set of audio clips to render the number as an audio variable in a number format, as defined by the localization bundle. Commas and periods are used as delimiters for this format. The localization bundle treats the rightmost delimiter as the decimal indicator. **Note:** Localization bundles, in some cases, can define numbers to be treated as integers, in which decimal and fractional parts are ignored. Make sure you know which number formats in your localization bundle support the use of decimals and fractional parts.	***Type***: String ***Input format***: [-] ***#***[, or .] ***###***[, or .] ***##*** ***Examples***: `102473` or `15,588.345` or `10.123,52`
Currency	Formats a number-digit string into a set of audio clips to render the number as an audio variable in a currency format, as defined by the localization bundle. Commas and periods are used as delimiters for this format. The localization bundle treats the rightmost delimiter as the decimal indicator.	***Type***: String ***Input format***: [-] ***#***[, or .] ***###***[, or .] ***##*** ***Examples***: `102473` or `15,588.345` or `10.123,52`
Character	Formats a string into a set of audio clips to render the string character by character. At a minimum, localization bundles support: • Letters A through Z • Numbers 0 through 9 • Star symbol (*) • Hash symbol (#) • Comma (,) • Space With this format, words are spelled out and numbers are read out one digit at a time. Localization bundles can also support additional characters, including non-Latin characters. For details about which additional characters are supported by your localization bundle, see the supplemental documentation that came with the bundle. Localization bundles ignore any characters that are not supported by that bundle.	***Type***: String ***Input format***: **`xxxxxx...`** ***Examples***: • **`what`** (rendered as “double-you aitch ay tee”) • **`10247`** (rendered as “one zero two four seven”) • **`401K`** (rendered as “four oh one kay”)
1. Characters in brackets [ ] are optional.		

Audio Field properties

Both the **date** and **time** system variables contain fields labeled **audio**. These fields are specifically designed and optimized to be used with audio variables in prompt segments. Therefore, to use these fields, a localization bundle and standard phrases must be installed. For more information about using audio variables, see Audio Variable on page 676.

When these fields are used in prompts, additional properties that govern how the information is presented to the caller can be specified. These properties are dependent on the **Format** property selected:

- When using the **audio** field of the **date** system variable in a prompt, use **Date** for the **Format** property.
- When using the **audio** field of the **time** system variable in a prompt, use **Time** for the **Format** property.

Assuming the correct **Format** property is selected, additional properties presented in the Avaya Properties view depend on which language localization bundle is installed and being used in Orchestration Designer. For details regarding these additional properties, see the supplemental documentation included with the localization bundle.

Chapter 36: Creating scripts for testing

Simulation scripts

Orchestration Designer uses the Avaya Voice Browser (AVB) to simulate speech applications for testing and debugging purposes. As described in Application testing by simulation on page 386, you can use the AVB to simulate a wide variety of scenarios. With the AVB, you can manually enter various inputs. You can also use scripts to automate the entry of inputs.

To work with the AVB, simulation scripts must be in the form of XML files. You can save and store them wherever you want, as long as the AVB is able to find and use them.

Avaya has included sample scripts for both these types within the respective sections. Copy and modify them for your own use as needed.

Using scripts to simulate caller responses

About this task

In cases where you do not want to manually simulate responses made by callers, whether the responses are ASR, DTMF, or action, you can use an XML script to automate the responses.

You can do this if you do not have a microphone attached to your computer, or you do not want to disturb your co-workers.

Note that, if you do not enable scripting in the Avaya Voice Browser (AVB), or if no configuration is defined for a particular command, the AVB performs the default action for that command. See Summary of connector commands on page 598.

Procedure

1. Create the caller response script.
2. Save the script file to the desired location.
3. On the **Avaya Voice Browser Script** tab, select the **Enable script file processing** check box.

   Tip:

   To prevent the AVB from trying to recognize any inputs from a microphone, Avaya recommends that you turn off your microphone when using a script to simulate audio input.

4. Perform one of the following actions:
   - In the **Script file name** field, enter the full path to the script, for example, `C:\temp\simulationScript.xml`.
   - Use the **Browse** button to navigate to and select the script file.
5. Run the application simulation.

Guidelines for creating a response script

To work correctly with the Avaya Voice Browser (AVB), the response script must be in the form of an XML file. You can use any good text editor to create the file.

When creating your script, use the following guidelines:

Note:

For a sample script that illustrates these guidelines, see Example response script on page 591.

- The response script must begin with the following code as the first line of the file:

```
<?xml version="1.0" encoding ="UTF-8"?>
```

- Each response that you want to simulate must adhere to one of the following forms:
  - For ASR recognition responses only:

```
<command name="name" type="type">
<item value="value1"/>
<item value="optionalValue2"/>
</command>
```

Note:

You can have one or multiple **item** entries for each ASR recognition response. If you want to simulate only a single response, use only one **item** entry. If you want to simulate multiple responses to a single ASR attempt, you can add as many **item** entries as you need.

  - For all other responses:

```
<command name="name" type="type" value="value" />
```

where:

  - *name* is the type of response you want to simulate.

Valid types include:

- **rec**: Simulates the results of an ASR or DTMF recognition attempt.
- **tel**: Simulates the response of the telephony system to various situations, mostly errors.
- **record**: Simulates the results of a caller recording attempt.

  - **quit**: Terminates the script and the simulation.

- *type* is the specific type of response. The valid values for this field depend on what you have in the **name** field.
- *value* is the actual content, or value, of the response returned. The valid values for this field depend on what you have in the **name** field.

  For more information, see Valid values for response scripts on page 589.

- For each response that the application expects during the course of the simulation, you must provide a simulated response. During simulation, the AVB uses each command entry in turn as it encounters nodes that require a response.

  If you want to simulate all responses for the entire application, therefore, you must have a separate command entry for each expected response.

  Tip:

  Avaya Voice Browser response scripts are useful when working on long and complex applications. An excellent way to use scripts is to enter only as many responses as needed to get to the point in the application that is currently being worked on.

  For example, you create an application that prompts the caller for twelve different responses. You have completed the application up to the sixth prompt and are working on the seventh and eighth prompts. A script can be created that provides responses for only the first six prompts. In other words, the Avaya Voice Browser uses the scripted responses for the first six prompts and then goes into normal interactive mode so that the seventh and eighth prompts can be tested manually.

Valid values for response scripts

The following table lists and describes the valid values for each of the fields in these formats:

Table 4: Valid values for use in scripts

Name	Type	Value	Comments
rec: Recognition responses			
	asr - spoken responses	Any expected spoken response.	Valid values include any item in an active grammar.
	dtmf - touchtone key presses	One or more key presses	Valid values include numbers, star (*), and pound sign (#) characters.
	err - error	**nomatch**	Simulates a situation in which no caller response matches an expected response.
		noinput	Simulates a situation in which the caller did not respond at all.

Table continues…

Name	Type	Value	Comments
tel: Telephony system responses			
	hangup	n/a	No **value** is required, because this type simulates a caller hanging up. You can either leave this field blank or leave it out entirely.
	err - error	**trans_noanswer**	Simulates an attempted transfer in which the destination number did not answer.
		trans_busy	Simulates an attempted transfer in which the destination number was busy.
		trans_netbusy	Simulates an attempted transfer in which the network was not able to reach the destination because all network lines were busy.
		trans_fardiscon	Simulates a successful call transfer in which the party at the other end hung up.
		trans_netdiscon	Simulates a successful call transfer in which the network for some reason disconnected the call prematurely.
		trans_maxdiscon	Simulates a successful call transfer which was terminated by the system because the length of the call exceeded the maximum allowable amount of time.
record: Responses recorded by the system **Note:** This option does not actually record any responses.			
.	**max** - maximum record time	n/a	Simulates a situation in which the caller exceeded the maximum allowable length of time for a recording and the system terminated the recording. You can either leave the **value** field blank or leave it out entirely.
	done	n/a	Simulates a situation in which the caller completed the recording and signaled that it was complete. You can either leave the **value** field blank or leave it out entirely.

Table continues…

Name	Type	Value	Comments
	err - error	Any positive integer	Simulates a situation in which there was an error in the recording. The value field contains a numeric code that identifies the cause of the error.
	dtmf - touchtone key press	One key press	Simulates the key the caller pressed to terminate the recording.
quit			
	n/a	n/a	Terminates the script and the simulation. Use this option only when you want the script to terminate the simulation. When you use this option, you can either leave the **type** and **value** fields blank or leave them out entirely.

Example response script

The following script is a sample of an caller response script. You can use this sample script as a model to create your own scripts.

The following script contains five command entries that correspond to five prompts in the application call flow:

- The first prompt asks what kind of car the caller would like to buy, gas-powered, electric, or a hybrid.
- The second prompt asks what color vehicle the caller would like to rent.
- The third prompt asks what color the caller would like for the vehicle interior.
- The fourth prompt asks if the caller would like to contact a sales representative.
- The last prompt asks the caller to press one to speak to a sales representative or two to leave a message for a sales representative.

The following response script simulates the appropriate responses to these prompts:

```
<?xml version="1.0" encoding ="UTF-8"?>
<callscript>
<call>
<command name="rec" type="asr">
<item value="hybrid"/>
</command>
<command name="rec" type="asr">
<item value="blue"/>
</command>
<command name="rec" type="asr">
<item value="tan"/>
</command>
<command name="rec" type="asr">
<item value="yes"/>
</command>
```

```
<command name="rec" type="dtmf" value="1"/>
<call>
</callscript>
```

Using scripts to simulate IC and AES connectors

About this task

When you use Interaction Center (IC) or Application Enablement Services (AES) connectors in your Orchestration Designer applications, you can simulate the actions and functions of those connectors with a connector script. A connector script can make it possible to test your application without having to connect to a live IC or system. For more information about IC and connectors, see AES and IC connectors on page 610.

To facilitate a useful simulation of the application, you can script the behavior of the connector simulator to respond with a specific action based on scripted input. For example, if you want to test your application to make sure the system responds properly when it encounters an error condition, you can write and use a script to simulate errors with connectors. Or you want to test the behavior of the IC connector to verify that it does what you intend.

Note that, if you do not enable scripting in the connector simulator, or if no configuration is defined for a particular command, the AVB performs the default action for that command. In most cases, that means the command executes successfully. For more information about default actions, see Summary of connector commands on page 598.

Procedure

1. Create the connector script.

   For more information, including the procedure to do this and a sample script, see Guidelines for creating a connector script on page 592.
2. Save the script file to the desired location, and make a note of where you saved it.
3. On the **Avaya Voice Browser Script** tab, select the **Enable connector simulation file processing** check box.
4. Perform one of the following actions:
   - In the **Simulation file name** field, enter the full path to the script, for example, `C:\temp\simulation.xml`.
   - Use the **Browse** button to navigate to and select the script file.
5. Run the application simulation.

Guidelines for creating a connector script

To work correctly with the Avaya Voice Browser (AVB), the connector script must be in the form of an XML file. You can use any good text editor to create the file.

When creating your script, use the following guidelines:

Note:

For sample scripts that illustrates these guidelines and that you can use as a model, or starting point, for your own scripts, see Example connector scripts on page 596.

- The script must begin with the following code as the first line of the file:

```
<?xml version="1.0" encoding ="UTF-8"?>
```

- The second and last lines of the file must read as follows:

```
<simulator>
</simulator>
```

- Nested between these two tags, you must identify the connector for which the script works, using the following format:

```
<connector name="connectorType"
</connector>
```

where *connectorType* represents the type of connector for which this part of the script works. Valid values for this field include:

- **IC**
- **CTI**

Note:

You can include one *connector* entry for each connector type between the *simulator* tags. That is, you can have one section of your script, between *connector* tags for each type of connector, both IC and CTI. You should not, however, include multiple connector tags for a single connector type within the same script. If you do, the simulator acts upon the first one and ignores any other connector configurations for that connector type.

- Nested between each set of *connector* tags, you configure the commands you want the script to support for that connector simulator.

  The *command* tag set uses the following format:

```
<command name="commandName">
</command>
```

  where *commandName* is the command you want to configure.

  For more information about the commands for each connector type, see Summary of connector commands on page 598.

  You can include multiple *command* entries between each set of *connector* tags.

- Within each *command* entry, you must have one or more sets of configurations. A configuration is a block of XML code in which you define the behavior of the connector simulator for that command. Each configuration is contained within a set of configuration tags that use the following format:

  <configuration>

  </configuration>

- Each *configuration* is made up of sets of *comparisons* and *actions*:
  - A *comparison* instructs the simulator to compare an input received during the call with a certain value. If the comparison evaluates as true, the simulator is to take the corresponding action or actions.

    A *comparison* uses the following format:

    <compare type="*type*" field="*field*" value="*value*" />

    where:

    - *type* represents the type of comparison the simulator is to make. Possible values include:
      - **EQUAL**: Defines a field and a value to compare. With this type, both the **field** and **value** elements are required. You can use the asterisk (*****) as a wildcard for the **value** element.
      - **ALL**: Instructs the simulator to take the action in every case. With this type, you do not use either the ***field*** or the ***value*** element.
    - *field* is a predefined value that must correspond with the command being configured. For more information, including the fields that are valid for each command, see Summary of connector commands on page 598.
    - *value* is the value that the field must have to result in an evaluation of true. If you want to allow any value for the field to evaluate as true, you can use the asterisk (*****) as a wildcard.
  - An *action* instructs the simulator what action to take when the requisite comparison or comparisons evaluate as true.

    An *action* uses the following format:

    <action type="*type*" field="*field*" value="*value*" />

    where:

    - *type* is a predefined action that the command can take. For more information, including the types that are valid for each command, see Summary of connector commands on page 598.
    - *field* is either blank or the name of a known variable or variable field in Orchestration Designer.
    - *value* is the value to be assigned to the variable or variable field designated in the field element.

      A special case action is the error condition. To simulate an error for any command, use the following general format:

      <action type="ERROR" field="" value="*errorMessage*"/>"

      Note that the **field** element is empty. When the simulator encounters an action with this format, the simulator returns the *errorMessage*.

For more information about other configuration options, see Additional configuration options in connector scripts on page 595.

Additional configuration options in connector scripts

In addition to the basic configuration options available with *comparison* and *action* tag sets, you can use other options to refine the way your scripts work.

See the following sections:

- Combining comparisons and actions on page 595
- Selecting one comparison/action set instead of another on page 595

Combining comparisons and actions

You can use multiple comparison and action tags to perform multiple comparisons or actions in the same configuration. Using multiple comparisons and actions is equivalent to performing a Boolean *and* Operation: For the actions to be performed, all comparisons must first evaluate as true. When all comparisons evaluate as true, then all actions are performed.

For example, see the following snippet of code:

```
<configuration>
 <compare type="EQUAL" field="VDUID" value="1122334455"/>
 <compare type="EQUAL" field="workflow" value="myworkflow"/>
 <aon type="SET_OUTPUT" field="workflowoutput" value="someValue"/>
 <aon type="SET_VDU" field="testfield" value="anotherValue"/>
</configuration>
```

In this configuration, the script first checks the **id** field of the **vdu** variable, to see if it contains the value **1122334455**. If it does not, then the command does not execute. If it does, the script continues to the next comparison. The second comparison checks to see if the **workflow** variable has a value of **myworkflow**. If it does, the script then performs both actions.

In this example, the actions the script takes are to:

- Set the **workflowoutput** variable to the value defined by the first action **`value`** element (*someValue*).
- Set the **testfield** field of the vdu variable to the value defined by the second action **`value`** element (*anotherValue*).

Selecting one comparison/action set instead of another

You can use multiple configurations to evaluate a series of comparisons and perform a particular action or set of actions. Using multiple configurations is equivalent to performing a Boolean OR

operation: For the actions to be performed, one particular comparison or set of comparisons must evaluate as true.

Configurations are evaluated in order, so you must take care when placing each configuration to be evaluated. As soon as the simulator evaluates a configuration as true, the simulator performs the corresponding actions. The simulator ignores all remaining configurations for that command.

For example, see the following snippet of code:

```
<command name="Workflow">
 <configuration>
  <compare type="EQUAL" field="VDUID" value="1122334455"/>
  <aon type="SET_OUTPUT" field="workflowoutput" value="someValue"/>
 </configuration>
 <configuration>
  <compare type="EQUAL" field="VDUID" value="*"/>
  <aon type="SET_OUTPUT" field="workflowoutput" value="anotherValue"/>
 </configuration>
</command>
```

In this example, the simulator evaluates the first configuration. If the value of the **vdu** variable **id** field is **1122334455**, the simulator sets the value of the **workflowoutput** field to the first value (*someValue*). The simulator then considers this command as completed and the application continues.

If the value of the **vdu** variable **id** field is anything *other than* **1122334455**, the simulator continues to the second configuration. The second configuration uses the asterisk (*) wildcard for the **value** element. This value means that the second configuration accepts any value for the **vdu** variable **id** field. In this case, the simulator sets the value of the **workflowoutput** field to the second value (*anotherValue*).

Tip:

In this example, because any other value for the vdu variable id field is acceptable, you can also use the key word ALL. In this example, that means you can substitute the code **<compare type="ALL"** in place of the line:

```
<compare type="EQUAL" field="VDUID" value="*"/>
```

You can use this substitute line of code as a “catch-all” for all those conditions that do not match with any specific value.

Example connector scripts

This section contains two sample scripts that illustrate the guidelines established in Guidelines for creating a connector script on page 592. You can use these sample scripts as models to create your own scripts. The scripts include:

- Example IC connector script on page 597
- Example AES connector script on page 597

Example IC connector script

The following script is a sample of an IC connector script. This script tells the simulator to check whether a new call is coming in on channel 10. If the call is coming in on channel 10, the script instructs the simulator to set the **id** field of the **vdu** variable to the value **1122334455**. If the call is coming in on any other channel, the script instructs the simulator to set the **id** field of the **vdu** variable to the value **6677889900**.

```
<?xml version="1.0" encoding="UTF-8"?>
<simulator>
 <connector name="IC">
  <command name="NewCall">
   <configuration>
    <compare type="EQUAL" field="channel" value="10"/>
    <aon type="SET_VDU_ID" field="" value="1122334455/>
   </configuration>
   <configuration>
    <compare type="EQUAL" field="channel" value="*"/>
    <aon type="SET_VDU_ID" field="" value="6677889900"/>
   </configuration>
  </command>
 </connector>
</simulator>
```

Example AES connector script

The following script is a sample of a connector script. You can use connector scripts only to check for error responses when commands are invoked. If no error response is defined in the script for a particular connector, the simulator uses the default action. The default action is usually the successful completion of the item.

The following script simulates various possible error responses when the simulator encounters the designated commands.

```
<?xml version="1.0" encoding="UTF-8"?>
<simulator>
  <connector name="">"
    <command name=Hold">"
      <configuration>
        <compare type="ALL"/>
        <aon type="ERROR" field="" value="Error invoking hold" />
      </configuration>
    </command>
    <command name="Dial">
      <configuration>
        <compare type="EQUAL" field="PhoneNumber" value="4085777734" />
        <aon type="ERROR" field="" value="busy" />
      </configuration>
      <configuration>
        <compare type="EQUAL" field="PhoneNumber" value="4085777735" />
        <aon type="ERROR" field="" value="noanswer" />
      </configuration>
    </command>
    <command name="Retrieve">
      <configuration>
        <compare type="ALL" />
```

```
        <aon type="ERROR" field="" value="Error invoking retrieve" />
      </configuration>
    </command>
    <command name="Disconnect">
      <configuration>
        <compare type="ALL"/>
        <aon type="ERROR" field="" value="Error invoking disconnect"/>
      </configuration>
    </command>
    <command name="Conference">
      <configuration>
        <compare type="ALL"/>
        <aon type="ERROR" field="" value="Error invoking conference"/>
      </configuration>
    </command>
    <command name="Transfer">
      <configuration>
        <compare type="ALL"/>
        <aon type="ERROR" field="" value="Error invoking transfer"/>
      </configuration>
    </command>
  </connector>
</simulator>
```

This script performs the following actions in applications that use connectors:

- If the command is Hold(AES) on page 718, the simulator returns an error message that says, “error invoking hold”.
- If the command is Dial(AES) on page 715, the simulator checks to see what number was dialed. If the number is 4085777734, the simulator returns an error of “busy”. If the number is 4085777735, the simulator returns an error of “noanswer”.
- If the command is Retrieve(AES) on page 719, the simulator returns an error message that says, “error invoking retrieve”.
- If the command is Disconnect (AES) on page 717, the simulator returns an error message that says, “error invoking disconnect”.
- If the command is Conference(AES) on page 711, the simulator returns an error message that says, “error invoking conference”.
- If the command is Transfer (AES) on page 720, the simulator returns an error message that says, “error invoking transfer”.

Summary of connector commands

This section lists and describes all the commands that can be used in conjunction with connector scripts. The commands are divided into the following groups:

- IC connector commands on page 599
- AES connector commands on page 600

IC connector commands

The following table lists and describes the IC connector commands that can be included in connector scripts.

Table 5: IC connector commands(continued)

Command Name	Valid Comparison Field Values	Valid Action Type Values	Default Action [1]
Alarm	alarmname - Contains the name of the alarm, as defined in the Alarm on page 674 item	ERROR	The simulator returns an indication of a successful alarm.
CallGone	VDUID - Contains the value of the **vdu** variable **id** field	ERROR	The simulator returns a successful “call gone” response
GetVDU	VDUID - Contains the value of the **vdu** variable **id** field Name - Contains the name of the workflow	ERROR	The simulator looks up in the simulator VDU table the requested **id** value and returns the value. If the value is not in the VDU table, the simulator returns an error.
NewCall	Channel - Contains the value of the channel on which the call is coming in.	ERROR SET_VDU_ID	The simulator returns the value of the SET_VDU_ID action type to the value of the **vdu** variable **id** field.
SetVDU	VDUID - Contains the value of the **vdu** variable **id** field Name - Contains the name of the workflow	ERROR	The simulator sets the value of the **value** element in the simulator VDU table and returns a response to the effect that the value was set correctly.
Transfer	VDUID - Contains the value of the **vdu** variable **id** field	ERROR	The simulator returns a successful call transfer response.

Table continues…

Command Name	Valid Comparison Field Values	Valid Action Type Values	Default Action [1]
InvokeWorkflow	VDUID - Contains the value of the **vdu** variable **id** field workflow - Contains the name of the workflow to invoke	ERROR SET_OUTPUT - The field for this type must be the name of a variable that matches the name of a variable on the IC system. SET_VDU - The field for this type must be the name of a custom field in the **vdu** variable.	The simulator returns a response to the effect that the workflow ran successfully. At the same time, the simulator sets the designated variables with the values from the SET_OUTPUT and SET_VDU action types.
1. The default action is the action that the simulator performs when no comparison evaluates as true.			

For more information about the IC connector, see IC connector on page 622.

AES connector commands

The following table lists and describes the connector commands you can include in connector scripts:

Table 6: connector commands(continued)

Command Name	Valid Comparison Field Values	Valid Action Type Values [1]	Default Action [2]
CallInfo	*	ERROR - The error code to be returned is defined in the *value* element.	The simulator returns a response that the call information was successfully obtained.
Conference	*	ERROR - The error code to be returned is defined in the *value* element.	The simulator returns a response that the call was successfully conferenced in.

Table continues…

Command Name	Valid Comparison Field Values	Valid Action Type Values [1]	Default Action [2]
Dial	PhoneNumber - Contains the value of the Dial Number variable from the Dial(AES) on page 715 item	ERROR - Error codes for return include: • busy • failed • noanswer • unreachable • unknown	The simulator returns a response that the number was successfully dialed (with an “established” state). Before transferring a call, the state of the Dialed call should be selected.
Disconnect	*	ERROR - The error code to be returned is defined in the *value* element.	The simulator returns a response that the call was successfully disconnected (with a “disconnected” state).
Hold	*	ERROR - The error code to be returned is defined in the *value* element.	The simulator returns a response that the call was successfully placed on hold (with a “held” state).
Retrieve	*	ERROR - The error code to be returned is defined in the *value* element.	The simulator returns a response that the call was successfully retrieved from hold status (with an “established” state).
Transfer	*	ERROR - The error code to be returned is defined in the *value* element.	The simulator returns a response that the call was successfully transferred (with a “transferred” state).
1. Use the value field to return the error code. 2. The default action is the action that the simulator performs when no comparison evaluates as true.			

For more information about the connectors, see AES connectors on page 610.

Chapter 37: Conditional operators

Conditional operator descriptions

Conditional operators are used within Orchestration Designer applications with If on page 739 and Else If on page 723 items.

The following table lists and describes the function of each conditional operator. The table also provides information about how the **Condition** and **If** items respond to each operator.

Table 7: Conditional operators(continued)

Operator	Function/Comments	Response
Equal	Compares the value of two variable values, the “left” value and the “right” to see if the values are equal. This operator treats the two values being compared as character or string values. This operator compares the value of alphabetic characters taking case into account. In other words, this operator treats the letter “A” and the letter “a” as the different values.	• In the **Condition** item, if the two values are equal, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the two values are equal, Orchestration Designer plays the associated prompt.

Table continues…

Operator	Function/Comments	Response
NotEqual	Compares the value of two variable values, the “left” value and the “right” to see if the values are *not* equal. This operator treats the two values being compared as character or string values. This operator compares the value of alphabetic characters taking case into account. In other words, this operator treats the letter “A” and the letter “a” as the different values.	• In the **Condition** item, if the two values are *not* equal, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the two values are *not* equal, Orchestration Designer plays the associated prompt.
GreaterThan	Compares the value of two variable values, the “left” value and the “right” to see whether the value of the left variable is greater than the value of the right variable. This operator treats the two values being compared as character or string values. This operator compares the value of alphabetic characters taking case into account. In other words, this operator treats the letter “A” and the letter “a” as the different values.	• In the **Condition** item, if the value of the left variable is greater than the value of the right variable, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the value of the left variable is greater than the value of the right variable, Orchestration Designer plays the associated prompt.
LessThan	Compares the value of two variable values, the “left” value and the “right” to see whether the value of the left variable is less than the value of the right variable. This operator treats the two values being compared as character or string values. This operator compares the value of alphabetic characters taking case into account. In other words, this operator treats the letter “A” and the letter “a” as the different values.	• In the **Condition** item, if the value of the left variable is less than the value of the right variable, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the value of the left variable is less than the value of the right variable, Orchestration Designer plays the associated prompt.

Table continues…

Operator	Function/Comments	Response
EqualsIgnoreCase	Compares the value of two variable values, the “left” value and the “right” to see if the values are equal. This operator treats the two values being compared as character or string values. This operator compares the value of alphabetic characters without regard to case. In other words, this operator treats the letter “A” and the letter “a” as the same value.	• In the **Condition** item, if the two values are equal, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the two values are equal, Orchestration Designer plays the associated prompt.
NotEqualsIgnoreCase	Compares the value of two variable values, the “left” value and the “right” to see if the values are *not* equal. This operator treats the two values being compared as character or string values. This operator compares the value of alphabetic characters without regard to case. In other words, this operator treats the letter “A” and the letter “a” as the same value.	• In the **Condition** item, if the two values are *not* equal, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the two values are *not* equal, Orchestration Designer plays the associated prompt.
=	Compares the value of two variable values, the “left” value and the “right” to see if the values are equal. This operator compares the two values as number values.	• In the **Condition** item, if the two values are equal, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the two values are equal, Orchestration Designer plays the associated prompt.
!=	Compares the value of two variable values, the “left” value and the “right” to see if the values are *not* equal. This operator compares the two values as number values.	• In the **Condition** item, if the two values are *not* equal, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the two values are *not* equal, Orchestration Designer plays the associated prompt.

Table continues…

Operator	Function/Comments	Response
<	Compares the value of two variable values, the “left” value and the “right” to see whether the value of the left variable is less than the value of the right variable. This operator compares the two values as number values.	• In the **Condition** item, if the value of the left variable is less than the value of the right variable, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the value of the left variable is less than the value of the right variable, Orchestration Designer plays the associated prompt.
<=	Compares the value of two variable values, the “left” value and the “right” to see whether the value of the left variable is less than or equal to the value of the right variable. This operator compares the two values as number values.	• In the **Condition** item, if the value of the left variable is less than or equal to the value of the right variable, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the value of the left variable is less than or equal to the value of the right variable, Orchestration Designer plays the associated prompt.
>	Compares the value of two variable values, the “left” value and the “right” to see whether the value of the left variable is greater than the value of the right variable. This operator compares the two values as number values.	• In the **Condition** item, if the value of the left variable is greater than the value of the right variable, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the value of the left variable is greater than the value of the right variable, Orchestration Designer plays the associated prompt.
>=	Compares the value of two variable values, the “left” value and the “right” to see whether the value of the left variable is greater than or equal to the value of the right variable. This operator compares the two values as number values.	• In the **Condition** item, if the value of the left variable is greater than or equal to the value of the right variable, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the value of the left variable is greater than or equal to the value of the right variable, Orchestration Designer plays the associated prompt.
HasMore	Checks a collection associated with a variable to see if there are any more items in the collection that have not been processed.	• In the **Condition** item, if there are more items to be processed, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if there are more items to be processed, Orchestration Designer plays the associated prompt.

Table continues…

Operator	Function/Comments	Response
HasNoMore	Checks a collection associated with a variable to see if there are any more items in the collection that have not been processed.	• In the **Condition** item, if there are no more items to be processed, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if there are more items to be processed, Orchestration Designer plays the associated prompt.
HasPrevious	Checks a collection associated with a variable to see if there are any previous items in the collection that have not been processed.	• In the **Condition** item, if there are previous items to be processed, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if there are previous items to be processed, Orchestration Designer plays the associated prompt.
HasNoPrevious	Checks a collection associated with a variable to see if there are no previous items in the collection that have not been processed.	• In the **Condition** item, if there are no previous items to be processed, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if there are previous items to be processed, Orchestration Designer plays the associated prompt.
IsEmpty	Checks a variable, the “left” variable, to see if it has any value assigned. If it does not have a value assigned, the operator returns a value of **true**.	• In the **Condition** item, if the variable has no value assigned, that is, it is an empty variable, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the variable is an empty variable, Orchestration Designer plays the associated prompt.
NotEmpty	Checks a variable, the “left” variable, to see if it has any value assigned. If it does have a value assigned, the operator returns a value of **true**.	• In the **Condition** item, if the variable has a value assigned, that is, it is not an empty variable, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the variable is not an empty variable, Orchestration Designer plays the associated prompt.

Table continues…

Operator	Function/Comments	Response
IsTrue	Checks a variable, the “left” variable, to see it if has a value of **true**. This operator processes the variable as a Boolean variable.	• In the **Condition** item, if the variable has a value of **true**, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the variable has a value of **true**, Orchestration Designer plays the associated prompt.
IsFalse	Checks a variable, the “left” variable, to see it if has a value of **false**. This operator processes the variable as a Boolean variable.	• In the **Condition** item, if the variable has a value of **false**, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the variable has a value of **false**, Orchestration Designer plays the associated prompt.
IsAfter	Checks to see if the “left” variable has a date/time value that is later than the “right” variable. This operator assumes that both variables are of the same date or time format.	• In the **Condition** item, if the left variable has a date/time value later than the right variable, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the left variable has a date/time value later than the right variable, Orchestration Designer plays the associated prompt.
IsBefore	Checks to see if the “left” variable has a date/time value that is earlier than the “right” variable. This operator assumes that both variables are of the same date or time format.	• In the **Condition** item, if the left variable has a date/time value earlier than the right variable, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the left variable has a date/time value earlier than the right variable, Orchestration Designer plays the associated prompt.
IsDigits	Checks to see if the variable is of type **Digits**. That is, it checks to see if all the characters of the variable are digits.	• In the **Condition** item, if the variable is of type **Digits**, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the variable is of type **Digits**, Orchestration Designer plays the associated prompt.

Table continues…

Operator	Function/Comments	Response
isNull	Checks to see if the value of the variable is "__DD_NULL".	• In the **Condition** item, if the variable is of type **setNull**, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the variable is of type **setNull**, Orchestration Designer plays the associated prompt.
IsCollection	Checks to see if the variable is of type **Collection**. That is, it checks to see if a collection is associated with a variable.	• In the **Condition** item, if the variable is of type **Collection**, the call flow proceeds to the node designated in the **Next Form** property. • In the **If** item, if the variable is of type **Collection**, Orchestration Designer executes the associate nodes.

Creating single and compound conditions

About this task

You can create the following types of conditions under the If on page 739 and Else If on page 723 condition items:

Single condition. In the **Properties** pane of the **If** or **Else If** item, select an operator, and set the required properties.

Compound condition. Ensure that the **Operator** property in the **Properties** pane of the **If** and **Else If** item is blank, and then use AND on page 675, OR on page 790, or NOT on page 772 to connect multiple expressions to form a compound condition. You can nest a compound condition in another compound condition to create a more complex condition.

The following are the examples for creating conditional logic under the **If** item:

- Use **AND** to create a conditional logic if the caller's age is more than 21 and gender is male.

  To create the preceding logic, in the data node editor, under **If**, drag **AND** from the **Palette** pane. Drag two Expression on page 728  items under **AND**, and then set values for the **Operator** and other properties.

- Use **OR** to create a conditional logic if the caller's age is more than 21, gender is male, and the caller either smokes or is sick.

  In the preceding **AND** condition, under **AND**, drag **OR**. Under **OR**, drag two **Expression** items, and then set values for the **Operator** and other properties.

- Use **NOT** to create a conditional logic if the caller's age is more than 21, gender is male, and the caller is neither sick nor smokes.

  In the preceding **AND** condition, under **AND**, drag **NOT**. Under **NOT**, drag **OR**. Drag two **Expression** items under **OR**, and then set values for the **Operator** and other properties.

Similarly, you can create conditional logic under the **Else If** item. You can use the **Else If** item to build a complex If/Else logic. Orchestration Designer executes the Else If expression only if the preceding If expression and any of the preceding Else If expressions evaluate to False, and the current Else If expression evaluates to True.

Chapter 38: AES and IC connectors

AES connectors

AES connectors

Orchestration Designer applications offer the capability to interface and integrate with two external types of systems: Avaya Interaction Center (IC) systems and Application Enablement Services (AES) systems.

Orchestration Designer applications can integrate with Application Enablement Services servers to extend the capabilities of VoiceXML 2.1 speech applications for H323 based communication (SIP is not supported). Among the limitations of the W3C VoiceXML 2.0 Recommendation is the fact that it has very limited call control functionality.

With connectors in Orchestration Designer applications, you can add call control functionality, such as:

- Placing a caller on hold
- Doing a controlled call transfer
- Conferencing a call

To use the extended call control functionality in Orchestration Designer, you must first add at least one computer telephony server and enable the AES connector functionality. See Enabling AES functionality in Orchestration Designer on page 612.

One of the most common uses of AES in speech applications is to perform call transfers in which you can transfer variable data along with the call. For a suggested procedure to construct a call and data transfer using connectors, see Example procedure for constructing a call and data transfer using AES connectors on page 613.

If you plan to deploy your application that uses AES connectors to an Avaya IVR system, such as Avaya IR or Avaya Experience Portal, you must perform some procedures to enable the Orchestration Designer application to interact correctly with the IVR system. See Configure Orchestration Designer applications to work with Avaya IVR systems on page 621.

For CTIC (AESC) deployment, tsapi.pro needs 2 additional properties for talking to **AES 4.2** through secure connection. The avayaprca,jks comes with cticonnector.war. You just need to change the path to match your deployment. For example:

trustStoreLocation=/opt/apps/cticonnector/WEB-INF/lib/avayaprca.jks

trustStorePassword=password

Limitation

1. You can hear the status of the call if you are doing a blind call or a consult call with transfer on ring set. Note that you cannot hear busy or invalid line (fast busy) because the switch disallows a transfer to a busy line.
2. If you have CTI (AES) nodes in your application, you can't run the application in HTML mode. The application will stop when it hits the CTI (AES) nodes.

Types of AES connector items in Orchestration Designer

Orchestration Designer provides several AES connectors to use when you want to incorporate functionality in your applications:

- **Add Conference Party**: Add a conference party to an already established conference call. See Add Conference Party (AES) on page 708.
- **Conference Party**: Makes it possible to add an additional caller to an existing conference. This additional caller is added to the conference without announcement (unannounced) to the existing callers. Sometimes, this type of call operation is also called *single-step conference*. See Conference Party (AES) on page 712.
- **Blind Call**: Used to transfer the caller to a destination number without knowing the call results. If the transfer fails, the call is lost and the application cannot retrieving it. See Blind Call (AES) on page 709.
- **Consultation Call**: Used to transfer the caller to a destination number while maintaining control of the call. If the call is not successful, the caller will be pulled back from being on hold and the call results can be analyzed for further action. See Consultation Call (AES) on page 712.
- **Call Info**: Collects information about the call and stores the information in a variable.

  This is most useful when passing call data information from an AES parent application to a module. See Call Info (AES) on page 709.
- **Dial**: Dials a telephone number. See Dial(AES) on page 715.
- **Conference**: Takes two telephone numbers passed to it and merges them in a single call session. See Conference(AES) on page 711.
- **Remove Conference Party**: Remove a conference party to an already established conference call. See Remove Conference Party (AES) on page 718.
- **Transfer**: Transfers a call and associated data to another number. See Transfer (AES) on page 720.
- **Disconnect**: Disconnects a designated number from the call. See Disconnect (AES) on page 717.
- **Hold**: Places the caller on a designated number on hold. See Hold(AES) on page 718.

- **Retrieve**: Releases a caller on hold from hold status. See Retrieve(AES) on page 719.

Note:

Data node on page 653 is the only node in which you can use connector items.

Enabling AES functionality in Orchestration Designer

About this task

To use AES connector functionality in Orchestration Designer applications, you must first enable the AES connector servlet in the Tomcat server engine.

After the AES connector is installed in Tomcat, when Tomcat is restarted, the AES connector attempts to establish a connection with the AES simulator in Orchestration Designer. This connection allows you to test and simulate AES functions in your Orchestration Designer applications.

When you enable AES for an application, Orchestration Designer automatically uses the AES connector servlet to collect call info data and store it in a special variable named *callinfo*. See Callinfo variable on page 620.

When you deploy a completed speech application, the AES connector servlet must be redirected so that it no longer attempts to establish a connection with the AES simulator in Orchestration Designer. Rather, it should establish a connection with the actual telephony server (T-server).

Procedure

1. In the Navigator view or the Avaya OD Navigator view, click the project for which you want to enable the AES connector functionality.
2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left navigation pane, click **Orchestration Designer**.
4. Click the **Pluggable Connectors** tab.

   Note:

   This tab is displayed only for speech application project properties.
5. In the **Category** field, click **Telephony**.
6. In the lower pane, select the check box corresponding to **Avaya AES Connector**.
7. Click **OK**.

Example procedure for constructing a call and data transfer using AES connectors

One of the most common uses of AES in speech applications is to perform call transfers in which variable data can be transferred along with the call itself. This type of transfer is different from the VXML call transfer capabilities of Orchestration Designer. The two standard VXML call transfer types, Blind Transfer and Bridged Transfer, both transfer only the call. Blind Transfer and Bridged Transfer do not provide the capability to transfer data with the call.

When constructing a call transfer using the AES connector, there are several guidelines to keep in mind. The following example is not the only way to construct such a call, but is one suggested way. The following process provides the general steps and guidelines to keep in mind when setting up a transfer call with AES connectors. The following process does *not* mention all the prompts, announcements, and other elements that are required to create a complete and functional speech application.

Note:

Support for call transfer functions can vary from one IVR system to another. For more information and details about support for transfer functions on your system, see the documentation for your IVR system.

Note:

This procedure can be simplified using Blind Call (AES) on page 709 and Consultation Call (AES) on page 712.

This example includes the following steps:

Step 1: Set up the call for transfer

Procedure

1. Make sure your application is enabled. Enable the AES connector functionality. For more information, see Enabling AES functionality in Orchestration Designer on page 612.
2. Drag a Data node on page 653 into your application call flow.

   For our example, name this as `XferSetup`.
3. To open the **XferSetup** subflow editor, double-click the node.
4. Drag a Hold(AES) on page 718 item after the **Next** item.

This item keeps the original caller on hold while the system dials the telephone number to transfer to.

Set this item with the following property: **Call ID Variable** = **callinfo**

5. Drag a try/catch around the **Hold** operation in Step c and catch com.avaya.sce.runtimecommon.SCERuntimeException. You should take appropriate action as per your application.
6. Drag a Dial(AES) on page 715 item after the **Hold** item.

   This item uses the value of the **Dial Number Variable** and field to dial the destination telephone number. (The **Dial Number Variable Field** is used only if **Dial Number Variable** is a complex variable)

   Note:

   If you have collected data that you want to transfer with the call, make sure you assign it to a variable. Then, assign that variable to the **UUI Variable** property of the **Dial** item, in the Avaya Properties view.

   Set this item properties. For more information, see Dial (AES) item property description for the example procedure on page 618.
7. Drag a try/catch around the Dial operation in Step e and catch the com.avaya.sce.runtimecommon.SCERuntimeException. Note that the caller is now on hold, so you must retrieve that caller from hold. Refer Step 3 for more details on Retrieve. You should take appropriate action as per your application.
8. Drag a Condition on page 699 item after the **Dial** item and set it up to direct the call flow to a node that will handle the call if the dialing action is completed successfully.

   For this example, name this condition as **CallCompleted**. Set the properties for this condition item. For more information, see CallCompleted condition item property description for the example procedure on page 619.

   It is important to note the difference between the condition statement after the dial operation and the try/catch around the dial operation. The try/catch is there to handle catastrophic errors. If the call does not connect to the destination because it is busy for instance, that is an expected outcome and the state of the call can then be determined, that is, the try/catch is not tripped. However, if the call stops due to unexpected issues, such as connection issues to the AES or network failures, then try/catch handles it.

Next steps

Step 2: Accomplish the call transfer on page 614

Step 2: Accomplish the call transfer

Before you begin

Step 1: Set up the call for transfer on page 613

Procedure

1. Drag a Transfer (AES) on page 720 item into the node subflow.

2. For the **Held Call Info Variable**, use the variable associated with the **Hold** item, in this case, the **callinfo** variable in **XferSetup**.
3. For the **Active Call Info Variable**, use the variable associated with the **DialCall** item, in this case, the **Dial Number Variable**.
4. Drag a try/catch around the **Transfer** operation in Step d and catch the com.avaya.sce.runtimecommon.SCERuntimeException.

   Note that the caller is still on hold, so you must retrieve that caller from hold. Refer Step 3 for more details on Retrieve. You should take appropriate action per your application.
5. Use the **Next** item to direct the call flow to the return node or to another node that will clean up after the call and terminate the call session.

   Note:

   After the call is successfully transferred, no further interaction with the caller is possible. As soon as the transfer is complete, this application proceeds to whatever node you designate in this step.

Next steps

Step 3: Set up the call flow to handle unsuccessful transfer attempts on page 615

Step 3: Set up the call flow to handle unsuccessful transfer attempts

Before you begin

Step 2: Accomplish the call transfer on page 614

About this task

The following procedure consists of all the steps necessary to transfer a call in. You can choose to use the Consultation and Blind Call operations to greatly simplify the process. These operations combine all the manual steps you performed into one operation. By using this you lose some of the control you get by doing them manually.

Why should you choose consultation over blind call? Consultation calls act like bridged calls. Consultation calls allow you to monitor the progress of the call and to retrieve control if it fails. You also have the option when using consultation calls of transferring the call upon a ring detection which allows you to maintain control over early dial errors, but still transfer for instance to a vector or announcement you want the caller to hear. Blind calls do not allow you to regain control of the call in any failure condition. For both types of calls, if a call reaches a queue, it will automatically be transferred into the queue.

Procedure

1. Drag a Retrieve(AES) on page 719 item into the node subflow.

   The **Retrieve** item takes the original caller off hold and allows you to redirect the call based on whatever reason the call was not transferred successfully.

   For the **Call Info Variable** property, select the variable associated with the **Hold** item, in this case, the *callinfo* variable in **XferSetup**.

2. Check the status of the dialed call if it failed by checking the state field of the dialed call variable. You can take appropriate actions per your application based on the failure state:
   - `noanswer`: Call was not answered before the maximum number of rings was reached
   - `unreachable`: Destination cannot be reached
   - `busy`: Destination was busy
   - `unknown`: Call failed for unknown reasons
   - `failed`: Call failed

Next steps

Step 4: Set up for a consultation call on page 616

Step 4: Set up for a consultation call

Before you begin

Step 3: Set up the call flow to handle unsuccessful transfer attempts on page 615

About this task

In the case of a consultation call, the hold/dial and transfer are all rolled up into one operation. When the operation completes successfully, the caller on hold should be in a *transferred* state. If the caller is not in that state, then you can go back and verify the dialed call state to determine the cause of failure that is contained in the **ConsultationCall variable** state field.

Note that you no longer need to retrieve the caller from hold. That operation is intrinsically part of the consultation call. In the event of a failure condition, the caller will be retrieved from hold automatically and you can continue to interact with them (play a prompt or retry the failed transfer for instance).

Procedure

1. Drag a Data Node into your application call flow. For our example, name this `ConsultCall`.
2. To open the ConsultCall subflow editor, double-click the node.
3. Drag a **Consultation Call** item into the editor and name it `ConsultationCall`. This will create a variable that will represent the state of the dialed call. Set the properties.

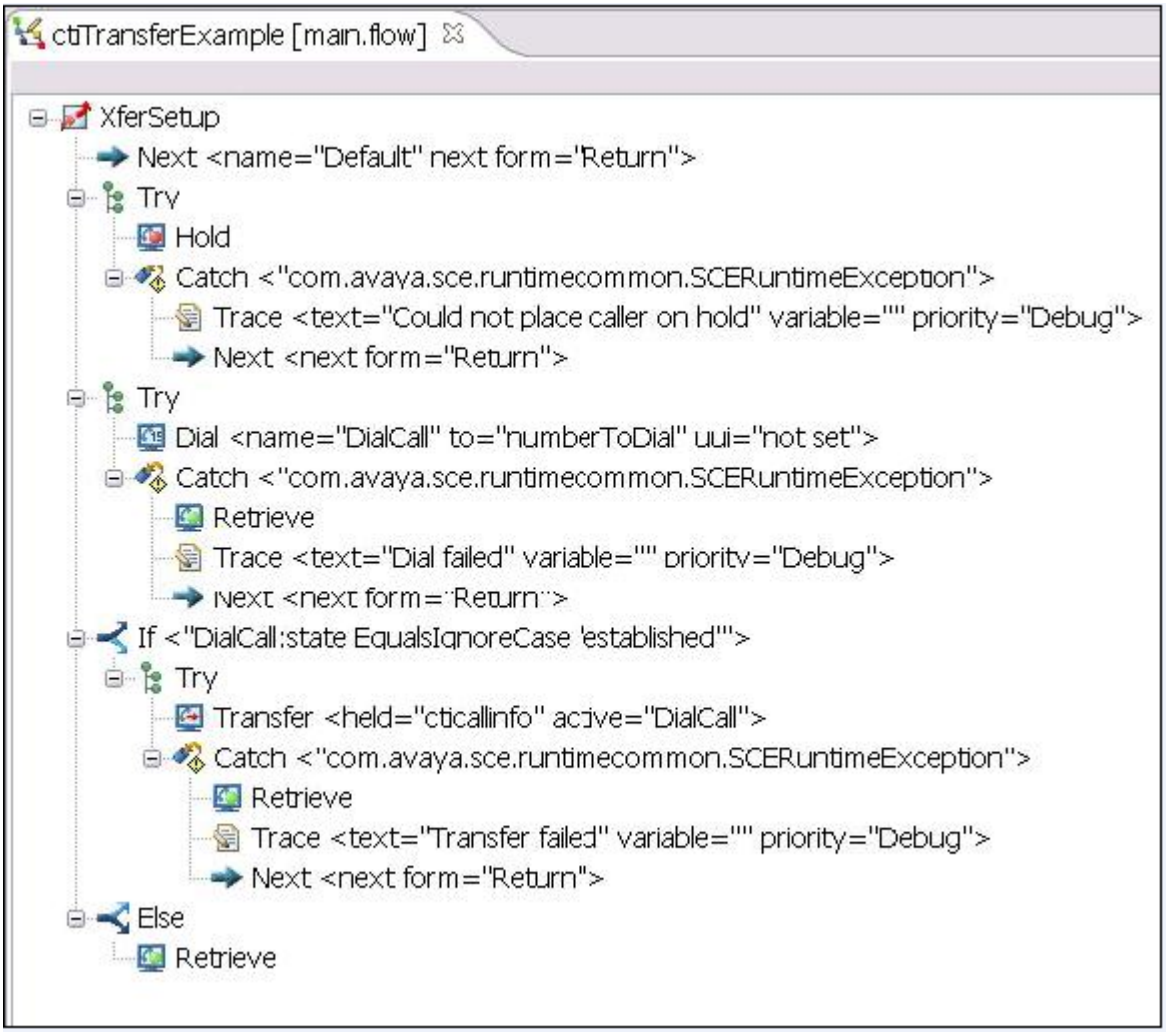
   For more information, see Consultation Call item property description for the example procedure on page 619
4. Drag a try/catch around the Consultation Call operation and catch the com.avaya.sce.runtimecommon.SCERuntimeException. You should take appropriate action as per your application.
5. Drag a Condition item after the Consultation Call item and set it up to direct the call flow to a node that will handle the call if the Consultation Call action is not completed successfully. For this example, name this condition `CallFailed`. Set the properties for this condition item.

   For more information, see CallFailed condition item property description for the example procedure on page 619.

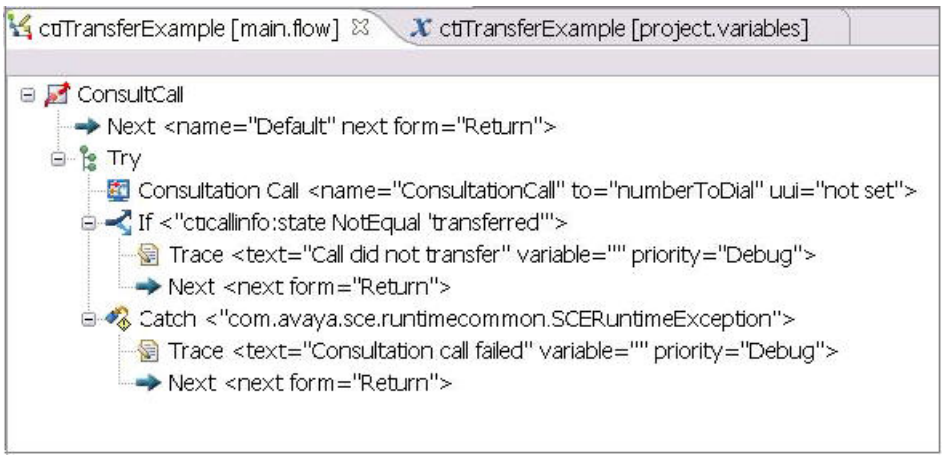
Example

The following examples illustrate how a transfer is done.

Example 1:



Example 2:



Dial (AES) item property description for the example procedure

Name	Description
Name	DialCall
Dial Number Variable	The variable that contains the telephone number to be dialed. If the selected variable is a complex variable, you must also select a **Dial Number Variable Field**.
Dial Number Variable Field	The variable field that contains the telephone number to be dialed. This field is available and required only if the variable selected for the **Dial Number Variable** is a complex variable.
UUI Variable	The variable that contains any user-to-user (UUI) information you want to pass to the transfer destination. UUI is text-based data that accompanies the call. For example, during the course of a call, you collect information from the caller such as an account number, customer preferences, customer identification data, and so forth. You can use transactions to pass this data to a call center agent, where this data can be used to populate a data window on the agent's computer screen. If the selected variable is a complex variable, you must also select a **UUI Variable Field**.
UUI Variable Field	The variable field that contains the UUI information you want to pass to the transfer destination. This field is available and required only if the variable selected for the **UUI Variable** is a complex variable.
UUI as ASCII	Should UUI be passed as ASCII or binary.

Table continues…

Name	Description
Ring max	Enter the number of rings to allow on the far end before the system considers the call a “no answer” and returns the caller to this application for further handling.
Ring max Variable	A variable indicating the ring max.
Ring max Variable Field	If Ring max Variable is complex, use this field.

CallCompleted condition item property description for the example procedure

Name	Description
Operator	EqualsIgnoreCase
Left variable	The variable associated with the **Dial** item (in this case, **DialCall**)
Left variable field	state
Next Form	The form to go to if the condition evaluates as ’true’ **Note:** The “right” value, whether it is a variable or constant, is the condition that must be ’true’ before the call passes to the **Next Form**. In this case, use the constant value so that, if the call is established, the application can pass control of the call on. If not, the default branch is used.
Right variable	blank
Right variable field	blank
Right Constant	established

CallFailed condition item property description for the example procedure

Name	Description
Operator	NotEqual
Left variable	callinfo (the call on hold)
Left variable field	state
Right Constant	transferred

Consultation Call item property description for the example procedure

Name	Description
Name	ConsultationCall.

Table continues…

Name	Description
Caller Call Info Variable	callinfo.
Dial Number Variable	The variable that contains the telephone number to be dialed. If the selected variable is a complex variable, you must also select a Dial Number Variable Field.
Dial Number Variable Field	The variable field that contains the telephone number to be dialed. This field is available and required only if the variable selected for the Dial Number Variable is a complex variable.
UUI Variable	The variable that contains any user-to-user (UUI) information you want to pass to the transfer destination. UUI is text-based data that accompanies the call. For example, during the course of a call, you collect information from the caller such as an account number, customer preferences, customer identification data, and so forth. You can use transactions to pass this data to a call center agent, where this data can be used to populate a data window on the agent's computer screen. If the selected variable is a complex variable, you must also select a UUI Variable Field.
UUI Variable Field	The variable field that contains the UUI information you want to pass to the transfer destination. This field is available and required only if the variable selected for the UUI Variable is a complex variable.
UUI as ASCII	Should UUI be passed as ASCII or binary.
Ring max	Enter the number of rings to allow on the far end before the system considers the call a "no answer" and returns the caller to this application for further handling.
Ring max Variable	A variable indicating the ring max.
Ring max Variable Field	If Ring max Variable is complex, this is the field to use.
Transfer On Ring	True if you want call to transfer when ring is detected.

Callinfo variable

When you enable AES, Orchestration Designer creates a complex variable, **callinfo**, which collects call data about the incoming call.

At run time, when a call is initiated to the Orchestration Designer application, the application automatically sends a **callinfo** command to the connector at the beginning of the call. The T-server then returns the data to populate the fields of the **callinfo** variable.

For information about the **callinfo** variable fields, see callinfo variable fields on page 621.

callinfo variable fields

The following table describes the fields associated with the **callinfo** variable.

For information about the **callinfo** variable, see About the callinfo variable on page 620.

Name	Description
ani	Returns the ANI, or the number that the caller is calling from.
callid	Returns the call identifier created by the system when the call is established. The connector uses this to keep track of call progress.
dnis	Returns the DNIS, or the number that the caller dialed.
state	Returns the current state of the call. Possible values include: • established • ringing • disconnected • hold • queued • failed • transferred
stationextension	This field is not currently used by Orchestration Designer.
ucid	Returns the universal call identifier. This is a unique call ID recognized by all systems. You can use this to query other systems for information about the call.
uui	Returns the contents of the **uui** (user-to-user information) variable field.

Configure Orchestration Designer applications to work with Avaya IVR systems

To deploy a speech application that uses connectors to an Avaya IVR system, either Avaya Interactive Response or Avaya Experience Portal, see Configuring AES settings on page 446.

Configure Channel to Extension Mapping Script for Avaya IR systems

To generate the channel to extension mapping for Avaya IR systems, see Configuring IR Channel Map on page 449.

IC connectors

IC connector

Orchestration Designer applications offer the capability to interface and integrate with two external types of systems: Avaya Interaction Center (IC) systems and Application Enablement Services (AES) systems.

Orchestration Designer IVR applications can integrate with Avaya Interaction Center (IC) systems, such as IC 6.1.3 to 7.2.1, through the IC VOX server, through the IC VRUSM (sold as an additional add-on to the IC), or through the HTTPVox (IC 7.2.1). This integration means that each system can leverage the strengths of the other system in a single speech application.

With the IC connector items in Orchestration Designer IVR applications, you can:

- Raise alarms on the IC system. See Alarm on page 674.
- Transfer calls to the IC system. See Transfer Call on page 849.
- Invoke a workflow on the IC system. See Invoke Workflow on page 746.
- Get and set values in the vdu fields on the IC VOX server. See The vdu and vdu_cache variables on page 623

Note:

For the Avaya Interaction Center VOX Server to recognize the IVR system, it must be appropriately configured. The configuration parameters for the VOX Server are described in the *Avaya Interaction Center VOX Server Programmer's Guide*. If you are using the VRUSM, you will be provided with the configuration details upon purchase of the add-on. Further, see Configuring VOX/VRUSM simulators on page 525 for details on creating a VOX server simulation environment.

Enabling IC functionality in Orchestration Designer

About this task

To use IC connector functionality in Orchestration Designer applications, you must first enable the IC connector servlet in the Tomcat server engine.

After the IC connector is installed in Tomcat, when Tomcat is restarted, the IC connector attempts to establish a connection with the IC simulator in Orchestration Designer. This connection means you can simulate and test IC functions in your Orchestration Designer applications.

When you deploy the completed application, you must redirect the IC connector so that it no longer attempts to establish a connection with the IC simulator in Orchestration Designer. Rather it must try to establish a connection with the actual VOX/VRUSM server on the IC system.

Procedure

1. In the Navigator view or the Avaya OD Navigator view, click the project for which you want to enable the AES connector functionality.
2. On the **Project** menu, click **Properties**.

   The system displays the Properties for <project name> dialog box.
3. In the left navigation pane, click **Orchestration Designer**.
4. Click the **Pluggable Connectors** tab.

   Note:

   This tab is displayed only for speech application project properties.
5. In the **Category** field, click **Contact Center**.
6. In the lower pane, select the check box corresponding to **Avaya IC Connector**.
7. Click **OK**.

The vdu and vdu_cache variables

When you enable IC, Orchestration Designer creates two complex variables, **vdu** and **vdu_cache**, which map to an EDU (Electronic Data Unit) on the IC system.

The **vdu** variable is created with one read-only variable field, labeled **id**. At run time, when a call is initiated to the Orchestration Designer application, the application sends a **new call** command to the IC system. The IC system then returns a unique EDU ID value, in the form of a **vduid** variable value, which is stored in the **id** field of the **vdu** variable in the Orchestration Designer application.

The **vdu_cache** variable mimics the **vdu** variable, but fields cannot be added, deleted, or renamed from in it. As changes are made to the **vdu** variable, the **vdu_cache** variable will also reflect these changes. The **vdu_cache** does not go to the VOX server to get the data *except* for the first time you access it.

You can add other fields to the **vdu** variable and use them to:

- Make variable values available to the IC system.

  When you create a custom field for the **vdu** variable, Orchestration Designer makes the field's value available to the IC system through the IC connector. The IC system obtains the value of the variable field with the **VOX.setvdu** method. For more information see the *Avaya Interaction Center VOX Server Programmer's Guide*.

- Obtain variable values from the IC system.

  The IC system uses the **VOX.getvdu** method to pass variable values to the Orchestration Designer application. For more information see the *Avaya Interaction Center VOX Server Programmer's Guide*.

  Note:

  - To get values from or set values to the IC system, you must have variable fields in your **vdu** variable whose names *exactly* match the names of **vdu** fields on the IC system. There is no way to query the IC system for these field values, so you must be familiar enough with the workflows on the IC system to have the exact names of the fields whose values you want to use.
  - You cannot convert the fields in the **vdu** and the **vdu_cache** variables used in a Web operation to a specific type such as integer, long, and so on. These fields always take String values. If you want to use a **vdu** variable within the Web operation that requires a type to be set, you must assign the value of that **vdu** variable to another local variable and then convert the local variable to a required type.

For more information about the use of the **vdu** or **vdu_variable** variable in Orchestration Designer, see Invoke Workflow on page 746 and Transfer Call on page 849.

For more information about variables in general, see Variables on page 226.

IC connector at run time

At run time, the IVR system that uses the Orchestration Designer speech application configured for IC and the IC system must both be configured to monitor the same channel for incoming calls. The IVR system can be either an Avaya IR or an Avaya Experience Portal system.

As a call comes in on a channel that the IC system is monitoring, the switch notifies the IC system. When the call is recognized on the IVR system, the Orchestration Designer application running on the IVR system sends a **`new call`** command to the IC system. The IC system, in return, creates a new EDU and returns the ID to the application on the IVR system, where it populates the **vdu id** variable field value. This establishes a pipeline to the IC system for the call and makes IC functionality available for use during that session.

Note:

The Avaya Experience Portal system does not use the concept of channels in routing calls, but rather of extensions. So, if you are using the IC connector in an application on a Avaya Experience Portal system, the channel that the IC system is monitoring on the switch must be mapped to a specific extension on the Avaya Experience Portal system. That is, if the IC system is using channel **1234**, the extension you use on theAvaya Experience Portal system must be **1234**. See theAvaya Experience Portal documentation library.

If this dual connection cannot be established within a period of time configurable on the IC system, the IVR system generates an error and discontinues the call.

At the end of the call, the speech application sends a **call gone** command to the IC system to signal that it can also disconnect the call.

IC will set the session.lasterror field when an error occurs during execution of an IC command. You can check this field to obtain detailed error information. If empty, no error has occurred.

You can also choose to disable run-time exceptions by setting the **sage.ic.throwexceptions** flag to false (default is true) in the speech application's Web descriptor parameters. For more information, see Configuring Orchestration Designer project development, runtime, and Java servlets settings on page 511. When an IC error has occurred, you can check the **lasterror** field to determine if the IC call was successful.

In Orchestration Designer 5.0, the preferred way to catch errors is to "catch" **SCERuntimeException**. See Catch on page 693.

Warning:

If IC exceptions are disabled, you must check this field after every call or you risk missing errors that occurred.

Virtual channel mapping in IR

Two or more IR systems may send calls to an appserver. Problems in both the IC and connectors will arise when the channels overlap. The calls may be sent to your application on the same channel, although IR maps them to different extensions. To resolve this problem, you can create a virtual channel mapping that will allow you to map the physical channel to a logical one of a particular IR system. See Configuring IR Channel Map on page 449.

IC failover

VOX specific failover

Fail over is implemented by configuring the IC Connector so that the VOX server initiates the connection to the IC Connector. On the IC server, multiple VOX servers are set up so in case one goes down, another one can be started up. When the IC Connector starts up, it waits for a VOX server to initiate a connection before it is able to serve calls.

If the currently connected VOX server goes down, the IC Connector detects that the VOX connection is inactive and waits for a new VOX to initiate a connection before IC Connector can serve calls again. The user must start a new VOX server, in case one VOX server goes down.

VRUSM/HTTPVox specific failover

The VRUSM supports automatic failover. If you configure multiple VRUSM servers, then the Interaction Center Connector (ICC) communicates with the servers in a round robin fashion, sending each command to a different VRUSM in the list.

If any VRUSM fails either by detection of a failed pong response or if the ICC is unable to communicate with it on a prior request, it is removed from the active server list. The ICC then pings the server on a periodic basis to determine if it is active. If it does become active, it is moved back to the active list of servers for further commands.

Invoke IC and AES connectors using localhost

If you are using a Load Balancer (LB) to direct calls to multiple application servers, you can choose to have your applications invoke the IC and AES Connectors using localhost.

In the earlier releases of Dialog Designer (Dialog Designer is now called Orchestration Designer), you had to use the header information in the Request Object to invoke the connectors. However with a LB, you had to configure the LB to pass through the IP and port of the client in the Request headers, otherwise the header would direct you back to the LB.

In order to bypass this issue, you can select to invoke the connectors using localhost. You must enter the port your Application Server resides on. The port you use is appended to the `web.xml` file and you can change the value by editing the deployed application's `web.xml` file without having to export the application again.

Chapter 39: Avaya Media Processing Server support

Media Processing Server support

Orchestration Designer supports Avaya Media Processing Server (MPS) which offers support for advanced speech, VoiceXML, Session Initiation Protocol, multiple version ISDN and SS7, and other packages and native interfaces enabling MPS to seamlessly integrate with numerous AES applications for fast, intelligent voice call routing and pleasing customer experiences.

For more information about MPS and installing MPS, refer to the MPS documentation on the Avaya support Web site at http://support.avaya.com

To invoke an MPS Developer application from an Orchestration Designer call flow, you can use the **MPS Developer Application** node, which is available in the Palette pane. The **MPS Developer Application** node helps to conveniently set up the com.nortel.ivr.appoffer object with the mandatory parameters.

Note:

You must deploy an Orchestration Designer application, in which you use the **MPS Developer Application** node, to the MPS platform. The **MPS Developer Application** node can successfully invoke the Orchestration Designer application only if you deploy the Orchestration Designer application to the MPS platform.

For more information about the types of nodes available in Orchestration Designer, see Nodes on page 138.

Setting up the MPS Developer Application node

When you drop the **MPS Developer Application** node into the call flow editor, the node that is created already contains the **Object** element with the class ID set to **com.nortel.ivr.appoffer**. The **Object** element contains the required input parameters. You must set the appropriate values for the input parameters to configure the node. To add more input parameters and output parameters, use the **Object Input** and **Object Output** nodes from the **Palette** pane. You can set either a constant value or variable in the **Object Input** node. For each Object Output node, Orchestration Designer creates a variable field in the implicit variable for the **Object** node.

For more information, see MPS Developer Application node on page 658.

When you execute the MPS Developer Application node, the system returns the output values to the corresponding variable fields of the implicit variable for the Object. You can add the following parameters using the Object Output node, and name them the same:

- **callconn**: Indicates whether the call is currently connected. It is true if the call is connected and false if the call is disconnected.
- **status**: Indicates the status of the handoff operation that was initiated through the object **comm.nortel.ivr.appoffer**. It is success on success and failure on failure.
- **appdatalen**: Number of bytes in the application data field **name.appdata**.
- **appdata**: Application specific data, where the format is agreed upon between Orchestration Designer and the MPS developer application.

For information about the parameters supported for MPS developer application, see Parameters supported for MPS developer application on page 628.

Parameters supported for MPS developer application

Parameter name	Optional	Description
poolname	No	Specifies the pool name or specific application that is invoked by VXML. The following formats are supported: **@serviceType** Invoke the application associated with the entry in the field **<serviceType>** on the same MPS and line plusoffset (see param offset) as the VXML application. For example, if the VXML application (running onMPS#3 line#23) specifies poolname="**@peripro**"offset="24", then the MPS Developer application onMPS#2 on line#47 is invoked. This parameter is equivalent to specifying poolname="**@peripro2.47**". Service Type can also be **vamp** because the VXML Interpreter automatically converts a service type of **peripro** to **vamp**. For example, the VXML Interpreter automatically converts a pool name of **@peripro3.5** to **@vamp3.5**. Either method (peripro or vamp) is valid when you specify a Service Type. **@serviceType** ServiceId Invoke the application that matches the service type that is specified in the field <serviceType> and service identifier <serviceId>. For example, specifying poolname="**@peripro3.5** " transfers the call to the MPS Developer application that runs on MPS#3 on line#5.

Table continues…

Parameter name	Optional	Description
appdata	Yes	Application specific data, where the format is agreed upon between the VXML and MPS Developer application. No default value is available. When no value is specified, the **Receive Application Data** data card on the **Line Operations block** of MPS Developer is not used.
timeout	Yes	Specifies the amount of time (in seconds) in which the invocation must be completed before it is stopped. The default value is 30 seconds.
offset	Yes	Specifies the offset value that is used when **@serviceType** is specified for the poolname. The default value is 0.

Example application for invoking MPS developer application from VXML

Orchestration Designer provides a sample application called CallMPSApp in the ISO image.

Chapter 40: Orchestration Designer integration

Transferring UUI data in a SIP configuration

Sending UUI data from a SIP configuration

About this task

You can use the VXML transfers such as Blind transfer, Bridged transfer, and Consultation transfer to send the UUI data and the AAI option to send the shared UUI data from a SIP configuration. The data must be in the C8,{hex string} format, where C8 represents the identifier of the UUI and hex string represents the data value.

For more information, see Blind Transfer on page 678, Bridged Transfer on page 684, and Consultation Transfer on page 699.

Use this procedure to pass **Hello** as the shared UUI.

Procedure

1. Convert **Hello** into hex format.

   The system converts **Hello** into **48656C6C6F** and stores the converted value in the destination variable.

   For more information, see Encoding string data in ASCII hex format on page 631.

2. Depending on the variable that you use to pass as the UUI/AAI parameter, perform one of the following actions:

   - Use the **Insert** operation and the **0** character position to insert the **C8**, identifier string at the beginning of the converted value.
   - Use the **Append** operation to append the converted hex data to a variable that contains the **C8,** identifier string.

     The system changes the destination variable value to **C8,48656C6C6F**.

Encoding string data in ASCII hex format

About this task

To send the UUI data from a SIP configuration, you must first encode the string data in the C8, {hex string} format, where C8 represents the identifier of the UUI and hex string represents the data value. For more information, see Sending UUI data from a SIP configuration on page 630.

Procedure

1. In the Data node editor, drag an Operation on page 777 item under the Data node.
2. In the Properties view, set the **Type** property to **String to ASCII Hex**.
3. In the **Destination Variable** field, click the variable in which you want to store the converted hex format data.
4. If you select a complex variable in the **Destination Variable** field, then in the **Destination Variable Field** field, you must select a complex variable field in which you want to store the converted hex format data.
5. In the **Source Variable** field, click the variable that contains the data that you want to convert in hex format.
6. If you select a complex variable in the **Source Variable** field, then in the **Source Variable Field** field, you must select a complex variable field that contains the data that you want to convert in hex format.

Receiving UUI data in a SIP configuration

Orchestration Designer automatically parses the shared UUI into a collection of ID-value pairs and stores them in the shareduui variable. The ID represents the identifier of the UUI data. For example, C8 is for UUI information and FA is for UCID information. You can iterate the shareduui variable to access each data item included in the shared UUI data.

If you want to view or modify the shared UUI data in a SIP configuration, you must first decode the ASCII hex encoded UUI data to string data. For more information, see Decoding ASCII hex format data to string format data on page 632

To set arbitrary SIP headers for Avaya Experience Portal:

- Using the **AACC Bridged Transfer**, and **AACC Consultation Transfer** nodes:

  Set two external property items for each SIP header, one to set the header name and the other to set the header value. For example,

  property name: AVAYA_SIPHEADER.session.connection.protocol.sip.unknownhdr[i].name

  property value: {header *name*}

  property name: AVAYA_SIPHEADER.session.connection.protocol.sip.unknownhdr[i].value

property value: {header *value*}

The element [i] indexes the array of headers. For example, 0 is for the first SIP header, 1 is for the second SIP header.

- Using the **AACC Blind Transfer** node:

  Append the value string to the end of `"P-Intrinsics="` in the value of **External Property** item property:

  property name:
  `AVAYA_SIPHEADER.session.connection.protocol.sip.referto.header[0] = "P-Intrinsics="`

  property value: `{"P-Intrinsics=*value*"}`

To set Avaya Experience Portal specific UUI/AAI parameters, see Simulation Profile: OD Runtime Session tab field description on page 531.

For more information about UUI data passed in SIP headers, see the “SIP application support” section in *Administering Avaya Experience Portal Guide*.

Decoding ASCII hex format data to string format data

About this task

To view or modify the shared UUI data in a SIP configuration, you must first decode the ASCII hex-encoded data to string data.

Procedure

1. In the Data node editor, drag an **Operation** item under the Data node.
2. In the Properties view, set the **Type** property to **ASCII Hex To String**.
3. In the **Destination Variable** field, click the variable in which you want to store the converted string format data.
4. If you select a complex variable in the **Destination Variable** field, then in the **Destination Variable Field** field, you must select a complex variable field in which you want to store the converted string format data.
5. In the **Source Variable** field, click the variable that contains the data that you want to convert to string format.
6. If you select a complex variable in the **Source Variable** field, then in the **Source Variable Field** field, you must select a complex variable field that contains the data that you want to convert to string format.

Chapter 41: Nodes and Palette options

Nodes and palette items

Orchestration Designer uses nodes in the Call Flow Editor as the basic building blocks for speech application projects. Each node represents a piece of code and a functionality. The Call Flow Editor palette provides access to the nodes and options you use to build call flows. For more information about the basic types of nodes, see About nodes on page 138.

Palettes are the lists, or menus, that appear on the left side of most editors in Orchestration Designer. Palette options are the options and node items that appear in one or more palettes of the editors in Orchestration Designer.

Limitaton

Node (templates), Audio file names, grammar names, project variables, folders and project names cannot contain double byte or extended ascii characters (E.g. Mandarin/Korean). This is because OD uses these names in urls/uris which must be composed of a subset of US ASCII characters. OD prevents the entry of unsupported characters.

Detailed node descriptions

The Call Flow Editor uses a palette to provide access to the nodes you use to build call flows in Orchestration Designer speech projects. Orchestration Designer offers three types of nodes, to build projects, that are available from the Call Flow Editor palette. These nodes are grouped depending on whether you have imported any module nodes into the Orchestration Designer workbench.

The following detailed information is provided for each of the nodes in Orchestration Designer:

- **Type**: Describes what type of node it is. For more information about the three types of nodes, see Nodes on page 138.
- **Purpose**: Describes what the node is designed to do.
- **Behavior**: Describes in detail how the node works, including interdependencies with other nodes, options, or items.
- **Properties**: Indicates what properties, or attributes, are associated with the option or item in the Avaya Properties view. Describes in detail valid values.
- **Default node structure**: Indicates what node options or items the node is populated with (in the node detail editor) when you select it and drag it into the call flow.

Note:

You can change the default flow in the node editor any way you want, but be careful; nodes can be inadvertently broken so that it no longer performs the function it was intended to perform.

The Call Flow Editor has the following nodes:

Application Items (basic nodes)	Templates (composite nodes)
• Data node on page 653 • Form node on page 655 • Menu node on page 656 • Return node on page 662 • Servlet node on page 663 • Sub Flow Begin node on page 664 • Sub Flow Reference node on page 665 • Sub Flow Return node on page 667 • Symbolic node on page 668 • Tracking node on page 669 • VXML Servlet node on page 670	• Announce node on page 634 • AACC Blind Transfer node on page 635 • AACC Bridged Transfer node on page 637 • AACC Consultation Transfer node on page 639 • AACC AML Blind Transfer node on page 641 • AACC AML Bridged Transfer node on page 643 • AACC AML Consultation Transfer node on page 644 • Blind Transfer node on page 646 • Bridged Transfer node on page 647 • Consultation Transfer node on page 648 • Disconnect node on page 654 • MPS Developer Application node on page 658 • Prompt and Collect node on page 659 • Record node on page 660

Announce node

Type

Template (composite) node

Purpose

In a speech application, the purpose of this node is to play an announcement to the caller.

In a message application the purpose of this node is to send an outbound message.

Behavior

When you drag this node into a call flow, you must define a prompt to play to the caller. This node, then, plays that prompt when the node is executed.

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to IQ for reporting.
- **Secure Fetch**: Set this property to true if you want the next node in the call flow or the message flow to get and post data by using HTTPS. If you set this property to true, the system generates the URL for the next node in the call flow as HTTPS and uses the configured fetch secure port.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.

  Tip:

  Orchestration Designer uses the text you enter in this property field for the pop-up hint that appears when your mouse hovers over the node icon in the workspace. If you leave this blank, Orchestration Designer uses the name of the node for the pop-up hint.

Default node structure (Node Detail Editor Flow)

- **Prompt** item: Indicates what prompt Orchestration Designer must use to play the announcement in a speech application and to send the outbound message in a message application.
- **Next** item: Indicates which node comes next in the call flow.

AACC Blind Transfer node

Type:

Template (composite) node

Available from:

Call flow editor of a speech application

Purpose:

Use the **AACC Blind Transfer** node to transfer a call to a SIP-enabled Avaya Aura® Contact Center without verifying that the transfer can be successfully completed or retaining control of the call. In a blind transfer, Orchestration Designer dials the Avaya Aura® Contact Center Controlled Directory Number (CDN) and then transfers the call without any further verification or action. After the call is transferred, the speech application continues to execute until finished, and the call does not return to the speech application.

Note:

Support for call transfer functions can vary from one IVR system to another. For more information about the support for transfer functions on your system, see the documentation for your IVR system.

Behavior:

When you drag the **AACC Blind Transfer** node into the workspace and save the speech project, Orchestration Designer automatically creates a variable with the same name as that of the **AACC Blind Transfer** node because the default flow in the **AACC Blind Transfer** node editor contains a **Blind Transfer** item.

Properties:

- **Name**: Type a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set the **Preferred Path** property to **true** to use the preferred path at the time of application execution. The default value is **true**. If you set the **Preferred Path** property to **false** and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset the value when executing a node that has **Preferred Path** set to **true**. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya Aura® Performance Center for reporting.
- **Secure Fetch**: Set the **Secure Fetch** property to **true** if you want the next node in the call flow to get and post data by using HTTPS. If you set the **Secure Fetch** property to **true**, the system generates the URL for the next node in the call flow by using HTTPS and uses the configured fetch secure port.
- **Comments**: Type any comment that you want to add as a reminder or description of the purpose or content of the node.

  Tip:

  Orchestration Designer uses the text that you enter in the **Comments** property for the pop-up hint that appears when you move the pointer over the node icon in the workspace. If you leave the **Comments** property blank, Orchestration Designer uses the name of the node for the pop-up hint.

Default node structure (Node Detail Editor Flow):

- **Blind Transfer** item: Use the **Blind Transfer** item to specify the Avaya Aura® Contact Center CDN, either hard-coded or contained within a variable, to which the speech application must transfer the call.

  The **Blind Transfer** item also contains a **Prompt** item to play a prompt to the caller before performing the transfer.

  You can also specify the user-to-user information (UUI) and UCID that you want to pass to Avaya Aura® Contact Center. By default, the **Blind Transfer** item within the **AACC Blind Transfer** node uses the **SIP** transfer protocol and **Shared UUI** transfer mode to transfer the call and pass the call-associated data to Avaya Aura® Contact Center.

Note:

UCID consumes 10 bytes of UUI data. Avaya Aura® Communication Manager can pass up to 96 bytes of hex- encoded UUI data.

- **External Property** item: The system adds the following **External Property** items that represent the **P-Intrinsics** SIP header name. Use the **External Property** items to pass the contact-intrinsic data to Avaya Aura® Contact Center along with the call.

  ```
  AVAYA_SIPHEADER.session.connection.protocol.sip.referto.header[0] =
  "P-Intrinsics="
  ```

  Where, you must append the **P-Intriniscs** value string to the end of `"P-Intrinsics="` in the value of **External Property** item property.

  Note:

  You can also pass CAD data by using the P-Intrinsics SIP header. P-Intrinsics can pass up to 1024 bytes of hex-encoded data.

- **Next** item: Use the **Next** item to specify the next node in the call flow that you want to execute.

AACC Bridged Transfer node

Type:

Template (composite) node

Available from:

Call flow editor of a speech application

Purpose:

Use the **AACC Bridged Transfer** node to transfer a call to a SIP-enabled Avaya Aura® Contact Center after verifying that the transfer can be successfully completed, while retaining control of the call. In a bridged transfer, Orchestration Designer dials the Avaya Aura® Contact Center Controlled Directory Number (CDN), waits for a response from the dialed CDN, and then transfers the caller.

If the dialed CDN is busy or there is no answer, the system responds in a prescribed manner, without losing control of the call. If the transfer is successful, the speech application remains in a suspended mode until the transferred call ends. At that point, control returns to the speech application, which continues executing.

Note:

Support for call transfer functions can vary from one IVR system to another. For more information about the support for transfer functions on your system, see the documentation for your IVR system.

Behavior:

When you drag the **AACC Bridged Transfer** node into the workspace and save the speech project, Orchestration Designer automatically creates a variable with the same name as that of the **AACC Bridged Transfer** node because the default flow in the **AACC Bridged Transfer** node editor contains a **Bridged Transfer** item.

Note:

Any audio prompt in or after a transfer node is not played before the call is transferred. To ensure that the prompt is played before the transfers happens, add a transitional audio prompt to the node that transfers the call.

Properties:

- **Name**: Type a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set the **Preferred Path** property to **true** to use the preferred path at the time of application execution. The default value is **true**. If you set the **Preferred Path** property to **false** and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset the value when executing a node that has **Preferred Path** set to **true**. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya Aura®Performance Center for reporting.
- **Secure Fetch**: Set the **Secure Fetch** property to **true** if you want the next node in the call flow to get and post data by using HTTPS. If you set the **Secure Fetch** property to **true**, the system generates the URL for the next node in the call flow by using HTTPS and uses the configured fetch secure port.
- **Comments**: Type any comment that you want to add as a reminder or description of the purpose or content of the node.

  Tip:

  Orchestration Designer uses the text that you enter in the **Comments** property for the pop-up hint that appears when you move the pointer over the node icon in the workspace. If you leave the **Comments** property blank, Orchestration Designer uses the name of the node for the pop-up hint.

Default node structure (Node Detail Editor Flow):

- **Bridged Transfer** item: Use the **Bridged Transfer** item to specify the Avaya Aura® Contact Center CDN, either hard-coded or contained within a variable, to which the speech application must transfer the call. By default, the **Bridged Transfer** item within the **AACC Bridged Transfer** node uses the **SIP** transfer protocol and **Shared UUI** transfer mode to transfer the call and pass the call-associated data to Avaya Aura® Contact Center.

  The **Bridged Transfer** item includes:

  - A **Prompt** item to play a prompt to the caller before performing the transfer.
  - The capability to have the application respond to a specific word, phrase, or touchtone key press. For example, you want to allow callers to return to the main application by making two consecutive pound (#) key presses or uttering a phrase such as “Avaya come back.”
  - Option to specify the user-to-user information (UUI) and UCID that you want to pass to Avaya Aura® Contact Center.

    Note:

    UCID consumes 10 bytes of UUI data. Avaya Aura® Communication Manager can pass up to 96 bytes of hex- encoded UUI data.

- **External Property** item: The system adds two **External Property** items that represent the name and value pair of the SIP header. Use the **External Property** items to pass the contact-intrinsic data to Avaya Aura® Contact Center along with the call.

  To set more SIP headers, in the **AACC Bridged Transfer** node editor, either copy the default **External Property** items that represent the SIP header name and value pair or drag two **External Property** items, one to set the header name and the other to set the header value.

  For example,

  - property name: AVAYA_SIPHEADER.session.connection.protocol.sip.unknownhdr[i].name
    property value: {header *name*}
  - property name: AVAYA_SIPHEADER.session.connection.protocol.sip.unknownhdr[i].value
    property value: {header *value*}

  The element [i] indexes the array of headers. For example, 0 is for the first SIP header, 1 is for the second SIP header.

  Note:

  If you want to pass more contact-intrinsic data, you can use the P-Intrinsics SIP header in the **External Property** items. You can also pass CAD data by using the P-Intrinsics SIP header. P-Intrinsics can pass up to 1024 bytes of hex-encoded data.

- **Next** item: Use the **Next** item to specify the next node in the call flow that you want to execute.

AACC Consultation Transfer node

Type:

Template (composite) node

Available from:

Call flow editor of a speech application

Purpose:

Use the **AACC Consultation Transfer** node to transfer a call to a SIP-enabled Avaya Aura® Contact Center while retaining control of the call if the transfer attempt is unsuccessful. In a consultation transfer, Orchestration Designer dials the Avaya Aura® Contact Center Controlled Directory Number (CDN) and monitors the progress of the transfer.

When the transfer is successfully complete, the caller is disconnected from the speech application and connected to Avaya Aura® Contact Center Manager Server, and the application continues to execute until finished. If the transfer attempt is unsuccessful, the call is brought back to the speech application and the speech application continues to interact with the caller, that is, play prompts, collect inputs, or attempt other transfers.

Note:

Support for call transfer functions can vary from one IVR system to another. For more information about the support for transfer functions on your system, see the documentation for your IVR system.

Behavior:

When you drag the **AACC Consultation Transfer** node into the workspace and save the speech project, Orchestration Designer automatically creates a variable with the same name as that of the **AACC Consultation Transfer** node because the default flow in the **AACC Consultation Transfer** node editor contains a **Consultation Transfer** item.

Properties:

- **Name**: Type a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set the **Preferred Path** property to **true** to use the preferred path at the time of application execution. The default value is **true**. If you set the **Preferred Path** property to **false** and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset the value when executing a node that has **Preferred Path** set to **true**. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya Aura® Performance Center for reporting.
- **Secure Fetch**: Set the **Secure Fetch** property to **true** if you want the next node in the call flow to get and post data by using HTTPS. If you set the **Secure Fetch** property to **true**, the system generates the URL for the next node in the call flow by using HTTPS and uses the configured fetch secure port.
- **Comments**: Type any comment that you want to add as a reminder or description of the purpose or content of the node.

  Tip:

  Orchestration Designer uses the text that you enter in the **Comments** property for the pop-up hint that appears when you move the pointer over the node icon in the workspace. If you leave the **Comments** property blank, Orchestration Designer uses the name of the node for the pop-up hint.

Default node structure (Node Detail Editor Flow):

- **Consultation Transfer** item: Use the **AACC Consultation Transfer** node to specify the Avaya Aura® Contact Center CDN, either hard-coded or contained within a variable, to which the speech application must transfer the call.

  The **Consultation Transfer** item also contains a **Prompt** item to play a prompt to the caller before performing the transfer.

  You can also specify the user-to-user information (UUI) and UCID that you want to pass to Avaya Aura® Contact Center.

  By default, the **Consultation Transfer** item within the **AACC Consultation Transfer** node uses the **SIP** transfer protocol and **Shared UUI** transfer mode to transfer the call and pass the call-associated data to Avaya Aura® Contact Center.

  Note:

  UCID consumes 10 bytes of UUI data. Avaya Aura® Communication Manager can pass up to 96 bytes of hex-encoded UUI data.
- **External Property** item: The system adds two **External Property** items that represent the name and value pair of the SIP header. Use the **External Property** items to pass the contact-intrinsic data to Avaya Aura® Contact Center along with the call.

To set more SIP headers, in the **AACC Consultation Transfer** node editor, either copy the default **External Property** items that represent the SIP header name and value pair or drag two **External Property** items, one to set the header name and the other to set the header value.

For example,

- property name: AVAYA_SIPHEADER.session.connection.protocol.sip.unknownhdr[i].name property value: {header *name*}
- property name: AVAYA_SIPHEADER.session.connection.protocol.sip.unknownhdr[i].value property value: {header *value*}

The element [i] indexes the array of headers. For example, 0 is for the first SIP header, 1 is for the second SIP header.

Note:

If you want to pass more contact-intrinsic data, you can use the P-Intrinsics SIP header in the **External Property** items. You can also pass CAD data by using the P-Intrinsics SIP header. P-Intrinsics can pass up to 1024 bytes of hex-encoded data.

- **Next** item: Use the **Next** item to specify the next node in the call flow that you want to execute.

AACC AML Blind Transfer node

Type:

Template (composite) node

Available from:

Call flow editor of a speech application

Purpose:

Use the **AACC AML Blind Transfer** node to transfer a call to an AML-based Avaya Aura® Contact Center without verifying that the transfer can be successfully completed or retaining control of the call.

In an AACC AML blind transfer, Orchestration Designer reserves a Landing Pad on Avaya Aura® Contact Center Manager Server by using the Open Interface Web Services, dials the Avaya Aura® Contact Center Controlled Directory Number (CDN), and then transfers the call to Avaya Aura® Contact Center without any further verification or action. After Orchestration Designer transfers the call, the speech application continues to execute until finished, and the call does not return to the speech application.

Note:

Support for call transfer functions can vary from one IVR system to another. For more information about the support for transfer functions on your system, see the documentation for your IVR system.

Behavior:

When you drag the **AACC AML Blind Transfer** node into the workspace and save the speech project, Orchestration Designer automatically creates a variable with the same name as that of the **AACC AML Blind Transfer** node because the default flow in the **AACC AML Blind Transfer** node editor contains a **Blind Transfer** item.

Properties:

- **Name**: Type a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set the **Preferred Path** property to **true** to use the preferred path at the time of application execution. The default value is **true**. If you set the **Preferred Path** property to **false** and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset the value when executing a node that has **Preferred Path** set to **true**. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya Aura® Performance Center for reporting.
- **Secure Fetch**: Set the **Secure Fetch** property to **true** if you want the next node in the call flow to get and post data by using HTTPS. If you set the **Secure Fetch** property to **true**, the system generates the URL for the next node in the call flow by using HTTPS and uses the configured fetch secure port.
- **Comments**: Type any comment that you want to add as a reminder or description of the purpose or content of the node.

  Tip:

  Orchestration Designer uses the text that you enter in the **Comments** property for the pop-up hint that appears when you move the pointer over the node icon in the workspace. If you leave the **Comments** property blank, Orchestration Designer uses the name of the node for the pop-up hint.

Default node structure (Node Detail Editor Flow):

- **Blind Transfer** item: The **Blind Transfer** item contains the Landing Pad reservation number that the speech application receives from the Avaya Aura® Contact Center Open Interface Web Services. The speech application uses the Landing Pad reservation number to transfer the call to Avaya Aura® Contact Center.

  The **Blind Transfer** item also contains a **Prompt** item to play a prompt to the caller before performing the transfer.

  You can also specify the user-to-user information (UUI) and UCID that you want to pass to Avaya Aura® Contact Center. By default, the **Blind Transfer** item within the **AACC AML Blind Transfer** node uses the **AML** transfer protocol and **Shared UUI** transfer mode to transfer the call and pass the call-associated data to Avaya Aura® Contact Center by using the Open Interface Web Services.

  Note:

  UCID consumes 10 bytes of UUI data. Avaya Aura® Communication Manager can pass up to 96 bytes of hex- encoded UUI data.

  AML-based solutions do not support SIP UUI data.
- **Next** item: Use the **Next** item to specify the next node in the call flow that you want to execute.

AACC AML Bridged Transfer node

Type:

Template (composite) node

Available from:

Call flow editor of a speech application

Purpose:

Use the **AACC AML Bridged Transfer** node to transfer a call to an AML-based Avaya Aura® Contact Center after verifying that the transfer can be successfully completed, while retaining control of the call.

In an AACC AML bridged transfer, Orchestration Designer reserves a Landing Pad on Avaya Aura® Contact Center Manager Server by using the Open Interface Web Services, dials the Avaya Aura® Contact Center Controlled Directory Number (CDN), waits for a response from the dialed CDN, and then transfers the call.

If the dialed CDN is busy or there is no answer, the system responds in a prescribed manner, without losing control of the call. If the transfer is successful, the speech application remains in a suspended mode until the transferred call ends. At that point, control returns to the speech application, which continues executing.

Note:

Support for call transfer functions can vary from one IVR system to another. For more information about the support for transfer functions on your system, see the documentation for your IVR system.

Behavior:

When you drag the **AACC AML Bridged Transfer** node into the workspace and save the speech project, Orchestration Designer automatically creates a variable with the same name as that of the **AACC AML Bridged Transfer** node because the default flow in the **AACC AML Bridged Transfer** node editor contains a **Bridged Transfer** item.

Note:

Any audio prompt in or after a transfer node is not played before the call is transferred. To ensure that the prompt is played before the transfers happens, add a transitional audio prompt to the node that transfers the call.

Properties:

- **Name**: Type a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set the **Preferred Path** property to **true** to use the preferred path at the time of application execution. The default value is **true**. If you set the **Preferred Path** property to **false** and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset the value when executing a node that has **Preferred Path** set to **true**. The exitPreferredPath value is stored

in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya Aura® Performance Center for reporting.

- **Secure Fetch**: Set the **Secure Fetch** property to **true** if you want the next node in the call flow to get and post data by using HTTPS. If you set the **Secure Fetch** property to **true**, the system generates the URL for the next node in the call flow by using HTTPS and uses the configured fetch secure port.
- **Comments**: Type any comment that you want to add as a reminder or description of the purpose or content of the node.

  Tip:

  Orchestration Designer uses the text that you enter in the **Comments** property for the pop-up hint that appears when you move the pointer over the node icon in the workspace. If you leave the **Comments** property blank, Orchestration Designer uses the name of the node for the pop-up hint.

Default node structure (Node Detail Editor Flow):

- **Bridged Transfer** item: The **Bridged Transfer** item contains the Landing Pad reservation number that the speech application receives from the Avaya Aura® Contact Center Open Interface Web Services. The speech application uses the Landing Pad reservation number to transfer the call to Avaya Aura® Contact Center.

  By default, the **Bridged Transfer** item within the **AACC AML Bridged Transfer** node uses the **AML** transfer protocol and **Shared UUI** transfer mode to transfer the call and pass the call-associated data to Avaya Aura® Contact Center by using the Open Interface Web Services.

  The **Bridged Transfer** item includes:

  - A **Prompt** item to play a prompt to the caller before performing the transfer.
  - The capability to have the application respond to a specific word, phrase, or touchtone key press. For example, you want to allow callers to return to the main application by making two consecutive pound (#) key presses or uttering a phrase such as “Avaya come back.”
  - Option to specify the user-to-user information (UUI) and UCID that you want to pass to Avaya Aura® Contact Center.

    Note:

    UCID consumes 10 bytes of UUI data. Avaya Aura® Communication Manager can pass up to 96 bytes of hex- encoded UUI data.

    AML-based solutions do not support SIP UUI data.
- **Next** item: Use the **Next** item to specify the next node in the call flow that you want to execute.

AACC AML Consultation Transfer node

Type:

Template (composite) node

Available from:

Call flow editor of a speech application

Purpose:

Use the **AACC AML Consultation Transfer** node to transfer a call to an AML-based Avaya Aura® Contact Center while retaining control of the call if the transfer attempt is unsuccessful. In an AACC AML consultation transfer, Orchestration Designer reserves a Landing Pad on Avaya Aura® Contact Center Manager Server by using the Open Interface Web Services, dials the Avaya Aura® Contact Center Controlled Directory Number (CDN), and then monitors the progress of the transfer.

When the transfer is successfully complete, the caller is disconnected from the speech application. The caller is now connected to Avaya Aura® Contact Center Manager Server and the speech application continues to execute until finished. If the transfer attempt is unsuccessful, the call is brought back to the speech application and the speech application continues to interact with the caller, that is, play prompts, collect inputs, or attempt other transfers.

Note:

Support for call transfer functions can vary from one IVR system to another. For more information about the support for transfer functions on your system, see the documentation for your IVR system.

Behavior:

When you drag the **AACC AML Consultation Transfer** node into the workspace and save the speech project, Orchestration Designer automatically creates a variable with the same name as that of the **AACC AML Consultation Transfer** node because the default flow in the **AACC AML Consultation Transfer** node editor contains a **Consultation Transfer** item.

Properties:

- **Name**: Type a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set the **Preferred Path** property to **true** to use the preferred path at the time of application execution. The default value is **true**. If you set the **Preferred Path** property to **false** and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset the value when executing a node that has **Preferred Path** set to **true**. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya Aura® Performance Center for reporting.
- **Secure Fetch**: Set the **Secure Fetch** property to **true** if you want the next node in the call flow to get and post data by using HTTPS. If you set the **Secure Fetch** property to **true**, the system generates the URL for the next node in the call flow by using HTTPS and uses the configured fetch secure port.
- **Comments**: Type any comment that you want to add as a reminder or description of the purpose or content of the node.

  Tip:

  Orchestration Designer uses the text that you enter in the **Comments** property for the pop-up hint that appears when you move the pointer over the node icon in the workspace. If you leave the **Comments** property blank, Orchestration Designer uses the name of the node for the pop-up hint.

Default node structure (Node Detail Editor Flow):

- **Consultation Transfer** item: The **Consultation Transfer** item contains the Landing Pad reservation number that the speech application receives from the Avaya Aura® Contact Center Open Interface Web Services. The speech application uses the Landing Pad reservation number to transfer the call to Avaya Aura® Contact Center.

  The **Consultation Transfer** item also contains a **Prompt** item to play a prompt to the caller before performing the transfer.

  You can also specify the UUI data and UCID that you want to pass to Avaya Aura® Contact Center.

  By default, the **Consultation Transfer** item within the **AACC AML Consultation Transfer** node uses the **AML** transfer protocol and **Shared UUI** transfer mode to transfer the call and pass the call-associated data to Avaya Aura® Contact Center by using the Open Interface Web Services.

  Note:

  UCID consumes 10 bytes of UUI data. Avaya Aura® Communication Manager can pass up to 96 bytes of hex- encoded UUI data.

  AML-based solutions do not support SIP UUI data.

- **Next** item: Use the **Next** item to specify the next node in the call flow that you want to execute.

Blind Transfer node

Type

Template (composite) node

Purpose

The purpose of this node is to transfer a caller without verifying that the transfer can be successfully completed or retaining control of the call. In a blind transfer, the system dials the number and then transfers the caller without any further checking or other action. After the caller is transferred, the application continues to execute until finished, and the caller does not return to the application.

Note:

Support for call transfer functions can vary from one IVR system to another. For more information and details about support for transfer functions on your system, see the documentation for your IVR system.

Behavior

When you drag this node into the workspace, Orchestration Designer automatically creates a variable with the same name. This is because the default flow in the node editor for this node contains a Blind Transfer item.

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to IQ for reporting.
- **Secure Fetch**: Set this property to true if you want the next node in the call flow to get and post data by using HTTPS. If you set this property to true, the system generates the URL for the next node in the call flow as HTTPS and uses the configured fetch secure port.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.

  Tip:

  Orchestration Designer uses the text you enter in this property field for the pop-up hint that appears when your mouse hovers over the node icon in the workspace. If you leave this blank, Orchestration Designer uses the name of the node for the pop-up hint.

Default node structure (Node Detail Editor Flow)

- **Blind Transfer** item: Indicates the telephone number, either hard-coded or contained within a variable, to which the application transfers the caller. Includes an option to play a prompt to the caller before performing the transfer.
- **Next** item: Indicates which node comes next in the call flow.

Bridged Transfer node

Type

Template (composite) node

Purpose

The purpose of this node is to transfer a caller after verifying that the transfer can be successfully completed, while retaining control of the call. In a bridged transfer, the system dials the number, waits for a response from the dialed number, and then transfers the caller.

If the dialed number is busy or there is no answer, the system responds in a prescribed manner, without losing control of the call. If the transfer is successful, the application remains in a suspended mode until the transfer call is ended. At that point, control returns to the application, which continues executing.

Note:

Support for call transfer functions can vary from one IVR system to another. For more information and details about support for transfer functions on your system, see the documentation for your IVR system.

Behavior

When you drag this node into the workspace, Orchestration Designer automatically creates a variable with the same name. This is because the default flow in the node editor for this node contains a **Bridged Transfer** item.

Note:

Any audio prompts in, or after a transfer node, will not be played before the call has been transferred. Putting the audio prompt as a transitional prompt when transitioning to the node that does the transfer ensures that the prompt is played before the transfer happens.

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya IQ for reporting.
- **Secure Fetch**: Set this property to true if you want the next node in the call flow to get and post data by using HTTPS. If you set this property to true, the system generates the URL for the next node in the call flow as HTTPS and uses the configured fetch secure port.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.

  Tip:

  Orchestration Designer uses the text you enter in this property field for the pop-up hint that appears when your mouse hovers over the node icon in the workspace. If you leave this blank, Orchestration Designer uses the name of the node for the pop-up hint.

Default node structure (Node Detail Editor Flow)

- **Bridged Transfer** item: Indicates the telephone number, either hard-coded or contained within a variable, to which the application attempts to transfer the caller. This item includes:
  - An option to play a prompt to the caller before performing the transfer.
  - The capability to have the application respond to a specific word, phrase, or touchtone key press. This is useful, for instance, if you wanted to allow callers to return to the main application by making two consecutive pound (#) key presses or uttering a phrase such as “Avaya come back.”. For more information, see Grammar on page 737.
- **Next** item: Indicates which node comes next in the call flow.

When you drag this node into the workspace, Orchestration Designer automatically creates a variable with the same name.

Consultation Transfer node

Type

Template (composite) node

Purpose

The purpose of this node is to transfer a caller while retaining control of the call if the transfer attempt is unsuccessful. In a consultation transfer, the system dials the number and monitors the progress of the transfer.

When the transfer is successfully completed, the caller is disconnected from the application (the caller is now connected with the called party) and the application continues to execute until finished. If the transfer attempt is unsuccessful, the call is brought back to the application and the application continues to interact with the caller, that is, play prompts, collect inputs, or attempt other transfers.

Note:

Support for call transfer functions can vary from one IVR system to another. For more information and details about support for transfer functions on your system, see the documentation for your IVR system.

Behavior

When you drag this node into the workspace, Orchestration Designer automatically creates a variable with the same name. This is because the default flow in the node editor for this node contains a **Consultation Transfer** item.

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to IQ for reporting.
- **Secure Fetch**: Set this property to true if you want the next node in the call flow to get and post data by using HTTPS. If you set this property to true, the system generates the URL for the next node in the call flow as HTTPS and uses the configured fetch secure port.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.

  Tip:

  Orchestration Designer uses the text you enter in this property field for the pop-up hint that appears when your mouse hovers over the node icon in the workspace. If you leave this blank, Orchestration Designer uses the name of the node for the pop-up hint.

Default node structure (Node Detail Editor Flow)

- **Consultation Transfer** item: Indicates the telephone number, either hard-coded or contained within a variable, to which the application transfers the caller. Includes an option to play a prompt to the caller before performing the transfer.
- **Next** item: Indicates which node comes next in the call flow.

Conversation 

Type:

Template (composite) node

Available from:

Call flow editor of a speech application.

Purpose:

Use the **Conversation** node to create a conversational application running on Nuance 11. This helps to create a mixed initiative form that supports Dragon Voice. This also supports multiple grammar elements at the form level, an Initial element for initial prompting and multiple fields for different slots.

Behavior:

When you drag the **Conversation** item into the flow editor, it sets up a node with one grammar element at the form level, an initial element with one prompt, and one input element with a prompt. You can subsequently add more grammar items to match the model files that you have created in Nuance Experience Studio (NES). You need to setup the prompt element under the initial element, and also add more prompts if necessary. You must rename the input element and configure the prompt. The input element name has to match the slot or concept name from the NES model. When executed, the node generates a mixed initiative form that works with Dragon Voice.

Properties:

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to IQ for reporting.
- **Secure Fetch**: Set this property to true if you want the next node in the call flow or the message flow to get and post data by using HTTPS. If you set this property to true, the system generates the URL for the next node in the call flow as HTTPS and uses the configured fetch secure port.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.

  Tip:

  Orchestration Designer uses the text you enter in this property field for the pop-up hint that appears when your mouse hovers over the node icon in the workspace. If you leave this blank, Orchestration Designer uses the name of the node for the pop-up hint.

Default node structure (Node Detail Editor Flow):

- **Grammar**: Configure grammar to refer to a model file from NES. You need more than one grammar element for the NES model. For more information, see *Nuance documentation*.

- **Initial**: One or more prompt elements are required in this element to setup the initial prompt played to the caller to start a conversation.
- **Input**: The name must match the concept or slot name. A prompt is required under this element so that the system plays it to the caller when the input is missing.

Collect node

Type

Template (composite) node

Purpose

The purpose of this node is to collect the response from the recipient of the email or SMS in the form of SMS files, email files, and attribute present in the message such as subject, email body, and SMS content.

Behavior

When you drag this node into the workspace, Orchestration Designer automatically creates a variable with the same name. This variable contains the following fields:

Tip:

The returns from all these variable fields can be used or manipulated later in the application. For example, if you wanted to ascertain the range of recognized utterances, you can have the contents of the utterance field written to a database. You can use this information later to refine and improve the functioning of your application.

- **confidence**: A number between 0.0 and 1.0 that expresses the degree of confidence the system has that it has correctly recognized the caller utterance. 0.0 indicates minimum confidence and 1.0 indicates maximum confidence. For message applications, confidence is always 1.0.
- **utterance**: The actual words or text in the message.
- **inputmode**: The mode in which the input was provided. For message application, inputmode is always text.
- **interpretation**: The interpretation, by the text browser. In other words, this is the result of how the text browser recognizes and maps the utterance. Interpretation can be different from utterance based on the tag of the grammar.

  Tip:

  When you want to use the response to control the flow of the message, use the **interpretation** field rather than the **utterance** field. Avaya recommends that you use the **interpretation** field in this type of conditional branching situation because the tag of the grammar item is not language dependent. This means that it does not matter whether the actual **utterance** was “yes,” “oui,” “ja,” or some other language form. If the grammar tag is set to interpret each of these responses as “true”, the **interpretation** field maps the response to a value of “true” in each of these cases. This tag mapping helps the application to use the response correctly, irrespective of the utterance.

- **noinputcount**:The number of times the application did not receive any input. This value is set on both successful and unsuccessful recognition of the inputs.
- **nomatchcount:** The number of times the application received input but it did not match the values specified in the application. This value is set on both successful and unsuccessful recognition of the inputs.
- **value**: If the text browser returns a value for the **interpretation** field, this field contains the same value. If the text browser does not return a value for the **interpretation** field, this field contains the value of the **utterance** field (or other optional fields depending on the **Record Utterance** and **Collect Mark Data** properties).

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya IQ for reporting.
- **Secure Fetch**: Set this property to true if you want the next node in the call flow to get and post data by using HTTPS. If you set this property to true, the system generates the URL for the next node in the call flow as HTTPS and uses the configured fetch secure port.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.

  Tip:

  Orchestration Designer uses the text you enter in this property field for the pop-up hint that appears when your mouse hovers over the node icon in the workspace. If you leave this blank, Orchestration Designer uses the name of the node for the pop-up hint.

Default node structure (Node Detail Editor Flow)

- **Input** item: Uses sub-items to indicate:
  - What prompt the message flow uses to collect input from the incoming message.
  - What means the message flow uses to collect the input (Text Grammar item).
  - How the system functions if the caller does not respond (**No Input** item, which has its own associated **Prompt** item).

    Note that this setting overrides any global or application-level settings that you assign in the **AppRoot** node. If you prefer to use the global settings, you can delete this item. For more information about global settings, see AppRoot node event handlers on page 137.
  - How the system functions if the system cannot match the response with any expected response (**No Match** item, which has its own associated **Prompt** item).

    Note that this setting overrides any global or application-level settings that you assign in the **AppRoot** node. If you prefer to use the global settings, you can delete this item. For more information about global settings, see AppRoot node event handlers on page 137.

- **Next** item: Indicates which node comes next in the message flow.

Data node

Type

Application Item (basic) node

Purpose

Use this node to perform a variety of operations on data obtained from sources internal or external to the application or the system on which the application is running. This node is available for data applications. This node is intended primarily to make it possible to:

- Manipulate application or system data.
- Make use of Web services or database operations.
- Perform conditional branching based on variable data.
- Invoke connectors like the IC or AES connectors.
- Invoke 3rd party connectors.
- Create simple and complex conditional expressions.

Behavior

When you drag this node into the call flow, this node sets up a generic form in which the palette is optimized for the purposes described in the previous section. In addition, this is the node in which you can set up and use functions with Interaction Center (IC) and Application Enablement Services (AES) connectors. For more information about IC and AES connectors and their use, see AES and IC connectors on page 610.

For a message application, if you enable the **Notification Connector (Email, SMS)** pluggable data connector (PDC), the **Send SMS** and **Send Email** items are available in the Palette pane of the **Data** node to send the SMS or email message notification.

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to IQ for reporting.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.

  Tip:

  Orchestration Designer uses the text you enter in this property field for the pop-up hint that appears when your mouse hovers over the node icon in the workspace. If you leave this blank, Orchestration Designer uses the name of the node for the pop-up hint.

- **Graphic**: Enter the path and name of the graphic file that you want to display on the web page. For example, `css/images/avaya-logo.png`.
- **Graphic URI Variable**: Select the variable that contains the path of the graphic file. Ensure that you copy the image files in the directory that is accessible through url or uri, and that the value of the variable assigned to the **Graphic URI Variable** property matches the url or uri.
- **Graphic URI Variable Field**: Select the field of the variable that contains the graphic file path.
- **Graphic Style**: Enter the CSS style property or properties for the image. For example, to make the image as wide as the browser window, you can enter the property `width:100%`.
- **Title Message**: Enter the message that you want to display on the web page, below the graphic.
- **Message Style**: Enter the CSS style property or properties for the way you want to display the message. For example, to make the message bold, you can enter the property `fontweight: bold`.

Default node structure (Node Detail Editor Flow)

- **Next** item: Indicates which node comes next in the call flow or the message flow.

Disconnect node

Type

Template (composite) node

Purpose

Use this node when you want to disconnect the caller from the system or a VXML session. This node is usually used when you must perform some type of “clean up” or follow-up actions. It is also used to return data back to other interpreters (for example, a CCXML processor).

Behavior

When you drag this node into the workspace, the application disconnects the caller and proceeds to the next node, as indicated in the **On Disconnect** item.

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to IQ for reporting.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.

Tip:

Orchestration Designer uses the text you enter in this property field for the pop-up hint that appears when your mouse hovers over the node icon in the workspace. If you leave this blank, Orchestration Designer uses the name of the node for the pop-up hint.

Default node structure (Node Detail Editor Flow)

- **On Disconnect** item: Indicates what node the application goes to after disconnecting the caller.

Form node

Type

Application Item (basic) node

Purpose

Use this multipurpose node to perform a wide variety of functions, from collecting and processing caller input to initiating call transfers and more. This node is the most generic of all the nodes and is actually the basis of many of the Templates (Composite nodes).

You can also use this node to specify whether you want the next node in the call flow or message flow to get and post data by using HTTPS.

Behavior

When you use this node into the call flow or message flow, this node sets up a generic form that you can use for a wide variety of purposes.

For a speech application, in terms of VoiceXML, this node sets up a generic form that meets the broad general guidelines for a form as defined in the W3C VoiceXML 2.0 Recommendation.

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya IQ for reporting.
- **Secure Fetch**: Set this property to true if you want the next node in the call flow or message flow to get and post data by using HTTPS. If you set this property to true, the system generates the URL for the next node in the call flow or message flow as HTTPS and uses the configured fetch secure port.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.

Tip:

Orchestration Designer uses the text you enter in this property field for the pop-up hint that appears when you move the mouse pointer over the node icon in the workspace. If you leave this blank, Orchestration Designer uses the name of the node for the pop-up hint.

Default node structure (Node Detail Editor Flow)

There is no default flow in the node editor for this node.

Menu node

Type

Application Item (basic) node

Purpose

The purpose of this node is to present the caller with a limited series, or menu of options. The caller must respond to these options using either touchtone (DTMF) key presses or spoken responses.

In message applications (SMS applications only), the user must select one of the menu options. The application flow executes based on the text that you receive from the user.

Behavior

When you drag this node into the call flow, this node sets up a framework to construct a menu of options for callers, mostly through the options offered on the node editor palette. One palette option, in particular, is optimized for use with this node: the **Choice** item. Use this item, along with the **Prompt** and **Grammar** items, to designate the options available to the caller and create the menu.

In SMS applications, you can create only a **Grammar** item in the **Choice** item. The use of grammar limits the list of recognizable words used for each choice.

When you drag a **Menu** item into the call flow, Orchestration Designer automatically creates a complex variable with the same name. A complex variable contains the following fields, which are populated upon completion of the node:

- **confidence**: A number between 0.0 and 1.0 that expresses the degree of confidence the system has that it has correctly recognized the caller utterance. The number 0.0 indicates minimum confidence, and the number 1.0 indicates maximum confidence. The specific interpretation of this number depends on the ASR platform.
- **utterance**: A text (raw string) summary of what the caller said or keyed in, as recognized by the ASR engine. In the case of a DTMF response, this is the digit sequence of keys that the caller pressed.
- **inputmode**: The mode in which the caller input was provided, either voice or DTMF.
- **interpretation**: The contents of the interpretation tag, as provided by the ASR engine. In other words, this is the result of how the ASR engine recognized and mapped the utterance. If the ASR engine is unable to provide an interpretation, this field remains undefined.

- **value**: If the ASR engine returned an interpretation, this field takes on the same value as the **interpretation** field. If not, this field takes on the value of the **utterance** field (or other optional fields depending on the **Record Utterance** and **Collect Mark Data** properties).

After the ASR engine returns a recognition result and populates these Input variable fields, you can use the outcome to further direct the call. For SMS applications, this field stores the recognized word that the user texts.

Tip:

It is always a good idea to give callers the option to use either a spoken reply or a DTMF key press to select a choice. For example, you can create a prompt for a choice that says, “To hear about your savings options, press one or say “savings.” Then, when setting up the Choice item, you both use an ASR grammar for “savings” and enter `1` in the DTMF field.

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya IQ for reporting.
- **Secure Fetch**: Set this property to true if you want the next node in the call flow to get and post data by using HTTPS. If you set this property to true, the system generates the URL for the next node in the call flow as HTTPS and uses the configured fetch secure port.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.

  Tip:

  Orchestration Designer uses the text you enter in this property field for the pop-up hint that appears when your mouse hovers over the node icon in the workspace. If you leave this blank, Orchestration Designer uses the name of the node for the pop-up hint.
- **Record Utterance**: (True/False). Used for capturing the caller’s spoken utterance. When true, additional fields are added to the variable that store the recording of the caller's utterance, the size of the recording (in bytes) and the duration of the recording (in milliseconds).
- **Collect Mark Data**: (True/False). Used for capturing the most recently encountered mark in a prompt. When true, additional fields are added to the variable that stores the name of the last **Mark** encountered by the VoiceXML platform as well as the time elapsed since the mark was processed.
  - **Bind Name to Node**: (True/False). When true, the name of the variable item is bound to the name of the node. When the parent node is renamed, the name of the variable item is renamed as well; keeping the name of the variable in sync with the node name.

Default node structure (Node Detail Editor Flow)

There is no default flow in the node editor for this node.

MPS Developer Application node

Type

MPS interoperability

Purpose

You can use the **MPS Developer Application** node to invoke an MPS developer application in Orchestration Designer. The **MPS Developer Application** node sets the minimum required parameters and values to call an MPS developer application. The **MPS Developer Application** node is based on VXML object element, hence you need not manually create a VXML code.

Note:

You must deploy an Orchestration Designer application, in which you use the **MPS Developer Application** node, to the MPS platform. The **MPS Developer Application** node can successfully invoke the Orchestration Designer application only if you deploy the Orchestration Designer application to the MPS platform.

Behavior

This item sets up a call flow node that contains the parent object element and its mandatory child parameters. The parent object element has the com.nortel.ivr.appoffer class ID. You can add more parameters. For information about input and output parameters in the **MPS Developer Application** node, see Setting up the MPS Developer Application node on page 627.

Properties

Name: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components.

Preferred Path: Set this property to true to use the preferred path at the time of application execution. The default value is true.

If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has Preferred Path set to true.

The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya IQ for reporting.

Secure Fetch: Set this property to true if you want the next node in the call flow to get and post data by using HTTPS. If you set this property to true, the system generates the URL for the next node in the call flow as HTTPS and uses the configured fetch secure port.

Comments: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.

Tip:

Orchestration Designer uses the text you enter in this property field for the pop-up hint that appears when your mouse hovers over the node icon in the workspace. If you leave this blank, Orchestration Designer uses the name of the node for the pop-up hint.

Prompt and Collect node

Type

Template (composite) node

Purpose

The purpose of this node is to prompt a caller for a response and collect that response in the form of a spoken reply or DTMF (touchtone) key presses.

Behavior

When you drag this node into the workspace, Orchestration Designer automatically creates a variable with the same name. This variable contains the following fields:

Tip:

The returns from all these variable fields can be used or manipulated later in the application. For example, if you wanted to ascertain the range of recognized utterances as received by the ASR server, you can have the contents of the utterance field written to a database. You can use this information later to refine and improve the functioning of your application.

- **confidence**: A number in the range 0.0 to 1.0 that indicates the degree to which the ASR server is confident that it has correctly recognized the spoken response. 0.0 indicates minimum confidence and 1.0 indicates maximum confidence.
- **utterance**: The raw string of words that the ASR server recognized. The presentation and spelling of these words is platform specific. For example, the ASR server can present the number 530 as “five hundred thirty,” “5 hundred 30,” or even “530.” If DTMF key presses were used, this field contains the string of matched digits.
- **inputmode**: The mode in which the caller input was provided, either voice or DTMF.
- **interpretation**: The interpretation, by the ASR server, of the spoken response. This interpretation is determined by the programming of the ASR server itself. This field also contains the tag of the grammar item that the ASR server recognized. The ASR server must be set to interpret the response, including the tag, correctly.

  Tip:

  When you want to use the spoken response of a caller to control the flow of the call, use the **interpretation** field rather than the **utterance** field. Avaya recommends that you use the **interpretation** field in this type of conditional branching situation because the tag of the grammar item is not language dependent. This means that it does not matter whether the actual **utterance** of the caller was “yes,” “oui,” “ja,” or some other language form. If the grammar tag is set to interpret each of these responses as “true,” the **interpretation** field maps the response to a value of “true” in each of these cases. This tag mapping helps the application to use the response correctly, no matter what form the caller utters.
- **noinputcount**:The number of times the application did not receive any caller input.This value is set on both successful and unsuccessful recognition of the inputs.
- **nomatchcount:** The number of times the application received the caller input but it did not match the values specified in the application. This value is set on both successful and unsuccessful recognition of the inputs.

- **value**: If the ASR server returns a value for the **interpretation** field, this field contains the same value. If the ASR server does not return a value for the **interpretation** field, this field contains the value of the **utterance** field (or other optional fields depending on the **Record Utterance** and **Collect Mark Data** properties).

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya IQ for reporting.
- **Secure Fetch**: Set this property to true if you want the next node in the call flow to get and post data by using HTTPS. If you set this property to true, the system generates the URL for the next node in the call flow as HTTPS and uses the configured fetch secure port.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.

  Tip:

  Orchestration Designer uses the text you enter in this property field for the pop-up hint that appears when your mouse hovers over the node icon in the workspace. If you leave this blank, Orchestration Designer uses the name of the node for the pop-up hint.

Default node structure (Node Detail Editor Flow)

- **Input** item: Uses sub-items to indicate:
  - What prompt the call flow uses to collect input from the caller (**Prompt** item).
  - What means the call flow uses to collect the input (ASR or DTMF **Grammar** item).
  - How the system functions if the caller does not respond (**No Input** item, which has its own associated **Prompt** item).

    Note that this setting overrides any global or application-level settings that you assign in the **AppRoot** node. If you prefer to use the global settings, you can delete this item. For more information about global settings, see AppRoot node event handlers on page 137.
  - How the system functions if the system cannot match the response by the caller with any expected response (**No Match** item, which has its own associated **Prompt** item).

    Note that this setting overrides any global or application-level settings that you assign in the **AppRoot** node. If you prefer to use the global settings, you can delete this item. For more information about global settings, see AppRoot node event handlers on page 137.
- **Next** item: Indicates which node comes next in the call flow.

Record node

Type

Template (composite) node

Purpose

The purpose of this node is to prompt the caller for a spoken reply and then to record the response by the caller. The node then saves the recording in the **data/temp** folder.

Behavior

When you drag this node into the workspace, Orchestration Designer automatically creates a variable with the same name. This variable contains the following fields, which are populated on completion of the node:

- **duration**: The length of the recording in milliseconds.
- **size**: The size of the recording in bytes.
- **termchar**: The value of the DTMF key the caller pressed to terminate the recording. If the caller did not press a DTMF key to terminate the recording, this field is undefined.
- **maxtime**: Boolean value that indicates whether the recording was terminated because the caller exceeded the maximum allowable time for the recording (**true**) or not (**false**).
- **utterance**: A text summary of what the caller said or keyed in to terminate the recording.
- **confidence**: A number in the range 0.0 to 1.0 that indicates the degree to which the ASR server is confident that it has correctly recognized the termination key word or phrase. 0.0 indicates minimum confidence and 1.0 indicates maximum confidence.
- **value**: The URL to the audio (*.wav) file that contains the recording of the response of the caller.

  Tip:

  The return values in these variable fields can be used or manipulated later in the application. For example, if you want to play the recording back for the caller to approve, you can create a prompt that uses a phrase variable with a format of *url*. You can then assign the **Record** variable's **value** field to that phrase variable. Or, if you want to assess the general confidence levels of the ASR server, you can record the return from the **confidence** field to a database for later report processing and analysis.

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components. For more information, see Conventions for naming Java components on page 59.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya IQ for reporting.
- **Secure Fetch**: Set this property to true if you want the next node in the call flow to get and post data by using HTTPS. If you set this property to true, the system generates the URL for the next node in the call flow as HTTPS and uses the configured fetch secure port.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.

Tip:

Orchestration Designer uses the text you enter in this property field for the pop-up hint that appears when your mouse hovers over the node icon in the workspace. If you leave this blank, Orchestration Designer uses the name of the node for the pop-up hint.

Default node structure (Node Detail Editor Flow)

- Record on page 810 item: Use this to set options for recording a spoken response. This item uses sub-items to indicate:
  - What prompt the application uses to obtain a spoken response from the caller (Prompt on page 797 item).
  - What node the application uses to handle the recording and finish the call transaction if the caller ends the recording by hanging up (On Disconnect on page 775 item).
  - How the system functions if the caller does not respond (No Input on page 769 item, which has its own associated Prompt on page 797 item).

    Note that this setting overrides any global or application-level settings that you assign in the **AppRoot** node. If you prefer to use the global settings, you can delete this item. For more information about global settings, see AppRoot node event handlers on page 137.
- Next on page 768 item: Indicates which node comes next in the call flow.

Return node

Type

Application Item (basic) node

For sub flow returns, see Sub Flow Return node on page 667.

Purpose

Use this node to exit the application. If you are using a speech, or message (SMS or email), or data project as a module for another application, this node can return the focus and pass parameters to the main application project or to yet another module.

Behavior

The behavior of this node depends on whether the speech or message project is intended for use as a standalone or parent application, or as a reusable module.

In a standalone or parent application, this node tells the VoiceXML browser or the TextXML browser to terminate the application.

In a reusable module, this node tells the VoiceXML to return to the parent application or to proceed to another module. Either way, this node also makes it possible to define output parameters that are passed to the receiving application or module.

For either use, this should be the last node in your call flow. If the speech project is a standalone or parent application, this node is not required, and the application should still successfully terminate. However, Avaya recommends that you include this node anyway. If the speech project is a reusable module, you *must* have this node as the final node in the module.

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components. For more information, see Conventions for naming Java components on page 59.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.

  Tip:

  Orchestration Designer uses the text you enter in this property field for the pop-up hint that appears when your mouse hovers over the node icon in the workspace. If you leave this blank, Orchestration Designer uses the name of the node for the pop-up hint.

The **Return** node of a web application has the following additional properties:

- **Graphic** : Enter the path and name of the graphic file that you want to display on top of the web page. For example, `css/images/avaya-logo.png`.
- **Graphic URI Variable**: Select the variable that contains the path of the graphic file. Ensure that you copy the image files in the directory that is accessible using url or uri, and that the value of the variable assigned to the **Graphic URI Variable** property matches the url or uri.
- **Graphic URI Variable Field**: Select the field of the variable that contains the graphic file path.
- **Graphic Style** : Enter any CSS style property or properties for the way you want the image to be displayed. For example, to make the image as wide as the browser window, you can enter the property “width:100%”.
- **Title Message** : Enter the message that you want to display on top of the web page, below the graphic.
- **Message Style** : Enter any CSS style property or properties for the way you want the message to be displayed. For example, to make he message bold, you can enter the property “**font-weight:bold**”.

Default node structure (Node Detail Editor Flow)

There is no default flow in the node editor for this node.

Servlet node

Type

Application Item (basic) node

Purpose

Use this node to insert custom Java code into your project. For example, you can insert a code written for a business logic or a code that calls other Java code, such as a Java bean, that Orchestration Designer does not support natively. This node is available for data applications.

Behavior

When you drag this node into the call flow or message flow, Orchestration Designer does nothing to process it. During code generation, however, Orchestration Designer generates the appropriate Java classes in the WEB-INF directory to contain and run the code “as is.”

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components. For more information, see Conventions for naming Java components on page 59.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya IQ for reporting.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.

  Tip:

  Orchestration Designer uses the text you enter in this property field for the pop-up hint that appears when your mouse hovers over the node icon in the workspace. If you leave this blank, Orchestration Designer uses the name of the node for the pop-up hint.

Default node structure (Node Detail Editor Flow)

There is no default flow in the node editor for this node.

Sub Flow Begin node

Type

Application Item (sub flow) node

Purpose

Sub flows are used for invoking another callflow in the project. Large projects can be broken up into multiple, smaller callflow files which are invoked by other project callflows using the Sub Flow node.

The Sub Begin Node is the entry point for a sub call flow. Somewhat like the AppRoot/Start node for the main call flow, but the Begin node cannot be edited and does not allow items to be added to it like the AppRoot.

There are two other parts or nodes to a sub flow multiple call flow feature:

- Sub Flow Reference node on page 665: This defines a reference to a sub flow that has been defined in the project (). A sub flow can be referenced multiple times within a call flow.
- Sub Flow Return node on page 667: This is the return point from a sub flow, returning control back to the call flow (called the sub flow). A sub flow can have multiple return nodes. This is similar to the Return node on page 662 from the main call flow, but the sub flow return does not allow items (such as output parameters) to be added to it like the main flow Return node does.

Behavior

Contents of sub flows are hidden to other flows (that is, “black boxes”). You cannot use a “goto” to go to other nodes in other flows.

Properties

- **Name**: The default value is “Begin.” You cannot modify this value.
- **Comments**: Type any comment that you want to add as a description of the purpose or content of the node. You can leave this field blank.

Sub Flow Begin node in a web project has the following additional properties:

- **Graphic**: Enter the path and name of the graphic file that you want to display on the web page. For example, `css/images/avaya-logo.png`.
- **Graphic URI Variable**: Select the variable that contains the path of the graphic file. Ensure that you copy the image files in the directory that is accessible using url or uri, and that the value of the variable assigned to the **Graphic URI Variable** property matches the url or uri.
- **Graphic URI Variable Field**: Select the field of the variable that contains the graphic file path.
- **Graphic Style**: Enter the CSS style property or properties for the image. For example, to make the image as wide as the browser window, you can enter the property `width:100%`.
- **Title Message**: Enter the message that you want to display on the web page, below the graphic.
- **Message Style**: Enter the CSS style property or properties for the way you want to display the message. For example, to make the message bold, you can enter the property `fontweight: bold`.

Default node structure (Node Detail Editor Flow)

There is no default flow in the node editor for this node.

Sub Flow Reference node 

Type

Application Item (sub flow) node

Purpose

Sub flows are used for invoking another callflow in the project. Large projects can be broken up into multiple, smaller callflow files which are invoked by other project callflows using the Sub Flow node. This node is available for data applications.

The Sub Flow Reference node defines a reference to a sub flow that has been defined in the project. A sub flow can be referenced multiple times within a sub flow.

There are two other parts or nodes to a sub flow multiple call flow feature:

- Sub Flow Begin node on page 664:This is the entry point for a sub call flow. Somewhat like the AppRoot/Start node for the main call flow, but the Begin node cannot be edited and does not allow items to be added to it like the AppRoot.

- Sub Flow Return node on page 667: This is the return point from a sub flow, returning control back to the call flow (called the sub flow). A sub flow can have multiple return nodes. This is similar to the Return node on page 662 from the main call flow, but the sub flow return does not allow items (such as output parameters) to be added to it like the main flow Return node does.

Behavior

Contents of sub flows are hidden to other flows (that is, “black boxes”). You cannot use a “goto” to go to other nodes in other flows.

Properties

- **Name**: Type a name for the Subflow reference node. Type a name that reflects the central purpose of the Subflow Reference node. Use naming conventions for Java components. For more information, see Conventions for naming Java components on page 59.
- **Flow Name**: Click the name of the subflow that you want invoke from the flow. This field contains a list of the subflows that are contained within the project.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya IQ for reporting.
- **Comments**: Type any comment that you want to add as a description of the purpose or content of the node. You can leave this field blank.

Sub Flow Reference node in a web project has the following additional properties:

- **Graphic**: Enter the path and name of the graphic file that you want to display on the web page. For example, `css/images/avaya-logo.png`.
- **Graphic URI Variable**: Select the variable that contains the path of the graphic file. Ensure that you copy the image files in the directory that is accessible using url or uri, and that the value of the variable assigned to the **Graphic URI Variable** property matches the url or uri.
- **Graphic URI Variable Field**: Select the field of the variable that contains the graphic file path.
- **Graphic Style**: Enter the CSS style property or properties for the image. For example, to make the image as wide as the browser window, you can enter the property `width:100%`.
- **Title Message**: Enter the message that you want to display on the web page, below the graphic.
- **Message Style**: Enter the CSS style property or properties for the way you want to display the message. For example, to make the message bold, you can enter the property `fontweight: bold`.

Default node structure (Node Detail Editor Flow)

There is no default flow in the node editor for this node.

Sub Flow Return node

Type

Application Item (sub flow) node

Purpose

Sub flows are used for invoking another call flow or message flow in the project. Large projects can be broken up into multiple, smaller call flow or message flow files which are invoked by other project call flow or message flow using the Sub Flow node.

The Sub Return Node is the return point from a sub flow, returning control back to the call flow or message flow (called the sub flow). This is similar to the Return node on page 662 from the main call flow or message flow, but the sub flow return does not allow items (such as output parameters) to be added to it like the main flow Return node does.

There are two other parts or nodes to a sub flow multiple call flow or message flow feature:

- Sub Flow Begin node on page 664:This is the entry point for a sub call flow or message flow. Somewhat like the AppRoot/Start node for the main call flow or message flow, but the Begin node cannot be edited and does not allow items to be added to it like the AppRoot.
- Sub Flow Reference node on page 665: This defines a reference to a sub flow that has been defined in the project (). A sub flow can be referenced multiple times within a call flow or a message flow.

Behavior

Contents of sub flows are hidden to other flows (that is, “black boxes”). You cannot use a “goto” to go to other nodes in other flows.

Properties

- **Name**: Type a name for the Sub Flow Return node. Use naming conventions for Java components. For more information, see Conventions for naming Java components on page 59.

  The system uses this name as the name for the exit point in a Sub Flow Reference node.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya IQ for reporting.
- **Comments**: Type any comment that you want to add as a description of the purpose or content of the node. You can leave this field blank.

Sub Flow Return node in a web project has the following additional properties:

- **Graphic**: Enter the path and name of the graphic file that you want to display on the web page. For example, `css/images/avaya-logo.png`.
- **Graphic URI Variable**: Select the variable that contains the path of the graphic file. Ensure that you copy the image files in the directory that is accessible using url or uri, and that the value of the variable assigned to the **Graphic URI Variable** property matches the url or uri.

- **Graphic URI Variable Field**: Select the field of the variable that contains the graphic file path.
- **Graphic Style**: Enter the CSS style property or properties for the image. For example, to make the image as wide as the browser window, you can enter the property `width:100%`.
- **Title Message**: Enter the message that you want to display on the web page, below the graphic.
- **Message Style**: Enter the CSS style property or properties for the way you want to display the message. For example, to make the message bold, you can enter the property `fontweight: bold`.

Default node structure (Node Detail Editor Flow)

There is no default flow in the node editor for this node.

Symbolic node

Type

Application Item (basic) node

Purpose

Use this node as a “goto” node, to redirect the application to another node. This node is useful when you want several nodes all to go to the same destination node or just to keep from cluttering up the workspace with a confusing web of connector lines. This node is available for data applications.

Behavior

When you drag the Symbolic node into the Call Flow or Message Flow Editor main workspace, you must set its one property to go to some other node during simulation and run time, then the application automatically jumps to the indicated node.

When you set the node for the Symbolic node to go to, Orchestration Designer automatically adds a Symbolic node icon element in the upper left area of the node icon to which the Symbolic node goes.

When you click a Symbolic node in the main workspace, Orchestration Designer automatically highlights in yellow the node that the Symbolic node is set to go to. This highlighting makes it easier to locate the destination of the Symbolic node.

Further, Symbolic node references can be quickly duplicated from a selected node in the Call Flow or Message Flow Editor by right-clicking on any existing node, and selecting **Create New Reference** from within the context menu.

Properties

Because the Symbolic node is just a pointer to another node, its properties, as displayed and set in the Avaya Properties view, are different from all the other nodes:

- **Node**: Indicates the node that the Symbolic node points to. You can use the drop-down list to select the node you want to point to.

Default node structure (Node Detail Editor Flow)

There is no default flow in the node editor for this node.

Tracking node

Type

Application Item (basic) node

Purpose

Use this node to log:

- Debug and tracing information about the application.
- Event and alarm data to a system log.
- Performance and other data that can be put into a report.

Behavior

In Avaya Experience Portal (VP) systems, error and report data are sent to a log file on the Avaya Experience Portal Management System (VPMS), from which you can generate reports to analyze application operations.

The VP system logs the trace/debug data locally on the application server. The path to the log file is:

`//applicationName/data/log/trace-YYYY-MM-DD.log`

where:

- *applicationName* is the name of your application project
- *YYYY-MM-DD* is the year-month-day that the log file was generated

On Avaya IR systems, all logs are local to the application server. The Orchestration Designer application stores them in the `//applicationName/data/log` directory. You can view the data and generate reports from the log files located there.

The trace for the current day is `//applicationName/data/log/trace.log`.

Note:

Open and review logs using a text editor, or import the data into a report generating application. Except for the Application Report on the Avaya Experience Portal system, Avaya does not provide reporting software in the current release.

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components. For more information, see Conventions for naming Java components on page 59.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to IQ for reporting.

- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.

  Tip:

  Orchestration Designer uses the text you enter in this property field for the pop-up hint that appears when your mouse hovers over the node icon in the workspace. If you leave this blank, Orchestration Designer uses the name of the node for the pop-up hint.

Tracking node in a web project has the following additional properties:

- **Graphic**: Enter the path and name of the graphic file that you want to display on the web page. For example, `css/images/avaya-logo.png`.
- **Graphic URI Variable**: Select the variable that contains the path of the graphic file. Ensure that you copy the image files in the directory that is accessible using url or uri, and that the value of the variable assigned to the **Graphic URI Variable** property matches the url or uri.
- **Graphic URI Variable Field**: Select the field of the variable that contains the graphic file path.
- **Graphic Style**: Enter the CSS style property or properties for the image. For example, to make the image as wide as the browser window, you can enter the property `width:100%`.
- **Title Message**: Enter the message that you want to display on the web page, below the graphic.
- **Message Style**: Enter the CSS style property or properties for the way you want to display the message. For example, to make the message bold, you can enter the property `fontweight: bold`.

Default node structure (Node Detail Editor Flow)

- Next on page 768 item: Indicates which node comes next in the call flow or the message flow.

VXML Servlet node

Type

Application Item (basic) node

Purpose

Use this node to insert custom VoiceXML code into your project. The purpose of this node is to make it possible to write and employ custom VoiceXML code for operations that Orchestration Designer does not natively handle.

Behavior

The framework for this node generates the VoiceXML header and footer (*<vxml>* and *</vxml>*) and the form header and footer (*<form>* and *</form>*). The custom code you add within the Java class for this node is inserted between the form header and footer.

After you drag this node into the call flow, you must edit the Java class for the node. Because this node permits you to write and employ your own VoiceXML, Avaya is not responsible for the results. It is possible to write code that will prematurely exit the application, transfer a caller out of the system, and perform other actions that the parent application is not aware of.

For information about editing the Java class of a node, see Editing the Java class of a node on page 155.

Caution:

Use this node with caution as this can cause your applications to stop responding or display run-time errors.

Note:

The ability to integrate variables into and from this node is not easily supported. Do not use this node to pass variable values.

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components. For more information, see Conventions for naming Java components on page 59.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to IQ for reporting.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.

  Tip:

  Orchestration Designer uses the text you enter in this property field for the pop-up hint that appears when your mouse hovers over the node icon in the workspace. If you leave this blank, Orchestration Designer uses the name of the node for the pop-up hint.

Default node structure (Node Detail Editor Flow)

- Next on page 768 item: Indicates which node comes next in the call flow.

Detailed Palette option descriptions

Most of the editors in Orchestration Designer use palettes as a way of providing the options and node items that can be used within each editor.

The following information is provided for each of the options and items in this section:

- **Type**: Identifies which type of option or item it is. This helps identify what the option or item is used for.
- **Available from**: Indicates where in Orchestration Designer the option or item can be used.
- **Purpose**: Describes what the option or item is designed to do.
- **Behavior**: Describes in detail how the option or item works, including interdependencies with other options or items.

- **Properties**: Indicates what properties, or attributes, are associated with the option or item in the Avaya Properties view. Describes in detail valid values.

The following options and items appear on various palettes within Orchestration Designer:

Coordination Sequence on page 714. Default on page 722. Dial (AES) on page 715 Disconnect (AES) on page 717	Input Parameter on page 745 Invoke Workflow on page 746 Join Condition on page 750 LaunchCCXMLCall on page 753 LaunchVXMLCall on page 758 Link on page 762. Location Input on page 870 Loop Collection on page 763 Loop Variable Collection on page 871 Mark on page 764 Media on page 764 Media Page on page 765	Record on page 810 Region on page 814 Report on page 815 Report Alarm on page 818 Return Event on page 820 Say As on page 821 Send Email on page 822 Send SMS on page 825 Send to AACC on page 828 Sequence on page 831 Set OD Date/ Time on page 828 Set VDU Fields on page 830 Simple Variable on page 832	Web Element on page 838 Web Service (Operation) on page 857 Web Service (REST) on page 857

Alarm

Type

IC item

Available from

Data node on page 653 only

Purpose

The **Alarm** item makes it possible to send an alarm from the speech application to the Alarm Manager on the Interaction Center (IC) system.

Behavior

Before you can use this or any other IC item, you must enable IC for Orchestration Designer. For more information about the ability of Orchestration Designer to interact with IC, and to enable IC in your Orchestration Designer applications, see About the IC connector on page 622.

When the speech application encounters the **Alarm** item, it passes the values for the properties to the IC VOX server, which relays them to the IC system.

Properties

- **Name** - Enter the name to assign the alarm.

  This name is passed along to the IC system, where the name is used as an identifier.

- **Priority** - Select the priority to assign the alarm. Options include:
  - **Low**
  - **High**
  - **Info**
  - **EMERGENCY**

    The priority you assign here determines the alarm that is generated on the IC system. For the interpretation of what these alarms mean, see your IC documentation.

- **Description**: Enter in this field a short description of what the alarm is for. This description is passed verbatim to the Alarm Manager on the IC system, as part of the alarm event.
- **Description Variable and Description Field**: Allows you to pass in a variable or field that contains the description instead of hardcoding the text.

AND 

Type

Boolean operator

Available from

Data node

Any level tab in the Prompt File Editor

Purpose

You can use the **AND** Boolean operator to evaluate if all of its child expressions are true.

Behavior

This item sets a compound condition that evaluates to true only if all of its child expressions are true. This item is used as a parent item and can be nested under If on page 739 or any other Boolean operator. This item connects all its child expressions logically. Its child expressions can be either a single expression or a compound condition.

For more information, see Creating single and compound conditions on page 608.

Properties

Boolean Operator: Shows the name of the Boolean operator. If you want to select any other Boolean operator, select that operator from the list.

Audio Variable

Type

Prompt Editor Palette item

Available from

Any level tab in the Prompt File Editor. For more information about the Prompt File Editor, see Prompt file editor in a speech application on page 207.

Purpose

Audio variables use prepackaged and prerecorded variables to play back standardized type of information, such as date, time, and currency, to the callers. You can use audio variables to use prerecorded speech, instead of text-to-speech, to play standardized type of dynamically changing information to the callers.

Behavior

The run-time framework treats audio variables in speech applications differently than other types of variables. When the Orchestration Designer runtime encounters an audio variable, the localization bundle parses the audio variable value into parts, and maps the parts to the predefined elements. The localization bundle then maps the elements to the appropriate pre-recorded .wav files. At run time, the system plays back the prerecorded files.

For example, your speech application has a variable that is defined to take the value of the current date. If you use this variable as an audio variable segment in a prompt, the localization bundle parses the date variable value into three parts and maps each part to the month, day, and year elements. It then maps each of these elements to the appropriate prerecorded audio files.

Before using audio variables, be sure that you install the localization bundle and standard phrases for the language that you want to use in the speech application.

To install the localization bundle, see Localization bundles on page 289.

To install a set of standard phrases, see Installing the standard phrasesets of a localization bundle on page 293.

For audio variables to work properly, be sure that:

- The format of the variable that is used as an audio variable matches the format that the localization bundle expects and can recognize. For more information about the formats that localization bundles can recognize, see Variable formats in localization bundles on page 583.
- You select the correct format for the audio variable when you use the audio variable in a prompt. For more information about formats for audio variables, see System variable fields and properties on page 563.

For information about how to use system variables as audio variables, see Audio Field properties on page 586.

Properties

- Variable: Select the variable you want to use. If the variable you select is a complex variable, you must also select a **Variable** field.
- Variable Field: Select the variable field you want to use.

  The contents of this list depend on which variable you selected in the Variable property: Only the fields associated with the selected variable are presented in this list.

  Note:

  If the variable you selected in the **Variable** field is a simple variable, this field is inactive.
- Format: Select the format you want to use for the variable.

  The options available depend on the language and the localization bundle you are using. For more information about formats available with localization bundles, see Variable formats in localization bundles on page 583.

Audio Constant

Type

Prompt Editor Palette item

Available from

Any level tab in the Prompt File Editor. For more information about the Prompt File Editor, see Prompt file editor in a speech application on page 207.

Purpose

Audio Constant makes use of a literal string to play standardized type of information to the callers. These standardized type of information include time, date, and currency.

Behavior

With Audio Variable, Orchestration Designer takes the contents of a variable at run time and converts it to a sequence of .wav files. With Audio Constant, you just have to enter the text that you want to convert to a literal string. For example, if you enter the value ‘101’ in the prompt segment properties, Orchestration Designer converts the string into `hundred.wav` and `one.wav` files and plays phrases one, hundred, one at run time.

The net effect of Audio Variable and Audio Constant is the same. The only difference is that Audio Variable gets data from an Orchestration Designer variable, and Audio Constant uses a literal string that cannot change.

Properties

- **Constant Value**: Type the string that you want to convert to a sequence of .wav files.
- **Format** : Select the format for the audio constant value. The options are: **Number**, **Date**, **Time**, **Currency**, and **Spelling characters**. The system displays other properties based on the format that you select in the **Format** field. Set the other properties as per your requirements. For example, if you select **Time** in the **Format** field, the system displays another property called as **Time Format**. You can set the format for the time in the **Time Format** field.

Blind Transfer

Type

Form item

Available from

The following nodes of a speech application:

- Announce node on page 634
- Blind Transfer node on page 646, AACC Blind Transfer node on page 635, and AACC AML Blind Transfer node on page 641. Pre-populated by default.
- Form node on page 655

Note:

Although the **Blind Transfer** item appears in the Palette pane of other nodes, such as **Prompt and Collect** node, **Bridged Transfer** node, **AACC Bridged Transfer** node, **AACC AML Bridged Transfer** node, and **Record** node, it is not actually available for use in these nodes without destroying the intended functionality of these nodes. The **Blind Transfer** item is intended primarily for use only in the **Blind Transfer** node, **AACC Blind Transfer** node, and **AACC AML Blind Transfer** node.

Purpose

The **Blind Transfer** item provides the elements that are required to set up a blind transfer.

If you use the **Blind Transfer** item in the **Blind Transfer** node, then the speech application transfers a call to the telephone number that you specify and exits without verifying whether the transfer was successful.

If you use the **Blind Transfer** item in the **AACC Blind Transfer** node, then the speech application transfers a call to the Avaya Aura® Contact Center destination Controlled Directory Number (CDN) that you specify and exits without verifying whether the transfer was successful.

If you use the **Blind Transfer** item in the **AACC AML Blind Transfer** node, then the speech application uses the Open Interface Web Services to reserve a Landing Pad on the target Avaya Aura® Contact Center. The speech application uses the Landing Pad reservation number to transfer the call to the Avaya Aura® Contact Center CDN that you specify in the `AACCLandingpadClient.properties` file. The speech application then exits without verifying whether the transfer was successful.

Behavior

When the system executes the **Blind Transfer** item, the system dials the telephone number or CDN, as appropriate, and then transfers control of the call to the dialed number. After the call is transferred, the speech application finishes and exits.

When you drag the **Blind Transfer** item into a node editor, Orchestration Designer automatically creates a variable by using the same name as that of the **Blind Transfer** item. The variable is undefined until the transfer is attempted. If the caller hangs up during the call transfer attempt and before the transfer is completed, then the variable is assigned the near_end_disconnect value. If for any other reason the system cannot successfully complete the call transfer, the variable is assigned the value unknown. In all other cases, the value remains undefined.

Tip:

You can use the return value of this variable later in your application. For example, if you want to provide data for a report on the success of transfer attempts, you can write the value of this variable to a database for further processing or manipulation. Or, you can use the value of this variable to redirect the call flow, based on what value is returned.

Note:

Support for call transfer functions can vary from one IVR system to another. The **Blind Transfer** item uses the VoiceXML <*transfer*> tag. For more information about support for the VoiceXML <*transfer*> tag and for transfer functions on your system, see the documentation for your IVR system.

Properties

- **Name**: Type a name that you want to assign to the **Blind Transfer** item and the associated variable. When you change the name of the **Blind Transfer** item in the Properties view and save the speech project, Orchestration Designer automatically changes the name of the associated variable.

  Note:

  The name must follow conventions for naming Java components. For more information, see Conventions for naming Java components on page 59.

- **Destination Number**: If you use the **Blind Transfer** item in the **Blind Transfer** node, then type the telephone number where the system must transfer the call.

  If you use the **Blind Transfer** item in the **AACC Blind Transfer** node, then in the **Destination Number** field, type the Avaya Aura® Contact Center destination CDN where the system must transfer the call.

  Use the **Destination Number** field when you want to hard code the number. For example, if you want to transfer the call at this point to an operator, you enter 00, which is the number to dial for the system operator.

  If you use the **Blind Transfer** item in the **AACC AML Blind Transfer** node, then the **Destination Number** field contains the Landing Pad reservation number that the system receives from the Avaya Aura® Contact Center Open Interface Web services at run time.

  Note:

  The system uses the Landing Pad reservation number to transfer the call to the Avaya Aura® Contact Center destination CDN that you specify in the AACCLandingpadClient.properties file. For more information, see Configuring the AACCLandingpadClient.properties file to transfer an inbound call to Avaya Aura Contact Center by using the Open Interface Web Services on page 381.

  The **Destination Number** field is unavailable if you select a variable in the **Destination Variable** field.

- **Destination Variable**: If you use the **Blind Transfer** item in the **Blind Transfer** node, then in the **Destination Variable** field, select the variable that contains the telephone number where the system must transfer the call.

If you use the **Blind Transfer** item in the **AACC Blind Transfer** node, then in the **Destination Variable** field, select the variable that contains the Avaya Aura® Contact Center destination CDN where the system must transfer the call.

Use the **Destination Variable** field when you want to perform dynamic transfers. For example, in dynamic transfers, callers can specify the telephone number to call. The system stores the telephone number in a variable. In other dynamic transfers, the system obtains the destination telephone number from some other source and stores it in a variable.

If you use the **Blind Transfer** item in the **AACC AML Blind Transfer** node, then in the **Destination Variable** field, select the variable in which you want to store the Landing Pad reservation number that the system receives from the Avaya Aura® Contact Center Open Interface Web services at run time. You can configure your database to store the Landing Pad reservation number. You can use the reservation number later in your application, for example, to debug the application.

If you select a complex variable in the **Destination Variable** field, then in the **Destination Variable Field** field, select a complex variable field.

The **Destination Variable** field is unavailable if you specify a constant in the **Destination Number** field.

- **Destination Variable Field**: If you select a complex variable in the **Destination Variable** field, then in the **Destination Variable Field** field, you must select the complex variable field that contains the telephone number or Avaya Aura® Contact Center CDN where the system must transfer the call, or the complex variable field in which you want to store the Landing Pad reservation number, as appropriate.
- **AAI Data**: If you use the **Blind Transfer** item in the **Blind Transfer** node, then in the **AAI Data** field, type the application-to-application (AAI) literal data that you want to pass from the speech application to the receiving end of the call transfer.

  If you use the **Blind Transfer** item in the **AACC Blind Transfer** node or **AACC AML Blind Transfer** node, then type the user-to-user information (UUI) that you want to pass to Avaya Aura® Contact Center.

  Note:

  Avaya Aura® Communication Manager can pass up to 96 bytes of hex- encoded UUI data.

  AML-based solutions do not support SIP UUI data.
- **AAI Variable**: If you use the **Blind Transfer** item in the **Blind Transfer** node, then in the **AAI Variable** field, select the variable that contains the AAI data that you want to pass to the receiving end.

  If you use the **Blind Transfer** item in the **AACC Blind Transfer** node or **AACC AML Blind Transfer** node, then select the variable that contains the UUI data that you want to pass to Avaya Aura® Contact Center.

  If you select a complex variable in the **AAI Variable** field, then in the **AAI Variable Field** field, select a complex variable field.

  The **AAI Variable** field is unavailable if you specify the AAI or UUI data in the **AAI Data** field.
- **AAI Variable Field**: If you select a complex variable in the **AAI Variable** field, then in the **AAI Variable Field** field, select the complex variable field that contains the AAI data or UUI data, as appropriate, that you want to pass to the receiving end.

- **Bind Name to Node**: (**true**/**false**) When **true**, the name of the variable item is bound to the name of the node. When you rename the parent node, the system renames the variable item, keeping the name of the variable in sync with the node name.
- **Transfer to AACC**: Option to transfer the call to Avaya Aura® Contact Center for assisted care. The options are:
  - **true**: If you want to transfer the call to Avaya Aura® Contact Center.

    Select **true** if you use the **Blind Transfer** item in the **AACC Blind Transfer** node or **AACC AML Blind Transfer** node.
  - **false**: If you do not want to transfer the call to Avaya Aura® Contact Center.

    Select **false** if you use the **Blind Transfer** item in the **Blind Transfer** node.
- **Transfer Protocol**: The transfer protocol that you want to use to transfer a call to Avaya Aura® Contact Center. The options are:
  - **SIP**: If you use the **Blind Transfer** item in the **AACC Blind Transfer** node, the transfer protocol is **SIP** to transfer a call to a SIP-enabled Avaya Aura® Contact Center.
  - **AML**: If you use the **Blind Transfer** item in the **AACC AML Blind Transfer** node, the transfer protocol is **AML** to transfer a call to Avaya Aura® Contact Center by using the Avaya Aura® Contact Center Open Interface Web Services.

  Note:

  The **Transfer Protocol** field is available only when you set the **Transfer to AACC** field to **true**.
- **Transfer Mode**: The transfer mode that you want to use to transfer the call-associated UUI and UCID to Avaya Aura® Contact Center. The options are:
  - **Shared**: The system hex encodes the UUI and UCID string, prefixes PD,00;C8 to the hex-encoded UUI and UCID string, and then passes the hex-encoded UUI and UCID string to Avaya Aura® Contact Center during a transfer.
  - **Service Provider**: The system hex encodes the UUI and UCID string, prefixes 04 to the hex-encoded UUI and UCID string, and then passes the hex-encoded UUI and UCID string to Avaya Aura® Contact Center during a transfer.

  Note:

  The **Transfer Mode** field is available only when you set the **Transfer to AACC** field to **true**.
- **UCID Variable**: The variable that contains the UCID that you want to pass to Avaya Aura® Contact Center.

  If you select a complex variable in the **UCID Variable** field, then in the **UCID Variable Field** field, select a complex variable field.

  Note:

  The **UCID Variable** field is available only when you set the **Transfer to AACC** field to **true**.

  UCID consumes 10 bytes of UUI data.

- **UCID Variable Field**: If you select a complex variable in the **UCID Variable** field, then in the **UCID Variable Field** field, select the complex variable field that contains the UCID that you want to pass to Avaya Aura® Contact Center.

  Note:

  The **UCID Variable Field** field is available only when you set the **Transfer to AACC** field to **true**.

Boolean

Type

Database Operation Condition Operator

Available from

Database Operation Editor Predicate Tab

Purpose

For joining two simple conditions with the Boolean operator (AND or OR) to create a Compound condition.

Behavior

The item must exist inside the Compound item. It can only be added after the first condition (Simple/Join/Compound) has been created.

Properties

- **Operator**: Choose either AND or OR.

Break

Type

SSML item

Available from

Any level tab in the Prompt File Editor. For more information about the Prompt File Editor, see Prompt file editor in a speech application on page 207.

Purpose

Speech synthesis engines that are SSML-compliant use a built-in algorithm to apply breaks, or pauses, in the reading of text. This algorithm is based on the linguistic context. If you want to override the automatic breaks of the speech synthesis engine, use the **Break** item to insert breaks or override automatic breaks in speech synthesis.

Note:

The **Break** item and its properties function correctly only with SSML-compliant speech synthesis engines. Microsoft SAPI Speech, which is used by Orchestration Designer during

application simulation, is *not* an SSML-compliant speech synthesis engine, so any settings you make with this item are ignored. For more information about the SSML standard, see the Speech Synthesis Markup Language (SSML) Version 1.0 W3C Recommendation.

Behavior

The specific behavior of the **Break** item depends on the properties you use.

In general, the **Break** item overrides an automatic break as interpreted and applied by the speech synthesis engine. The **Break** item also inserts a break where the speech synthesis engine does not insert one.

For example, if you added the following text into a TTS prompt segment:

“Here comes a break and now the break is over.”

The speech synthesis engine renders this with a very slight pause before the word “and.” If you want a longer pause at that point, you can divide the TTS text into two segments:

- “Here comes a break”
- “and now the break is over.”

Then you can insert a **Break** item between them, either to increase the relative length of the pause with the **Strength** property, or to set an exact length for the pause with the **Time** property. For more detailed information about these properties, see the following section, “Properties.”

Properties

- **Strength**: With the **Strength** property, you can change the length of an automatic break relative to the default interpretation of the speech synthesis engine. For example, if the speech synthesis engine inserts a break at a point in the text where you do not want one, you can override it by using a **Strength** setting of **none**.

  Select one of the following settings:

Strength Setting	Description
none	This setting suppresses any automatic breaks that the speech synthesis engine applies to the text.
x-weak **weak** **medium** **strong** **x-strong**	These settings represent a range of break options. The weaker the setting, the shorter the break is at the point where the **Break** item is inserted. The stronger the setting the longer the break is. The specific application of these settings varies according to the TTS server.

- **Time**: With the **Time** property, you can specify exactly the amount of time that the break lasts, in milliseconds or seconds. For example, if you set the **Time** property to **200ms**, the system inserts a 200 millisecond (or 0.2 second) pause at that point in the prompt.

  Select one of the following settings:

Time Setting	Description
250ms **500ms** **750ms** **1s** **2s** **3s** **4s** **5s**	This setting applies a pause of the exact duration selected at the point where it is inserted. - **ms** = milliseconds - **s** = seconds
custom	With this setting, you can define the exact length of time that you want the pause to last. When you select this setting, Orchestration Designer automatically adds the **Custom Break Time [s or ms]** property to the Property view.
Custom Break Time [s or ms]	This setting is available only when the **custom** setting for **Time** is selected. With it, you can determine exactly how long a pause you want to insert. The acceptable format for this setting is a number followed, with no space, by: - **ms** = milliseconds - **s** = seconds For example, if you want to insert a pause of exactly one and one-fourth seconds, enter in this field `1250ms`. **Note:** You cannot use decimals in this field. If you want to use a fraction of a second as your time, you must enter it as milliseconds (**ms**).

Bridged Transfer

Type

Form item

Available from

- Announce node on page 634
- Bridged Transfer node on page 647, AACC Bridged Transfer node on page 637, and AACC AML Bridged Transfer node on page 643. Pre-populated by default.
- Form node on page 655

Note:

Although the **Bridged Transfer** item appears in the Palette pane of other nodes, such as **Prompt and Collect** node, **Blind Transfer** node, **AACC Blind Transfer** node, **AACC**

AML Blind Transfer node, and **Record** node, it is not actually available for use in these nodes without destroying the intended functionality of these nodes.

The **Bridged Transfer** item is intended primarily for use only in the **Bridged Transfer** node, **AACC Bridged Transfer** node, and **AACC AML Bridged Transfer** node.

Purpose

The **Bridged Transfer** item provides the elements needed to set up a bridged transfer.

If you use the **Bridged Transfer** item in the **Bridged Transfer** node, then the speech application transfers a call to the telephone number that you specify but does not lose control of the call.

If you use the **Bridged Transfer** item in the **AACC Bridged Transfer** node, then the speech application transfers a call to the Avaya Aura® Contact Center destination Controlled Directory Number (CDN) that you specify but does not lose control of the call.

If you use the **Bridged Transfer** item in the **AACC AML Bridged Transfer** node, then the speech application uses the Open Interface Web Services to reserve a Landing Pad on the target Avaya Aura® Contact Center. The speech application uses the Landing Pad reservation number to transfer the call to the Avaya Aura® Contact Center CDN that you specify in the `AACCLandingpadClient.properties` file. The speech application does not lose control of the call.

Behavior

When the system executes the **Bridged Transfer** item, the system dials the telephone number or CDN, as appropriate, and then waits for a response.

Note:

Support for call transfer functions can vary from one IVR system to another. The **Bridged Transfer** item uses the VoiceXML <*transfer*> tag. For more information about support for the VoiceXML <*transfer*> tag and for transfer functions on your system, see the documentation for your IVR system.

When you drag the **Bridged Transfer** item into a node editor, Orchestration Designer automatically creates a variable with the same name as that of the **Bridged Transfer** item. The variable contains the following fields, which are populated upon completion of the node:

- **duration**: The length of time that the call transfer lasted, in milliseconds.
- **inputmode**: The mode in which the caller input was provided to return control of the call to the application, either voice or DTMF.
- **utterance**: A text, in raw string, summary of what the caller said or keyed in to terminate the call transfer.
- **value**: The value of the **value** field depends on the result of the attempt to transfer the call. The following table shows the possible values the **value** field can take, along with the actions that cause the value to be assigned.

If:	Then this variable field is:
The caller hangs up either before or after the transfer is completed.	Undefined, and the system throws a `connection.disconnect.hangup` event.

Table continues…

If:	Then this variable field is:
The caller uses a voice or DTMF command to end the call transfer, either before or after the transfer is complete.	Assigned a value of near_end_disconnect.
The party being called ends the transferred call by hanging up.	Assigned a value of far_end_disconnect.
The transfer does not complete because the number being called is busy.	Assigned a value of busy.
The transfer does not complete because an intermediate network refuses the call.	Assigned a value of network_busy.
The transfer does not complete because the party being called does not answer within the time set by the **Connect Timeout** field of the **Bridged Transfer** item.	Assigned a value of noanswer.
The transfer does not complete, but the reason for the failure is not known. Or the transfer completes but is ended for reasons that are not known.	Assigned a value of unknown.
The system ends the transferred call because the transfer exceeds the maximum allowable time as set in the **Maximum Time** field of the **Bridged Transfer** item.	Assigned a value of maxtime_disconnect.
The network ends the transferred call by disconnecting the two parties.	Assigned a value of network_disconnect.

Tip:

You can use or manipulate the return values from these variable fields later in the application. For example, if you want to generate a report that includes information about the average length of time call transfers take, which ties up channels, you can set up the speech application to write the return value of the duration field to a database. You can then extract the information from the database for your report. Or, you want the system to wait for five seconds before attempting to redial the telephone number after getting a busy signal. You can use the busy or network_busy return in the value field to reroute the call back through the **Bridged Transfer** node after a five-second pause.

Exception errors can be converted into the following return codes:

- `noauthorization`
- `baddestination`
- `noroute`
- `noresource`
- `protocol.nnn`
- `unsupported.transfer.blind`
- `unsupported.transfer.bridge`
- `unsupported.uri`

Properties

- **Name**: Type a name that you want to assign to the **Bridged Transfer** item and the associated variable.

  When you change the name of the **Bridged Transfer** item in the Properties view and save the speech project, Orchestration Designer automatically changes the name of the associated variable.

  Note:

  The name of the **Bridged Transfer** item must follow conventions for naming Java components. For more information, see Conventions for naming Java components on page 59.

- **Transfer Audio**: Select the prompt that you want to use for the call transfer.

  The **Transfer Audio** prompt must be a transitional audio prompt. The transitional audio prompt is the audio (*.wav) file that the system plays to the callers while transferring the call.

  For more information about transitional audio prompts, see Transitional audio prompts in speech applications on page 205.

  Note:

  Any audio prompts in or after a transfer node are not played before the call is transferred. To ensure that the prompt is played before the transfers happens, add a transitional audio prompt to the node that transfers the call.

- **Connect Timeout**: The number of seconds that the system must wait for a response from the dialed party. If the system receives no response within this time, control of the call returns to the speech application. In this case, the system also returns a value of `noanswer` to the **value** field of the Bridged Transfer variable.

  Enter a positive integer.

  The **Connect Timeout** field is unavailable if you select a variable in the **Connect Timeout Variable** field.

- **Connect Timeout Variable**: The variable that contains the number of seconds that the system must wait for a response from the dialed party. If the system receives no response within this time, control of the call returns to the speech application. In this case, the system also returns a value of `noanswer` to the **value** field of the Bridged Transfer variable.

  The **Connect Timeout Variable** field is unavailable if you specify a constant value in the **Connect Timeout** field.

  If you select a complex variable in the **Connect Timeout Variable** field, then you must select a complex variable field in the **Connect Timeout Field** field.

- **Connect Timeout Field**: If you select a complex variable in the **Connect Timeout Variable** field, then in the **Connect Timeout Field** field, you must select a complex variable field that contains the number of seconds that the system must wait for a response from the dialed party.

- **Maximum Time**: The maximum number of seconds that a transferred call can last. When a transferred call reaches this number of seconds, the system ends the transfer and returns control of the call to the original application. In this case, the system also returns a value of `maxtime_disconnect` to the **value** field of the Bridged Transfer variable.

Enter a positive integer.

The default value is zero (**0**), which means that the system imposes no time limit on call transfers.

- **Destination Number**: If you use the **Bridged Transfer** item in the **Bridged Transfer** node, then in the **Destination Number** field, specify the telephone number where the system must transfer the call.

  If you use the **Bridged Transfer** item in the **AACC Bridged Transfer** node, then in the **Destination Number** field, specify the Avaya Aura® Contact Center destination CDN where the system must transfer the call.

  Use the **Destination Number** field when you want to hard code the number. For example, if you want to transfer the caller at this point to an operator, enter 00, which is the number to dial for the system operator.

  If you use the **Bridged Transfer** item in the **AACC AML Bridged Transfer** node, then the **Destination Number** field contains the Landing Pad reservation number that the system receives from the Avaya Aura® Contact Center Open Interface Web services at run time.

  Note:

  The system uses the Landing Pad reservation number to transfer the call to the Avaya Aura® Contact Center destination CDN that you specify in the `AACCLandingpadClient.properties` file. For more information, see Configuring the AACCLandingpadClient.properties file to transfer an inbound call to Avaya Aura Contact Center by using the Open Interface Web Services on page 381.

  The **Destination Number** field is unavailable if you select a variable the **Destination Variable** field.

- **Destination Variable**: If you use the **Bridged Transfer** item in the **Bridged Transfer** node, then in the **Destination Variable** field, select the variable that contains the telephone number where the system must transfer the call.

  If you use the **Bridged Transfer** item in the **AACC Bridged Transfer** node, then in the **Destination Variable** field, select the variable that contains the Avaya Aura® Contact Center destination CDN where the system must transfer the call.

  Use the **Destination Variable** field when you want to perform dynamic transfers. For example, in dynamic transfers, callers can specify the telephone number to call. The system stores the telephone number in a variable. In other dynamic transfers, the system obtains the destination telephone number from some other source and stores it in a variable.

  If you use the **Bridged Transfer** item in the **AACC AML Bridged Transfer** node, then in the **Destination Variable** field, select the variable in which you want to store the Landing Pad reservation number that the system receives from the Avaya Aura® Contact Center Open Interface Web services at run time. You can configure your database to store the Landing Pad reservation number. You can use the reservation number later in your application, for example, to debug the application.

  If you select a complex variable in the **Destination Variable** field, then in the **Destination Variable Field** field, select a complex variable field.

  The **Destination Variable** field is unavailable if you specify a constant value in the **Destination Number** field.

- **Destination Variable Field**: If you select a complex variable in the **Destination Variable** field, then in the **Destination Variable Field** field, you must select the complex variable field that contains the telephone number or Avaya Aura® Contact Center CDN where the system must transfer the call, or the complex variable field in which you want to store the Landing Pad reservation number, as appropriate.
- **AAI Data**: If you use the **Bridged Transfer** item in the **Bridged Transfer** node, then in the **AAI Data** field, type the application-to-application (AAI) literal data that you want to pass from the speech application to the receiving end of the call transfer.

  If you use the **Bridged Transfer** item in the **AACC Bridged Transfer** node or **AACC AML Bridged Transfer** node, then in the **AAI Data** field, type the user-to-user information (UUI) that you want to pass to Avaya Aura® Contact Center.

  Note:

  Avaya Aura® Communication Manager can pass up to 96 bytes of hex- encoded UUI data.

  AML-based solutions do not support SIP UUI data.
- **AAI Variable**: If you use the **Bridged Transfer** item in the **Bridged Transfer** node, then in the **AAI Variable** field, select the variable that contains the AAI data that you want to pass to the receiving end.

  If you use the **Bridged Transfer** item in the **AACC Bridged Transfer** node or **AACC AML Bridged Transfer** node, then in the **AAI Variable** field, select the variable that contains the UUI data that you want to pass to Avaya Aura® Contact Center.

  If you select a complex variable in the **AAI Variable** field, then you must select a complex variable field in the **AAI Variable Field** field.
- **AAI Variable Field**: If you select a complex variable in the **AAI Variable** field, then in the **AAI Variable Field** field, select the complex variable field that contains the AAI or UUI data, as appropriate, that you want to pass to the receiving end.
- **Bind Name to Node**: (**true**/**false**). When **true**, the name of the variable item is bound to the name of the node. When you rename the parent node, the system renames the variable item, keeping the name of the variable in sync with the node name.
- **Convert Errors**: If you set the **Convert Errors** field to **true**, then the names and values of any error events that the system generates are automatically captured and stored in the variable that is associated with the **Bridged Transfer** node, **AACC Bridged Transfer** node, or **AACC AML Bridged Transfer** node that you use.
- **Transfer to AACC**: Option to transfer the call to Avaya Aura® Contact Center for assisted care. The options are:
  - **true**: If you want to transfer the call to Avaya Aura® Contact Center.

    Select **true** if you use the **Bridged Transfer** item in the **AACC Bridged Transfer** node or **AACC AML Bridged Transfer** node.
  - **false**: If you do not want to transfer the call to Avaya Aura® Contact Center.

    Select **false** if you use the **Bridged Transfer** item in the **Bridged Transfer** node.

- **Transfer Protocol**: The transfer protocol that you want to use to transfer the call. The options are:
  - **SIP**: If you use the **Bridged Transfer** item in the **AACC Bridged Transfer** node, the transfer protocol is **SIP** to transfer a call to a SIP-enabled Avaya Aura® Contact Center.
  - **AML**: If you use the **Bridged Transfer** item in the **AACC AML Bridged Transfer** node, the transfer protocol is **AML** to transfer a call to Avaya Aura® Contact Center by using the Avaya Aura® Contact Center Open Interface Web Services.

  Note:

  The **Transfer Protocol** field is available only when you set the **Transfer to AACC** field to **true**.
- **Transfer Mode**: The transfer mode that you want to use to transfer the call-associated UUI and UCID to Avaya Aura® Contact Center. The options are:
  - **Shared**: The system hex encodes the UUI and UCID string, prefixes PD,00;C8 to the hex-encoded UUI and UCID string, and then passes the hex-encoded UUI and UCID string to Avaya Aura® Contact Center during a transfer.
  - **Service Provider**: The system hex encodes the UUI and UCID string, prefixes 04 to the hex-encoded UUI and UCID string, and then passes the hex-encoded UUI and UCID string to Avaya Aura® Contact Center during a transfer.

  Note:

  The **Transfer Mode** field is available only when you set the **Transfer to AACC** field to **true**.
- **UCID Variable**: The variable that contains the UCID that you want to pass to Avaya Aura® Contact Center. If you select a complex variable in the **UCID Variable** field, then in the **UCID Variable Field** field, you must select a complex variable field.

  Note:

  The **UCID Variable** field is available only when you set the **Transfer to AACC** field to **true**.

  UCID consumes 10 bytes of UUI data.
- **UCID Variable Field**: If you select a complex variable in the **UCID Variable** field, then in the **UCID Variable Field** field, select the complex variable field that contains the UCID that you want to pass to Avaya Aura® Contact Center.

  Note:

  The **UCID Variable Field** field is available only when you set the **Transfer to AACC** field to **true**.

Capture Expression

Type

Call flow or message flow item

Available from

- Announce node on page 634
- Prompt and Collect node on page 659
- Collect node on page 651
- Form node on page 655
- Menu node on page 656
- **AppRoot** node (see About the AppRoot node on page 132)

Purpose

The purpose of the Capture Expression item is to capture the value of ECMA script objects from the Voice XML or the TextXML page and store them in an Orchestration Designer session variable.

Behavior

The Capture Expression item is used in the following different ways:

- **App Root**: In a call flow application, on the AppRoot, Capture Expression is used to capture ECMA script values as well as supplemental session variables when the voice dialog starts up. It can be used for capturing the value of SIP headers, for example, or other VoiceXML platform variables that are not automatically captured by Orchestration Designer when a dialog starts.

  Refer to the documentation for your Avaya Experience Portal platform for further details on headers and other variables that are exposed.

  In a message flow, on the Approot, Capture Expression is used to capture ECMA script values and all session related variables, including the message properties and the UCID to the inbound message.
- **Event handlers**: Capture Expression can be added to goto items used in Event handlers for capturing the event type and event message. The event type/message can then be used in another section of the call flow or message flow for handling errors differently.
- **Forms/Menus** - Can be added to goto items on links for capturing values from the **application.lastresult$** variable such as capturing the **utterance** that activated the link, or the **confidence**.

  Note:

  Orchestration Designer does not validate the ECMA script expression that is entered. Be sure that you consult the VXML specification and understand which ECMA script objects are valid at different places of the generated VXML page.

Properties

- **Variable**: The variable that will store the captured data. Select the variable you want to use. If the variable you select is a complex variable, you must also select a **Variable Field**.
- **Variable Field**: Select the variable field you want to use. The contents of this list depend on which variable you selected in the **Variable** property.
- **Expression**: The ECMA script expression that will be evaluated by the VoiceXML or TextXML browser. The result is stored in the variable. The ECMA script object must be simple (for example, strings or numbers).

- **Get Value in Json Format** : This property is useful for getting the whole json string of an object such as the session or any of its child objects.

Case

Type

Condition item

Available from

Data node in speech and message applications

Prompt file editor in speech and message applications

Purpose

The Case statements provide options for the Switch statement to compare the case values with the values of the switch variable in a multi-choice pattern.

Note:

A case statement must have at least one child element.

Behavior

Each Case statement consists of a value to compare with the value of the switch variable and a set of statements to execute if the compared values match. Each Case must contain at least one operation to execute when the Case statement is executed.

You can specify either a constant value or a variable for the Case statement.

If the value of the switch variable matches the constant case value or the value of the case variable, then the Switch statement executes the block of statements that follow the Case statement.

Properties

- **Value**: Type a value that you want to compare with the value of the switch variable.

  The **Value** field is unavailable if you specify a variable in the **Variable** field.

- **Variable**: Select a variable that contains the value that you want to compare with the value of the switch variable.

  The **Variable** field is unavailable if you specify a constant value in the **Value** field.

- **Variable Field**: If you select a complex variable in the **Variable** field, then in the **Variable Field** field, select the variable field that contains the value that you want to compare with the value of the switch variable.

Catch (Exception)

Type

Data node on page 653 exception handler

Available from

Data node on page 653

Purpose

To catch Java exceptions that are thrown at run time.

Behavior

Try/Catch items in the Data node on page 653 *try* to execute items under the Try on page 850 item. To handle exceptions, add Catch (Exception) on page 692 items (under the parent Try on page 850). For every Try item, you need at least one Catch.

By default, Catch (Exception) on page 692 catches *all* exceptions. Catch (Exception) on page 692 can also be used to catch specific exception types for handling specific errors differently. Comma separated exception types can also be specified for catching different exception types, but handling them in the same way.

List of common exceptions are provided in the Avaya Properties view. Currently this list contains:

- SQLException
- IOException
- SCERuntimeException

A new variable called **ddLastException** automatically captures the exception caught by a Catch item (includes the following data members: errorcode, message, object, stacktrace, and type).

Following is an example of the Try/Catch mechanism.

Catch

Type

Event handler item

Available from

- **AppRoot** node (see AppRoot node on page 132)
- Announce node on page 634
- Prompt and Collect node on page 659
- Collect node on page 651

- Blind Transfer node on page 646
- Bridged Transfer node on page 647
- Record node on page 660
- Menu node on page 656
- Form node on page 655

Purpose

The **Catch** event handler is a generic handler for events you create. Catch (VXML or TextXML Events) catches VXML or TextXML events that are thrown in the application or by the VXML or TextXML platform.

Note:

Orchestration Designer has built-in event handlers for most of the default events. The Catch event handler is designed to handle all other, custom, events. For more information about the built-in event handlers, see:

- No Input on page 769
- No Match on page 770
- On Disconnect on page 775

Behavior

Before you can use the Catch event handler, you must create the event for it to catch. Within the scope of this event handler, you must also use a Throw on page 844 item in your application.

The Catch event handler responds to a specified event anywhere within the scope of the handler. For example, if you drag the Catch event handler into the **AppRoot** node, it acts as a global event handler for the specified events. It can catch these specified events no matter where in the application the events are thrown. If you drag the Catch event handler into another node, however, it can catch the specified event only if it is thrown while in that node. If you drag the Catch event handler underneath a node item, it can catch the specified event only if it is thrown within that node item.

A Catch event handler at the node item level overrides a Catch event handler at the node level. Likewise, a Catch event handler at the node level overrides a Catch event handler at the **AppRoot**, or global, level.

For more information about creating events, throwing events, and catching events, see Events on page 314.

The following are the examples on handling VXML events:

- Catch (event=“myEvent”), Return Event

  In this scenario, when the “myEvent” event is caught, the module will exit, returning control back to the calling application, and it will return the “myEvent” that was caught.

  In other words, it propagates or re-throws the event to let the calling application handle the event.

- Catch (event=“error.runtime”), Return Event, Throw (event=“myReallyBadEvent”, message=“Something happened...”)

In this scenario, the application catches an “error.runtime” event and the module will exit returning control back to the calling application.

Instead of propagating the “error.runtime” event, however, it returns “myReallyBadEvent” instead, with the message “Something happened...”. The calling application can catch the “myReallyBadEvent” event, but does not know that the cause of the event was an “error.runtime” event.

- Catch (“error.badfetch”), Exit

  In this case, the application catches an “error.badfetch” event and then exits the application. Exiting the application terminate the session.

  It does not matter if it is a module or a stand-alone application, the application will end. This is typically used when there is some fatal error with the application or system.

Properties

- **Event**: Select the event that you want the Catch event handler to handle.

  For the Catch event handler to work, you must specify the event for it to catch when the event is thrown. You must also use one or more of the following options with it:

- Prompt on page 797 item: This item plays the designated prompt before the application continues.
- Goto on page 736 item: This item directs the application to the designated node.
- Throw on page 844 item: Throws an event that you choose. Select a user-defined event that you have created.

  By throwing an event within an event handler, you can transfer control to a different part of the application, for example, to the **AppRoot** node. For more information about throwing events and handling events, see Throw on page 844 or Events.

Choice

Type

Menu item

Available from

Menu node on page 656

Purpose

When you use a Menu node in your call flow, you must offer callers at least one choice, or it does not make sense to use a menu. **Choice** items exist for the purpose of providing callers with options on a menu, from which the callers can choose.

Double click the **Choice** item to open the node specified in the next form attribute in the node editor.

Behavior

The **Choice** item uses either a spoken reply or a DTMF response to direct the call flow based on the response of a caller. Each **Choice** item in a Menu node represents a single option available to

the caller. So, if you want to offer callers a menu with four options, you must use four **Choice** items, one for each option.

If you want to use ASR, you must assign a grammar to each **Choice** item.

Note:

When you assign grammars to choices, any recognized response in a grammar activates the choice to which it is assigned. For this reason, you must use a separate grammar for each choice in the menu. Each of these grammars can contain only one or two words for recognition. For more information about creating and using grammar, see Grammars on page 242.

Properties

Enter values for the following properties:

- **Name**: Enter a descriptive name to indicate what this choice represents.
- **Accept Choice**: Select whether you want the choice recognition to be:
  - **Exact**: (the default) In this case, the system recognizes the response of the caller successfully only if the response is exactly what the system expects. For example, if the system is set to respond to “Weekly News in Review” exactly, the system recognizes only that exact phrase with all words in the proper order.
  - **Approximate**: In this case, the system can recognize a range of responses, usually a subset of the expected response. For example, if the system is set to respond to “Weekly News in Review” approximately, the system can recognize “Weekly News,” “News Review,” and “Weekly Review” all as valid responses.

    Note:

    This option does not work well with Microsoft SAPI Speech, which Orchestration Designer uses to test ASR in applications. For this reason, you cannot test this option until you deploy the application to a live system.
- **Next Form**: Indicates what node this choice goes to when selected. You can either:
  - Select the node you want this choice to go to.
  - Use the graphic Connector tool in the main.flow window to connect this choice to the desired node.
- **DTMF**: (Optional) Enter in this field the DTMF key press or key press combination that the caller can use to select this choice.
- **DTMF Variable/DTMF Variable Field**: Use a variable for the DTMF key that activates the choice.

  Note:

  If you designate a key press or series of key presses, Orchestration Designer displays these key presses in the Call Flow Editor main workspace as part of the branch arrows that lead away from the node. If multiple key presses are indicated for a menu option, Orchestration Designer lists the key presses separated by commas. For an example, see the figure on page 697.

Tip:

It is always a good idea to give callers the option to use either a spoken reply or a DTMF key press to select a choice. For example, you can create a prompt for a choice that says, “To hear about your savings options, press one or say “savings.” Then, when setting up this Choice item, you both use an ASR grammar for “savings” and enter **1** in the DTMF field.

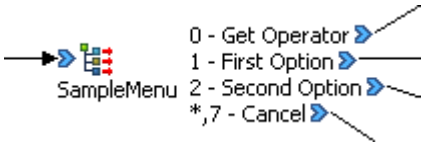


Figure 2: Sample Menu node, showing DTMF menu options

Column Operand

Type

Database Operation Condition Operand

Available From

Database Operation Editor Predicate Tab

Purpose

For specifying the table name and the column name on the left side or right side of a condition. It applies to both Simple and Join conditions.

Behavior

When adding a Simple condition item, both the Column item and Comparison item are automatically added inside the Simple condition item. Similarly, adding a Join condition would automatically create both Column items and the Comparison item inside it.

Properties

- **Table Name**: Choose the table that has been added to the database operation.
- **Column Name**: Choose the column from the list for the table selected.

Comparison Operator

Type

Database Operation Condition Operator

Available From

Data node on page 653

Database Operation Editor Predicate Tab

Purpose

For setting up the comparison between the left side and the right side of a condition. It is needed when using both simple and join conditions

Behavior

When adding a Simple condition item, both the Column item and Comparison item are automatically added inside the Simple condition item. Similarly, adding a Join Condition on page 750 would automatically create both Column items and the Comparison item inside it.

Properties

- **Operator**: Choose from (=, <, <=, >, >=, <>, LIKE, NOT LIKE)

Complex Variable

Type

Variable

Available from

Variable Editor

Purpose

A complex variable is a variable that has several attributes, called “fields”. Complex variables make it possible to use a single variable to pass along multiple values. To be valid, a complex variable must have at least one Field on page 733 assigned to it.

Behavior

A complex variable must have at least one variable field to be valid. For more information about complex variable fields, see Field on page 733.

When you create a complex variable, Orchestration Designer automatically assigns a temporary name to it and adds one **Field** item. You can rename both the variable and the field. You can also add as many additional **Field** items as you want.

In some cases, Orchestration Designer creates complex variables automatically when nodes or items are used in the call flow or message flow. For example, when you drag a Bridged Transfer item into the call flow or the message flow, Orchestration Designer automatically creates a complex variable with the same name as the item.

For more information about the use of variables in Orchestration Designer, see Variables on page 226.

Properties

- **Name**: Enter in this field the name to identify the variable.

  Note:

  This name must follow conventions for naming Java components as described in Conventions for naming Java components on page 59.

Condition

This item has been deprecated.

- In the Data Node, use right-click to replace by If on page 739+Next on page 768.
- In the Prompt Editor, use right-click to remove the condition parent item. The If on page 739 and Else on page 722 child items will be preserved.

Consultation Transfer

Type

Form item

Available From

The following nodes of a speech application:

- Announce node on page 634
- Consultation Transfer node on page 648, AACC Consultation Transfer node on page 639, and AACC AML Consultation Transfer node on page 644. Pre-populated by default.
- Form node on page 655

Note:

Although the **Consultation Transfer** item appears in the Palette pane of other nodes, such as **Prompt and Collect** node, **Bridged Transfer** node, **AACC Bridged Transfer** node, **AACC AML Bridged Transfer** node, and **Record** node, it is not actually available for use in these nodes without destroying the intended functionality of these nodes. The **Consultation Transfer** item is intended primarily for use only in the **Consultation Transfer** node, **AACC Consultation Transfer** node, and **AACC AML Consultation Transfer** node.

Purpose

The **Consultation Transfer** item provides the elements needed to set up a consultation transfer.

If you use the **Consultation Transfer** item in the **Consultation Transfer** node, then the speech application transfers a call to the telephone number that you specify without releasing the call from the voice platform until the call is successfully transferred.

If you use the **Consultation Transfer** item in the **AACC Consultation Transfer** node, then the speech application transfers a call to the Avaya Aura® Contact Center destination Controlled Directory Number (CDN) that you specify without releasing the call from the voice platform until the call is successfully transferred.

If you use the **Consultation Transfer** item in the **AACC AML Consultation Transfer** node, then the speech application uses the Open Interface Web Services to reserve a Landing Pad on the target Avaya Aura® Contact Center. The speech application uses the Landing Pad reservation number to transfer the call to the Avaya Aura® Contact Center CDN that you specify in the `AACCLandingpadClient.properties` file. The speech application does not release the call from the voice platform until the call is successfully transferred.

If the connection cannot be established to the called number, then the call is returned back to the speech application which can attempt to handle the call in other ways.

Behavior

When you drag the **Consultation Transfer** item into a node editor, Orchestration Designer automatically creates a variable by using the same name as that of the **Consultation Transfer** item. The variable is undefined until the transfer is attempted.

Tip:

You can use the return value of this variable later in your application. For example, if you want to provide data for a report on the success of transfer attempts, you can write the value of this variable to a database for further processing or manipulation. Or, you can use the value of this variable to redirect the call flow, based on what value is returned.

Note:

Support for call transfer functions can vary from one IVR system to another. The **Blind Transfer** item uses the VoiceXML <*transfer*> tag. For more information about support for the VoiceXML <*transfer*> tag and for transfer functions on your system, see the documentation for your IVR system.

A consultation transfer is similar to a **Blind Transfer** except that the outcome of the call transfer is known and the call is not dropped as a result of an unsuccessful transfer attempt. When performing a consultation transfer, the platform monitors the progress of the transfer until the connection is established between caller and the called person.

If the connection cannot be established because of reasons, such as no answer or line busy, the session remains active and returns control to the speech application. As in the case of a blind transfer, if the connection is established, the interpreter disconnects from the session, the system throws the `connection.disconnect.transfer` event, and document interpretation continues normally.

If the connection to the called number is successfully established, then the call is transferred and the speech application platform releases the call and the speech application ends. The application can execute further to perform activities such as logging and resource cleanup, but the call is no longer connected to the speech application.

If the transfer is successful, the speech application gets a `connection.disconnect.transfer` event and the VXML session ends, or the session continues with session termination operations. If the transfer fails, then the variable item stores the error code and the speech application can then evaluate the error and perform the appropriate action. For more information, see the VXML specification for return codes. Define event types by using the Event Type Editor, and use the VXML event handlers to **Catch** (VXML Events) the event.

Exception errors can be converted into the following return codes:

- `noauthorization`
- `baddestination`
- `noroute`
- `noresource`
- `protocol.nnn`

- `unsupported.transfer.consultation`
- `unsupported.uri`

Properties

- **Name**: Type a name that you want to assign to the **Consultation Transfer** item and the associated variable.

  When you change the name of the **Consultation Transfer** item in the Properties view and save the speech project, Orchestration Designer automatically changes the name of the associated variable.

  Note:

  The name of the **Consultation Transfer** item must follow conventions for naming Java components. For more information, see Conventions for naming Java components on page 59.

- **Transfer Audio**: Select the prompt that you want to use for the call transfer.

  The **Transfer Audio** prompt must be a transitional audio prompt. The transitional audio prompt is the audio (*.wav) file that the system plays to the caller while transferring the call.

  For more information about transitional audio prompts, see Transitional audio prompts in speech applications on page 205.

  Note:

  Any audio prompt in or after a transfer node is not played before the call is transferred. To ensure that the prompt is played before the transfers happens, add a transitional audio prompt to the node that transfers the call.

- **Connect Timeout**: The maximum number of seconds to allow before the terminal connection is established.

  The **Connect Timeout** field is unavailable if you select a variable in the **Connect Timeout Variable** field.

- **Connect Timeout Variable**: The variable that contains the maximum number of seconds to allow before the terminal connection is established.

  If you select a complex variable in the **Connect Timeout Variable** field, then in the **Connect Timeout Field** field, you must select a complex variable field.

  The **Connect Timeout Variable** field is unavailable if you specify a constant value in the **Connect Timeout** field.

- **Connect Timeout Field**: If you select a complex variable in the **Connect Timeout Variable** field, then in the **Connect Timeout Field** field, you must select a complex variable field that contains the maximum number of seconds to allow before the terminal connection is established.

- **Destination Number**: If you use the **Consultation Transfer** item in the **Consultation Transfer** node, then in the **Destination Number** field, type the telephone number where the system must transfer the call.

  If you use the **Consultation Transfer** item in the **AACC Consultation Transfer** node, then in the **Destination Number** field, specify the Avaya Aura® Contact Center destination CDN where the system must transfer the call.

Use the **Destination Number** field when you want to hard code the number. For example, if you want to transfer the caller at this point to an operator, enter 00, which is the number to dial for the system operator.

If you use the **Consultation Transfer** item in the **AACC AML Consultation Transfer** node, then the **Destination Number** field contains the Landing Pad reservation number that the system receives from the Avaya Aura® Contact Center Open Interface Web services at run time.

Note:

The system uses the Landing Pad reservation number to transfer the call to the Avaya Aura® Contact Center destination CDN that you specify in the `AACCLandingpadClient.properties` file. For more information, see Configuring the AACCLandingpadClient.properties file to transfer an inbound call to Avaya Aura Contact Center by using the Open Interface Web Services on page 381.

The **Destination Number** field is unavailable if you select a variable in the **Destination Variable** field.

- **Destination Variable**: If you use the **Consultation Transfer** item in the **Consultation Transfer** node, then in the **Destination Variable** field, select the variable that contains the telephone number where the system must transfer the call.

  If you use the **Consultation Transfer** item in the **AACC Consultation Transfer** node, then in the **Destination Variable** field, select the variable that contains the Avaya Aura® Contact Center destination CDN where the system must transfer the call.

  Use the **Destination Variable** field when you want to perform dynamic transfers. For example, in dynamic transfers, callers can specify the telephone number to call. The system stores the telephone number in a variable. In other dynamic transfers, the system obtains the destination telephone number from some other source and stores it in a variable.

  If you use the **Consultation Transfer** item in the **AACC AML Consultation Transfer** node, then in the **Destination Variable** field, select the variable in which you want to store the Landing Pad reservation number that the system receives from the Avaya Aura® Contact Center Open Interface Web services at run time. You can configure your database to store the Landing Pad reservation number. You can use the reservation number later in your application, for example, to debug the application.

  If you select a complex variable in the **Destination Variable** field, then in the **Destination Variable Field** field, you must select a complex variable field.

  The **Destination Variable** field is unavailable if you specify a constant value in the **Destination Number** field.

- **Destination Variable Field**: If you select a complex variable in the **Destination Variable** field, then in the **Destination Variable Field** field, you must select a complex variable field that contains the telephone number or Avaya Aura® Contact Center CDN where the system must transfer the call, or the complex variable field in which you want to store the Landing Pad reservation number, as appropriate.
- **AAI Data**: If you use the **Consultation Transfer** item in the **Consultation Transfer** node, then in the **AAI Data** field, type the application-to-application (AAI) literal data that you want to pass from the speech application to the receiving end of the call transfer.

If you use the **Consultation Transfer** item in the **AACC Consultation Transfer** node or **AACC AML Consultation Transfer** node, then in the **AAI Data** field, type the user-to-user information (UUI) that you want to pass to Avaya Aura® Contact Center.

Note:

Avaya Aura® Communication Manager can pass up to 96 bytes of hex- encoded UUI data.

AML-based solutions do not support SIP UUI data.

- **AAI Variable**: If you use the **Consultation Transfer** item in the **Consultation Transfer** node, then in the **AAI Variable** field, select the variable that contains the AAI data that you want to pass to the receiving end.

  If you use the **Consultation Transfer** item in the **AACC Consultation Transfer** node or **AACC AML Consultation Transfer** node, then in the **AAI Variable** field, select the variable that contains the UUI data that you want to pass to Avaya Aura® Contact Center.

  If you select a complex variable in the **AAI Variable** field, then in the **AAI Variable Field** field, you must select a complex variable field.
- **AAI Variable Field**: If you select a complex variable in the **AAI Variable** field, then in the **AAI Variable Field** field, select the complex variable field that contains the AAI or UUI data, as appropriate, that you want to pass to the receiving end.
- **Bind Name to Node**: (**true**/**false**). When **true**, the name of the variable item is bound to the name of the node. When you rename the parent node, the system renames the variable item, keeping the name of the variable in sync with the node name.
- **Convert Errors**: If you set the **Convert Errors** field to **true**, then the names and values of any error events that the system generates are automatically captured and stored in the variable that is associated with the **Consultation Transfer** node.
- **Transfer to AACC**: Option to transfer the call to Avaya Aura® Contact Center for assisted care. The options are:
  - **true**: If you want to transfer the call to Avaya Aura® Contact Center.

    Select **true** if you use the **Consultation Transfer** item in the **AACC Consultation Transfer** node or **AACC AML Consultation Transfer** node.
  - **false**: If you do not want to transfer the call to Avaya Aura® Contact Center.

    Select **false** if you use the **Consultation Transfer** item in the **Consultation Transfer** node.
- **Transfer Protocol**: The transfer protocol used to transfer the call. The options are:
  - **SIP**: If you use the **Consultation Transfer** item in the **AACC Consultation Transfer** node, the transfer protocol is **SIP** to transfer a call to a SIP-enabled Avaya Aura® Contact Center.
  - **AML**: If you use the **Consultation Transfer** item in the **AACC AML Consultation Transfer** node, the transfer protocol is **AML** to transfer a call to Avaya Aura® Contact Center by using the Avaya Aura® Contact Center Open Interface Web Services.

Note:

The **Transfer Protocol** field is available only when you set the **Transfer to AACC** field to **true**.

- **Transfer Mode**: The transfer mode that you want to use to transfer the call-associated UUI and UCID to Avaya Aura® Contact Center. The options are:
  - **Shared**: The system hex encodes the UUI and UCID string, prefixes PD,00;C8 to the hex-encoded UUI and UCID string, and then passes the hex-encoded UUI and UCID string to Avaya Aura® Contact Center during a transfer.
  - **Service Provider**: The system hex encodes the UUI and UCID string, prefixes 04 to the hex-encoded UUI and UCID string, and then passes the hex-encoded UUI and UCID string to Avaya Aura® Contact Center during a transfer.

Note:

The **Transfer Mode** field is available only when you set the **Transfer to AACC** field to **true**.

- **UCID Variable**: The variable that contains the UCID that you want to pass to Avaya Aura® Contact Center.

  If you select a complex variable in the **UCID Variable** field, then in the **UCID Variable Field** field, you must select a complex variable field.

Note:

UCID consumes 10 bytes of UUI data.

The **UCID Variable** field is available only when you set the **Transfer to AACC** field to **true**.

- **UCID Variable Field**: If you select a complex variable in the **UCID Variable** field, then in the **UCID Variable Field** field, select the complex variable field that contains the UCID that you want to pass to Avaya Aura® Contact Center.

Note:

The **UCID Variable Field** field is available only when you set the **Transfer to AACC** field to **true**.

Convert Date String

Type

Datetime function

Available from

Data node

Purpose

You can use the **Convert Date String** function to convert a date and time string of a specified format into a Java date object. This function requires the input in the date and time format. For

more information about special characters used for the format, see the SimpleDateFormat documentation.

Behavior

This function is used for converting a date and time string value into a Java date object. It requires either a constant or a variable input string value, datetime format string, and the variable where the converted Java date object has to be stored.

Properties

Function: Shows the name of the datetime function. If you want to use some other datetime function, select that function from the list.

To Variable: Click the variable where you want to store the converted Java date object.

To Variable Field: If you select a complex variable in the **To Variable** field, then you must select a complex variable field in the **To Variable Field** field. This field is unavailable if you select a simple variable in the **To Variable** field.

Datetime String Format: Type the datetime string format. The system converts the date and time strings based on the format you specify in this field.

To convert the date and time strings based on the time zone, use the letter “Z.”

For information about the pattern letters that represent the components of date and time strings, see Pattern letters representing the components of date and time strings on page 706.

From Constant String: Type the date string, for example, 04/14/2011. Specify a sample date in this field. The system uses this sample date to validate the format of the converted date.

From Variable: Select the date string variable that you want to convert to a Java date object.

From Variable Field: If you select a complex variable in the **From Variable** field, then you must select a complex variable field in the **From Variable Field** field. This field is unavailable if you select a simple variable in the **From Variable** field.

Example usage of the Convert Date String function

About this task

To convert a date string with the 04/01/2011 format from the MyDateStringVar project variable into a Java date object, and store the converted Java date object in the MyDateObjVar project variable, perform the following actions:

1. From the **Palette** pane, drag **Convert Date String** into the data node editor.
2. In the Properties view, set the following properties:
    a. In the **To Variable** field, click **MyDateObjVar**.
    b. In the **Datetime String Format** field, type **MM/dd/yyyy**.
    c. In the **From Constant String** field, type **04/01/2011**.
    d. In the **From Variable** field, click **MyDateStringVar**.

You can also convert the date and time string based on the time zone.

To convert a constant date string with 04/01/2011 2:01 pm -0700 date and time, and time zone into a Java date object and store the converted Java date object in the MyDateObjVar project variable, perform the following actions:

1. From the **Palette** pane, drag **Convert Date String** into the data node editor.
2. In the Properties view, set the following properties:
    a. In the **To Variable** field, click **MyDateObjVar**.
    b. In the **Datetime String Format** field, type **MM/dd/yyyy h:mm a Z**.
    c. In the **From Constant String** field, type **04/01/2011 2:01 pm -0700**.

Pattern letters representing the components of date and time strings

The following table contains the pattern letters that represent the components of date and time strings:

Letter	Date and time component	Presentation	Example
G	Era designator	Text	AD
y	Year	Year	1999; 99
M	Month in year	Month	April; Apr; 11
w	Week in year	Number	14
W	Week in month	Number	3
D	Day in year	Number	251
d	Day in month	Number	15
F	Day of week in month	Number	5
E	Day in week	Text	Wednesday; Wed
a	Am/pm marker	Text	AM
H	Hour in day (0-23)	Number	0
k	Hour in day (1-24)	Number	24
K	Hour in am/pm (0-11)	Number	0
h	Hour in am/pm (1-12)	Number	12
m	Minute in hour	Number	15
s	Second in minute	Number	45
S	Millisecond	Number	728
z	Time zone	General time zone	Pacific Standard Time; PST; GMT-07:00
Z	Time zone	RFC 822 time zone	-0700

All the characters from 'A' through 'Z' and from 'a' through 'z' are reserved.

Convert OD Date/Time

Type

Datetime function

Available from

Data node

Purpose

You can use the **Convert OD Date/Time** function to convert the date and time values from the OD date and time variables into a single Java date object. You can specify either the date or time variable or both as source values. You can use this function when you have already split the input date and time values, and populated the values in the OD date and time variables. For more information, see Set OD Date/Time on page 828.

After you convert the date and time values from the OD date and time variables into a Java date object, you can compare the Java date object with the Java date objects that you created using the Convert Date String on page 704 function. Java date objects make comparison accurate because of their numeric value with millisecond precision.

If you want to convert an OD date and time variable value into a Java date object based on time zone, ensure that the OD date variable has the time zone field and offset time assigned. For example, -0800 for Pacific time in a complex variable.

Behavior

This function converts the date and time values from the OD date and time variables into a Java date object.

You can set either the date or time variable or both in its properties. The OD date variable must contain at least year, month, and day of month fields. The OD time variable must contain at least hour, minute, and second fields.

Properties

- **Function**: Shows the name of the datetime function. If you want to use some other datetime function, select that function from the list.
- **To Variable**: Click the variable where you want to store the converted Java date object.
- **To Variable Field** : If you select a complex variable in the **To Variable** field, then you must select a complex variable field in the **To Variable Field** field. This field is unavailable if you select a simple variable in the **To Variable** field.
- **From OD Date Variable**: Select the OD date variable that you want to convert into a Java date object.
- **From OD Time Variable**: Select the OD time variable that you want to convert into a Java date object.

Example usage of the Convert Date/Time function

About this task

To convert an OD date and time variable into a Java date object and store the converted Java date object in the MyDateObjVar project variable, perform the following actions:

Procedure

1. From the **Palette** pane, drag **Convert OD Date/Time** into the data node editor.
2. In the Properties view, set the following properties:
   a. In the **To Variable** field, click **MyDateObjVar**.
   b. In the **From OD Date Variable** field, click **date**.
   c. In the **From OD Time Variable** field, click **time**.

Add Conference Party (AES)

Type

AES connector item

Available from

Data node on page 653

Purpose

Add a conference party (up to 5 or 6) to a conference call already established.

Behavior

Before you can use this or any other AES connector item, you must enable AES connectors for Orchestration Designer. For more information about the ability of Orchestration Designer to interact with AES, and to enable AES in your Orchestration Designer applications, see AES connectors on page 610.

After a conference call is established with multiple parties (using Conference(AES) on page 711, you can use the **Add Conference Party** in a Data node to add additional parties to the already established conference.

Properties

- **Active Call Info Variable**: A variable for the currently active call.
- **Party Number Variable**: The variable that contains the phone number of the party to add.
- **Party Number Variable Field**: The variable field that contains the phone number of the party to add (if the variable is complex).

Blind Call (AES)

Type

AES connector item

Available From

Data node on page 653 (AES section)

Purpose

Used to transfer the caller to a destination number without knowing the call results. If the transfer fails, the call is lost and the application cannot retrieving it.

Behavior

When this node is invoked, the caller is put on hold, then the number to dial out is attempted. The call will be transferred to the destination automatically in all cases except call failure where the switch will not allow the transfer. In the case of failure, the call is dropped. If successful and UUI data has been indicated, it will be sent to the destination.

Note:

Avaya recommends that you use a **Try/Catch** around the operation to capture catastrophic errors. In all cases, the call is terminated if it fails because it is a blind call.

Properties

- **Name**: Unique name of the node. Will be used to create a call info variable that holds call information about the newly dialed call.
- **Caller Call Info Variable**: Call info variable that represents the current caller. Typically, this is the **callinfo** variable.
- **Dial Number Variable/Field**: Variable and field used to hold the number to dial.
- **UUI Variable/Field**: Variable and field used to hold the UUI information that is passed in with the new call. UUI is user to user information and allows you to pass up to 96 bytes of character data.

Call Info (AES)

Type

AES connector item

Available from

Data node on page 653 only

Purpose

In speech applications that use Application Enablement Services (AES) to extend call control capabilities, the **Call Info** item collects a variety of data about the call and assigns this data to variable fields. You can then use these variable values to define the way the call is handled in the AES environment.

Behavior

Before you can use this or any other AES connector item, you must enable AES connectors for Orchestration Designer. For more information about the ability of Orchestration Designer to interact with AES, and to enable AES in your Orchestration Designer applications, see AES connectors on page 610.

When you drag the **Call Info** item into a Data node, Orchestration Designer automatically creates a complex variable with the same name as the item, and populates it with the following fields:

- **ani**: Contains the telephone number the caller is calling from, as provided by the Automatic Number Identification (ANI) service.
- **callid**: Contains the call ID number for the call, as assigned by the telephony server.
- **dnis**: Contains the telephone number the caller dialed, as provided by the Dialed Number Identification Service (DNIS).
- **stationextension**: Contains the number of the port on the switch that the call came into.
- **ucid**: Contains the universal call ID number.

  The universal call ID number is an identification number that the network assigns to the call when it enters the network. The system uses this number to track the call no matter what other locations or sites in the network the call is routed to.
- **uui**: Contains any user-to-user information (UUI) that exists with the call.

  UUI is text-based data that accompanies the call. For example, during the course of a call, you collect information from the caller such as an account number, customer preferences, customer identification data, and so forth. You can use AES transactions to pass this data to a call center agent, where this data can be used to populate a data window on the agent's computer screen.

  Note:

  Orchestration Designer does not display the **Call Info** variable and its fields until you save the application. After you save the application project, the variable and its fields are available from the Variable Editor. See Variables on page 226.

callinfo is populated before your first node runs. This palette item is used mostly for submodules. You have to pass in a call ID to obtain the **callinfo** data. When you run a submodule, the **callinfo** variable is not populated, however, so **callinfo** is necessary there.

Properties

- **Call ID Variable**: Select the variable to use. Avaya recommends that you use the **callinfo.callid** field that the system creates when you enable AES. Typically this is passed to a module as an input value.
- **Name**: Enter in this field the name you want to assign to the **Call Info** item and its associated variable.

  Orchestration Designer assigns a default name automatically when you drag the **Call Info** item into the node, and you can use the default name or rename it.

  Note:

  This name must follow conventions for naming Java components as described in Conventions for naming Java components on page 59.

Conference(AES)

Type

AES connector item

Available from

Data node on page 653

Purpose

In speech applications that use Application Enablement Services (AES) to extend call control capabilities, the **Conference** item makes it possible to take two telephone calls on the same port and merge them into a single conference call.

This additional caller is added to the conference with announcement (announced) to the existing callers. Sometimes, this type of call operation is also called “three-step conference”.

Behavior

Before you can use this or any other AES connector item, you must enable AES connectors for Orchestration Designer. For more information about the ability of Orchestration Designer to interact with AES, and to enable AES in your Orchestration Designer applications, see AES connectors on page 610.

Before you can conference a call, you must keep the incoming call on hold. For information about keeping calls on hold, see Hold(AES) on page 718. You must then use a Dial(AES) on page 715 item to dial the number to conference the incoming call with. The call you are trying to conference with, then, is the active call. After the system successfully establishes a connection with the number dialed, then it merges the call on hold with the new connection.

To extend the conference beyond three callers and up to a maximum of six callers, you can add callers using either additional Conference(AES) on page 711 items or Conference Party (AES) on page 712 items. Beyond six callers, the Avaya Experience Portal or Interactive Response port drops out and no additional callers are allowed to be added to the conference.

The active **callinfo** variable is the one that is used for the last Dial(AES) on page 715. The active **CallInfo** variable must be used when adding another caller to the conference using Conference(AES) on page 711.

Return status’ are found in the state field of the active **callinfo** variable. There are two possible states: conferenced or fail.

For more information, see AES connectors on page 610.

Properties

- **Held Call ID Variable**: Select the variable that contains the name of the **CallInfo** variable associated with the call that is on hold. You must also select the **callid** field for the **Held Call ID Variable Field**.
- **Held Call ID Variable Field**: Select the **callid** field item.
- **Active Call ID Variable**: Select the variable that contains the name of the **Dial** variable associated with the call that is being dialed. You must also select the **callid** field for the **Active Number Variable Field**.

- **Active Call ID Variable**: Select the **callid** field item.

Conference Party (AES)

Type

AES connector item

Available From

Data node on page 653

Purpose

In speech applications that use Application Enablement Services (AES) to extend call control capabilities, the **Add Conference Party** item makes it possible to add an additional caller to an existing conference.

This additional caller is added to the conference without announcement (unannounced) to the existing callers. Sometimes, this type of call operation is also called “single-step conference”.

Behavior

Assuming that you already have an existing conference call, **Conference Party (AES)** is used to add an additional party to the call. This is sometimes called a “single-step conference” because a party is conferenced into a Call directly. The telephone address string provided as the argument must be complete and valid.

Properties

- **Active Call Info Variable**: The variable that contains the call info for the active call.
- **Party Number Variable**: Holds the predefined variable (phone number) of the conference that the additional party is trying to conference into.
- **Party Number Variable Field**: Select the **Party Number Variable** from the predefined list of fields.

Consultation Call (AES)

Type

AES connector item

Available From

Data node on page 653 (AES section)

Purpose

Used to transfer the caller to a destination number while maintaining control of the call. If the transfer is not successful, the caller will be taken off from hold and the call results can be analyzed for further action.

Behavior

When this node is invoked, the caller is put on hold, then the number to dial out is attempted with a newly dialed call. If the call is successfully transferred, the caller will be transferred as well as any UUI data that has been indicated. If the call fails, the caller is taken off from hold and is able to interact with the application again. If a call enters a queue, it will be automatically transferred when the queued event occurs.

When a consultation call item is used, a call info variable with the name of that consultation call item will be created. The status of the call can be determined from the <Name>.state field where Name is the unique name property given to the consultation call item.

The status of the call can be determined from the **<*Name*>.state** field where Name is the unique name property given to this node and created as a call info variable.

Note:

Avaya recommends that you use a **Try/Catch** around the operation to capture catastrophic errors.

In terms of call states, for the caller on hold, the states include: **transferred**, **disconnected**, or **established**. The **established** state means the caller was not able to transfer and has been taken off hold for an alternative operation. If the “Transfer on Ring” property is set, the noanswer state is not encountered because the operation will not check to determine if the called person has picked up.

Properties

- **Name**: Unique name of the node. Will be used to create a call info variable that holds call information about the newly dialed call and to allow analysis of call.
- **Caller Call Info Variable**: Call info variable that represents the current caller. Typically, this is the **callinfo** variable.
- **Dial Number Variable/Field**: Variable and field used to hold the number to dial.
- **UUI Variable/Field**: Variable and field used to hold the UUI information that is passed in with the new call. UUI is user to user information and allows you to pass up to 96 bytes of character data.
- **Ring Max**: Maximum number of rings before determining the call is ring no answer. Each ring will take approximately 6 seconds.
- **Transfer on Ring**: When this property is set to true, the consultation call transfers the call when it detects a ring event. All errors that occur before the ring are handled properly. The ring timer does not have any affect when this flag is set because the call is transferred on first ring.

Coordination Parallel

Type

Coordination

Available From

Root level of the prompt

Purpose

The Coordination Parallel is a media container. It defines a simple time grouping in which multiple elements can play at the same time. These elements must be of a media type (media page or SMIL Link) and an audio type. The elements of coordination parallel cannot be of the same type.

Behavior

By default, the elements within a parallel container play at the same time. However, the child elements nested within a parallel container must be from different groups.

For example, the audio group consists of TTS, Phrase, Text Variable, Phrase Variable, Audio Variable, and Audio Constant, while the video group consists of the Media Page element. You can nest the MediaPage element to play in parallel with the TTS, Phrase, Text Variable, Phrase Variable, Audio Variable, and Audio Constant elements. However, you cannot nest Mediapage element with another Mediapage element and similarly TTS element with another TTS, Phrase, or other elements from the same group to play in parallel.

You can also nest Coordination Parallel and Coordination Sequence elements under this container.

Properties

- **End Sync**: Determines how to complete the prompt end condition. The End Sync option can have a value of “first” or “last”.
  - A value of “first” indicates the prompt end condition after the first media stream finishes playing.
  - A value of “last” indicates the prompt end condition after the last prompt finishes playing. For the media that has completed playing and is in waiting mode, the behavior is to play silence for audio, and freeze the last frame for video.

Coordination Sequence

Type

Coordination

Available From

Root level of the prompt

Purpose

The Coordination Sequence is a media container. It defines a simple time grouping in which multiple elements play in sequence.

Behavior

By default, elements within a sequence container play one after the other. You can nest more than one Coordination Sequence elements under this container. However, you cannot nest Coordination Parallel on page 713 as a child of the Coordination Sequence element in this container.

Note:

If you group 2 items one after another, Orchestration Designer plays them in sequence by default. For example:

```
   TTS
Media Page
```

is equivalent to

```
Sequence
   TTS
Media Page
```

Properties

None.

Dial(AES)

Type

AES Item

Available from

Data node on page 653 only

Purpose

In speech applications that use Application Enablement Services (AES) to extend call control capabilities, the **Dial** item requests that a call be placed from the IVR system to a destination telephone number.

Behavior

Before you can use this or any other AES connector item, you must enable AES connectors for Orchestration Designer. For more information about the ability of Orchestration Designer to interact with AES, and to enable AES in your Orchestration Designer applications, see AES connectors on page 610.

Note:

During an AES dial operation, when a queue is hit, the AES connector stops the ring timer and returns a “queued” state back to the application. The application checks for that state and transfers the call. In the case of an AES Consultation Call (AES) on page 712, the call is automatically transferred when it hits a queue.

Note:

Avaya recommends that you use a **Try/Catch** around the operation to capture catastrophic errors.

When you drag the **Dial** item into a Data node, Orchestration Designer automatically creates a complex variable with the same name as the item, and populates it with the following fields:

- **ani**: Contains the telephone number the original caller is calling from, as provided by the Automatic Number Identification (ANI) service.

- **callid**: Contains the call ID number for the original call, as assigned by the telephony server.
- **dnis**: Contains the telephone number the original caller dialed, as provided by the Dialed Number Identification Service (DNIS).
- **state**: Returns the current state of the call. Possible values include the following values: **established**, **disconnected**, **failed**, **queued**, **unreachable**, **noanswer**, and **busy**.
- **stationextension** - Contains the number of the original port on the switch that the call came into.
- **ucid**: Contains the universal call ID (UCID) number. This field is only valid if it is enabled on the PBX/switch.

  The universal call ID number is an identification number that the network assigns to the call when it enters the network. The system uses this number to track the call no matter what other locations or sites in the network the call is routed to.
- **uui**: Contains any user-to-user information (UUI) that exists with the call.

  UUI is text-based data that accompanies the call. For example, during the course of a call, you collect information from the caller such as an account number, customer preferences, customer identification data, and so forth. You can use AES transactions to pass this data to a call center agent, where this data can be used to populate a data window on the agent's computer screen.

  Note:

  Orchestration Designer does not display the **Dial** variable and its fields until you save the application. After you save the application project, the variable and its fields are available from the Variable Editor. For more information about variables, see Variables on page 226.

This item is used in conjunction with a Transfer (AES) on page 720 or Conference(AES) on page 711 operation. For more information, see AES connectors on page 610.

Properties

- **Name**: Enter in this field the name you want to assign to the **Dial** item and its associated variable.

  Orchestration Designer assigns a default name automatically when you drag the **Dial** item into the node, and you can accept the default name or rename it.

  Note:

  This name must follow conventions for naming Java components as described in Conventions for naming Java components on page 59.
- **Caller Call Info Variable**: Call info variable that represents the current caller. Typically, this is the **callinfo** variable.
- **Dial Number Variable**: Select the variable that contains the telephone number for the system to dial.

  If this variable is a complex variable, you must also select a **Dial Number Variable Field**.
- **Dial Number Variable Field**: If the **Dial Number Variable** is a complex variable, select the variable field that contains the telephone number for the system to dial.

- **UUI Variable**: Select the variable that contains the user-to-user information (UUI) that you want to pass to the destination telephone number.

  If this variable is a complex variable, you must also select a **UUI Variable Field**.

  This variable can contain up to 96 bytes of data.
- **UUI Variable Field**: If the UUI Variable is a complex variable, select the UUI variable field to which you want to pass to the destination telephone number.
- **UUI as ASCII**: Decides whether to pass the UUI as ASCII or binary.
  - **True**: Pass the UUI as ASCII.
  - **False**: Pass the UUI as binary.
- **Ring max**: Enter in this field the number of rings the system should wait for the destination telephone number to respond before the system terminates the attempt. Whenever the destination number exceeds this number of rings, then the system returns a state of **noanswer** and moves to the next item in the subflow.
- **Ring max Variable**: A variable indicating the ring max.
- **Ring max Variable Field**: If Ring max Variable is complex, use this field.

Disconnect (AES)

Type

AES connector item

Available from

Data node on page 653 only

Purpose

In speech applications that use Application Enablement Services (AES) to extend call control capabilities, the **Disconnect** item terminates a call that has been transferred or conferenced using AES.

Behavior

Before you can use this or any other AES connector item, you must enable AES connectors for Orchestration Designer. For more information about the ability of Orchestration Designer to interact with AES, and to enable AES in your Orchestration Designer applications, see About AES connectors on page 610.

When you drag the **Disconnect** item into a Data node, Orchestration Designer uses it to terminate a AES connection that has been established either in this node or another.

Properties

- **Call ID Variable**: Select the variable to use. Avaya recommends that you use the **callinfo** variable that the system creates when you enable AES.

Hold(AES)

Type

AES connector item

Available from

- Data node on page 653

Purpose

In speech applications that use Application Enablement Services (AES) to extend call control capabilities, the **Hold** item puts a caller on hold while the call is transferred or conferenced.

Behavior

Before you can use this or any other AES connector item, you must enable AES connectors for Orchestration Designer. For more information about the ability of Orchestration Designer to interact with AES, and to enable AES in your Orchestration Designer applications, see AES connectors on page 610.

When you drag the **Hold** item into a Data node, the Orchestration Designer application keeps the original call in a hold state until one of the following is true:

- The call transfer or conference has been established.
- The system determines that the transfer or conference cannot be established and a Retrieve(AES) on page 719 item takes the call out of the hold state.

Properties

- **Call ID Variable**: Select the variable to use. Avaya recommends that you use the **callinfo** variable that the system creates when you enable AES.

Remove Conference Party (AES)

Type

AES connector item

Available from

- Data node on page 653

Purpose

Remove a conference party from the conference call already established.

Behavior

Before you can use this or any other AES connector item, you must enable AES connectors for Orchestration Designer. For more information about the ability of Orchestration Designer to interact with AES, and to enable AES in your Orchestration Designer applications, see About AES connectors on page 610.

After you establish a conference call with multiple parties, you can use the **Remove Conference Party** in a Data node to remove any party in an already established conference.

Properties

- **Active Call Info Variable**:A variable for the currently active call.
- **Party Number Variable**: The variable that contains the phone number of the party to remove.
- **Party Number Variable Field**: The variable field that contains the phone number of the party to remove (if the variable is complex).

Retrieve(AES)

Type

AES connector item

Available from

- Data node on page 653

Purpose

In speech applications that use Application Enablement Services (AES) to extend call control capabilities, the **Retrieve** item puts a caller that has been on hold back in an active state.

Behavior

Before you can use this or any other AES connector item, you must enable AES connectors for Orchestration Designer. For more information about the ability of Orchestration Designer to interact with AES, and to enable AES in your Orchestration Designer applications, see AES connectors on page 610.

When you drag the **Retrieve** item into a Data node, the Orchestration Designer application takes an AES call that is in a Hold(AES) on page 718 state and puts it back in an active state within the application. This item is usually used when an AES call transfer or conference has been attempted but not successfully established.

Properties

- **Caller Call Info Variable**: Call info variable that represents the current caller. Typically, this is the **callinfo** variable.

Separator

Type

Node item

Available from

Data node on page 653

Purpose

Allows you to add whitespace between items in the data tree. Can be used to visually group items together by separating items from other items.

Behavior

There is no behavior associated with the separator except in a visual sense of the data tree. It is used to insert a visible and physical space between two items in the data tree.

Transfer (AES)

Type

AES connector item

Available from

Data node on page 653 only

Purpose

In speech applications that use Application Enablement Services (AES) to extend call control capabilities, the **Transfer** item makes it possible to transfer not only the call, but also associated data.

Note:

This type of transfer is different from the standard call transfer capabilities of Orchestration Designer. The two standard call transfer types, Blind Transfer and Bridged Transfer, both transfer only the call. They do not provide the capability to transfer data with the call.

Behavior

Before you can use this or any other AES connector item, you must enable AES connectors for Orchestration Designer. For more information about the ability of Orchestration Designer to interact with AES, and to enable AES in your Orchestration Designer applications, see AES connectors on page 610.

Note:

Support for call transfer functions can vary from one IVR system to another. For more information and details about support for transfer functions on your system, see the documentation for your IVR system.

Before you can transfer a call and its associated data to another number, you must keep the incoming call on hold. For information about placing calls on hold, see Hold(AES) on page 718. You must then use a Dial(AES) on page 715 item to dial the number to transfer to. The number you are trying to transfer to, then, is the active call. After the system successfully establishes a connection with the number dialed, then it connects the call on hold with the new connection.

When an exception is caught on a Transfer, you can do a Blind Transfer on page 678, which is helpful when doing batch fetches.

For more information, including a suggested procedure to create a conference call using AES connectors in Orchestration Designer, see Example procedure for constructing a call and data transfer using AES connectors on page 613.

Properties

- **Call ID Variable**: Select the variable to use. Avaya recommends that you use the **callinfo** variable that the system creates when you enable AES.
- **Hold Call Info Variable**: Use the variable associated with the **Hold** item. In this case, the **callinfo** variable in **XferSetup**.
- **Active Call Info Variable**: Use the variable associated with the **DialCall** item. In this case, the **Dial Number Variable**.

Database Operation

Type

Node item

Available from

Data node on page 653

Purpose

The **Database** item makes it possible to use a predefined database operation. The **Database** item in itself only invokes the database operation and tells the call flow or the message flow where to use it.

Behavior

Before you can use the **Database** item in the Data node, you must have a data source defined for the application. For more information about defining and using data sources, see Data source management on page 295.

You must also have at least one database operation created for the application. For information about setting up and using database operations, see Creating a database operation file on page 300.

The way the database operation actually behaves in the Data node depends on how the database operation is defined. When you drag the **Database** item into the Data node flow, all Orchestration Designer does is invoke the selected database operation.

Properties

- **Name**: Select the database operation you want to invoke.

Database Transaction

Type

Node item

Available from

Data node on page 653

Purpose

Used for grouping multiple database operations in a single transaction. If one step fails in a transaction, all the changes are rolled back. Changes are only committed when all operations within the transaction succeed.

Behavior

By combining a Database Operation on page 721 item and Operation on page 777 item into a Data node on page 653, the items can be grouped within the Database Transaction on page 721 node to make a single transaction.

Properties

None.

Default

Type

Condition item

Available from

Data node in speech and message applications

Prompt file editor in speech and message applications

Purpose

This is an optional statement. You can use a Default statement to execute an alternative operation if none of the case values matches the value of switch variable.

Note:

A Default statement must have at least one child element.

Behavior

The Default statement is equivalent to the Else on page 722 statement. If none of the case values matches the value of the switch variable, the Switch statement executes the operations following that Default statement.

You can add a Default statement after the last case statement.

Properties

None.

Else

Type

Condition item

Available from

Any level tab in the Prompt File Editor

Data node on page 653 in the Call Flow Editor

Purpose

The Condition on page 699 item makes it possible to select and play different prompt or data segments based on the condition being tested. It works in conjunction with If on page 739 and **Else** items to test conditions. The **Else** item defines a default for cases where no conditions of an **If** item are met.

In the Call Flow Editor, the Else item is used to conditionally execute different operations based on the evaluation of logical statements. It works in conjunction with an If item to execute operations when the “If” condition evaluates to a “false” value.

Behavior

The **Else** item cannot be used independently but must be used in conjunction with the If on page 739 item.

If there are multiple **If** items, all the **If** items are tested first. If none of the conditions being tested for are met in any of the **If** items, the system returns the result assigned to the **Else** item.

Properties

The **Else** item has no inherent properties. It can only have one or more prompt segments assigned to it.

Else If

Type

Condition item

Available from

Any level tab in the Prompt File Editor

Data node on page 653 in the Call Flow Editor

Purpose

The **Else If** item is used for building complex if/else logic. With the Condition on page 699 item, you can select and play different prompt or data flow segments based on various conditions. The **Else If** introduces a new “nested if” condition. If the If on page 739 condition fails, the **Else If** item makes it possible to compare the value of a selected variable, the **Left Variable** or **Left Variable Field** with:

- A condition
- A constant
- The value of another variable or variable field

Based on the result of the comparison, the system responds according to what is assigned to the **Else If** item, or continues on.

You can create the following types of conditions:

- **Single condition**. In the Properties view of the **If** or **Else If** item, select an operator, and set the required properties.
- **Compound condition**. Ensure that the **Operator** property in the Properties view of the **If** and **Else If** item is blank, and then use AND on page 675, OR on page 790, or NOT on page 772 to connect multiple expressions to form a compound condition. You can nest a compound condition in another compound condition to create a more complex condition.

For more information, see Creating single and compound conditions on page 608.

Behavior

The behavior of this item depends largely on the conditional **Operator** you select. When the condition satisfies, the application plays the prompt segments assigned to the **Else If** item. For more information about conditional operators and how to use them, see Conditional operators on page 602.

Orchestration Designer executes the **Else If** statement only if the preceding **If** expression and any of the preceding **Else If** expressions are evaluated to FALSE, and the current **Else If** expression is evaluated to TRUE.

Properties

- **Left Variable**: Select the variable you want to use to make the comparison.

  If you select a simple variable, use the variable for the comparison.

  If you select a complex variable, you must also select a variable field from the **Left Variable Field** list for the comparison.

  Note:

  For more information about simple and complex variables, see Variables on page 226.
- **Left Variable Field**: If you have selected a complex variable in the **Left Variable** field, select the field that you want to use to make the comparison. If you have selected a simple variable in the **Left Variable** field, this field is inactive and no choices are available.
- **Compare**: Select the conditional operator you want to use to make the comparison. For more information about conditional operators and how to use them, see Conditional operators on page 602.
- **Value**: Enter a literal or a constant value you want to compare the **Left Variable** or **Left Variable Field** with.

  For example, if you want to compare the account balance to see if it has a minimum value of at least $2000, and if it has, you want to play another prompt. To do this, enter $2000 in the **Value** field and the appropriate conditional operator in the **Compare** field.

  Note:

  If you use the **Value** field, you cannot use the **Right Variable** field.
- **Right Variable**: Select the variable you want to compare with the **Left Variable** or **Left Variable Field**.

  If you select a simple variable, use the variable for the comparison.

If you select a complex variable, you must also select a variable field from the **Right Variable Field** list for the comparison.

For more information about simple and complex variables, see Variables on page 226.

Note:

If you use the **Right Variable** field, you cannot use the **Value** field.

- **Right Variable Field**: If you have selected a complex variable in the **Right Variable** field, select the field that you want to use to make the comparison. If you have selected a simple variable in the **Right Variable** field, this field is inactive and no choices are available.

Email

Type

Prompt Editor Palette item

Available from

Prompt Contents tab of the prompt file editor in an email channel message application.

For more information about the Prompt File Editor, see Prompt file editor in a message application on page 207.

Purpose

Use the **Email** item in an email channel message application to send an outbound email message.

For example, you want to send an outbound email message to your customer for an upcoming offer from an email channel message application.

You can specify the email message that you want to send in an email file. Create a prompt file and add the **Email** item to the prompt file. Assign the email file to the **Email** item. You can then add the **Prompt** item to the **Announce** node from where you want to send the email message and assign the prompt file to the **Prompt** item.

When the system executes the **Announce** node, the system sends the email message to the text browser that is located on Avaya Experience Portal Media Processing Platform (MPP) for further processing and delivery.

Behavior

The **Email** item sends an outbound email message from an email channel message application.

Properties

- **Email Constant**: Click the email file that contains the email message that you want to send. The **Email Constant** field is unavailable if you select a variable in the **Email Variable** field.
- **Email Variable**: Click the variable that contains the email message that you want to send. If you select a complex variable in the **Email Variable** field, then you must select a complex variable field in the **Email Variable Field** field. The **Email Variable** field is unavailable if you select an email file in the **Email Constant** field.
- **Email Variable Field**: If you select a complex variable in the **Email Variable** field, then in the **Email Variable Field** field, select the complex variable field that contains the email message

that you want to send. The **Email Variable Field** field is unavailable if you select a simple variable in the **Email Variable** field.

Emphasis

Type

SSML item

Available from

Any level tab in the Prompt File Editor. For more information about the prompt file editor, see Prompt file editor in a speech application on page 207.

Purpose

In natural speech, some words are emphasized more than others. That is, a greater stress is given on some words than the other words. The Emphasis item causes the speech synthesis engine to emphasize on selected text to be rendered.

Note:

The Emphasis item and its properties function correctly only with SSML-compliant speech synthesis engines. Microsoft SAPI Speech, which is used by Orchestration Designer during application simulation, is *not* an SSML-compliant speech synthesis engine, so any settings you make with this item are ignored. For more information about the SSML standard, see the Speech Synthesis Markup Language (SSML) Version 1.0 W3C Recommendation.

Behavior

When you drag an Emphasis item into a prompt, you can choose to either:

- Drag it into the root level of the prompt.
- Attach it to a specific TTS or text variable segment.

If you drag the Emphasis item into the root level of the prompt, the setting of the Emphasis item affects all the TTS and text variables segments from that point forward. It remains in effect for the rest of the prompt or up to the point where you drag another Emphasis item into the root level.

If you attach the Emphasis item to a particular TTS or text variable segment, the setting of the Emphasis item affects only that TTS or text variable segment.

Properties

- **Level**: Select one of the following settings:

Emphasis Setting	Description
strong	Applies a great amount of emphasis to the selected text.
medium	(default) Applies a moderate amount of emphasis to the selected text.
none	Applies no emphasis to the selected text. You can use this setting, in effect, to cancel a prior Emphasis item setting.
reduced	In effect, acts as the opposite of emphasizing a word. This setting actually reduces the normal amount of emphasis on the selected text. For example, “going to” with a setting of **reduced** renders as “gonna.”

Note:

The actual amount of emphasis applied to the text depends on the speech synthesis engine. The amount of emphasis also depends on the language being spoken. This is because each language indicates emphasis differently, based on a combination of pitch changes, timing changes, loudness, and other acoustic differences.

Example usage of the Emphasis item

About this task

To create a TTS sentence that says, “That car is a *beautiful* one,” with a strong emphasis on the word “beautiful”:

Procedure

1. Create a prompt with three TTS prompt segments:
   - “That car is a”
   - “beautiful”
   - “one”
2. Attach an **Emphasis** item to the second segment with a **Level** setting of **strong**.

Exit

Type

Event handler

Available from

- **AppRoot** node (see AppRoot node on page 132)
- Form node on page 655

Purpose

When events are caught, at the **Field** level, **Form** level, or **AppRoot**, there are various ways to handle the event such as re-playing prompts, or going to different forms. In the case where there are more serious errors, use the **Exit** event handler to exit the application.

Behavior

The exit item is used to disconnect the caller and terminate the session. The exit item is intended to be used when “fatal errors” occur and there is no recovery.

A typical use of the Exit item is as a child of Catch (“error.badfetch”). Usually, when the application encounters these “error.badfetch” errors, it means that the application server is unavailable or the session on the application server is no longer valid.

In this scenario, it is not possible to fetch other VXML or Text XML pages to handle the event because this usually results in additional “error.badfetch” events. To prevent endless bad fetch loops, using Exit will allow the platform to terminate the session.

See Catch on page 693 for other ways to handle VXML events.

Properties

- **Threshold**: A variable that must have a positive integer value.

  Note:

  This field is available only when the **Exit** item is dragged into an event handler or a node.

  This property is used to disable the **Exit** item until this value is met. For example, if the **Exit** property is set to 3, the system ignores the Throw item the first two times the specified event is encountered. Only when the event occurs for the third time, the system will throw the event.

Expression

Used to embed an ECMA script expression within a prompt. This can be used for playing event details or other ECMA script values.

For example, when a prompt is played in an Event handler, the prompt can use the Expression element to play “_message” (the event message).

Using Expression elements in a prompt requires knowledge of the valid ECMA script objects in a VXML document. The value of the Expression property is literally embedded in the generated VXML, so if the Expression is invalid, there can be VXML parse or semantic errors.

Expression

Type

Boolean operator

Available from

Data node

Any level tab in the Prompt File Editor

Purpose

You can use the Expression Boolean operator to create Boolean expressions under the AND on page 675, OR on page 790, and NOT on page 772 Boolean operators. For more information, see Creating single and compound conditions on page 608.

Behavior

This item creates a Boolean expression that contains two operands with a comparison operator in between. This item evaluates to either true or false. This item is used as a child of the AND, OR, and NOT Boolean operators to create a compound condition.

Properties

Operator: Select the conditional operator and set the other properties accordingly. For more information about conditional operators, see Conditional operators on page 602.

External Property

External Property item

Type

Variable item

Available from

- **AppRoot** node (see AppRoot node on page 132)
- Form node on page 655

Purpose

The W3C VoiceXML 2.0 Recommendation uses the <property> element to set values that affect system behavior. These values can include properties like timeouts, recognition processes, and caching policies. Orchestration Designer uses the **Property** item to set many of these common properties, but some IVR systems or text processing systems make it possible to set other properties that are not part of the standard Orchestration Designer set.

In addition to the standard properties that Orchestration Designer supports, some IVR systems or text processing systems have custom properties available. Use the **External Property** item to set those properties on the IVR system or text processing system.

Note:

For more information about the standard properties supported in Orchestration Designer, see Property on page 799.

Behavior

The **External Property** item takes a named property value and passes it to the IVR system or the text processing system. The format, content, and resultant behavior of the value depend on what the IVR system or the text processing system expects and can process.

An example of using an **External Property** item can be described in the usage of the VoiceXML property **fetchtimeout** as an External Property item. In this case, the voice browser (AAS) waits a given amount of time (that is, the fetch timeout time) to get a page response back from the application. This timeout is configurable on the platform and applies to all fetches done for all applications. Some applications can exceed this timeout during long operations. Such operations can be database access, an AES dial operation that has a long ring timeout, Web service calls.

Instead of changing the platform timeout, set the **fetchtimeout** property in a node or in the AppRoot (for an application-wide change) and change the timeout to a more appropriate value. For example, set Name to fetchtimeout and value to 30 seconds (15 seconds more than the default timeout).

Note:

A very long **fetchtimeout** can have a negative effect on the efficiency of the browser in cases where the application has a catastrophic error and the browser is waiting for a page response or exceeding the timeout.

Properties

- **Name**: Enter the name to identify the property. This name is sent, along with the **Value**, to the IVR system or the text processing system on which the property is to be set.
- **Value**: Enter the exact value to be sent to the IVR system or the text processing system. This must be in the form and format that the IVR system or the text processing system expects and can process.
- **Variable**: Select the variable you want to use. If the variable you select is a complex variable, you must also select a **Variable Field**.
- **Variable Field**: Select the variable field you want to use. The contents of this list depend on which variable you selected in the **Variable** property.
- **Hex-encode Value**: Select whether you want to hex encode the data to be sent to the IVR system or the text processing system.

External Property item in AACC Blind Transfer, AACC Bridged Transfer, and AACC Consultation Transfer nodes

Type

Variable item

Available from

AACC Blind Transfer node on page 635

AACC Bridged Transfer node on page 637

AACC Consultation Transfer node on page 639

Purpose

Use the **External Property** item in the **AACC Blind Transfer**, **AACC Bridged Transfer**, and **AACC Consultation Transfer** nodes to pass the contact-intrinsic data and Call Attached Data (CAD) in SIP headers to a SIP-enabled Avaya Aura® Contact Center.

Behavior

When you add the **AACC Blind Transfer**, **AACC Bridged Transfer**, and **AACC Consultation Transfer** nodes to a call flow, the system creates a default structure for the node and adds the following items to the node:

- **Blind Transfer**, **Bridged Transfer**, or **Consultation Transfer** item depending on the node.
- **Prompt** item.
- Two **External Property** items that represent the name and value pair of the SIP header.
- **Next** item.

You can use the **External Property** items to specify the SIP headers that you want to pass to a SIP-enabled Avaya Aura® Contact Center. You can specify the contact-intrinsic data in the SIP headers.

If you want to add more SIP headers, either copy the default **External Property** items that represent the SIP header name and value pair or drag two **External Property** items from the **Palette** pane, one to set the header name and the other to set the header value.

- For the **AACC Bridged Transfer**, and **AACC Consultation Transfer** nodes, you can configure the following properties
  - `AVAYA_SIPHEADER.session.connection.protocol.sip.unknownhdr[i].name property value: {header *name*}`
  - `AVAYA_SIPHEADER.session.connection.protocol.sip.unknownhdr[i].value property value: {header *value*}`

    Where, the element [i] indexes the array of headers. For example, 0 is for the first SIP header, 1 is for the second SIP header.
- For the **AACC Blind Transfer** node, you can configure the following property:

  `AVAYA_SIPHEADER.session.connection.protocol.sip.referto.header[0] = "P-Intrinsics="`

  Where, you must append the P-intrinsics value string to the end of `"P-Intrinsics="` in the value of **External Property** item property.

If you want to pass more contact intrinsic data, you can use the **P-Intrinsics** SIP header name in the **External Property** item. You can also pass CAD data along with the contact-intrinsic data by using the **P-Intrinsics** SIP header. **P-Intrinsics** can pass up to 1024 bytes of hex-encoded data.

Note:

P-Intrinsics is the default SIP header name of the default **External Property** items that the system adds to the AACC Transfer nodes.

If you use the **P-Intrinsics** SIP header, you must the set the contact-intrinsic and CAD data that you want to send to Avaya Aura® Contact Center in the following XML format:

```
<cc>
 <i>key1=value1</i>
 <i>key2=value2</i>
 <i>key3=value3</i>
 <i>key4=value4</i>
 <i>key5=value5</i>
 <cad>CAD</cad>
</cc>
```

Each <i> tag contains the key name and value pair of the contact-intrinsic data. Key name must be unique and can contain up to 25 characters. Value can contain up to 80 characters. You can specify up to five key name and value pairs of contact-intrinsic data. You can use the <cad> tag to specify the CAD data.

Note:

To construct the XML string by using variables and constant values, use the **Concatenate** operation in the **Operation** item within a **Data** node.

Ensure that there is no space or carriage return in the data that you specify in the XML string for the **P-Intrinsics** SIP header.

The system automatically hex-encodes the data contained in the XML string when passing the data to a SIP-enabled Avaya Aura® Contact Center.

Properties

The following table shows the description of the **External Property** item properties depending on whether you use the **External Property** item to specify the SIP header name or header value.

Property name	Description if you use the External Property item for the SIP header name	Description if you use the External Property item for the SIP header value

Name	Type the SIP header for the header name.
	• For the **AACC Bridged Transfer**, and **AACC Consultation Transfer** nodes, the default header in the default **External Property** item is:
	`AVAYA_SIPHEADER.session.connection.protocol.sip.unknownhdr[0].name`
	• For the **AACC Blind Transfer** node, the default header in the default **External Property** item is:
	`AVAYA_SIPHEADER.session.connection.protocol.sip.referto.header[0] = "P-Intrinsics="`
		Type the SIP header for the header value.
		• The default header in the default **External Property** item is:
		`AVAYA_SIPHEADER.session.connection.protocol.sip.unknownhdr[0].value`
		• For the **AACC Blind Transfer** node, the default header in the default **External Property** item is:
		`AVAYA_SIPHEADER.session.connection.protocol.sip.referto.header[0] = "P-Intrinsics="`

Value	Type a value for the SIP header name. **Note:** For the **AACC Bridged Transfer**, and **AACC Consultation Transfer** nodes, **P-Intrinsics** is the default SIP header name in the default **External Property** item. For the **AACC Blind Transfer** node, you must append the value string to the end of `"P-Intrinsics="` in the value of **External Property** item property.	Type the contact-intrinsic data that you want to send in the SIP header. If you use the **P-Intrinsics** SIP header name, then you can specify the CAD data along with the contact-intrinsic data. For the **AACC Blind Transfer** node, you must append the value string to the end of `"P-Intrinsics="` in the value of **External Property** item property. **Note:** **P-Intrinsics** can pass up to 1024 bytes of hex-encoded data.

Table continues…

Property name	Description if you use the External Property item for the SIP header name	Description if you use the External Property item for the SIP header value
Value Variable	Select the variable that contains the value for the SIP header name.	Select the variable that contains the contact-intrinsic data that you want to send in the SIP header. If you use the **P-Intrinsics** SIP header name, then you can select the variable that contains the contact-intrinsic data and CAD data.
Value Variable Field	If you select a complex variable in the **Value Variable** field, then in the **Value Variable Field** field, you must select a complex variable field that contains the value for SIP header name.	If you select a complex variable in the **Value Variable** field, then in the **Value Variable Field** field, you must select a complex variable field that contains contact-intrinsic data and CAD data, as appropriate.
Hex-encode Value	Option to specify whether you want to hex encode the SIP header name that you want to pass to Avaya Aura® Contact Center. The options are: • **true**: If you want to hex encode the SIP header name that you want to pass to Avaya Aura® Contact Center. • **false**: If you do not want to hex encode the SIP header name that you want to pass to Avaya Aura® Contact Center.	Option to specify whether you want to hex encode the data that you want to pass in the SIP header to Avaya Aura® Contact Center. The options are: • **true**: If you want to hex encode the contact-intrinsic data and CAD data that you want to pass in the SIP header to Avaya Aura® Contact Center. The default is **true**. • **false**: If you do not want to hex encode the contact-intrinsic data and CAD data that you want to pass in the SIP header to Avaya Aura® Contact Center.

Field

Type

Variable item

Available from

Variable Editor, with complex variables only. For more information about variables and the Variable Editor, see Variables on page 226.

Purpose

A Complex Variable on page 698 is a variable that has several attributes, called *fields*. Complex variables make it possible to use a single variable to pass along multiple values. To be valid, a complex variable must have at least one **Field** item assigned to it.

Behavior

In a sense, a **Field** item acts like a simple variable within a complex variable. In other words, each **Field** item contains a single variable value.

When you create a complex variable, Orchestration Designer automatically assigns a temporary name to it and adds one **Field** item. You can rename both the variable and the field. You can also add as many additional **Field** items as you want.

In some cases, Orchestration Designer creates complex variables automatically when you use certain nodes or items in the call flow or message flow. For complex variables that Orchestration Designer creates automatically, the fields are also created automatically. As a general rule, you cannot add **Field** items to these variables. For example, when you drag a Bridged Transfer item into the call flow or the message flow, Orchestration Designer automatically creates a complex variable with the same name as the item. This complex variable automatically has four fields created and assigned to it: **duration**, **inputmode**, **utterance**, and **value**.

Properties

- **Name**: Enter in this field the name you want to identify the variable field.

  Note:

  This name must follow conventions for naming Java components as described in Conventions for naming Java components on page 59.

- **Value**: Enter in this field the default value you want to assign to the variable field, if any. If you leave this field blank, the default value is undefined.
- **Private**: (True/False). When you set the **Private** attribute to **true**, the value of this variable is not traced or logged in the framework reporting. However, the value of a private variable is visible only if you add an application trace or application report. If you use a private variable as in/out params for VXML subdialogs, the variable value is encrypted. For example, <var name="untitledOutput1" expr="'___AvayaEncrypted_dJklKMkrNmg='"/>. The variable value is unencrypted on submission. If your application returns a private variable on exit, the value is not encrypted. However, the value is encrypted on a subdialog return. Private values that are passed to an OSDM are not encrypted. If a prompt contains a private variable, the variable value does not appear in the VXML trace because the page is not written to the trace.log.

Flush Prompts

Type

Form item

Available from

- Announce node on page 634
- Prompt and Collect node on page 659

- Blind Transfer node on page 646
- Bridged Transfer node on page 647
- Record node on page 660
- Form node on page 655

Purpose

The **Flush Prompts** item causes queued output to be played before transitioning to the next node.

Behavior

When executed, the **Flush Prompts** item directs the voice browser to play all queued output immediately.

Only one **Flush Prompts** item can be added in any one node. An example of where the **Flush Prompts** is useful is when doing an AES transfer. When doing a transfer one would typically play a prompt “please wait while I transfer your call”. If a caller does not hear a particular prompt, a Flush Prompts item can be added just before the prompt item that is not playing. This forces any queued prompts to play before moving on.

It does not matter where the **Flush Prompts** item appears in the subflow of a node.

Note:

If you use the **Flush Prompts** item in an **AppRoot** node, the system plays the prompts that are configured in the node that is currently executed, and then fetches the next node. Avaya recommends that you use the **Flush Prompts** item on individual nodes, where you want the system to fetch and play prompts, rather than applying the **Flush Prompts** item globally in the application.

Properties

None

Get VDU Fields

Type

IC Connector item

Available From

Data node on page 653 (IC section)

Purpose

Get VDU Fields allows you to get up to 20 vdu variables in one round trip to the VOX.

Behavior

When this node is invoked, the data will be retrieved from the VOX and added into the **vdu_cache** complex variable into the field names that match the names of the vdu fields indicated. You can then access each variable out of the cache.

Properties

- **vdu_field1 - vdu_field20**: Indicates the name of the vdu field to retrieve. You must first add the field to the vdu complex variable.

Goto

Type

Form item

Available from

- **AppRoot** node (see About the AppRoot node on page 132)
- Announce node on page 634
- Prompt and Collect node on page 659
- Collect node on page 651
- Blind Transfer node on page 646
- Bridged Transfer node on page 647
- Record node on page 660
- Menu node on page 656
- Form node on page 655
- Servlet node on page 663

Purpose

The **Goto** item makes it possible to redirect a call, message, or web flow to a specific node in response to an event.

For example:

In a speech application, if a caller is having trouble making a response that the system can match, you want to redirect the caller to a live attendant. You can set the **No Match** event handler to go to a Blind Transfer node, which transfers the caller to the attendant.

In a message application, if a user is having trouble making a response that the system can match and you want to redirect the user to another node, you can set the **No Match** event handler to go to the other node.

Double click the **Goto** node to open the specified node in the node editor.

Behavior

When the system catches an event to which the **Goto** item is assigned, the application directs the call or the message to the node specified in the **Goto** item.

With the exception of the Servlet node, the **Goto** item cannot be used independently. It is always used as a sub-item of an event handler.

Note:

In the Servlet node, the **Goto** item is used independently.

The **Goto** item cannot be used with same event handler as a **Throw** item.

Properties

- **Goto Form**: Select the node to which you want to direct the call when the specified event is caught.
- **Name**: Display name for the item.
- **Threshold**: A variable that must have a positive integer value.

  The **Threshold** setting determines how many times the specified event must be caught before application redirects the call flow to the **Goto Form**. Until this threshold is reached, the system plays the prompt assigned to the event handler, if any, and then waits for another response from the user.
- **Threshold Variable**: The variable that contains the number of time the event must be caught before redirecting the call flow or the message flow.
- **Threshold Variable Field**: The variable that contains the number of time the event must be caught before redirecting the call flow or the message flow.

Grammar

Type

Form item

Available from

Any node in a speech application in which a spoken or DTMF response from a caller is possible.

Any node in a message application in which an SMS or email message from a customer is possible.

- **AppRoot** node (see AppRoot node on page 132)
- Announce node on page 634
- Prompt and Collect node on page 659
- **Collect** node
- Blind Transfer node on page 646
- Bridged Transfer node on page 647
- Record node on page 660
- Form node on page 655

Purpose

An integral part of a self-service speech application is the ability to collect responses from callers and use them to direct the call flow. These caller responses can be in the form of DTMF key presses or spoken commands.

Similarly, depending on the channel type of a message application, the message application collects SMS or email messages from customers and uses the inbound messages to direct the message flow.

Grammars are used at those points in a speech application and message application where you anticipate a spoken input, DTMF key press, SMS message, or email message from a customer. Grammars define what words, phrases, key presses, and text the system can recognize and use as input. **Grammar** items make it possible to select from among all the grammars defined for an application, to use at specific places in the application.

You can use variables for grammar, so that grammar can vary dynamically. In speech applications, if you use variables for grammar, you cannot use slots.

You can use the **Language** property of the **Grammar** item to create multilingual speech applications or message applications.

Note:

In the **AppRoot** node of a speech application, a grammar can be used only in conjunction with a **Link** item. This grammar is active on a global basis. That is, the system responds to any utterance that matches this grammar, no matter where the caller is in the call flow. For more information, see Link on page 762 and AppRoot node on page 132.

For more information about grammar, see Grammars on page 242.

Behavior

Grammar items cannot be used independently in nodes. Grammar items must always be used as sub-items of:

- Bridged Transfer on page 684 items
- Input on page 741 items
- Link on page 762 items
- Record on page 810 items

In short, you can use grammars only in places where it makes sense to collect and use spoken, DTMF, or message responses from customers. You must create the grammar file itself before you can use it in a node.

When a call reaches a node in which there is an active grammar, the system monitors input from the customer. Whenever a caller utterance, DTMF key press, or text in an inbound message matches an item in the grammar, the system processes the input according to how the node has been set to respond to that input.

If you copy a grammar variable from one speech application or message application to another, the system copies the variable, but does not copy the source to which the variable references.

For more information about creating and using grammar, see Grammars on page 242.

Properties

- **Name Variable**: Select the grammar variable. If you select a complex variable in this field, then you must select a grammar variable field in the **Name Variable Field** field. This field is unavailable if you select a value in the **Name Constant** field.
- **Name Variable Field**: If you select a complex variable in the **Name Variable** field, then select a grammar variable field in the **Name Variable Field** field. This field is unavailable if you select a simple variable in the **Name Variable** field.

Note:

In a speech application, if you use a variable instead of a constant, grammar slot values are unavailable at run time.

- **Name Constant**: Select the grammar you want to use.

  Note:

  You can select only one grammar per **Grammar** item. If you want to use multiple grammars at any point, you must use one **Grammar** item for each grammar you want to use. For example, if you want to offer callers the option to respond either with a key press of **1** or by saying the word “one,” you can use two grammars: one to recognize the DTMF input and the other to recognize the spoken response.

- **Weight**: (Default: 1.0) Optional. Enter in this field any floating point number of the format *n*, *n*., *.n*, or *n.n*, where *n* represents any length sequence of digits. A weight of 1.0 is considered neutral. That is, the grammar with a weight of 1.0 does not bias the recognition one way or the other. A weight of greater than 1.0 biases the grammar in a positive way. A weight of less than 1.0 biases the grammar in a negative way.

  The exact application of weights depends on the ASR engine or text processing system being used. Therefore, a weight that works well on a particular platform can produce different results on other platforms.

  Weights have no effect on DTMF grammars.

- **Language**: The language in which you anticipate a response from a customer.

  Use the **Language** property to interpret a spoken input, DTMF key press, or inbound SMS or email message that is received in any of the languages supported by Orchestration Designer. You can use the **Language** property to build a multilingual speech application or message application.

  Note:

  The default language is the starting language of an Orchestration Designer project.

  To use a language other than the starting language of an Orchestration Designer project, you must add the project language that you want to use in the Orchestration Designer project, and then translate the grammar files to that language. For more information, see Project languages on page 277 and Adding a project language on page 278.

  For example, you set the starting project language as English in a speech application. In the **Prompt and Collect** node of the speech application, you anticipate a spoken response in French from a caller. In the **Grammar** item, select the grammar file that you want to use to interpret the caller response, and set the **Language** property to French. The system automatically uses the French translated version of the grammar file to interpret the response from the caller.

If

Type

Condition item

Available from

Any level tab in the Prompt File Editor

Data node on page 653 in the Call Flow Editor

Purpose

The Condition on page 699 item makes it possible to select and play different prompt or data flow segments based on the condition being tested. It works in conjunction with **If** and Else on page 722 items to test conditions and return a result. The **If** item makes it possible to compare the value of a selected variable, the **Left Variable** or **Left Variable Field** with:

- A condition
- A constant
- The value of another variable or variable field

Then, based on the result of the comparison, the system responds according to what is assigned to the **If** item, or continues on.

You can create the following types of conditions:

- **Single condition**. In the Properties view of the **If** or Else If on page 723 item, select an operator, and set the required properties.
- **Compound condition**. Ensure that the **Operator** property in the Properties view of the **If** and **Else If** item is blank, and then use AND on page 675, OR on page 790, or NOT on page 772 to connect multiple expressions to form a compound condition. You can nest a compound condition in another compound condition to create a more complex condition.

For more information, see Creating single and compound conditions on page 608.

Behavior

The behavior of this item depends largely on which conditional **Operator** is selected. In general, when the value of the **Left Variable** is compared and satisfies the condition, then the application plays any prompt segments assigned to the **If** item. For more information about conditional operators and how to use them, see Appendix F: Conditional operators.

You can use multiple **If** items with a single **Condition** item. If you do so, then the application tests for each **If** item in turn, in the order the **If** items appear under the **Condition** item. This process continues until the conditions for one **If** item are met, or there are no more conditions to test for, that is, there are no more **If** items to test.

Properties

- **Left Variable**: Select the variable you want to use to make the comparison.

  If the variable you select is a simple variable, this is the variable to use for the comparison. If the variable you select is a complex variable, you must select a variable field from the **Left Variable Field** list to use for the comparison.

  Note:

  For more information about simple and complex variables, see Variables on page 226.
- **Left Variable Field**: If the variable you selected in the **Left Variable** field is a complex variable, select the field that you want to use to make the comparison. If the variable you selected in the **Left Variable** field is a simple variable, this field is inactive and no choices are available.

- **Compare**: Select the conditional operator you want to use to make the comparison. For more information about conditional operators and how to use them, see Conditional operators on page 602.
- **Value**: Use this field when you want to compare the value of the **Left Variable** or **Left Variable Field** with a literal, or constant, value. Enter in this field the literal value you want to compare the **Left Variable** or **Left Variable Field** with.

  For example, if you want to compare the account balance to see if it has a minimum value of at least $2000, and if it has, you want to play this prompt instead of another. To do this, you enter $2000 in this field and the appropriate conditional operator in the **Compare** field.

  If you use the **Value** field, you cannot use the **Right Variable** field.
- **Right Variable**: Select the variable you want to compare with the Left Variable or Left Variable Field.

  If the variable you select is a simple variable, this is the variable to use for the comparison. If the variable you select is a complex variable, you must select a variable field from the **Right Variable Field** list to use for the comparison.

  Note:

  For more information about simple and complex variables, see Variables on page 226.

  If you use the **Right Variable** field, you cannot use the **Value** field.
- **Right Variable Field**: If the variable you selected in the **Right Variable** field is a complex variable, select the field that you want to use to make the comparison. If the variable you selected in the **Right Variable** field is a simple variable, this field is inactive and no choices are available.

Input

Type

Form item

Available from

- Announce node on page 634
- Prompt and Collect node on page 659
- Collect node on page 651
- Form node on page 655

Note:

The **Input** item cannot be used in the same node as the Blind Transfer on page 678 item, the Bridged Transfer on page 684 item, or the Record on page 810 item. So, although the **Input** item appears on palettes for other nodes, such as the Blind Transfer node, the Bridged Transfer node, and the Record node, it is not actually available for use in those nodes without destroying their intended functionality. It is intended primarily for use in the Prompt and Collect node or the Collect node.

Purpose

In a speech application, the **Input** item is used to collect and process responses from callers. The responses can be in the form either of DTMF key presses or of spoken responses. In a message application, the **Input** item collects the response from the recipient of the email or sms in the form of sms files, email files, and attribute present in the message such as subject, email body, and sms content. The collected input is then used in some way to direct the flow of the call or the flow of message. The collected input can also be used as data that can be used by the application in a variety of ways.

Note:

The **Input** item in Orchestration Designer corresponds with the *field* within a *form* in the VoiceXML Version 2.0 W3C Recommendation. See the W3C VoiceXML 2.0 Recommendation.

Behavior

In Orchestration Designer, the primary means of collecting and using inputs from callers is through the Prompt and Collect node and collecting inputs from the recipient of the email or sms is through the Collect node. In a typical scenario, the system plays a prompt to prompt the user what type of response is expected, and then it waits for the user to respond. The **Input** item is the node element that is responsible for accepting and processing the user response. When it receives an input from a user , it submits that input to the ASR engine or the text browser and then waits for a return result from the ASR engine or the text browser.

The one essential sub-item of the **Input** item is the Grammar on page 737 item. Grammars determine what type of response the system is listening for and can recognize. DTMF grammars determine what touchtone key press or combination of key presses can be recognized and processed. Voice grammar and text grammar determine what words or phrases can be recognized and processed. You can use DTMF grammars, voice grammars, and text grammars with a single **Input** item. For more information about creating and using grammar, see Grammars on page 242.

In addition to the **Grammar** sub-item, the **Input** item can also take any of the standard event handlers. The Prompt and Collect node automatically includes the No Input on page 769 and **No Match on page 770** event handlers, but you can add any others you want.

When you drag an **Input** item into the call flow, Orchestration Designer automatically creates a complex variable with the same name.

Note:

Because the Prompt and Collect node and the Collect node, by design, includes an **Input** item, when you drag any of these nodes in a call flow or a message flow, Orchestration Designer automatically creates a complex variable with the same name as the **Input** item.

This complex variable contains the following fields, which are populated upon completion of the node:

- **confidence**: A number between 0.0 and 1.0 that expresses the degree of confidence the system has that it has correctly recognized the utterance. The number 0.0 indicates minimum confidence, and the number 1.0 indicates maximum confidence. In speech applications, the specific interpretation of this number depends on the ASR platform. For message applications, confidence is always 1.0.
- **utterance**: A text (raw string) summary of what the caller said or keyed in, as recognized by the ASR engine. In the case of a DTMF response, this is the digit sequence of keys that the

caller pressed. For message applications, utterance is the actual words or text in the message.

- **inputmode**: The mode in which the input was provided. For speech applications, the inputmode can either be voice or DTMF. For message application, inputmode is always text.
- **interpretation**: The contents of the interpretation tag, as provided by the ASR engine or the text browser. In other words, this is the result of how the ASR engine or the text browser recognizes and maps the utterance. If the ASR engine or the text browser is unable to provide an interpretation, this field remains undefined.
- **value**: If the ASR engine or the text browser returns an interpretation, this field takes on the same value as the **interpretation** field. If not, this field takes on the value of the **utterance** field.

After the ASR engine or the text browser returns a recognition result and populates these Input variable fields, you can use the outcome to further direct the call or message flow.

Properties

- **Name**: Enter the name you want to assign to the **Input** item and its corresponding variable.

  If you change the name of this item in the Avaya Properties view, Orchestration Designer automatically changes the name of the associated variable as well. If the **Bind name to node** attribute is set, then renaming the parent node will rename this field. This applies to all implicit variable items.

  Note:

  This name must follow conventions for naming Java components as described in Conventions for naming Java components on page 59.

- **N-Best Constant**: Enter the maximum number of recognition results to allow in the case of multiple recognition results from the ASR engine or the text browser.

  The default value for this property is **1**. This field is unavailable if you specify an N-best variable in the **N-Best Variable** field.

- **N-Best Variable**: Select the variable that contains the N-best value. This field is unavailable if you specify the maximum number of recognition results in the **N-Best** field. If you select a complex variable in this field, then you must select a complex variable field in the **N-Best Variable Field** field.
- **N-Best Variable Field**: If you select a complex variable in the **N-Best Variable** field, then select a complex variable field in the **N-Best Variable Field** field. This field is unavailable if you select a simple variable in the **N-Best Variable** field.

  Note:

  If the variable value is less than or equal to 0, the system sets it to 1 at run time.

- **Json Result Variable**: If you assign a variable to this property, the Json data will be automatically stored in the variable. You get the response object which contains the payload data in Json format.
- **Json Result Variable Field**: If you select a complex variable in the **Json Result Variable** field, then select a complex variable field in the **Json Result Variable** field. This field is unavailable if you select a simple variable in the **Json Result Variable** field.

- **Modal**: Select your choice from the drop-down list. Options are:
  - **True**: The system ignores all other grammars that are active at this point, except the grammar associated with this **Input** item.

    Setting this property to **True** can be helpful when concerned about a false match that redirects the call flow to another part of the application when you do not want it to. In a case like this, you want to make sure that a caller response matches a grammar associated with the **Input** item, if at all possible.
  - **False** (default): The system recognizes all active grammars. If the system matches a caller response with an item in a grammar other than the grammars associated with this **Input** item, control passes to the item associated with the other grammar, and any data collected within the current node is lost.
- **Record Utterance**: (True/False). When true, additional fields are added to the variable that store the recording of the utterance, the size of the recording (in bytes) and the duration of the recording (in milliseconds).
- **Collect Mark Data**: (True/False). When true, additional fields are added to the variable that store the name of the last Mark on page 764 encountered by the VoiceXML or TextXML platform as well as the time elapsed since the mark was processed.
- **Bind Name to Node**: (True/False). When true, the name of the variable item is bound to the name of the node. When the parent node is renamed, the name of the variable item is renamed as well; keeping the name of the variable in sync with the node name.
- **Recognition Target**: Select a value to specify the content you want to the message application to process. The message application uses the value selected in the **Recognition Target** field as the content to use for recognition of an inbound SMS or email message

  Note:

  This property is available only for email and SMS messages.

  You can select from one of the following values:
  - **to**: Variable that contains the email address or phone number of the receiver to whom you want to send the email or SMS message.
  - **from**: Variable that contains the email address or phone number of the sender of the email or SMS message.
  - **subject**: Variable that contains the subject line that you want to use in the email message.

    Note:

    This option is available only for an email message.
  - **body-text**: Content of the message body without any supporting metadata such as information about the formatting, or tags to format text.
  - **body-original**: Content of the message body with or without any supporting metadata.

    For a plain text and rich text message, the **body-original** is same as **body-text**. For an HTML message, **body-original** contains the html tags and other metadata other than the actual message. This option is available only for an email message.

Input Parameter

Type

Application item

Available from

AppRoot node only (see About the AppRoot node on page 132)

Purpose

When using a modular approach to application design, it is often necessary to pass variable or constant values from one module to another. The **Input Parameter** item makes it possible for a speech, a message, or a web project used as a module node to accept such input from another speech, message, or web project. For more information about creating and using application modules, see Module nodes on page 143.

Behavior

Use the **Input Parameter** item when you want to define values to be passed in to a speech, a message, or a web project that you want to use as a module for other speech, message, or web project . To pass values in to that module, you must drag into the **AppRoot** node one **Input Parameter** item for each value that you want to pass to the module.

For each **Input Parameter** that you drag into the **AppRoot** node, Orchestration Designer automatically creates a simple variable with the same name. You can use these variables the same way as you would use any other variable. If a default value is not assigned to the **Input Parameter** item, Orchestration Designer generates an error. When you test the module project, the Avaya Application Simulator (AAS) displays these **Input Parameter** variables in the **Input Parameter** display pane of the **Call** tab or in the display pane of **Messages** tab. For more information about testing projects, see Testing applications by simulation on page 386.

When you are satisfied that your module works the way you want it to work, generate and save it. You must then deploy it as an Orchestration Designer module. For more information about this and the procedure to deploy it, see Deploying a speech, or a message flow, or a web project as an Orchestration Designer reusable module on page 431.

Later, when you use the speech or web project as a module in another project, Orchestration Designer automatically populates the module node with one **Module Input** item for each **Input Parameter** item in the module. For more information, see Module Input on page 766.

Properties

- **Name**: Enter the name you want to use to identify the value being passed in and its corresponding variable.

  Orchestration Designer assigns a default name automatically when you drag the **Input Parameter** item into the node, and you can accept the default name or rename it.

  Note:

  This name must follow conventions for naming Java components as described in Conventions for naming Java components on page 59.

- **Description**: (Optional) Enter in this field a brief description of what the input parameter value is used for or expected to contain. Although this is optional, Avaya recommends that

you write a good description. The reason is that it can be helpful as a reminder of what the input parameter is for, later on when you or others use the module in another speech project.

- **Default Value**: Enter in this field the default value you want the variable to have. If you do not enter a value in this field, Orchestration Designer generates an error when you save or generate the project.
- **Variable**: Select the variable whose value you want to pass on to the parent module. If the variable you select is a complex variable, you must also select a **Variable** Field.
- **Variable Field**: Select the variable field you want to use. The contents of this list depend on which variable you selected in the **Variable** property.
- **Pass in as Value or Expression**: By default, this property is set to **Value**. This means the value in the Constant property is passed as constant string. In terms of VXML, the `"value"` attribute of the Param element is used.

  If this property is set to **Expression** , the value in the Constant property is passed as an expression. In terms of VXML, the `"expr"` attribute of the Param element is used.

Invoke Workflow 

Type

IC connector item

Available from

Data node on page 653 only

Purpose

The **Invoke Workflow** item allows you to invoke and use a workflow on an Avaya Interaction Center (IC) system without exiting the speech application on the IVR system. With the **Invoke Workflow** item, you can pass variable values to the workflow on the IC system. The workflow on the IC system can process these variable values and then return the results to the speech application on the IVR system.

Behavior

Before you can use this or any other IC item, you must enable IC for Orchestration Designer. For more information about the ability of Orchestration Designer to interact with IC, and to enable IC in your Orchestration Designer applications, see About the IC connector on page 622.

When the speech application encounters the **Invoke Workflow** item, the application:

- Sends a command to the IC system to invoke the workflow identified in the **Workflow Name** property.

  For more information about this property, see the “Properties” section of this item.
- Passes the values of any variables selected in the input variables and input variable fields *in order* to the IC workflow.

Note:

These values are passed in the order listed in the Avaya Properties view means that you must know the order in which variable values are expected in the workflow.

- Waits for the IC workflow to finish and return any results.

  Note that the amount of time that the Orchestration Designer application waits for a return from the IC system is determined by the **Max Timeout in ms** property. See the description of this property in the “Properties” section that follows.

  The **Return Immediately** property can be set to true or false (default) to have the invoked call return without waiting for the workflow to finish. If this flag is set, you would not be able to enter output variables or a **max Timeout** value.

  Note:

  The **vox.tr1** call allows you to perform a send and forget when invoking a workflow. The IC Connector does not wait for the results of the running workflow and the application continues to execute. Enable this feature by setting the **Return Immediately** property in the IC Invoke Workflow data node to true.

- Receives the results passed back to the application by the IC workflow and assigns them to the designated output variables and output variable fields.

  Important:

  When passing variable values to and from the IC system, keep in mind that, whereas the IC system allows dots (.), pluses (+), stars (*), and marks(**!**)in variable names, Orchestration Designer, being Java-based, does not. So, for example, a variable name of **account.number** is valid in the IC system, but it violates Java naming conventions, which do not allow the dot in a name.

  Therefore, when creating variables to pass values to known variables on the IC system or to receive variable values from known variables on the IC system, you must use the following naming convention:

  Replace the dot (**.**) by the following: **__DOT__**

  Replace the plus (**+**) by the following: **__PLUS__**

  Replace the star (*****) by the following: **__STAR__**

  Replace the mark (**!**) by the following: **__MARK__**

  This mnemonic replacement consists of two underscore (**_**) characters, followed by the word **DOT**, **PLUS**, **STAR**, or **MARK** (all uppercase), as applicable, followed by two more underscore characters.

  At run time, the Orchestration Designer application translates this mnemonic replacement for the dot, whether for variable values being passed to the IC system or received from the IC system.

  Tip:

  There are two ways that variable values can be passed to and from the IC system. One is to use the **Invoke Workflow** item, and the other is to create custom variable fields in the **vdu** variable and assign values to them. The second method can be useful if you want to pass more than four variable values at a time. This is because the **Invoke**

Workflow item has a limit of four output and four input variable values. By creating and assigning custom variable fields in the **vdu** variable *before* the **Invoke Workflow** item is initiated, you can send additional variable values to the IC system for use with the **Invoke Workflow** item.

Properties

- **Workflow Name**: Enter the name of the workflow you want to invoke on the IC system.

  This name must match *exactly* the name of a workflow on the IC system.

- **Max Timeout in ms**: Enter the amount of time in milliseconds to allow the IC system to invoke the workflow, complete its process, and return any results.

  If the IC system takes longer than this amount of time to return a response, the IVR system throws a run-time error and responds appropriately.

  Tip:

  This setting requires you to know the normal amount of time it takes the IC system to complete the workflow you are invoking. If, for example, you are invoking a workflow that normally takes from 4 to 7 seconds to process a request, and you have set this timeout to 5000 milliseconds (5 seconds), there is a good chance that the IVR application will time out before the request can be processed. Avaya recommends that you set this field to approximately 5 seconds more than the normal processing time that the workflow requires. In this example, you should set this field to **12000**.

- **Input Variable1** through **4**: For each input variable, select the variable whose value you want to pass to the IC workflow. You must assign the variables in the order the IC workflow expects to receive them.

  If a selected variable is a complex variable, you must also select a corresponding **Input Variable Field** for that variable.

- **Input Variable Field1** through **4**: If a given **Input Variable** is a complex variable, select the corresponding variable field whose value you want to pass to the IC workflow.

- **Output Variable1** through **4**: For each output variable, select the variable to which you want the IC workflow to return the results of processing.

  Note:

  The IC system creates output values using *name-value* pairs. Orchestration Designer applications can retrieve those name-value pairs only if it knows the names. For this reason, the name of the variable or variable field to which you want to assign the return from the IC system must *exactly* match the name of one of the names in these name-value pairs.

  If a selected variable is a complex variable, you must also select a corresponding **Output Variable Field** for that variable.

- **Output Variable Field1** through **4**: If a given **Output Variable** is a complex variable, select the corresponding variable field to which you want the IC workflow to return the results of processing.

- **Return Immediately**: Set this property to true or false (default) to have the invoked call return without waiting for the workflow to finish. If this flag is set, you would not be able to enter output variables or a **Max Timeout in ms** value.

Is Mobile

Type

Data Access item

Available from

Data Access palette of a **Data** node.

Purpose

Use the **Is Mobile** item to check whether the phone number is a mobile number.

Behavior

Is Mobile takes “to” and “from” phone numbers as input. If the “to” number is a mobile number, then **Is Mobile** returns “true”, else it returns “false”. In case of any errors, then **Is Mobile** returns “unknown”.

Properties

- **Result Variable**: Select the variable that contains the return value. The return value is one of the following:
  - true: If the number is a mobile number.
  - false: If the number is not a mobile number.
  - unknown: In case of any error.
- **Result Variable Field**: Select the field of the variable that contains the return value.
- **Result Message Variable**: Select the variable that contains the details of the result. The result message can be success or an error message.
- **Result Message Variable Field**: Select the field of the variable that contains the details of the result.
- **Provider Variable**: Select the variable that contains the entity (SMS short or long number) providing the lookup service results. You must configure the SMS short or long number in the EPM HTTP connections.
- **Provider Variable Field**: Select field of the variable containing the provider.
- **Carrier Variable**: Select the variable that contains the network to which the mobile phone is subscribed. For example, TMobile, ATT.
- **Carrier Variable Field**: Select field of the variable containing the carrier.
- **Carrier Id Variable**: Select the variable that contains the identification number of the carrier.
- **Carrier Id Variable Field**: Select field of the variable containing the Carrier Id.
- **Country Code Variable**: Select the variable that contains the country code that the phone is using.
- **Country Code Variable Field**: Select field of the variable containing the country code.
- **Raw JSON Result Variable**: Select the variable that contains the raw JSON data provided by the carrier.

- **Raw JSON Result Variable Field** Select field of the variable for raw JSON data.
- **From Number Variable**: Select the variable that contains the configured SMS short or long code. This is a mandatory field.
- **From Number Variable Field**: Select field of the variable that contains the configured SMS short or long code.
- **From Number Constant**: Select the constant value for SMS short or long code.
- **To Number Variable**: Select the variable that contains the number that you want to verify is a mobile number. This is a mandatory field.
- **To Number Variable Field**: Select field of the variable for **To Number Variable**.
- **To Number Constant**: Select the constant value of the number that you want to verify is a mobile number.

Join Condition

Type

Data item (Database Operation Condition)

Available From

Data node on page 653

Database Operation Editor Predicate Tab

Purpose

For setting up a condition that specifies a join between two tables. You can use the combination of the Simple and Join condition to create a complex search criteria.

Behavior

When adding a Join item, two Column item and a Comparison item are automatically created inside the Join item. You just need to specify the properties of the three nested items, and you have a join condition.

Properties

None

Language ID

Type

Notification connector item

Available from

The **Language ID** item is available in the Palette pane of a Data node in a message application only after you enable the **Text Processing (Language ID)** pluggable data connector.

For more information, see Enabling Orchestration Designer project pluggable data connectors on page 503.

Purpose

Use the **Language ID** item in a message application to accept a text string and the language hint and return the ISO-639-1 language code for the text.

Behavior

The **Language ID** item detects language in text as:

- If the text contains mixed language, the **Language ID** item returns the language code for the most dominant language in the text.
- If the dominant language is not different than the hinted language or exceeds the hinted language by more than 2%, the **Language ID** item returns the hinted language. If the dominant language is different than the hinted language, the **Language ID** item returns the code of the dominant language.
- If text is in an unsupported language, the **Language ID** item returns the code of one of the known languages.

Note:

The language identification code supports only the default languages. You cannot set the value of the language identification code in this PDC. The default language is set to **en**.

The following languages are recognized by character set:

Language	Code returned
Armenian	hy
Hebrew	he
Arabic	ar
Thaana	dv
Devanagari (Hindi)	hi
Bengali	bn
Gurmukhi	pa
Gujarati	gu
Oriya	or
Tamil	ta
Telugu	te
Kannada	kn
Malayalam	ml
Sinhala	si
Thai	th
Lao	lo
Tibetan	bo
Burmese&Myanmar	my
Georgian	ka

Table continues…

Language	Code returned
Ethiopic	am
Tagalog	tl
Khmer	km
Mongolian	mn
Sundanese	su

The following languages are recognized by character n-gram models:

Language	Code Returned
Danish	da
German	de
English	en
Spanish	es
Finnish	fi
French	fr
Croatian	hr
Indonesian	id
Icelandic	is
Italian	it
Japanese	ja
Korean	ko
Dutch	nl
Norwegian	no
Polish	pl
Portugeuse	pt
Russian	ru
Serbian	sr
Swedish	sv
Turkish	tr
Chinese (Simplified)	zh-cn
Chinese (Traditional)	zh-tw

Properties

- **Result Variable**: Click the variable that contains the result of the language recognition. If you select a complex variable in the **Result Variable** field, then you must select a complex variable field in the **Result Variable Field** field.
- **Result Variable Field**: If you select a complex variable in the **Result Variable** field, then in the **Result Variable Field** field, select the complex variable field that contains the result of language recognition.

- **Text Value**: Click the variable that contains the text to be analyzed for a particular language. If you select a complex variable in the **Text Value** field, then you must select a complex variable field in the **Text Value Field** field.
- **Text value Field**: If you select a complex variable in the **Text Value** field, then in the **Text Value Field** field, select the complex variable field that contains the text to be analyzed for a particular language.
- **Hint Variable**: Click the variable that contains the language code of the expected language of the input text. If you select a complex variable in the **Hint Variable** field, then you must select a complex variable field in the **Hint Variable Field** field.
- **Hint Variable Field**: If you select a complex variable in the **Hint Variable** field, then in the **Hint Variable Field** field, select the complex variable field that contains the language code of the expected language of the input text.
- **Hint Constant**: Type the language code of the expected language of the input text.

LaunchCCXMLCall

Type

Telephony item

Available from

The **LaunchCCXMLCall** item is available in the Palette pane of a **Data** node in a speech, message and web application only after you enable the **AAEP Outbound Call** pluggable data connector.

For more information, see Enabling Orchestration Designer project pluggable data connectors on page 503.

Purpose

Use the **LaunchCCXMLCall** item in a speech, message, or web application to initiate an outbound CCXML call from a speech, or a message, or a web application.

Behavior

When Orchestration Designer executes the **LaunchCCXMLCall** item, Orchestration Designer initiates the outbound CCXML call on Avaya Experience Portal Media Processing Platform (MPP) by launching the CCXML application that is configured on Experience Portal Manager (EPM). If the CCXML application contains the information to make a call, Experience Portal places the call from the Experience Portal system to the destination telephone number.

When the call connects, the CCXML application returns a successful launch code. If the call is unsuccessful, the CCXML application returns an appropriate error code.

Properties

- **result Variable**: Select the variable in which you want to store the message that the **LaunchCCXMLCall** item returns. If the CCXML call launch is successful, then the variable contains the value success. If the CCXML call launch is unsuccessful, then the variable contains the error message that the Web service returns.

- **result Variable Field**: If you select a complex variable in the **result Variable** field, then in the **result Variable Field** field, select the complex variable field in which you want to store the message that the **LaunchCCXMLCall** item returns.
- **applicationName**: Select the variable that contains the name of the CCXML application that you want to use to launch the outbound CCXML call.

  Note:

  The application name must be sane as the application name you add to Experience Portal through EPM.

  The **applicationName** field is unavailable if you specify a constant in the **applicationName Constant** field.
- **applicationName Field**: If you select a complex variable in the **applicationName** field, then in the **applicationName Field** field, select the complex variable field that contains the name of the CCXML application that you want to use to launch the outbound CCXML call.

  The **applicationName Field** field is unavailable if you select a simple variable in the **applicationName** field.
- **applicationName Constant**: Type the name of the CCXML application that you want to use to launch the outbound CCXML call.

  Note:

  The application name must be sane as the application name you add to Experience Portal through EPM.

  The **applicationName Constant** field is unavailable if you specify a variable in the **applicationName** field.

  Note:

  You must specify either a constant or a variable application name.
- **applicationURL**: Select the variable that contains the parameters that you want to append to the URL that is specified on EPM for the CCXML application.

  You can append the parameters to the URL of the CCXML application to invoke the CCXML application with different arguments as needed.

  The **applicationURL** field is unavailable if you specify a constant in the **applicationURL Constant** field.
- **applicationURL Field**: If you select a complex variable in the **applicationURL** field, then in the **applicationURL Field** field, select the complex variable field that contains the parameters that you want to append to the URL that is specified on EPM for the CCXML application.

  The **applicationURL Field** field is unavailable if you select a simple variable in the **applicationURL** field.
- **applicationURL Constant**: Type the parameters that you want to append to the URL that is specified on EPM for the CCXML application.

  You can append the parameters to the URL of the CCXML application to invoke the CCXML application with different arguments as needed. You must first append the ? character at the

end of the application URL. You can then append the required parameters after the ? character in the application URL. For example, *http:/www.avaya.com/avaya?query_string*.

The **applicationURL Constant** field is unavailable if you specify a variable in the **applicationURL** field.

- **toURI**: Select the variable that contains the telephone number or destination that provides a hint to the Application Interface Web service about the resources that the CCXML application requires.

  The options are:

  - Blank (no input): Indicates that the CCXML application does not require any resources.
  - tel: Use tel for an H.323 or SIP connection, or a mix of both H.323 and SIP connection.
  - sip: Use sip for a standard SIP connection.
  - sips: Use sips for a secure SIP connection.

  The **toURI** field is unavailable if you specify a constant in the **toURI Constant** field.

- **toURI Field**: If you select a complex variable in the **toURI** field, then in the **toURI Field** field, select the complex variable field that contains the telephone number or destination that provides a hint to the Application Interface Web service about the resources that the CCXML application requires.

  The **toURI Field** field is unavailable if you select a simple variable in the **toURI** field.

- **toURI Constant**: Type the telephone number or destination that provides a hint to the Application Interface Web service about the resources that the CCXML application requires.

  The options are:

  - Blank (no input): Indicates that the CCXML application does not require any resources.
  - tel: Use tel for an H.323 or SIP connection, or a mix of both H.323 and SIP connection.
  - sip: Use sip for a standard SIP connection.
  - sips: Use sips for a secure SIP connection

  Note:

  You must specify either a constant or a variable telephone number or destination.

  The **toURI Constant** field is unavailable if you specify a variable in the **toURI** field.

- **LaunchTimeout**: Select the variable that contains the maximum amount of time, in seconds, that the system must wait for the CCXML application to start. After the timeout is reached, the system logs an error message.

  The **LaunchTimeout** field is unavailable if you specify a constant in the **LaunchTimeout Constant** field.

- **LaunchTimeout Field**: If you select a complex variable in the **LaunchTimeout** field, then in the **LaunchTimeout Field** field, select the complex variable field that contains the maximum amount of time, in seconds, that the system must wait for the CCXML application to start. After the timeout is reached, the system logs an error message.

  The **LaunchTimeout Field** field is unavailable if you select a simple variable in the **LaunchTimeout** field.

- **LaunchTimeout Constant**: Type the maximum amount of time, in seconds, that the system must wait for the CCXML application to start. After the timeout is reached, the system logs an error message.

  The **LaunchTimeout Constant** field is unavailable if you specify a variable in the **LaunchTimeout** field.

- **parameters**: Select the variable that contains one or more parameter name and value pairs that you want to pass to the CCXML application when the system invokes the CCXML application.

  Each pair must be in the parametername=value format. Use a semi-colon (;) to separate multiple pairs. For information about the list of parameter names, see the *Administering Avaya Experience Portal* .

  Note:

  When the Web service passes the parameter to the CCXML application, Experience Portal appends the session.values.avaya.ParameterMap namespace to the parameter. Hence, you must reference the variable in the application as session.values.avaya.ParameterMap.<*parametername*>.

  For example, if you specify the UserCounter=0 parameter in the Web service, you must reference the UserCounter=0 parameter as session.values.avaya.ParameterMap.UserCounter in the application.

  The **parameters** field is unavailable if you specify a constant in the **parameters Constant** field.

- **parameters Field**: If you select a complex variable in the **parameters** field, then in the **parameters Field** field, select the complex variable field that contains one or more parameter name and value pairs that you want to pass to the CCXML application when the system invokes the CCXML application.

  The **parameters Field** field is unavailable if you select a simple variable in the **parameters** field.

- **parameters Constant**: Type one or more parameters in name and value pairs that you want to pass to the CCXML application when the system invokes the CCXML application.

  Each pair must be in the parametername=value format. Use a semi-colon (;) to separate multiple pairs. For information about the list of parameter names, see the *Administering Avaya Experience Portal* .

  When the Web service passes the parameter to the CCXML application, Experience Portal appends the session.values.avaya.ParameterMap namespace to the parameter. Hence, you must reference the variable in the application as session.values.avaya.ParameterMap.<parametername>.

  For example, if you specify the UserCounter=0 parameter in the Web service, you must reference the UserCounter=0 parameter as session.values.avaya.ParameterMap.UserCounter in the application

  The **parameters Constant** field is unavailable if you specify a variable in the **parameters** field.

- **UUI**: Select the variable that contains the user-to-user information that you want to pass to the platform telephony layer that is included in the outbound CCXML call.

The platform telephony layer passes the UUI information to the destination that receives the call.

The **UUI** field is unavailable if you specify a constant in the **UUI Constant** field.

- **UUI Field**: If you select a complex variable in the **UUI** field, then in the **UUI Field** field, select the complex variable field that contains the user-to-user information that you want to pass to the platform telephony layer that is included in the outbound CCXML call.

  The **UUI Field** field is unavailable if you select a simple variable in the **UUI** field.

- **UUI Constant**: Type the user-to-user information that you want to pass to the platform telephony layer that is included in the outbound CCXML call.

  The platform telephony layer passes the UUI information to the destination that receives the call

  The **UUI Constant** field is unavailable if you specify a variable in the **UUI** field.

- **Zone**: Select the variable which contains the zone ID to which the CCXML application is assigned as configured on EPM.

  Each zone represents the resources such as Media Server, ASR, TTS, and auxiliary EPMs that are assigned to that zone. For more information about zones, see *Administering Avaya Experience Portal.*

  The **Zone** field is unavailable if you specify a constant in the **Zone Constant** field.

- **Zone Field**: If you select a complex variable in the **Zone** field, then in the **Zone Field** field, select the complex variable field that contains the zone ID to which the VoiceXML application is assigned as configured on EPM.

  The **Zone Field** field is unavailable if you select a simple variable in the **Zone** field.

- **Zone Constant**: Type the zone ID to which the VoiceXML application is assigned as configured on EPM.

  Each zone represents the resources such as Media Server, ASR, TTS, and auxiliary EPMs that are assigned to that zone. For more information about zones, see *Administering Avaya Experience Portal.*

  The **Zone Constant** field is unavailable if you specify a variable in the **Zone** field.

Limitation

In simulation, since there is only one input (microphone) and one output (speakers) for multiple call appearances, dialogs must be “serialized” to use the resources. Once a dialog starts a form, it must complete before the next form with a call appearance may use the resources. This generally provides a reasonable simulation of an application, however, in a CCXML application you have multiple call appearances and if one has input, a subsequent call appearance playing an informational message will be blocked until the input is completed and exists. This can occur in the ConferencePredefined application when a conference attendee hangs up. Since the incoming call has a dialog running, listening for an add or drop command, the message “XXX left the conference” will not be heard until the input on the incoming call completes. This does not happen on the platform.

LaunchVXMLCall

Type

Telephony item

Available from

The **LaunchVXMLCall** item is available in the Palette pane of a **Data** node in a speech, message, and web application only after you enable the **AAEP Outbound Call** pluggable data connector.

For more information, see Enabling Orchestration Designer project pluggable data connectors on page 503.

Purpose

Use the **LaunchVXMLCall** item to launch an outbound VoiceXML call from a speech, or a message, or a web application.

For example, a message application is unable to interpret an inbound message because the required grammars are not configured in the message application. For such an event, you can use the **LaunchVXMLCall** item in the message application to make an automated outbound call to the customer.

Behavior

When Orchestration Designer executes the **LaunchVXMLCall** item, Orchestration Designer initiates the outbound VoiceXML call on the Avaya Experience Portal Media Processing Platform (MPP) by launching the VoiceXML application that is configured on Experience Portal Manager (EPM).

The actual launch of the VoiceXML application is handled by the default CCXML application that resides on Experience Portal. The default CCXML application also returns the success and failure events to the Application Interface web service.

When Orchestration Designer invokes the VoiceXML application on MPP, Orchestration Designer generates the VoiceXML page. Experience Portal launches the outbound call, and then places the call from the Experience Portal system to the destination telephone number that you specify in the **toURI** field of the **LaunchVXMLCall** item.

When the call connects, the VoiceXML dialog starts without error. The VoiceXML application returns a successful launch code. If the call is unsuccessful, the VoiceXML application returns an appropriate error code.

Properties

- **result Variable**: Select the variable in which you want to store the message that the **LaunchVXMLCall** item returns. If the VXML call launch is successful, then the variable contains the value **success**. If the VXML call launch is unsuccessful, then the variable contains the error message that the Web service returns.
- **result Variable Field**: If you select a complex variable in the **Result Variable** field, then in the **Result Variable Field** field, select the complex variable field in which you want to store the message that the **LaunchVXMLCall** item returns.
- **applicationName**: Select the variable that contains the name of the VoiceXML application that you want to run after the outbound call is connected.

Note:

The application name must be sane as the application name you add to Experience Portal through EPM.

The **applicationName** field is unavailable if you specify a constant in the **applicationName Constant** field.

- **applicationName Field**: If you select a complex variable in the **applicationName** field, then in the **applicationName Field** field, select the complex variable field that contains name of the VoiceXML application that you want to run after the outbound call is connected.
- **applicationName Constant**: Type the name of the VoiceXML application that you want to run after the outbound call is connected.

  The **applicationName Constant** field is unavailable if you select a variable in the **applicationName** field.

  Note:

  The application name must be sane as the application name you add to Experience Portal through EPM.

  You must specify either a constant or a variable application name.
- **toURI**: Select the variable that contains the telephone number or destination that the VoiceXML application must call.

  You can prefix the value in the **toURI** field with one of the following strings:

  - tel: Use tel for an H.323 or SIP connection, or a mix of both H.323 and SIP connection.
  - sip: Use sip for a standard SIP connection.
  - sips: Use sips for a secure SIP connection.

  The **toURI** field is unavailable if you specify a constant in the **toURI Constant** field.
- **toURI Field**: If you select a complex variable in the **toURI** field, then in the **toURI Field** field, select the complex variable field that contains the telephone number or destination that the VoiceXML application must call.

  The **toURI Field** field is unavailable if you select a simple variable in the **toURI** field.
- **toURI Constant**: Type the telephone number or destination that the VoiceXML application must call.

  You can prefix the value in the **toURI Constant** property with one of the following strings:

  - tel: Use tel for an H.323 or SIP connection, or a mix of both H.323 and SIP connection.
  - sip: Use sip for a standard SIP connection.
  - sips: Use sips for a secure SIP connection.

  The **toURIConstant** field is unavailable if you specify a variable in the **toURI** field.

  Note:

  You must specify either a constant or a variable telephone number or destination that the VoiceXML application must call.

- **applicationURL**: Select the variable that contains the parameters that you want to append to the URL that is specified on EPM for the VoiceXML application.

  You can append the parameters to the URL of the VoiceXML application to invoke the VoiceXML application with different arguments as needed.

  You must first append the ? character at the end of the application URL. You can then append the required parameters after the ? character in the application URL.

  For example, http:/www.avaya.com/avaya?query_string

  The **applicationURL** field is unavailable if you specify a constant in the **applicationURL Constant** field.
- **applicationURL Field**: If you select a complex variable in the **applicationURL** field, then in the **applicationURL Field** field, select the complex variable field that contains the parameters that you want to append to the URL that is specified on EPM for the VoiceXML application.

  The **applicationURL Field** field is unavailable if you select a simple variable in the **applicationURL** field.
- **applicationURL Constant**: Type the parameters that you want to append to the URL that is specified on EPM for the VoiceXML application.

  You can append the parameters to the URL of the CCXML application to invoke the CCXML application with different arguments as needed. You must first append the ? character at the end of the application URL. You can then append the required parameters after the ? character in the application URL. For example, *http:/www.avaya.com/avaya?query_string*.

  The **applicationURL Constant** field is unavailable if you specify a variable in the **applicationURL** field.
- **fromURI**: Select the variable that contains the telephone number or destination from where the call originates.

  EPM passes this information with the outbound VoiceXML call to the destination that receives the call.

  The **fromURI** field is unavailable if you specify a constant in the **fromURI Constant** field.
- **fromURI Field**: If you select a complex variable in the **fromURI** field, then in the **fromURI Field** field, select the complex variable field that contains the telephone number or destination from where the call originates.

  The **fromURI Field** field is unavailable if you select a simple variable in the **fromURI** field.
- **fromURI Constant**: Type the telephone number or destination from where the call originates.

  EPM passes this information with the outbound VoiceXML call to the destination that receives the call.

  The **fromURI Constant** field is unavailable if you specify a variable in the **fromURI** field.
- **ConnectTimeoutSecs**: Select the variable that contains the maximum amount of time, in seconds, that the system must wait for the outbound call to be connected.

  Valid values are 0 to 2,147,483.

  If you do not set the timeout value or if you set the timeout value to 0 (zero), then Experience Portal uses the default value of 120 seconds.

The **ConnectTimeoutSecs** field is unavailable if you specify a constant in the **ConnectTimeoutSecs Constant** field.

- **ConnectTimeoutSecs Field**: If you select a complex variable in the **ConnectTimeoutSecs** field, then in the **ConnectTimeoutSecs Field** field, select the complex variable field that contains the maximum amount of time, in seconds, that the system must wait for the outbound call to be connected.

  The **ConnectTimeoutSecs Field** field is unavailable if you select a simple variable in the **ConnectTimeoutSecs** field.

- **ConnectTimeoutSecs Constant**: Type the maximum amount of time, in seconds, that the system must wait for the outbound call to be connected. The valid values are 0 to 2, 147, 483. If you do not set the timeout value, or if you set the timeout value to 0, then Experience Portal uses the default value of 120 seconds.

  The **ConnectTimeoutSecs Constant** field is unavailable if you specify a variable in the **ConnectTimeoutSecs** field.

- **parameters**: Select the variable that contains one or more parameter name and value pairs that you want to pass to the VoiceXML application when the system invokes the VoiceXML application.

  Each pair must be in the parametername=value format. Use a semi-colon (;) to separate multiple pairs.

  For information about the list of parameter names, see the *Experience Portal Administration Guide*.

  The **parameters** field is unavailable if you specify a constant in the **parameters Constant** field.

- **parameters Field**: If you select a complex variable in the **parameters** field, then in the **parameters Field** field, select the complex variable field that contains one or more parameter name and value pairs that you want to pass to the VoiceXML application when the system invokes the VoiceXML application.

  The **parameters Field** field is unavailable if you select a simple variable in the **parameters** field.

- **parameters Constant**: Type one or more parameter name and value pairs that you want to pass to the VoiceXML application when the system invokes the VoiceXML application. Each pair must be in the parametername=value format. Use the semi-colon (;) to separate multiple pairs.

  The **parameters Constant** field is unavailable if you specify a variable in the parameters field.

- **UUI**: Select the variable that contains the user-to-user (UUI) information that you want to pass to the platform telephony layer that is included in the outbound VoiceXML call.

  The platform telephony layer passes the UUI information to the destination that receives the call.

  The **UUI** field is unavailable if you specify a constant in the **UUI Constant** field.

- **UUI Field**: If you select a complex variable in the **UUI** field, then in the **UUI Field** field, select the complex variable field that contains the UUI information that you want to pass to the platform telephony layer that is included in the outbound VoiceXML call.

The **UUI Field** field is unavailable if you select a simple variable in the **UUI** field.

- **UUI Constant**: Type the UUI information that you want to pass to the platform telephony layer that is included in the outbound VoiceXML call. The platform telephony layer passes the UUI information to the destination that receives the call.

  The **UUI Constant** field is unavailable if you specify a variable in the **UUI** field.

- **Zone**: Select the variable that contains the zone ID to which the VoiceXML application is assigned as configured on EPM.

  Each zone represents the resources such as Media Server, ASR, TTS, and auxiliary EPMs that are assigned to that zone. For more information about zones, see the *Administering Avaya Experience Portal*.

  The **Zone** field is unavailable if you specify a constant in the **Zone Constant** field.

- **Zone Field**: If you select a complex variable in the **Zone** field, then in the **Zone Field** field, select the complex variable field that contains the zone ID to which the VoiceXML application is assigned as configured on EPM.

  The **Zone Field** field is unavailable if you select a simple variable in the **Zone** field.

- **Zone Constant**: Type the zone ID to which the VoiceXML application is assigned as configured on EPM.

  Each zone represents the resources such as Media Server, ASR, TTS, and auxiliary EPMs that are assigned to that zone. For more information about zones, see *Administering Avaya Experience Portal.*

  The **Zone Constant** field is unavailable if you specify a variable in the **Zone** field.

Link

Type

Form item

Available from

- **AppRoot** node (see About the AppRoot node on page 132)
- Announce node on page 634
- Prompt and Collect node on page 659
- Blind Transfer node on page 646
- Bridged Transfer node on page 647
- Record node on page 660
- Menu node on page 656
- Form node on page 655

Purpose

The **Link** item is, essentially, a grammar-activated **Goto** item. The **Link** item makes it possible to redirect the call flow based on a particular spoken word or phrase or a combination of DTMF key presses. This functionality can be helpful, for example, when you want to allow callers to go to a

live operator at any point in the call flow. In this case, you can drag a **Link** item into the **AppRoot** node with grammars to recognize both the word “operator” and a key press of `0`. Either response by a caller would redirect the call flow to a call transfer node to dial the extension for the live attendant.

Behavior

The **Link** item has built-in DTMF recognition capability. This means that you do not have to use a DTMF grammar for the application to recognize a DTMF response. If you want the **Link** item to recognize spoken responses, you must attach an ASR **Grammar** item to the **Link** item.

You must use either one **Goto** item or one **Throw** item with the **Link** item. You can only use one or the other, not both.

If you use a **Goto** item, when the **Link** is activated, the application passes control to the node designated by the **Goto** item. For more information, see Goto on page 736.

If you use a **Throw** item, when the **Link** is activated, the application throws the event designated by the **Throw** item. For more information, see Throw on page 844.

Properties

- **Name**: Enter in this field the name you want to use to identify the **Link** item.

  For **Link** items that use a **Goto** item, this name appears as a connection point in the call flow.

  Note:

  This name must follow conventions for naming Java components as described in Conventions for naming Java components on page 59.

- **DTMF**: Enter in this field the DTMF key press or key press combination that you want to use to activate the **Link** item.

  Note:

  The advantage of using a DTMF grammar is that you can use multiple DTMF key press combinations to activate the **Link** item.

- **DTMF Variable/ DTMF Variable Field**: Use a variable for the DTMF key that activates the choice.

Loop Collection

Type

Conditional item

Available from

Data node

Purpose

You can use the **Loop Collection** item to build iteration logic to iterate over a collection of variable values in your application.

Note:

Alternatively, you can use the **Next In Collection** option in the **Operation** item. For more information, see Operation on page 777.

Behavior

The **Loop Collection** item creates a loop that iterates through a variable collection. After you set the **Variable** property of the **Loop Collection** item, you can add any operation that you want to execute for each item in the variable collection.

Properties

Variable: Select the variable that you want to use.

Mark

Adds a marker into a text/tag sequence. An SSML processor must either allow the VoiceXML interpreter to retrieve or must inform the interpreter when a <mark> is executed during audio output.

Type

SSML item

Available from

Any level tab in the prompt file editor. For more information, see Prompt file editor in a speech application on page 207.

Purpose

The **Mark** item is a Synthesized Speech Markup Language (SSML) element that is used to add markers in a prompt that plays TTS, TTS segment, or prompt. The markers can be used for detecting barge-in during prompt play back as well as tracking how much of a prompt the caller heard based on mark names and times.

Behavior

Marks are processed by the TTS engine as they are encountered during the speech synthesis. When a mark is encountered, the mark name is sent to the VoiceXML platform which tracks the name of the last mark encountered and the time since it was processed.

Media

Type

Media item

Available from

Parallel

Sequence

Purpose

Media is the main content used in IVVR play back.

Behavior

A Media item defines the media content like audio, video, image, text and configures playback properties for that content.

Properties

- **Media**: Name of media object from project.
- **Variable/Field**: Variable/Field pointing to the media object. This field is mutually exclusive with Media property.
- **Region**: Name of region added to the page to map media.
- **Begin**: Time offset in seconds to begin playing the media element.
- **End**: Time offset in seconds to end playing the media element.
- **Duration**: Duration in seconds to play media element.
- **Repeat Duration**: Total duration in seconds to repeat media element.
- **Repeat Count**: Total number of repeat media elements.
- **Clip Begin**: Beginning offset in seconds from the start of the media element.
- **Clip End**: Ending offset in seconds from the start of the media element.
- **Fill**: Extends the appearance of a media element to fill gaps in a presentation.

  For example, a prompt has video and audio elements playing simultaneously and the video is of shorter length than the audio element, if the fill attribute is set to freeze, the last video frame continues to appear until the audio prompt is complete.

Media Page 

Type

Media item

Available from

Root level of the prompt

Purpose

A Media Page is a container for the media elements. You must add a new Media Page to the prompt before adding any Media Items. You can also preview or play back the media page in a prompt during design time. For more details on previewing a media page, refer Previewing a Media Page on page 196.

Behavior

When dragged into a prompt, the media page creates 2 sections: head and body. The head is used to define elements that body can reference. Currently, these elements are limited to regions.

The media page contains the media content that is passed to the IVVR for play back. You can have separate additional media pages within the same prompt.

Properties

- **Name**: Name of the media page.
- **Background Color**: CSS2 system color.
- **Height**: Height of the root element.
- **Width**: Width of the root element.
- **Unit**: Unit of height and width of the Media page in pixel units (px) or percentage (%). The default is px.

Module Input

Type

Application item

Available from

Module nodes on page 143 only

Purpose

When using a modular approach to application design, it is often necessary to pass variable or constant values from one module to another. The **Module Input** item makes it possible for a speech or a message project to pass values into another speech or message project used as a module. For more information about creating and using application modules, see Module nodes on page 143.

Behavior

When you drag a module node into the call flow or the message flow, Orchestration Designer automatically populates it with one **Module Input** item for each input that the module can accept. If the module was created with Orchestration Designer, this means that there is one **Module Input** item for each **Input Parameter** item in the speech or the message project being used as the module. For more information, see Input Parameter on page 745.

If the module was not created with Orchestration Designer, you can specify inputs to the module when you import the module into Orchestration Designer. Then, when you use the module in an Orchestration Designer speech or message application, Orchestration Designer automatically populates it with the required **Module Input** items, much like it does with Orchestration Designer-created modules.

Note:

You can import any third-party VoiceXML subdialog in Orchestration Designer. However, Orchestration Designer can automatically recognize only Nuance Open Speech Dialog Module (OSDM) and Nuance Dialog Module (NDM) VoiceXML subdialogs. Orchestration Designer cannot automatically recognize the other third-party VoiceXML subdialogs.

Orchestration Designer supports OSDM versions: Address OSDM 2.0.3, Core OSDM 2.0.4, and Name OSDM 2.0.1, and NDM versions 5.0 and later.

To use these items in the speech project, you must assign a value to the **Module Input** item in the module node. This value can be either a variable value or a constant, but if you do not assign a value in the module node properties, no value is passed to the module node at run time.

Properties

- **Name**: (Display only) This name is the name of the item in the module that can accept input. If the module was created with Orchestration Designer, this is the name assigned to the **Input Parameter** item. You cannot change it here.
- **Description**: (Display only) This field is the description entered into this field when the module project was created.
- **Default Value**: (Display only) This field is the default value entered into this field when the module project was created.
- **Variable**: Select the variable whose value you want to pass in to the module. If the variable you select is a complex variable, you must also select a **Variable Field**.

  If you enter a **Constant** value, you cannot also use this field.
- **Variable Field**: If the variable you selected in the **Variable** field is a complex variable, you must select from this drop-down list the field whose value you want to pass in to the module. If the variable you selected in the **Variable** field is a simple variable, this field is blank.
- **Constant**: Enter in this field the exact string or value you want to pass in to the module.

  If you use this field, you cannot also use the **Variable** field.

  Important:

  If you use the **Constant** property, be careful that the form and format you enter match the form and format expected in the module that is receiving it.

Module Output

Type

Application item

Available from

Module nodes on page 143

Purpose

When using a modular approach to application design, it is often necessary to pass variable values from one module to another. The **Module Output** item makes it possible for a speech or a message project used as a module node to pass such data back to the parent speech or message project.

Behavior

When you drag a module node into the call flow or a message flow, Orchestration Designer automatically populates it with one **Module Output** item for each output value that the module passes to the parent application. If the module was created with Orchestration Designer, this means that there is one **Module Output** item for each **Output Parameter** item in the speech or

message project being used as the module. For more information, see Output Parameter on page 791.

If the module was not created with Orchestration Designer, you can specify outputs from the module when you import the module into Orchestration Designer. Then, when you use the module in an Orchestration Designer speech or message application, Orchestration Designer automatically populates it with the required **Module Output** items, much like it does with Orchestration Designer-created modules.

Note:

You can import any third-party VoiceXML subdialog in Orchestration Designer. However, Orchestration Designer can automatically recognize only Nuance Open Speech Dialog Module (OSDM) and Nuance Dialog Module (NDM) VoiceXML subdialogs. Orchestration Designer cannot automatically recognize the other third-party VoiceXML subdialogs.

Orchestration Designer supports OSDM versions: Address OSDM 2.0.3, Core OSDM 2.0.4, and Name OSDM 2.0.1, and NDM versions 5.0 and later.

At the same time, Orchestration Designer automatically creates:

- A complex variable with the same name as the module node.
- A field for that variable with the same name as the name of the **Module Output** item in the module node.

You can use the returns to this variable and variable field as you would use any other variable and variable field.

Properties

- **Name**: (Display only) This field is the name assigned to the **Module Output** item when the module was named. If the module was created with Orchestration Designer, this name is the same as the name of the **Output Parameter** item in the speech or message project being used as the module. You cannot change it here.
- **Description**: (Display only) This field is the description entered into this field when the module project was created.
- **Default Value**: (Display only) This field displays the value of the variable assigned to the **Variable** property of the **Output Parameter** item in the module that is passing the variable information to the parent application.

Next

Type

Form item

Available from

- Announce node on page 634
- Prompt and Collect node on page 659
- Blind Transfer node on page 646
- Bridged Transfer node on page 647

- Record node on page 660
- Form node on page 655
- Data node on page 653
- Tracking node on page 669

Purpose

The **Next** item determines or indicates the next node in the application flow.

Behavior

When executed, the **Next** item directs the flow to the node indicated in the **Next Form** field. Double click the **Next** node to open the specified node in the node editor.

You can have only one **Next** item in any one node. Most nodes that require the use of a **Next** item include it by default. This includes all the nodes listed in the previous section except the Form node.

No matter where the **Next** item appears in the subflow of a node, it is always executed last.

Properties

- **Next Form**: Select the node to which you want the flow to proceed when the current node is finished executing.

  Note:

  You can also populate this field by drawing a **Connection** line in the main workspace of the Call Flow Editor. For more information, see Connecting nodes in a workspace on page 147.

No Input

Type

Event handler

Available from

- **AppRoot** node (see AppRoot node on page 132
- Announce node on page 634
- Prompt and Collect node on page 659
- Collect node on page 651
- Blind Transfer node on page 646
- Bridged Transfer node on page 647
- Record node on page 660
- Menu node on page 656
- Form node on page 655

Purpose

When a user does not respond to a request for input within a specified period of time, the system throws a **No Input** event. The **No Input** event handler makes it possible to specify how the system responds when these No Input events are thrown.

Behavior

Within the scope defined for it, the **No Input** event handler waits for a No Input event to be thrown. When such an event is thrown, the **No Input** event handler responds according to the options you use with it. See the next section, “Properties.”

The **No Input** event handler responds to a No Input event anywhere within the scope of the handler. For example, if you drag the **No Input** event handler into the **AppRoot** node, it acts as a global event handler for No Input events. It can catch No Input events no matter where in the application the events are thrown. If you drag the **No Input** event handler into another node, however, it can catch a No Input event only if it is thrown while in that node. If you drag the **No Input** event handler underneath a node item, it can catch the No Input event only if it is thrown within that node item.

A **No Input** event handler at the node item level overrides a **No Input** event handler at the node level. Likewise, a **No Input** event handler at the node level overrides a **No Input** event handler at the **AppRoot**, or global, level.

Note:

This event handler is one of the default event handlers defined by the W3C VoiceXML 2.0 Recommendation and it is identified as a **No Input** event handler.

Properties

The **No Input** event handler has no inherent properties, but you must use one or more of the following options with it:

- **Prompt on page 797** item: Plays the designated prompt to the caller before the application continues. Use the Prompt item in an SMS channel or an email channel message application, to send an outbound SMS or email message respectively.
- **Goto on page 736** item: Directs the application to the designated node.
- **Throw on page 844** item: Throws an event that you choose. Select a user-defined event that you have created.

  By throwing an event within an event handler, you can transfer control to a different part of the application, for example, to the **AppRoot** node. For more information about throwing events and handling events, see Throw on page 844 or Events on page 314.

No Match

Type

Event handler

Available from

- **AppRoot** node (see About the AppRoot node on page 132)

- Announce node on page 634
- Prompt and Collect node on page 659
- Collect node on page 651
- Blind Transfer node on page 646
- Bridged Transfer node on page 647
- Record node on page 660
- Menu node on page 656
- Form node on page 655

Purpose

When a user does not respond to a request for input with a response that matches any that the system is set to recognize, the system throws a No Match event. The **No Match** event handler makes it possible to specify how the system responds when these No Match events are thrown.

Behavior

Within the scope defined for it, the **No Match** event handler waits for a No Match event to be thrown. When such an event is thrown, the **No Match** event handler responds according to the options you use with it. See the next section, “Properties.”

The **No Match** event handler responds to a No Match event anywhere within the scope of the handler. For example, if you drag the **No Match** event handler into the **AppRoot** node, it acts as a global event handler for No Match events. It can catch No Match events no matter where in the application the events are thrown. If you drag the **No Match** event handler into another node, however, it can catch a No Match event only if it is thrown while in that node. If you drag the **No Match** event handler underneath a node item, it can catch the No Match event only if it is thrown within that node item.

A **No Match** event handler at the node item level overrides a **No Match** event handler at the node level. Likewise, a **No Match** event handler at the node level overrides a **No Match** event handler at the **AppRoot**, or global, level.

Note:

This event handler is one of the default event handlers defined by the W3C VoiceXML 2.0 Recommendation. There it is identified as a *nomatch* event handler.

Properties

The **No Match** event handler has no inherent properties, but you must use one or more of the following options with it:

- Prompt on page 797 item: Plays the designated prompt to the caller before the application continues. Use the Prompt item in an SMS channel or an email channel message application, to send an outbound SMS or email message respectively.
- Goto on page 736 item: Directs the application to the designated node.
- Throw on page 844 item: Throws an event that you choose. Select a user-defined event that you have created.

  By throwing an event within an event handler, you can transfer control to a different part of the application, for example, to the **AppRoot** node. For more information about throwing events and handling events, see Throw on page 844 or Event.

NOT AND OR NOT

Type

Boolean operator

Available from

Data node

Any level tab in the Prompt File Editor

Purpose

You can use the **NOT** Boolean operator to evaluate the opposite of what its child expressions or compound conditions evaluate.

Behavior

This item creates a condition that evaluates to the opposite of what its child expressions or compound conditions evaluate. This item is used as a parent item and can be nested under If on page 739 or any other Boolean operator. It can have only one child expression which can be either a single expression or a compound condition.

For more information, see Creating single and compound conditions on page 608.

Properties

- **Boolean Operator**: Shows the name of the Boolean operator. If you want to select any other Boolean operator, select that operator from the list.

Object

Type

Application item

Available from

- Announce node on page 634
- Prompt and Collect node on page 659
- Blind Transfer node on page 646
- Bridged Transfer node on page 647
- Record node on page 660
- Form node on page 655

Purpose

The **Object** item makes it possible to extend the functionality of VoiceXML beyond its normal capabilities. For example, the VoiceXML standard does not support conference calling. However, if your system supports conference calling, you can write a platform-specific script or application that can be used to take advantage of your system's capabilities. You then use this script or application using the **Object** item.

Behavior

The **Object** item takes the values of the properties specified in the following section and uses them to implement and execute the object specified.

To pass data to the object from the application, use the **Object Input** item. For more information, see Object Input on page 773.

If the object returns data that you want to use in the application, use the **Object Output** item. For more information, see Object Output on page 774.

For more information about using objects in VoiceXML, see “object element” in the W3C VoiceXML 2.0 Recommendation.

Properties

Of the following properties for the Object item, only the Class ID property is required for all objects. The need to use other properties depends on what object you are calling.

- **Name**: Enter the name to identify the object in the call flow.
- **Class ID**: Enter the URI for the object you want to execute.

  This URI can be either an absolute or a relative URI. If it is a relative URI, it is interpreted relative to the **Codebase** property.
- **Data**: Enter the URI that specifies the location of the data for the object.

  This URI can be either an absolute or a relative URI. If it is a relative URI, it is interpreted relative to the **Codebase** property.
- **Type**: Enter the content type of the data specified in the **Data** property.
- **Archive**: Enter a space-separated list of URIs for archives containing resources relevant to the object.

  These resources can include the resources specified by the **Class ID** and **Data** properties.

  Any relative URIs are interpreted relative to the **Codebase** property.
- **Codebase**: Enter the base path to use when resolving relative URIs specified by the **Class ID**, **Data**, and **Archive** properties.

  The default for this property is the URI of the current document.
- **Codetype**: Enter the content type of the data the system expects when it downloads the object specified by the **Class ID** property.

  If you leave this field blank, the system defaults to the content type specified in the **Type** property.

Object Input 

Type

Application item

Available from

- Announce node on page 634

- Prompt and Collect node on page 659
- Blind Transfer node on page 646
- Bridged Transfer node on page 647
- Record node on page 660
- Form node on page 655

Purpose

When you use an Object on page 772 item to extend the functionality of VoiceXML, you need to pass variable values from the application to the object. The **Object Input** item makes it possible to pass these values to the object.

Behavior

The **Object Input** item takes the value assigned and passes it to the object when the object is implemented. For information about the values that can be passed, see the following section, “Properties”.

The **Object Input** item must be associated with an **Object** item. For more information about **Object** items, see Object on page 772.

Properties

- **Name**: Enter the name you want to identify the **Object Output** item in the node subflow.
- **Constant Value**: Enter the constant value you want to pass to the object.
- **Expression Value**: Enter the ECMAscript expression you want to pass to the object.

  Caution:

  Because the expected content for this property is an ECMAscript expression, Orchestration Designer applications pass the value of this property directly to the Avaya Voice Browser (AVB). Avaya recommends that you use this property with caution.
- **Variable Value**: Select the variable whose value you want pass to the object.

  If the selected variable is a complex variable, you must also select a **Variable Field**.
- **Variable Field Value**: If the variable selected in the **Variable** field is a complex variable, select the associated variable field whose value you want to pass to the object.

  Note:

  If the variable selected in the **Variable** field is a simple variable, this list is not populated.

Object Output

Type

Application item

Available from

- Announce node on page 634

- Prompt and Collect node on page 659
- Blind Transfer node on page 646
- Bridged Transfer node on page 647
- Record node on page 660
- Form node on page 655

Purpose

When you use an Object on page 772 item to extend the functionality of VoiceXML, you need to pass variable values back from the object to the application. The **Object Output** item makes it possible to pass these values back to the application.

Behavior

The **Object Output** item must be attached to an **Object** item. For more information about **Object** items, see Object on page 772.

When you attach an **Object Output** item to an **Object** item, Orchestration Designer automatically creates:

- A complex variable with the same name as the associated **Object** item.
- A field for that variable with the same name as the **Object Output** item.

Tip:

If you want to assign a default value to this variable field, do so in the Variable Editor. For more information, see Variables on page 226.

You can attach multiple **Object Output** items to a single **Object** item. Orchestration Designer creates a new variable field for each **Object Output** item you attach to the **Object** item.

Tip:

After you attach the **Object Output** item to the **Object**, the new variable and variable field do not appear in the Variable Editor until you save the project. After you save the project, you must close the Variable Editor, if it is open, and reopen it.

You can use these variables and variable fields the same as you would use any other variable and variable field.

Properties

- **Name**: Enter the name that identifies the **Object Output** item in the node subflow and its corresponding variable field.

Note:

The name you enter in this property field must match the name of the object variable being returned from the module.

On Disconnect

Type

Event handler

Available from

- **AppRoot** node (see AppRoot node on page 132
- Announce node on page 634
- Prompt and Collect node on page 659
- Blind Transfer node on page 646
- Bridged Transfer node on page 647
- Record node on page 660
- Menu node on page 656
- Form node on page 655

Purpose

When a caller hangs up or otherwise gets disconnected from the system before the application is done executing, the system throws an On Disconnect event. The **On Disconnect** event handler makes it possible to specify how the system responds when these On Disconnect events are thrown.

Behavior

Within the scope defined for it, the **On Disconnect** event handler waits for an On Disconnect event to be thrown. When such an event is thrown, the **On Disconnect** event handler responds according to the options you use with it. See the next section, “Properties.”

The **On Disconnect** event handler responds to an On Disconnect event anywhere within the scope of the handler. For example, if you drag the **On Disconnect** event handler into the **AppRoot** node, it acts as a global event handler for On Disconnect events. It can catch On Disconnect events no matter where in the application the events are thrown. If you drag the **On Disconnect** event handler into another node, however, it can catch an On Disconnect event only if it is thrown while in that node. If you drag the **On Disconnect** event handler underneath a node item, it can catch the On Disconnect event only if it is thrown within that node item.

An **On Disconnect** event handler at the node item level overrides an **On Disconnect** event handler at the node level. Likewise, an **On Disconnect** event handler at the node level overrides an **On Disconnect** event handler at the **AppRoot**, or global, level.

Note:

This event handler is one of the default event handlers defined by the W3C VoiceXML 2.0 Recommendation and is identified as a *connect.diconnect* event handler.

The **On Disconnect** event handler does not allow any child items to be added to it.

Properties

- **Next**: Directs the application to the designated node.
- **Submit**: (True/False). When true, the application variables are submitted to the application server. If **On Disconnect** is added on the AppRoot node, then the **Submit** property is not available.

  By throwing an event within an event handler, you can transfer control to a different part of the application, for example, to the **AppRoot** node. For more information about throwing events and handling events, see Throw on page 844 or Events on page 314.

Operation

Type

Data item

Available from

Data node on page 653 only

Purpose

The **Operation** item, labeled **Undefined Data Operation** in the subflow, makes it possible to manipulate the values of variables in a variety of ways.

Behavior

The **Operation** item takes the value of a variable or variable field and manipulates it according to the selection of the **Type** property. When you select the type, the name of the **Operation** item changes to match the type. The Avaya Properties view also changes to reflect the additional properties associated with that type.

The specific behavior of each defined type is described in the “Operation Item Property Types” table that follows.

Properties

The properties of the **Operation** item depend on which **Type** you assign to the item. Only one property is common to all variants of the **Operation** item: the **Type** property. What is selected for this property determines what other properties are available. The following table lists all the available types, along with the other properties associated with each type:

Table 8: Operation Item Property Types

Type	Associated properties	Comments/Remarks
Each of the following types performs an arithmetic operation on two number values and returns the result to the destination variable. **Add**: Adds the value of the left operand and the right operand. **Subtract**: Subtracts the value of the right operand from the left operand.	**Destination Variable (and Field)**	Select the variable to which you want to assign the return value of the operation. If this variable is a complex variable, you must also select a **Destination Variable Field**.
	Left Operand Variable (and Field)	Select the variable to use as the left operand. If this variable is a complex variable, you must also select a **Left Operand Variable Field**. If you use this property, you cannot use the **Left Operand Constant** property.
	Left Operand Constant	Enter in this field the integer value to use as the left operand. If you use this property, you cannot also use the **Left Operand Variable** property.

Table continues…

Type	Associated properties	Comments/Remarks
Multiply: Multiplies the value of the left operand and the right operand. **Divide**: Divides the value of the left operand by the right operand. **Modulus**: Divides the value of the left operand by the right operand and returns the value of the remainder, if any.	**Right Operand Variable (and Field)**	Select the variable to use as the right operand. If this variable is a complex variable, you must also select a **Right Operand Variable Field**. If you use this property, you cannot also use the **Right Operand Constant** property.
	Right Operand Constant	Enter in this field the integer value to use as the right operand. If you use this property, you cannot also use the **Right Operand Variable** property.
Append To Collection: Appends the value of a source variable to the end of a collection in the destination variable. For information about collections in variables, see Variables on page 226.	**Destination Variable**	Select the variable containing the collection to which you want to append the new entry.
	Source Variable	Select the variable containing the value you want to append to the collection.
Assign: Takes the value of the source and assigns it to the destination. If no source value is specified, that is, all three **Source..**. fields are left blank, the system assigns the value of an empty string to the destination variable.	**Destination Variable (and Field)**	The variable that the source value is assigned to. If the destination is a complex variable, a field must be selected.
	Source Variable (and Field)	Select the variable to assign to the destination. If this variable is a complex variable, a **Source Variable Field** must be selected whose value should be assigned to the destination. If this property is used, the **Source Constant** property cannot be used.
	Source Constant	Enter the constant value you want to assign to the destination variable. If this property is used, the **Source Variable** property cannot be used.
Concatenate: Concatenates to, or adds to the end of the value of, the destination variable or variable field. Usually, the variable or variable field value is of type String. Because this is a String-based operation, you can concatenate numbers,	**Destination Variable (and Field)**	Select the variable to which the source value is to be concatenated. If the destinationis a complex variable, you must also select a **Destination Variable Field**.
	Source Variable (and Field)	Select the variable (and field if **Source Variable** is a complex variable) whose value is to be concatenated to the destination. If the sourceis a complex variable, you must also select a **Source Variable Field**.

Table continues…

Type	Associated properties	Comments/Remarks
but they are then string-based numbers. For example, if you concatenate a **3** to a **4**, you end up with the value **34**.	**Source Constant**	Enter the constant value you want to concatenate to the destination variable. If this property is used, the **Source Variable** property cannot be used.
Decrement: Subtracts one (1) from the value of the destination variable or variable field. The destination variable must be of the type integer.	**Destination Variable (and Field)**	Select the variable whose value is to be decremented. If this variable is a complex variable, you must also select a **Destination Variable Field**.
Delete All From Collection: Removes all items from a collection variable. For information about collections in variables, see Variables on page 226.	**Destination Variable**	Select the collection variable. All entries in the collection will be cleared.
Delete From Collection: In a collection, deletes the entry that is at the current position of the collection pointer. For information about collections in variables, see Variables on page 226.	**Destination Variable**	Select the variable containing the collection from which you want to delete the entry.
Increment: Adds one (1) to the value of the destination variable or variable field. The destination variable must be of the type integer.	**Destination Variable (and Field)**	Select the variable whose value is to be incremented. If this variable is a complex variable, you must also select a **Destination Variable Field**.
Index Of: Returns the index of a substring from a larger string.	**Destination Variable (and Field)**	The variable (or field if variable is complex) that stores the index of the “String”. If the “String” value does not exist in the **Source Variable** (or field if complex), then this value will be -1. **Note:** index is 0-based—that is the first character is index 0.

Table continues…

Type	Associated properties	Comments/Remarks
	Source Variable (and Field)	The **source variable**—the value of this is searched to find the start index of “String” if it exists.
	String	The string to search the **source variable** for. For example, Index of “Fran” in “San Francisco” would return 4. Index of “Den” in “San Francisco” would return -1.
	String Variable (and Field)	The variable (or field if variable is complex) used to search another string in the **Source Variable** for an occurrence of either a user defined constant character or string, or for an occurrence of the value of the **String Variable**.
	Start From Constant **Start From Variable (and Field)**	This is used to skip over preceding characters in the string before searching for the first occurrence of “String”. You can either use a constant value, or supply a variable that contains a valid index to start from. This will throw a runtime exception if the “start from” value is invalid. For example: Index of “a” in “Hawaii” with start from “2” would return 3. Index of “a” in “Hawaii” with start from “10” would throw an exception because the start from index exceeds the length of the string.
Insert Into Collection: Inserts the value of a source variable into a collection in the destination variable at the current position of the collection pointer. For information about collections in variables, see Variables on page 226.	**Destination Variable**	Select the variable containing the collection into which you want to insert the new entry.
	Source Variable	Select the variable containing the value you want to insert into the collection.
Insert String: This function is similar to concatenate, but where concatenate appends one string to the end of another, Insert String will insert the contents of one string into another at a specified position. For example if Destination Variable has the value “car” and	**Destination Variable (and Field)**	The variable whose value will have the string contents of Source (variable or constant) inserted. If the **Destination Variable** is a complex variable, select the field whose value is to be incremented.
	Source Variable (and Field)	The variable whose value will be inserted into the destination variable’s value. If the source is a complex variable, a field must also be selected.
	Source Constant	The constant value (hardcoded) that will be inserted into the destination variable’s value.

Table continues…

Type	Associated properties	Comments/Remarks
Source Variable contains the value “race”, and Index Constant is 0, then the value of Destination Variable would result in “racecar”.	**Index Variable (and Field)**	the variable that contains the index (zero-based) where the source var/const will be inserted into the dest. var. This must be an integer. If the value is less than zero, it is automatically adjusted to be zero. If the value is larger than the length of the destination variable, then it is automatically adjusted to add the string to the end of the destination variable; effectively becoming a concatenation operation. If the index variable is complex, a field must also be selected.
	Index Constant	The constant value (hardcoded) that is used for the index.
Length: Returns the number of characters in a source variable string, and assigns that number to the destination variable.	**Destination Variable (and Field)**	Select the variable to which the source length is to be assigned. If this variable is a complex variable, a **Destination Variable Field** must also be selected (the field to which the source length is to be assigned).
	Source Variable (and Field)	Select the variable whose length you want to assign to the **destination variable**. If this variable is a complex variable, you must also select a **Source Variable Field** (the field whose length you want to assign to the destination variable).

Table continues…

Type	Associated properties	Comments/Remarks
Reset Collection: Sets the item marker to the beginning of the collection. After you set the item marker to the beginning of the collection, other variable operations can access the variable and the variable field, and use the value that is pointed by the item marker. If the collection is created for the first time, then in the data node editor, add the **Next in Collection** item after the **Reset Collection** item to set the item marker to the beginning of the collection. For information about collections in variables, see Variables on page 226.	**Destination Variable**	Select the variable whose collection you want to reset to the beginning.

Table continues…

Type	Associated properties	Comments/Remarks
Next in Collection: Returns the value of the next item in the collection. For information about collections in variables, see Variables on page 226. **Note:** Alternatively, you can use the **Loop Collection** item to build an iteration logic to iterate over a collection of variable values in your application. For more information, see Loop Collection on page 763.	**Destination Variable**	Select the variable that contains the collection. At run time, when the application reaches the end of the collection and there are no more items to return, the system would normally throw a run-time exception. To prevent this, drag a Condition item into the node subflow *before* this item, and: • Set the **Operator** property to **HasNoMore**. • For the **Left variable**, select the variable that contains the collection. • For the **Next Form** field, select the next node to go to when there are no more items left in the collection.
Parse: Uses the separator character to break a string into smaller strings. Returns the resulting smaller strings as a collection in the destination variable. For information about collections in variables, see Variables on page 226.	**Destination Variable**	Select the variable to which you want to assign the collection that results from the parsing action.
	Source Variable (and Field)	Select the variable that contains the string you want to parse. If this variable is a complex variable, you must also select a **Source Variable Field** (the variable that contains the string you want to parse).
	Separator Character	Enter the character you want to use as the trigger to break the string into smaller strings. For example, if your original string is “Parse this variable” and you use a space as the separator, this operation returns the three words “Parse”, “this”, and “variable”. It then returns the results as a collection with the three words as items in the collection.
	Include Empty Strings	This option allows you to include empty tokens. For example, if you use ‘|’ as a separator and your string is 1111|2222||3333, then, it returns 3 tokens (“1111”, “2222”, “3333”) if set to ‘false’ and returns 4 tokens (“1111”, “2222”, “”, “3333”) if set to ‘true’.

Table continues…

Type	Associated properties	Comments/Remarks
Previous In Collection: This is the inverse of Next in Collection. It moves the collection iterator to the item prior to the current item.	**Destination Variable (and Field)**	The variable that will contain the HEX encoded value of the **source variable**. If this variable is a complex variable, a **Destination Variable Field** must also be selected.
	Source Variable (and Field)	The **source variable** whose contents will be encoded where each character will be represented with its ASCII character code. If this variable is a complex variable, a **Source Variable Field** must also be selected.
Remove Character: Removes all occurrences of a designated character from the source variable and writes the new value of the variable to the destination.	**Destination Variable (and Field)**	Select the variable to which you want to write the new value. If the **Destination Variable** is a complex variable, select the field to which you want to write the new value.
	Source Variable (and Field)	Select the variable from which you want to remove all occurrences of the designated character. If the **Source Variable** is a complex variable, select the field from which you want to remove all occurrences of the designated character.
	Character	Enter in this field the character you want to remove from the source variable or variable field.
Set DateTime: Takes the value of the date (and optionally, time) variable and assigns it to the destination.	**Destination Variable (and Field)**	Allows users to assign the date and time variables to a destination variable. The destination variable then automatically converts the values contained in the date and time variables into a Java Date object, which is carried throughout the session.
	Date Variable	A complex variable that must contain the day, month, and year fields. You can create this variable and assign numeric values to its fields or you can use the built-in date variable for system date.
	Time Variable	A complex variable that must contain the hour, minute, and second fields. You can create this variable and assign numeric values to its fields or you can use the built-in time variable for system time. This property is optional.

Table continues…

Type	Associated properties	Comments/Remarks
Set Null: Sets the value of the variable to null. **Note:** Session variables cannot be null, so by default, they are initialized as empty strings (“”). In order to pass null values to database operations or Web services, “Set Null” will assign a special value (“__DD_NULL”) to the session variable which is interpreted as null by the runtime for Web services and databases. Likewise, if a database or web service operation returns a null value, then the runtime will set the value of the session variable to “__DD_NULL” to prevent Null Pointer Exceptions.	**Destination Variable (and Field)**	Select the variable to be set to Null. If this is a complex variable, a **Destination Variable Field** must also be selected. If the **Destination Variable** is a complex variable, select the field to be set to null. If the **Destination Variable** is a simple variable, this list is not populated.
Size: If the source variable is a collection, returns the size of the collection. That is, this type returns the number of items in the collection. If the source variable is not a collection, returns **1**. For information about collections in variables, see Variables on page 226.	**Destination Variable (and Field)**	Select the variable to receive the return value. If this is a complex variable, you must also select a **Destination Variable Field**. If the **Destination Variable** is a complex variable, select the field to receive the return value. If the **Destination Variable** is a simple variable, this list is not populated.
	Source Variable	Select the variable that contains the collection to be counted.

Table continues…

Type	Associated properties	Comments/Remarks
Sort: Takes the contents of a variable or variable field and sorts the collection in ascending or descending order. The variable or variable field must contain a collection for this operation to be meaningful. For information about collections in variables, see Variables on page 226.	**Destination Variable (and Field)**	Select the variable whose contents you want to sort. If the variable is a complex variable, you must select a **Destination Variable Field** as well (the field whose contents you want to sort).
	Sort Order	Select one of the following: • **ascending**: Arranges the designated items in ascending order. For example, if the operation is sorting a list of names, the names are arranged in alphabetic order, from A to Z. • **descending**: Arranges the designated items in descending order. For example, if the operation is sorting a list of numbers, the numbers are arranged in order from greatest to least.
String To ASCII Hex: Converts a string to its corresponding ASCII Hex character code. It is primarily intended as a utility method when sending data that must be hex-encoded through SIP headers such as when sending AAI data in a transfer. For example: the value “7654321” would result in “37363534333231” (7->37, 6->36). For more information, see Encoding string data in ASCII hex format on page 631. Refer to an ASCII character chart to see the full list of hex ASCII character codes. **Note:** Networks may handle SIP data differently; hex encoding is recommended, but may not work in all	**Destination Variable (and Field)**	Select the variable that will contain the HEX encoded value of the **source variable**. If the **Destination Variable** is a complex variable, a field must also be selected.

Table continues…

Type	Associated properties	Comments/Remarks
environments depending on the switch and other elements in the network.	**Source Variable (and Field)**	Select the variable whose contents will be encoded where each character will be represented with its ASCII character code. If the **Source Variable** is a complex variable, a field must also be selected.
ASCII Hex To String: Converts the ASCII hex-encoded data to string data. It is primarily intended as a utility method when receiving data that must be hex decoded through SIP headers such as when receiving AAI data in a transfer. For example, the value “37363534333231” (7-<37, 6->36) would result in “7654321”. For more information, see Decoding ASCII hex format data to string format data on page 632. Refer to an ASCII character chart to see the full list of hex ASCII character codes.	**Destination Variable**	Select the variable in which you want to store the converted string data. If you select a complex variable in the **Destination Variable** field, then you must select a complex variable field in the **Destination Variable Field** field.
	Destination Variable Field	If you select a complex variable in the **Destination Variable** field, then in the **Destination Variable Field** field, select the complex variable field in which you want to store the converted string data.
	Source Variable	Select the variable that contains the hex-encoded data that you want to convert to string format. If you select a complex variable in the **Source Variable** field, then you must select a complex variable field in the **Source Variable Field** field.
	Source Variable Field	If you select a complex variable in the **Source Variable** field, then in the **Source Variable Field** field, select the complex variable field that contains the hex-encoded data that you want to convert to string format.
Sub String: Returns a part of a string in the source variable and assigns it to the destination variable. The source variable must be of type string. **Note:** The index for a Sub String operation is zero-based. This means that you count the first character as zero	**Destination Variable (and Field)**	Select the variable to which you want to assign the substring. If the **Destination Variable** is a complex variable, select the field to which you want to assign the substring.
	Source Variable (and Field)	Select the variable from which you want to select and return the substring. If the **Source Variable** is a complex variable, select the field from which you want to select and return the substring.

Table continues…

Type	Associated properties	Comments/Remarks
(0) and go on from there. For example if you wanted to take the substring “truck” from the word “firetruck,” you must use a **Start Index** of **4** (the fifth character) and **Number of Characters** of **5**. Note that, in this case, for **Number of Characters**, you can also enter `-1` (see **Number of Characters** description).	**Start From** **Start From Variable (and Field)**	Enter in this field the point in the string at which the substring starts. For example, if you want the substring to start at the seventh character in the string, enter `6`. You can use a variable for the Start From rather than a hard coded value. If Start From variable is complex, then provide a field.
	Number of Characters **Num Chars Variable (and Field)**	Enter in this field the maximum number of characters to include in the substring that is returned. If the string has fewer than this number, it returns a shorter substring. If you enter `-1` in this field, the operation returns all characters starting with and following the index. For example, if you want to remove the first three digits from an account number that have a variable number of digits, set the **Start Index** to **3** and set the **Number of Characters** to `-1`. The default for this field is `-1`. You can use a variable for the number of characters to extract rather than a constant. If the Num Chars variable is complex, then you must select a field.
To Lower Case: Converts string values to all lower case.	**Destination Variable (and Field)**	The variable that receives the lower case converted string. If the **Destination Variable** is complex, a field must also be selected.
	Source Variable (and Field)	The variable whose value will be converted to all lower case. The converted result is stored in the **Destination Variable**. If the Source Variable is complex, a field must also be selected.
To Upper Case: Converts string values to all upper case.	**Destination Variable (and Field)**	The variable that receives the upper case converted string. If the **Destination Variable** is a complex variable, a field must also be selected.
	Source Variable (and Field)	The variable whose value will be converted to all upper case. The converted result is stored in the **Destination Variable**. If the **Source Variable** is a complex variable, a field must also be selected.

Table continues…

Type	Associated properties	Comments/Remarks
Trim Whitespace: Removes the whitespace from the beginning and end of strings. For example: “ Hello World ” would become “Hello World” after the trim whitespace operation.	**Destination Variable (and Field)**	Select the variable that will contain the results from the trimming action. If the destination is a complex variable, a field must be selected.
	Source Variable (and Field)	Select the variable (and field if **Source Variable** is a complex variable) that contains the string before the trim operation.
Set Current Language: Sets the project language. If you have entered a language that does not exist or does not match with any project language, then a SCERuntimeException is generated.	**Language**	Select the project language from the drop down list.
	Variable (and Field)	Select the variable that stores the project language.
Capture Last Node Data: Captures information about the call or message and the last node visited in the call flow or message flow during application execution before the operation is called.	**Destination Variable**	Select a variable that will be updated with the name of the previously executed node.
	Destination Variable Field	If the selected **Destination Variable** is a complex variable, select the field that will be updated with the name of the previously executed node.
	Count	The number of node names to capture during application execution before this operation is called. The default count is one. If the count is greater than one, then the call stack is captured in reverse order. That is, Last node, Second to last, Third to last, and so on.

Table continues…

Type	Associated properties	Comments/Remarks
	Store in Collection	Defines how you want to capture the information about last node that was visited during application execution before this operation was called. The options are: • **false**: (Default) The names of the nodes are stored in the destination variable as a delimited string in reverse order. For example, if the count is set to three and you set the value of **Store in Collection** to **false**, the names of the nodes are stored as “last|secondToLast|thirdToLast”. • **true**: The destination variable is converted into a collection variable to store these values in reverse order. **Note:** If you set the value of **Store in Collection** to **true**, you must select a simple variable in the **Destination Variable** field.

OR AND OR OR

Type

Boolean operator

Available from

Data node

Any level tab in the Prompt File Editor

Purpose

You can use the **OR** Boolean operator to evaluate if one of its child expressions is true.

Behavior

This item sets a compound condition that evaluates to true only if one of its child expressions is true. The item is used as a parent item and can be nested under If on page 739 or any other Boolean operator. This item connects all its child expressions logically. Its child expressions can be either a single expression or a compound condition.

For more information, see Creating single and compound conditions on page 608.

Properties

Boolean operator: Shows the name of the Boolean operator. If you want to select any other Boolean operator, select that operator from the list.

Output Parameter

Type

Application item

Available from

Return node on page 662 only

Purpose

When using a modular approach to application design, it is often necessary to pass variable or constant values from one module to another. With **Output Parameter** item, a speech, message, or web project used as a module node can pass data back to the parent speech, message, or web project.

For more information about creating and using application modules, see Module nodes on page 143.

Behavior

Use the **Output Parameter** item when you want to pass a variable value from a module node to the parent speech, message, or web project. To pass values from the module back to the parent application, you must drag one **Output Parameter** item into the Return node for each value that you want to pass back.

After you generate and save your module, you must deploy it as an Orchestration Designer module. For more information about the procedure to deploy Orchestration Designer modules, see Deploying a speech, or a message flow, or a web project as an Orchestration Designer reusable module on page 431.

When you use your speech, message, or web project as a module in another project, Orchestration Designer automatically populates the module node with one **Module Output** item for each **Output Parameter** item in the module. For more information, see Module Output on page 767.

Properties

- **Name**: Enter a name for the **Output Parameter** item.

  Orchestration Designer automatically assigns a default name when you drag the **Output Parameter** item into the node. You can use the default name or rename it.

  Note:

  The name must follow conventions for naming Java components as described in Conventions for naming Java components on page 59.

- **Description**: (Optional) Enter a brief description of what the output parameter value is used for. Avaya recommends you to write a good description so that when you use the module in another speech project, you can find out what the output parameter is used for.
- **Variable**: Select the variable whose value you want to pass back to the parent application. If the variable you select is a complex variable, you must also select a **Variable Field**.

- **Variable Field**: If you have selected a complex variable in the **Variable** field, you must select the field whose value you want to pass back to the parent application from the drop-down list. If you have selected a simple variable in the **Variable** field, this field is blank.
- **Expression**: Enter the ECMAscript expression you want to pass to the object.
- **Json String Variable:** If the returned value of the output parameter is an object, the variable will have the json string of the object. This allows developers to further process the content of the object in the OD application that invokes the module.

Parallel 

Type

Media item

Available from

Body, Parallel, Sequence

Purpose

The parallel container defines a simple time grouping in which multiple elements can play at the same time.

Behavior

By default, elements within a parallel container play at the same time. You can set various properties on the media itself to control the play back such as timing offsets, duration, repeats. You can also nest parallel and sequence elements under this container.

Properties

- **End Sync**: Determines how to end the sync as a function of child media elements. The End Sync option can have one of the following values:
  - **first**: The par, excl, or media element's implicit duration ends with the earliest active end of all the child elements. This does not refer to the lexical first child, or to the first child to start, but rather refers to the first child to end its (first) active duration.
  - **last**: The par, excl, or media element's implicit duration ends with the last active end of the child elements. This does not refer to the lexical last child, or to the last child to start, but rather refers to the last active end of all children that have a resolved, definite begin time. If the time container has no children with a resolved begin time, the time container ends immediately. If child elements have multiple begin times, or otherwise restart, the child elements must complete all instances of active durations for resolved begin times. This is the default value for par and excl elements.
  - **all**: The par, excl, or media element's implicit duration ends when all of the child elements have ended their respective active durations. Elements with indefinite or unresolved begin times will keep the simple duration of the time container from ending. When all elements have completed the active duration one or more times, the parent time container can end.
  - **media**: The time container element's implicit duration ends when the intrinsic media duration of the element ends. This must be defined by a host language. If the time container element does not define an intrinsic media duration, the host language must

define the simple duration for the element. This is the default value for media time container elements.

- **Fill**: Determines the action to take after an active duration. The Fill option can have one of the following values:

  Note:

  If you do not select any value, the Fill behavior defaults to “auto”.

  - **remove**: Specifies that the element will not extend past the end of the last instance of the simple duration.
  - **freeze**: Specifies that the element will extend past the end of the last instance of the simple duration by “freezing” the element state at that point. The parent time container of the element determines how long the element is frozen (as described immediately below).
  - **hold**: Setting this to “hold” has the same effect as setting to “freeze”, except that the element is always frozen to extend to the end of the simple duration of the parent time container of the element (independent of the type of time container). For profiles that support a layered layout model (e.g., SMIL 2.0 Language Profile), held elements (elements with fill=“hold”) will refresh their display area when a layer is added on top then later removed.
  - **transition**: Setting this to “transition” has the same effect as setting to “freeze”, except that the element is removed at the end of the transition. This value is only allowed on elements with media directly associated with them. If specified on any other element (for example, a time container element in the SMIL language profile), the attribute is ignored. See the SMIL Transitions module.
  - **auto**: The fill behavior for this element depends on whether the element specifies any of the attributes that define the simple or active duration. If none of the attributes dur, end, repeatCount or repeatDur are specified on the element, then the element will have a fill behavior identical to that if it were specified as “freeze”. Otherwise, the element will have a fill behavior identical to that if it were specified as “remove”.

Phrase

Type

Prompt Segment item

Available from

Any level tab in the prompt file editor. For more information about the prompt file editor, see Prompt file editor in a speech application on page 207.

Purpose

The Phrase item plays the selected audio file in a prompt. This item makes it possible to use prerecorded audio files in prompts.

Behavior

When the system plays a prompt containing a Phrase item, it calls and plays the designated audio phrase file in its turn.

Before the Phrase item is functional, you must create the audio phrase file to use in it. For more information about creating audio phrase files, see Phrases and phrasesets on page 165.

Properties

- **Phraseset**: Name of the phraseset that contains the phrase that you want to play.
- **Phrase in phraseset**: Name of the phrase within the selected phraseset.
- **Standalone Phrase**: Name of phrase that you want to play (if standalone).
- **No Cache**: The option to disable caching for this particular **Phrase**. The default value is **false**. You can change the value to **true** to disable caching. When you change the value to **true**, the audio elements in the generated VXML for those phrases will have two new attributes — **maxage** and **maxstale**. Both these attributes are set to “0”, disabling caching.

Phrase Variable

Type

Prompt Segment item

Available from

Any level tab in the prompt file editor. For more information about the prompt file editor, see Prompt file editor in a speech application on page 207.

Purpose

The **Phrase** variable Item can play an audio phrase where the variable value references a phraseset or a standalone phrase or an audio URL.

Behavior

A **Phrase** variable can point to a phrase in phraseset, a standalone phrase or an external audio file. A **Phrase** variable supports collection of values. If underlying variable value is a collection, then multiple <audio> elements are generated in the VXML.

Collection values must all be same type (phraseset-phrase, audio URL, or phrase ID). This feature can be useful when getting audio file information dynamically from a data source such as a database.

For more information about creating audio phrase files, see Phrasesets on page 167.

Properties

- **Type**: When the Type is Phraseset:Phrase ID, in the Variable Field, specify a variable name in the Phraseset:Phrase ID format.
  - When the Type is Phrase ID, in the Variable Field, specify a variable name which refers to the standalone phrase name.
  - When the Type is Audio URL, in the Variable Field, specify a variable name which contains the external audio file URL.
  - When the Type is DTMF Tones, then the prompt will generate DTMF tone pulses to match the value of the variable (convert the string value of a variable into a list of DTMF recordings that will be played). For example, if the variable value is “#1234”, then the prompt will pulse in-band 5 DTMF tones for “#”, “1”, “2”, “3”, “4”.

Also see the Setting Orchestration Designer project general properties on page 490 for enabling DTMF variables. If not enabled, then the Prompt Editor marks errors if DTMF Tone Phrase Variables are used. There can be run-time errors because the audio files for the DTMF recordings will not be part of the project.

- **Variable**: A variable name which points to a phrase in phraseset, a standalone phrase or an external URL.
- **Variable Field**: A field contained by a Variable. If you use simple Variable, leave this field empty.
- **Backup Text**: The alternative text played if Orchestration Designer cannot find the audio file in the phrase or phraseset.
- **Backup Variable**: The Variable which contains the alternative text played if Orchestration Designer cannot find the audio file in the phrase or phraseset.
- **Backup Field**: A field contained by a Backup Variable. If you use simple Variable, leave this field empty.
- **No Cache**: The option to disable caching for this particular **Phrase Variable**. The default value is **false**. You can change the value to **true** to disable caching. When you change the value to **true**, the audio elements in the generated VXML for those phrases will have two new attributes — **maxage** and **maxstale**. Both these attributes are set to “0”, disabling caching.

PrepareAAI

Type

Operation item

Available from

Data node

Note:

The **PrepareAAI** operation item is available only in speech projects.

Purpose

To generate application-to-application (AAI) data by using the application-specific data and UCID, and convert the AAI data from string format to ASCII hex format. You can then use the hex-encoded AAI data in a transfer.

Behavior

Depending on the mode that you use to send the AAI data, the **PrepareAAI** operation item uses the application data and UCID to generate the AAI data. The **PrepareAAI** operation item then encodes the AAI data in a hex format and adds a protocol descriptor and application ID to the hex-encoded AAI data. You can use the hex-encoded AAI data in a transfer.

For more information, see Blind Transfer on page 678, Bridged Transfer on page 684, Consultation Transfer on page 699, and Transfer on page 848.

Properties

- **Mode**: The option to specify the mode that the system must use to generate the AAI data. The system also adds a protocol descriptor and application ID to the hex-encoded AAI data.

The options are:

- **Shared**: To generate the hex-encoded AAI data by using the **Shared** mode.

  In a **Shared** mode, the format of the configured value is fixed and as specified by Avaya Experience Portal. Orchestration Designer sends the AAI data into an array of IDs and their corresponding values.

  If the **Mode** is **Shared**, you can specify the UCID along with the application data that you want to format and send as AAI data.

  The following is a sample hex-encoded data for the **Shared** mode:

  ```
  Shared Mode
  PD,04;C8,<hex encoded data>;FA,<ucid>
   PD,00;FA,<ucid>
  PD,04;C8,<hex encoded data>
  ```

- **Service Provider**: To generate the hex-encoded AAI data by using the **Service Provider** mode.

  In the **Service Provider** mode, Orchestration Designer sends the AAI data in a single block without interpreting the data.

  If the **Mode** is **Service Provider**, you can specify the application data that you want to send as AAI data.

  The following is a sample hex-encoded data for the **Service Provider** mode:

  ```
  Service Provider Mode
  04<hex encoded data>
  ```

- **UCID Variable**: The variable that contains the UCID that you want to use in the AAI data and convert to ASCII hex format.

  Note:

  The **UCID Variable** field is available only if you select the **Mode** as **Shared**.

  If you select a complex variable in the **UCID Variable** field, then you must select a complex variable field in the **UCID Variable Field** field.

- **UCID Variable Field**: If you select a complex variable in the **UCID Variable** field, then in the **UCID Variable Field** field, select the complex variable field that contains the UCID that you want to use in the AAI data and convert to ASCII hex format.

  Note:

  The **UCID Variable Field** field is available only if you select the **Mode** as **Shared**.

- **Application Data Variable**: The variable that contains the application-specific data that you want to use in the AAI data and convert to ASCII hex format.

  If you select a complex variable in the **Application Data Variable** field, then you must select a complex variable field in the **Application Data Variable Field** field.

- **Application Data Variable Field**: If you select a complex variable in the **Application Data Variable** field, then in the **Application Data Variable Field** field, select the complex variable field that contains the application-specific data that you want to use in the AAI data and convert to ASCII hex format.

- **VdN**: A vector directory number (VdN) is a “soft” extension number on an automatic call distributor that directs an incoming call to a “vector”. This number is not assigned to an

equipment location and must be set up according to the customer's dial plan, and the optional vectoring software must be enabled.

- **Collected digits**: Collected digits are digits (DTMF key presses) that an application collects and passes to another application using UUI. Collected digits can be up to 16 digits.
- **Result Variable**: The variable in which you want to store the formatted AAI string. Depending on the mode that you use to generate the AAI data, the system uses the application data and UCID to generate the AAI data. The system then encodes the AAI data in hex format, adds the protocol descriptor and application ID to the hex-encoded AAI data, and stores the formatted AAI string in the variable that you specify in the **Result Variable** field.

  If you select a complex variable in the **Result Variable** field, then you must select a complex variable field in the **Result Variable Field** field.
- **Result Variable Field**: If you select a complex variable in the **Result Variable** field, then in the **Result Variable Field** field, select the complex variable field in which you want to store the formatted AAI string.

Prompt

Type

Form item

Available from

- **AppRoot** node (see About the AppRoot node on page 132)
- Announce node on page 634
- Prompt and Collect node on page 659
- Collect node on page 651
- Blind Transfer node on page 646
- Bridged Transfer node on page 647
- Record node on page 660
- Disconnect node on page 654
- Menu node on page 656
- Form node on page 655
- Return node on page 662

Purpose

In a speech application, the **Prompt** item makes it possible to play a selected prompt. For example, in a **Prompt and Collect** node, use the **Prompt** item to prompt the caller for a response. In a Record node, use the **Prompt** item to instruct the caller how to make the recording and what the post-recording options are.

Use the **Prompt** item in an SMS channel or an email channel message application, to send an outbound SMS or email message respectively.

You can use the **Language** property of the **Prompt** item to create multilingual speech applications or message applications.

You can use variables for the prompts, so that prompts can vary dynamically.

Behavior

In a speech application, when you drag the **Prompt** item into a node, the **Prompt** item is generally played in the order it appears with other Prompt items at the same level within the node. Also, higher level **Prompt** items are played before lower level **Prompt** items.

For example, you have five **Prompt** items within a single Form node. Two of those **Prompt** items are assigned to an **Input** item. The subflow of the node is as follows:

- Prompt 1
- Prompt 2
- Input
  - Prompt 3
  - Prompt 4
- Prompt 5

In this example, the system would play the prompts in the following order: 1, 2, 5, 3, 4. The reason for this play order is that the higher level prompts (1, 2, and 5) are played in order before the lower level prompts (3 and 4).

If a **Prompt** item is a sub-item to another item, however, it plays before the item is executed. For example, if you dragged a **Prompt** item as a sub-item into a **Blind Transfer** item, the system plays the prompt before transferring the call.

In a message application, the **Prompt** item sends an outbound SMS or email message.

If you copy a prompt variable from one speech application or message application to another, the system copies the variable, but does not copy the source to which the variable references.

In the **AppRoot** node, the **Prompt** item can only be used as a sub-item to one of the event handlers.

Note:

Before you can select and use a prompt in a **Prompt** item, you must first create and save the prompt.

Properties

- **Name Variable**: Select the prompt variable. If you select a complex variable in this field, then you must select a prompt variable field in the **Name Variable Field** field. This field is unavailable if you specify a value in the **Name Constant** field.
- **Name Variable Field**: If you select a complex variable in the **Name Variable** field, then select a prompt variable field in the **Name Variable Field** field. This field is unavailable if you select a simple variable in the **Name Variable** field.
- **Name Constant**: In a speech application, select the prompt you want to play.

  In a message application, select the prompt that contains the outbound email or SMS message that you want to send.

- **Language**: The language in which you want to play a prompt or send an outbound SMS or email message to a customer. You can use the **Language** property to build a multilingual speech application or message application.

  Note:

  The default language is the starting language of an Orchestration Designer project.

  To use a language other than the starting language of an Orchestration Designer project, you must first add the project language that you want to use in the Orchestration Designer project, and then translate the phrases and prompts to that language. For more information, see Project languages on page 277and Adding a project language on page 278.

  For example, you set the starting project language as English in a speech project. In the **Prompt and Collect** node of the speech application, you want to play a prompt in French. In the **Prompt** item, select the prompt file that you want to play, and then set the **Language** property to French. The system automatically uses and plays the French translated version of the prompt.

Property

Type

Form item

Available from

- **AppRoot** node (see AppRoot node on page 132)
- Announce node on page 634
- Prompt and Collect node on page 659
- Blind Transfer node on page 646
- Bridged Transfer node on page 647
- Record node on page 660
- Menu node on page 656
- Form node on page 655

Purpose

The **Property** item is used to set one or more values that affect system behavior. These values include properties like timeouts and recognition processes.

Behavior

When dragged into the AppRoot node, the **Property** item sets application-wide defaults for the designated properties. When dragged into a node, the **Property** item sets the designated properties at the node level. When attached to a node item, the **Property** item sets the designated properties at the item level.

When a given property is set using the **Property** item, that setting affects anything below it in the application. For example, if you set the **Timeout Complete** property in a node, that setting affects everything that comes after it in the node. It does not, however, affect other nodes. At a given

level, the property value remains in effect until it is reset at a later point. For example, if you have a node with two property value settings for a particular property within the same node, the first stays in effect until the application reaches the second property setting.

A property setting at a lower level overrides a property setting for the same property at a higher level. For example, if you have the DTMF **Terminating Character** property set in both the AppRoot node and another DTMF **Terminating Character** property set in a node, the property at the node level overrides the one set at the AppRoot node level.

You can set as many or as few property settings as you want in a single **Property** item.

Note:

To set properties that are not defined within the **Property** item, use the **External Property** item. For more information, see External Property item on page 729.

Properties

The properties for the **Property** item are separated into three general groups according to their functions:

- **Speech**: These property settings determine various aspects of how the ASR engine responds to spoken inputs.

Speech (ASR) Property Setting	Description
Confidence Level	Enter the minimum required level of confidence for an ASR engine to return a recognition of a speech utterance. The system rejects any utterance that returns a confidence level lower than this value and throws a No Match event. The range of valid values for this property is 0.0 through 1.0. A value of 0.0 means that minimum confidence is required for a recognition. A value of 1.0 means that a maximum confidence is required for a recognition. Default = **0.5**
Sensitivity	Enter the sensitivity level for speech recognition to start. This setting determines how loud an utterance must be for the ASR engine to start speech recognition. The range of valid values for this property is 0.0 through 1.0. The higher the number you enter in this field, the more sensitive the ASR engine is, and the less amount of input volume it takes to start speech recognition. Default = **0.5**
Speed vs. Accuracy	Enter the number to determine the balance between speed of recognition and accuracy of recognition. The range of valid values for this property is 0.0 through 1.0. A lower value tips the balance in favor of speed of recognition. A higher value tips the balance in favor of accuracy of recognition. Default = **0.5**

Table continues…

Speech (ASR) Property Setting	Description
Timeout Complete	Enter the number of seconds of silence the system must wait before finalizing an utterance. Note that, if no utterance was ever recognized with a match, the system uses this timeout period to determine when to throw a No Match event. If an utterance was successfully recognized, this is the period of silence the system waits before returning the result. Valid values for this property are positive real numbers. Numbers can be of the format “*n*”, “*n.*”, “. *n*” or “*n.n*”, where *n* represents any sequence of one or more digits. Take care when setting this property. If you set the number too high, it can slow down system response time, even for successful recognitions. If you set the number too low, it can lead to an utterance being cut off prematurely. Reasonable values for this property are usually in the range 5 to 15 seconds. This property has no default value. The default value for this is set on the ASR server. If you set a value in this field, this value overrides the default on the ASR server. See also the next property, **Timeout Incomplete**.
Timeout Incomplete	Enter the number of seconds of silence the system must wait before finalizing an incomplete recognition of an utterance. An incomplete recognition is defined as an incomplete match of all active grammars. In this case, after the timeout is reached, the system rejects the utterance and throws a No Match event. An incomplete timeout can also occur when there is a complete match of an active grammar but it is possible to speak further and still match the grammar. By contrast, a complete timeout (see previous property, **Timeout Complete**) occurs when there is a complete match of an active grammar and no further input can be recognized. Valid values for this property are positive real numbers. Numbers can be of the format “*n*”, “*n.*”, “. *n*” or “*n.n*”, where *n* represents any sequence of one or more digits. Take care when setting this property. If you set the number too high, it can slow down system response time, even for successful recognitions. If you set the number too low, it can lead to an utterance being cut off prematurely. This property is usually set lower than the **Timeout Complete**, but high enough to allow callers to pause mid-utterance, for instance, for a breath. Reasonable values for this property are usually in the range 2 to 5 seconds. This property has no default value. The default value for this is set on the ASR server. If you set a value in this field, this value overrides the default on the ASR server. See also the previous property, **Timeout Complete**.

Table continues…

Speech (ASR) Property Setting	Description
Maximum Speech	Enter the number of seconds you want to allow the caller to continue speaking before the system ends the recognition attempt and throws a **maxspeechtimeout** event. Valid values for this property are positive real numbers. Numbers can be of the format “*n*”, “*n.*”, “. *n*” or “*n.n*”, where *n* represents any sequence of one or more digits. This property has no default value. The default value for this is set on the ASR server. If you set a value in this field, this value overrides the default on the ASR server.

- **General**: These property settings determine various aspects of how the system responds to DTMF (touchtone) key press inputs.

General Property Setting	Description
Interdigit timeout	Enter the number of seconds the system must wait for an additional DTMF input when such an input is expected. This property becomes active when a DTMF expects and requires a certain number of DTMF input digits. For example, when collecting a telephone number from a caller, the DTMF grammar is set up to require a ten-digit number. If the caller enters eight digits and then stops, this property is triggered at the end of the timeout setting, and the system throws a No Match event. Valid values for this property are positive integers. This property has no default value. The default value for this is set on the server that handles DTMF responses. If you set a value in this field, this value overrides the default on that server.
Private	When enabled, this property omits certain pieces of data from the logs. For example, you want to suppress a user’s credit card number, which is private information, from the logs. When enabled, it enables the privacy option on Avaya Experience Portal only. For more information about privacy option, refer the “Privacy feature support for VoiceXML applications” section of *Planning for Avaya Experience Portal guide*. **Note:** This property is not applicable for Interactive Response.

Table continues…

General Property Setting	Description
Terminating timeout	Number of seconds the system must wait for an additional DTMF input when such an input is possible. This property becomes active when a DTMF can accept a variable number of DTMF input digits. For example, when collecting an account number from a caller, the account is either seven or eight digits long. If the caller enters seven digits and then stops, this property is triggered at the end of the timeout setting, and the system returns a successful recognition result. Valid values for this property are positive real numbers. Numbers can be of the format "*n*", "*n.*", ". *n*" or "*n.n*", where *n* represents any sequence of one or more digits. This property has no default value. The default value for this is set on the server that handles DTMF responses. If you set a value in this field, this value overrides the default on that server.
Terminating character	The DTMF key press that the system must use to end a DTMF input. When this key is pressed, the system ends the DTMF input recognition and returns the appropriate result. Valid values for this property are the numbers **0** through **9**, the star (*****) key, and the pound (**#**) key. The default is **#**

- **Fetch**: These property settings pertain to the fetching of new documents and resources.

Fetch Property Setting	Description
fetchTimeout	The timeout for fetches. The value is a Time Designation. The default value is platform-dependent.
grammarmaxage	Tells the platform the maximum acceptable age, in seconds, of cached grammars. The default is platform-specific.
grammarmaxstale	Tells the platform the maximum acceptable staleness, in seconds, of expired cached grammars. The default is platform-specific.

- **Transitional Audio**: These property settings determine what prompt is to be used as a transitional audio prompt and how it is to behave. For more information about the use and behavior of transitional audio prompts, see Transitional audio prompts in speech applications on page 205.

Transitional Audio Property Setting	Description
Audio Prompt	Select the prompt to use as the transitional audio prompt. Note that a prompt must be defined as a transitional audio prompt when it is created. Only transitional audio prompts can be selected for this property.

Table continues…

Transitional Audio Property Setting	Description
Delay seconds	Enter the number of seconds the system must wait before starting to play the transitional audio prompt. The idea is that, if the system response time is short enough, it is better to have a short period of silence, rather than starting to play an audio file that is immediately cut off. Valid values for this property are positive real numbers. Numbers can be of the format “*n*”, “*n.*”, “. *n*” or “*n.n*”, where *n* represents any sequence of one or more digits. This property has no default value. The default value for this is set on the IVR system. If you set a value in this field, this value overrides the default on the IVR system.
Minimum play seconds	Enter the minimum number of seconds of the prompt that you want the system to play before proceeding. The idea is that, after the caller does begin to hear the prompt, it should not be stopped too quickly. This can be useful, for instance, when there is a short message you want callers to hear before the system proceeds. Valid values for this property are positive real numbers. Numbers can be of the format “*n*”, “*n.*”, “. *n*” or “*n.n*”, where *n* represents any sequence of one or more digits. This property has no default value. The default value for this is set on the IVR system. If you set a value in this field, this value overrides the default on the IVR system.

Prosody

Type

SSML item

Available from

Any level tab in the prompt file editor. For more information about the prompt file editor, see Prompt file editor in a speech application on page 207.

Purpose

The **Prosody** item is a Synthesized Speech Markup Language (SSML) element that makes it possible to control various aspects of Text-to-Speech (TTS) synthesis. This element means you can get more natural-sounding speech synthesis from the TTS engine.

Behavior

Based on the properties you set, the **Prosody** item alters the rendering of TTS speech synthesis. All properties are optional, but if you use the **Prosody** item, you must use at least one property. The **Prosody** item has six basic properties:

Note:

The specific effects of many of these properties can vary from one TTS engine to another.

- **Pitch**: This property controls the baseline pitch of the speech output. Increasing this property raises the baseline pitch. Decreasing this property lowers the baseline pitch.
- **Range**: This property controls the pitch range, or variability, of the speech output. Increasing this property increases the range of pitches the TTS engine produces. Decreasing this property decreases the pitch range.
- **Rate**: This property controls the speaking rate at which the TTS engine produces the output. Increasing this property speeds up the synthesized speech. Decreasing this property slows down the synthesized speech.
- **Duration**: This property is the desired amount of time it takes the TTS engine to read the contents of the TTS element. The value is stated in seconds (s) or milliseconds (ms). The text to be spoken is either compressed or expanded to fit into the duration you select. For example, if the text would normally take five seconds to speak, you set this property to four seconds, the system increases the rate to make the reading of the text fit into four seconds.

  The **Duration** property takes precedence over the **Rate** property.
- **Volume**: This property controls the volume, or loudness, of the speech output. Increasing this property makes the TTS output louder. Decreasing this property makes the TTS output quieter.
- **Contour**: This property sets the actual pitch contour for the speech output. This contour is accomplished by means of a series of percent/pitch value pairs.

  The **Contour** property takes precedence over both the **Pitch** and **Range** properties.

For information about the properties and behavior of properties, see the next section, “Properties.”

Note:

The **Prosody** item and its properties function correctly only with SSML-compliant speech synthesis engines. Microsoft SAPI Speech, which is used by Orchestration Designer during application simulation, is *not* an SSML-compliant speech synthesis engine, so any settings you make with this item are ignored. For more information about the SSML standard, see the Speech Synthesis Markup Language (SSML) Version 1.0 W3C Recommendation.

Properties

All properties are optional, but if you use the **Prosody** item, you must use at least one property. Note that all units, such as **Hz** and **st**, are case-sensitive.

- **Pitch**: Select one of the following settings:

Pitch Setting	Description
default	This setting has no effect on the output. This setting uses the default baseline pitch of the TTS server.

Table continues…

Pitch Setting	Description
x-low **low** **medium** **high** **x-high**	These settings represent a range of pitch options. The specific application of these properties varies according to the TTS server.
custom	With this setting, you can define the baseline pitch that you want, as a modification of the TTS server default. When you select this setting, Orchestration Designer automatically adds the **Custom Pitch [Hz or st]** property to the Property view.
Custom Pitch [Hz or st]	This setting is available only when the **custom** setting for **Pitch** is selected. With it, you can fine tune the baseline pitch by raising or lowering the pitch relative to the server default: - To raise or lower the pitch based on frequency, enter a number followed by **Hz**. For example, if you set this to **+8000Hz** the system raises the baseline pitch by 8000 cycles per second. If you set this to **-500Hz**, the system lowers the pitch by 500 cycles per second. - To raise or lower the pitch by semitones, enter a positive or negative number followed by **st**. Each whole number represents a semitone on the diatonic scale. Positive numbers raise the pitch. Negative numbers lower the pitch. For example, if you enter **+3st** in this field, the baseline pitch is raised by three semitones. If you enter **-1.5st**, the baseline pitch is lowered by one and a half semitones. The correct format to enter these values is a positive (+) or negative (-) sign, followed by a number, followed by **Hz** or **st**, with no spaces. Numbers can be of the format “*n*”, “*n.*”, “. *n*” or “*n.n*”, where *n* represents any sequence of one or more digits.

- **Range** - Select one of the following settings:

Range Setting	Description
default	This setting has no effect on the output. This setting uses the default pitch range of the TTS server.
x-low **low** **medium** **high** **x-high**	These settings represent a range of pitch range options. The specific application of these properties varies according to the TTS server.
custom	With this setting, you can define the pitch range that you want, as a modification of the TTS server default. When you select this setting, Orchestration Designer automatically adds the **Custom Range [Hz or st]** property to the Property view.

Table continues…

Range Setting	Description
Custom Range [Hz or st]	This setting is available only when the **custom** setting for **Range** is selected. With it, you can fine tune the pitch range by increasing or decreasing the range relative to the server default: - To increase or decrease the pitch range based on frequency, enter a number followed by **Hz**. For example, if you set this to **+8000Hz** the system increases the pitch range by 8000 cycles per second. If you set this to **-500Hz**, the system lowers the pitch range by 500 cycles per second. - To increase or decrease the pitch by semitones, enter a positive or negative number followed by **st**. Each whole number represents a semitone on the diatonic scale. Positive numbers increase the pitch range. Negative numbers decrease the pitch range. For example, if you enter **+3st** in this field, the pitch range is increased by three semitones. If you enter **-1.5st**, the baseline pitch is decreased by one and a half semitones. The correct format to enter these values is a positive (+) or negative (-) sign, followed by a number, followed by **Hz** or **st**, with no spaces. Numbers can be of the format “*n*”, “*n*.”, “. *n*” or “*n.n*”, where *n* represents any sequence of one or more digits.

- **Rate**: Select one of the following settings:

Rate Setting	Description
default	This setting has no effect on the output. This setting uses the default speaking rate of the TTS server.
x-slow **slow** **medium** **fast** **x-fast**	These settings represent a range of speaking rate options. The specific application of these properties varies according to the TTS server.
custom	With this setting, you can define the speaking rate that you want, as a modification of the TTS server default. When you select this setting, Orchestration Designer automatically adds the **Custom Rate [positive float]** property to the Property view.
Custom Rate [positive float]	This setting is available only when the **custom** setting for **Rate** is selected. With it, you can fine tune the speaking rate by increasing or decreasing the rate relative to the server default. The number you enter in this field acts as a multiplier on the default rate. For example, a setting of **1** or **100%** in this field means there is no change to the default rate. A setting of **2** or **200%** in this field makes the speaking rate twice as fast as the default rate. A setting of **0.5** or **50%** in this field makes the speaking rate half as fast as the default rate. The correct format to enter these values is “*n*”, “*n*.”, “. *n*” or “*n.n*”, where *n* represents any sequence of one or more digits.

- **Duration**: Select one of the following settings:

Duration Setting	Description
250ms **500ms** **750ms** **1s** **2s** **3s** **4s** **5s**	These settings represent a range of duration options in milliseconds (ms) or seconds (s). This setting overrides any **Rate** setting you have.
custom	With this setting, you can define the exact duration that you want. When you select this setting, Orchestration Designer automatically adds the **Custom Duration [s or ms]** property to the Property view.
Custom Duration [s or ms]	This setting is available only when the **custom** setting for **Duration** is selected. With it, you can set the exact duration for the text to be spoken. The correct format to enter these values is "*n*", "*n.*", ". *n*" or "*n.n*", where *n* represents any sequence of one or more digits. The number must be followed, with no space in between, by **s** for seconds or **ms** for milliseconds.

- **Volume**: Select one of the following settings:

  Note:

  The **Volume** property uses a range of 0.0 (silent) to 100.0 (full volume).

Volume Setting	Description
default	This setting is the same as setting the volume to 100.0, or full volume.
silent	This setting is the same as setting the volume to 0.0.
x-soft **soft** **medium** **loud** **x-loud**	These settings represent a range of volume options. The specific application of these properties varies according to the TTS server.
custom	With this setting, you can define the exact volume that you want. When you select this setting, Orchestration Designer automatically adds the **Custom Volume [float]** property to the Property view.
Custom Volume [float]	This setting is available only when the **custom** setting for **Volume** is selected. With it, you can set the exact volume you want for the text to be spoken. The correct format to enter these values is a positive (+) or negative (-) sign, followed by a number. Numbers can be of the format "*n*", "*n.*", "*.n*" or "*n.n*", where *n* represents any sequence of one or more digits.

- **Contour**: Enter in this field the pitch contour settings you want to use for the text to be spoken.

A pitch contour is defined by a set of *time-position/target-pitch* pairs. These pairs must in the form: **`(time-position,target-pitch)`**. The *time-position* value is a percentage of the total time period for the text to be spoken. The *target-pitch* value is the relative pitch for the text to be spoken at that point in time. The algorithm for interpolating the pitch from one target to the next is specific to each TTS engine.

For example, you want the TTS server to speak the phrase, “Good morning.” In normal conversation, the relative pitch of the syllable “morn-” is higher than the word “Good.” The pitch of the final syllable, “-ing,” then, drops below the initial word “Good.” So, to create a pitch contour for this phrase, you can create the following (*time-position*, *target-pitch*) pairs:

`(0%,+20Hz)(40%,+30%)(75%,-10Hz)`

In this example, the initial pitch for the phrase, and the word “Good,” is set at 20 cycles per second above the baseline pitch. At a time position 40% of the way into the phrase, the pitch rises to 30% above where the phrase started. This takes place at about the point where the server speaks the syllable “morn-”. The final syllable, “-ing,” starts about 75% of the way into the phrase, in terms of time. This final syllable drops to a point, then, 10 cycles per second below the baseline pitch.

For more information about the Contour property of the Prosody item, see the Speech Synthesis Markup Language (SSML) Version 1.0 W3C Recommendation.

Publish Realtime Event

Available from

Data node

Purpose

You can use the **Publish Realtime Event** item to send real time events to Avaya Analytics™.

Behavior

The **Publish Realtime Event** takes attribute - value pairs. The attribute can be a variable or a constant. The value can be a variable or constant. You can add multiple attribute - value pairs.

Properties

Event Name Variable: Select the variable name for the event.

Event Name Field: Select the field name for the selected **Event Name Variable**.

Event Name Constant: Enter the constant name for the event.

1 - Attribute Name Variable: Select the name for the variable attribute.

1 - Attribute Name Field: Select the field for the **1 - Attribute Name Variable** selected.

1 - Attribute Name Constant: Enter the constant name for the attribute.

1 - Value Name Variable: Select the value for the **1 - Attribute Name Variable**.

1 - Value Name Field: Select the value for the **1 - Attribute Name Field**.

1 - Value Name Constant: Enter the value for **1 - Attribute Name Constant**.

Note:

When you enter data into the attribute - value pair, a new set of attribute - value pair gets created.

Publish Realtime Event Variable

Available from

Data node

Purpose

You can use the **Publish Realtime Event Variable** item to send real time events to Avaya Analytics™.

Behavior

The **Publish Realtime Event** takes a variable name. Orchestration Designer converts this variable name into JSON at runtime.

Properties

Event Name Variable: Select the variable name for the event.

Event Name Field: Select the field name for the selected **Event Name Variable**.

Event Name Constant: Enter the constant name for the event.

Variable Name: Select the variable name. The variable can be a simple variable, a complex variable, or a collection of simple and complex variables.

Record

Type

Form Item

Available from

- Announce node on page 634
- Record node on page 660
- Form node on page 655

Note:

The **Record** item cannot be used in the same node as the Blind Transfer on page 678 item, the Bridged Transfer on page 684 item, or the Input on page 741 item. So, although the **Record** item appears on palettes for other nodes, such as the Blind Transfer node, the Bridged Transfer node, and the Prompt and Collect node, it is not actually available for use in those nodes without destroying their intended functionality. It is intended primarily for use in the Record node.

Purpose

There are times when you want to collect information from callers that is not easily collected using DTMF or ASR input. For example, you want to get the address of the caller, to enter into a database later on. Or you want to allow callers to leave voice messages for the parties they are trying to reach. The **Record** item is the call flow element that is responsible for collecting recorded input and saving it to a file for retrieval later.

Behavior

The **Record** item:

- Optionally, plays a prompt to the caller, instructing the caller what to do and how to signal that the recording is finished.
- Optionally, plays a beep tone to signal the start of the recording.
- Records the message from the caller, starting after the tone, if the beep is turned on. If the beep is not turned on, the **Record** item starts immediately after the prompt, if there is one. If there is no prompt, the system starts recording as soon as the **Record** item starts to execute.

  Note:

  If the caller does not start recording within the time as set in the **Max Silence** property, the system throws a **No Input** event. For more information about **Record** item properties, see the following section, “Properties.”
- Ends the recording when one of the following conditions is met:
  - A predetermined period of silence.
  - The caller signals with a DTMF key press that the caller is done recording.
  - The recording reaches the maximum allowed period of time for a recording.
  - The caller hangs up.
  - The system throws an event from which the application cannot continue.
  - The caller utters something or enters a DTMF key press that triggers an active grammar.
- Stores the recording as an audio file.
- Returns a series of variable values to the application, based on data taken from the recording session.

When you drag this item into a node subflow, Orchestration Designer automatically creates a complex variable with the same name. This variable contains the following fields, which are populated upon completion of the node at run time:

- **confidence**: This field returns a value only if the caller uses an active grammar “hotword” to terminate the recording. When that happens, this field returns a value that indicates the ASR server confidence level that the hotword was uttered and recognized. Note that not all IVR

systems support simultaneous speech recognition and recording, in which case this field is never populated.

- **duration**: This variable field is the length of time that the recording lasted, in milliseconds. For example, if the recording lasted for just over 24 seconds, this can have a value of **24389**.
- **maxtime**: If the caller exceeded the maximum allowable time for the recording, the system automatically ends the recording. If this happens, the system returns a value of **true** to this variable field. Otherwise, this variable field has the default value of **false**.
- **size**: This field is the size of the recorded audio file, in bytes.
- **termchar**: The value of this field is the DTMF character, if any, that was used to end the recording. If the caller did not end the recording with a DTMF key press, this variable field is undefined.
- **utterance**: This field returns a value only if the caller uses an active grammar “hotword” to terminate the recording. When that happens, this field returns a value that indicates what the caller said that the ASR server recognized as being the hotword to terminate the recording. Note that not all IVR systems support simultaneous speech recognition and recording, in which case this field is never populated.
- **value**: This variable field contains the URL to the recorded audio file.

  The **value** field of the **Record** item variable contains the URL of the audio file that was recorded. When the Avaya Application Simulator comes to this variable field which uses the URL format, the system retrieves the audio file from the designated URL and plays it back. For more information, see Playing a recorded audio file on page 813.

You can also use other functions to assign the recorded audio file to a database, from which you can retrieve it later. This can be done, for instance, to have an administrator later retrieve an address and convert it to a text-based format for later use.

Properties

- **Name**: Enter the name you want to assign to the **Record** item and its associated variable.

  If you do not enter a name, Orchestration Designer automatically assigns both the item and the variable a default name.
- **Play Beep**: If you want the system to play a short beep before starting the recording, set this field to **true**, which is the default. If you do not want the system to play a beep, set this field to **false**.
- **Max Length**: Enter the number of seconds that you want to allow for the maximum duration of a recording. For example, if you want to allow callers a maximum of two minutes of recording time, set this to **120**.

  The maximum allowable value for this field depends on the limitations and settings on the IVR system. The default is **60**.
- **Max Silence**: Enter the number of seconds to allow silence at the end of the recording before the system ends the recording. When the system detects a period of silence, the system ends the recording after it reaches this number of seconds of silence.

  If no recording was ever started, the system throws a No Input event when it reaches this number of seconds of silence.

  The default for this field is **3**.

- **Modal**: To deactivate any non-local grammars while a recording is in progress, set this to **true**, the default. To allow any non-local grammars to remain active while the recording is in progress, set this field to **false**.

  Often, any global grammars you have active are used to throw events from anywhere in the application. With this setting, you can override those global grammars while a recording is in progress. This means that something the caller says does not accidentally throw an event and prematurely end the recording.
- **DTMF Terminate**: To allow callers to end the recording by pressing a touch tone key on the key pad, set this to **true**. If this property is set to **true**, the system recognizes any DTMF key press that is not part of a local DTMF grammar as a signal to end the recording.

  To cause the system to ignore DTMF key presses that are not specified in a local DTMF grammar, set this to **false**.

  Note:

  If you set this property to **true**, the system treats it, essentially, as a local grammar. This setting overrides any other local DTMF grammars that is active.
- **Bind Name to Node**: (true/false). When true, the name of the variable item is bound to the name of the node. When the parent node is renamed, the name of the variable item is renamed as well; keeping the name of the variable in sync with the node name.
- **Audio Type**: Select one of the following formats:

Audio format	Description
audio/x-wav	WAV format: This format is a WAV (RIFF header) 8kHz 8-bit mono mu-law or A-law [PCM] single channel format.
audio/x-alaw-basic	G.711-compliant format: This format is a raw (headerless) 8kHz 8-bit mono A-law [PCM] single channel format.
audio/basic	G.711-compliant format: This format is a raw (headerless) 8kHz 8-bit mono mu-law [PCM] single channel format.
audio/L16;rate=8000	When the platform is Avaya Experience Portal, then Orchestration Designer uses **audio/L16;rate=8000** on record, and **audio/x-wav** for simulation.

Playing a recorded audio file

About this task

You can play an audio file of the recording back later in the application.

Procedure

1. Create a prompt in which to play the audio file.
2. Add a text variable in the prompt.

   For more information about text variables, see Text Variable on page 840.
3. In the **Variable** field for the text variable, select the **Record** item variable.

Note:

The **Record** item variable has the same name as the **Record** item.

4. In the **Variable Field** field for the text variable, select **value**.
5. In the **Format** field for the text variable, select **url**.

Region

Type

Media item

Available from

Within a media page under the head node.

Purpose

The region element controls the position, size and scaling of media object elements.

Behavior

Regions are bound to the media page after they are added to the head node.

Elements such as media in the body of a media page can reference the region to further define their physical properties. Changes to a region such as renaming, deleting, adding have a dynamic effect on those elements.

Properties

- **Name**: Name of the region.
- **Background Color**: CSS2 system color.
- **Height**: Height of the region.
- **Width**: Width of the region.
- **Unit for Height, Width**: Unit of height and width of the region in pixel unit (px) or percentage (%). The default is px.
- **Left**: Left dimension of bound box.
- **Right**: Right dimension of bound box.
- **Top**: Top dimension of bound box.
- **Bottom**: Bottom dimension of bound box.
- **Unit for Top, Bottom, Left, Right**: Unit of top, bottom, left, and right dimensions of bound box in pixel unit (px) or percentage (%). The default is px.
- **Fit**: Fit to box.
- **Z-Index**: Stacking order.
- **Show Background**: Option to show background when media content is not displayed.
- **Font Family**: List of fonts in a drop down box.

- **Font Family Custom**: Option to provide any font which is not listed in Font Family.
- **Font Size**: Size of the font.
- **Font Color**: Foreground color of the font.
- **Font Background Color**: Background color of the font.

Report 

Type

Tracking item

Available from

- Announce node on page 634
- Prompt and Collect node on page 659
- Collect node on page 651
- Blind Transfer node on page 646
- Bridged Transfer node on page 647
- Record node on page 660
- Disconnect node on page 654
- Menu node on page 656
- Form node on page 655
- Data node on page 653
- Tracking node on page 669

Purpose

The primary purpose of the **Report** item is to track and submit application data that you can use to analyze and evaluate application performance in the run-time environment. You can use the **Report** item to:

- Define the type of data the item is reporting.
- Declare the severity of alarm that is associated with each item.
- Give the reason the item is being reported.
- Record the value of variables or variable fields at that point in the application.

Behavior

During simulation or run time, the **Report** item takes the value of the various fields and properties and writes these values to a log file. The application locates this log file in the `ProjectName/data/log` directory, where *ProjectName* represents the name of the speech or message application project.

Note:

If you are using an Avaya Experience Portal system, you can view an automatically generated Application Report of this log data. See “Application Report” in your Avaya Experience Portal documentation.

Tip:

The name of the log file for the current day is always `report.log`. Each time you run a simulation, Orchestration Designer appends the log data to the end of this file. After the end of each day on which data has been written to the log file, Orchestration Designer renames the log file to: `report.log.`*`yyyy-mm-dd`*, where *yyyy-mm-dd* represents the date on which the log file was compiled.

Report log data is written using *log4j* logging technology. Because this is new technology and available information constantly changes, for more information about log4j technology, Avaya recommends that you perform an Internet search with the search term “log4j”.

Orchestration Designer applications are set at deployment whether they are to operate in an Avaya Interactive Response (IR) or an Avaya Experience Portal environment.

- If in an IR environment, the **Report** item data is written to a log file on the application server. This makes it easier to use standard reporting or database software to retrieve the information for reports which you can analyze more easily.
- If in an Avaya Experience Portal environment, the **Report** item data is written to an SQL database on the Avaya Experience Portal Management System (EPM) server. The Avaya Experience Portal Management System administration tool provides an option to generate and view an Application Report that uses this data. See “Application Report” in your Avaya Experience Portal documentation library.

Because the **Report** item is primarily a reporting tool for analysis and evaluation of applications, there is no need to enable or disable it. When you use it in an application, it is always enabled.

Properties

- **Type**: Select the type you want to assign to this Report item.

  Orchestration Designer writes the type you select to the log file as part of the report data. This makes it possible to search the log file or database for specific types with which to perform an analysis.

  For example, if you want to analyze the success rate of a car rental application in getting customers to reserve a rental car over the telephone, you can drag a **Report** item at the start of the actual transaction part of the call flow and select the type as **Start**.

  Then, at the end of the rental transaction, when the customer has verified and approved the reservation, you can drag another **Report** item and select **End** for the type.

  Later, after the application has been in service for some time, you can retrieve the report data and compile it into a report that compares the number of transactions that were started versus the number of transactions that were successfully completed.

  Note:

  For this example to work, you must use the same name in the **Name** field for both the **Start Report** item and the **End Report** item.

  The following table lists and describes the types available.

Type	Use this alarm descriptor to indicate:
Start	The beginning of a section of the call flow or the message flow on which you want to report data. To be used effectively, the **Name** field for this **Report** item must exactly match the **Name** field of a **Report** item of type **End** that comes later in the call flow.
End	The end of a section of the call flow or the message flow on which you want to report data. To be used effectively, the **Name** field for this **Report** item must exactly match the **Name** field of a **Report** item of type **Start** that occurs earlier in the call flow or the message flow.
Cancel	A cancellation of an **Report** item of type **Start** that occurs earlier in the call flow or the message flow. To be used effectively, the **Name** field for this **Report** item must exactly match the **Name** field of a **Report** item of type **Start** that precedes it in the call flow or the message flow.
Cancel All	A cancellation of all **Report** items of type **Start** that occur earlier in the call flow or the message flow.
In Progress	Report data on a type already in progress. In other words, use this type somewhere between a **Start** type and an **End** type to collect data in the interim.

- **Level**: Select the level of alarm you want to generate with this **Report** item.

  Orchestration Designer writes the alarm level you select to the log file as part of the report data. This makes it possible to search the log file for specific problems based on the alarm level of the report entry. It also makes it possible to use other software to generate alarms on the system by reading and responding to the contents of the `report.log` file.

  Tip:

  In Avaya IR systems, to do this, you must write or acquire an application to monitor the log file and then generate a system alarm when certain key words are encountered. In Avaya Experience Portal systems, to do this you must write or acquire a similar application to monitor the database to which the data is being written.

  The following table lists the alarm levels for **Report** items in the `report.log` file, and provides a brief description of what each level means.

Alarm Level	Description
Fatal	A situation in which the application would crash.
Error	A situation that would cause the application to generate an error.
Warning	A situation that would cause the application to generate a warning.
Info	A situation that would cause the application to generate an informational message.

Tip:

On Avaya Experience Portal systems, the Application Report displays this setting for entries in the **Level** column.

- **Activity Name**: Type a name that you want to use to identify the **Report** item entry in the `report.log` file. For example, if you want to use the **Report** item to record the data related to a customer transaction to reserve a rental car, you can name this **Report** item as RentalTrans.

  You can also select an activity name that you specified previously.

  Tip:

  On Avaya Experience Portal systems, the Application Report displays this setting for entries in the **Activity** column.
- **Message**: Enter in this field text describing what the purpose of this Report item is at this point in the call flow.

  For example, if you are using the Report item at the beginning of a car rental transaction section of the code, you can have a **Type** of **Start**, a **Name** of **RentalTrans**, and in this field you can then enter: **Customer starts the reservation transaction**.

  The report item writes this string to the log file or database exactly as you enter it here.

  Tip:

  On Avaya Experience Portal systems, the Application Report displays this setting for entries in the **Message** column.
- **Variable**: Select the variable whose value you want to write as an entry in the log file or database.

  If this variable is a complex variable, you must also select a **Variable Field**.

  Tip:

  On Avaya Experience Portal systems, the Application Report does not display this setting, but the value of this variable can be viewed in the Message Details page for the Application Report message. For more details on Application Report, see “Reports” chapter of *Administering Avaya Experience Portal guide*.
- **Variable Field**: If the Variable you selected is a complex variable, select the variable whose value you want to write as an entry in the log file or database.

  Tip:

  On Avaya Experience Portal systems, the Application Report does not display this setting, but the value of this variable can be viewed in the Message Details page for the Application Report message. See “Application Report” in the Avaya Experience Portal documentation library.

Report Alarm

Type

Tracking item

Available from

- Announce node on page 634
- Prompt and Collect node on page 659
- Collect node on page 651
- Blind Transfer node on page 646
- Bridged Transfer node on page 647
- Record node on page 660
- Disconnect node on page 654
- Menu node on page 656
- Form node on page 655
- Data node on page 653
- Tracking node on page 669

Purpose

The Report Alarm item allows you to issue an alarm to the Avaya Experience Portal. It is a mechanism to notify an administrator when error conditions are detected by the application.

Behavior

The Report Alarm item invokes the Avaya Experience Portal logAlarm function to allow you to generate alarms from the application. Avaya Experience Portal implements a filter mechanism to protect against alarm overloads.

Avaya Experience Portal ensures that only one alarm is generated by an application within a certain amount of time. If an application reports alarms with escalating severities, Avaya Experience Portal issues a new alarm. For more information on Avaya Experience Portal events and associated alarms, see Avaya Experience Portal documentation.

Properties

- **Level**: Select the severity of the alarm. The allowed values are **Minor**, **Major**, and **Critical**.
- **Message**: Enter a message to report with the alarm.
- **Variable**: Select the variable you want to report with the alarm. If the variable you select is a complex variable, you must also select a **Variable Field**.
- **Variable Field**: Select the variable field you want to use.

The contents of this list depend on which variable you selected in the Variable property. Only the fields associated with the selected variable are presented in this list.

Note:

If the variable you selected in the **Variable Field** is a simple variable, this field is inactive.

- **Log Alarm**: (True/False). If true, the message and variable are logged to the Application report log in addition to the alarm. If false, only an alarm is issued.

Return Event

Type

Event handler

Available from

- **AppRoot** node
- Form node on page 655

Purpose

Returns an event to the calling application. Can be used to propagate an event caught in a module to the calling application, or can be used to throw a different event back to the calling application.

Behavior

Following is an example for when to use the Return Event item, in collaboration with the Exit item.

- Catch (event="myEvent"), Return Event. In this scenario, when the "myEvent" event is caught, the module will exit, returning control back to the calling application, and it will return the "myEvent" that was caught.

  In other words, it propagates or re-throws the event to let the calling application handle the event.

- Catch (event="error.runtime"), Return Event, Throw (event="myReallyBadEvent", message="Something happened..."). In this scenario, the application catches an "error.runtime" event and the module will exit returning control back to the calling application.

  Instead of propagating the "error.runtime" event, however, it returns "myReallyBadEvent" instead, with the message "Something happened...". The calling application can catch the "myReallyBadEvent" event, but does not know that the cause of the event was an "error.runtime" event.

- Catch ("error.badfetch"), Exit. In this case, the application catches an "error.badfetch" event and then exits the application. Exiting the application will terminate the session.

  It does not matter if it is a module or a stand-alone application, the application will end. This is typically used when there is some fatal error with the application or system.

Properties

- **Threshold**: A variable that must have a positive integer value.

  This field is available only when the Throw item is dragged into an event handler or a node. This field is not available when the Throw item is dragged into a Link item.

  This property is used to disable the Throw item until this value is met. For example, if the Threshold property is set to 3, the system ignores the Throw item the first two times the specified event is encountered. Only the third time that the event is encountered does the system actually throw the event.

Say As

Type

SSML item

Available from

Any level tab in the prompt file editor. For more information about the prompt file editor, see Prompt file editor in a speech application on page 207.

Purpose

At times, the interpretation of a particular word, phrase, or number depends on the context and the intention for how the word, phrase, or number is to be used. The **Say As** property makes it possible to specify the way in which a designated prompt segment is to be spoken.

For example, the text “Dr.” can be interpreted a couple different ways, depending on the context. When used before a person’s name, it is interpreted as “Doctor.” When used at the end of a street name, it is interpreted as “Drive.” To specify to the speech synthesis engine how this is to be interpreted and spoken, use a setting of **name** for the first case and a setting of **address** for the second case.

To give another example, a seven-digit number, 5551234, can have a variety of interpretations, depending what the number represents in context. This number can represent a telephone number, an amount of money, or a simple number:

- If it is to represent a telephone number, you would select **telephone**. In this case, the speech synthesis engine can speak it as “five five five (pause) one two three four.”
- If it is to represent an amount of money, you would select **currency**. In this case, in U.S. currency, the speech synthesis engine can speak it as “five million, five hundred and fifty-one thousand, two hundred and thirty-four dollars.”
- If it is to represent a simple number, you would select **number**. In this case, the speech synthesis engine can speak it as “five million, five hundred fifty-one thousand, two hundred thirty-four.”

Behavior

The exact behavior of the setting for the **Say As** item depends on a number of factors.

One factor is the language in which the application is developed. A currency amount, for instance, would be rendered as dollars with the U.S. English language as the default application language. If the application language, however, is U.K. English, the same currency amount would be rendered as pounds.

The default application language also affects which **Interpret-As** formats are supported. Different languages require and support different formats.

Probably the biggest factor that affects the behavior of the **Say As** item is the **Interpret-As** formats that the speech synthesis engine supports. For example, if you use an **Interpret-As** format of **net**, but the speech synthesis engine does not support that format, the system ignores the **Say As** setting. For the **Interpret-As** formats supported by your speech synthesis engine, see the documentation for your speech synthesis engine.

Properties

- **Interpret-As**: Select the format the system is to use to render the text.

Send Email 

Type

Notification connector item

Available from

The **Send Email** item is available in the Palette pane of a Data node in a speech application and in a message application only after you enable the **Notification Connector (Email, SMS)** pluggable data connector.

For more information, see Enabling Orchestration Designer project pluggable data connectors on page 503.

Purpose

Use the **Send Email** item in a speech application or in a message application that is created for a channel other than the email channel to send an outbound email message also called as a notification.

For example, you receive an inbound call from a customer requesting the account statement of the checking account. You can use the **Send Email** item in the speech application to send an email message notification that provides the account statement to the customer.

The system uses the platform Web service to further process and deliver the email message that is contained within the **Send Email** item.

You can also use the **Send Email** item in an email channel message application. The difference between using an **Email** item and a **Send Email** item in an email channel message application is that the **Send Email** item is specifically designed to send an outbound email message from channels other than the email channel. Hence, when you use the **Send Email** item, you must specify the information such as the email address of the sender and receiver of the email message.

For example, you want to send an email notification from a speech application. The system stores the data of the voice call. The system does not have the data related to the email message such as the email address of the receiver to whom the system must send the email message notification. Hence, you must specify this information in the Properties view of the **Send Email** item.

When you use the **Email** item in an email channel message application, the system sends an outbound email message response by using the data, such as the email address of the sender and receiver of the email message, contained in the inbound email message. For more information, see Email on page 725.

You can also use the **Send Email** item in an email channel message application when you want to use the platform Web service to further process and deliver the email message.

Behavior

The **Send Email** item sends an outbound email message, also called as notification, from a speech application or a message application.

Properties

- **Email Constant**: Click the email file that contains the email message that you want to send. The **Email Constant** field is unavailable if you select a variable in the **Email Variable** field.
- **Email Variable**: Click the variable that contains the email message that you want to send. If you select a complex variable in the **Email Variable** field, then you must select a complex variable field in the **Email Variable Field** field. The **Email Variable** field is unavailable if you select an email file in the **Email Constant** field.
- **Email Variable Field**: If you select a complex variable in the **Email Variable** field, then in the **Email Variable Field** field, select the complex variable field that contains the email message that you want to send. The **Email Variable Field** field is unavailable if you select a simple variable in the **Email Variable** field.
- **To Variable**: Click the variable that contains the email address of the receiver to whom you want to send the email message. If you select a complex variable in the **To Variable** field, then you must select a complex variable field in the **To Variable Field** field.
- **To Variable Field**: If you select a complex variable in the **To Variable** field, then in the **To Variable Field** field, select the complex variable field that contains the email address of the receiver to whom you want to send the email message. The **To Variable Field** field is unavailable if you select a simple variable in the **To Variable** field.
- **From Constant**: Type the email address of the sender of the email message. The **From Constant** field is unavailable if you select a variable in the **From Variable** field.
- **From Variable**: Click the variable that contains the email address of the sender of the email message. If you select a complex variable in the **From Variable** field, then you must select a complex variable field in the **From Variable Field** field. The **From Variable** field is unavailable if you type an email address in the **From Constant** field.
- **From Variable Field**: If you select a complex variable in the **From Variable** field, then in the **From Variable Field** field, select the complex variable field that contains the email address of the sender of the email message. The **From Variable Field** field is unavailable if you select a simple variable in the **From Variable** field.
- **Cc Constant**: Type the email address of the receiver to whom you want to send a copy of the email message. The **Cc Constant** field is unavailable if you select a variable in the **Cc Variable** field. Use a semicolon (;) to separate multiple email addresses.
- **Cc Variable**: Click the variable that contains the email address of the receiver to whom you want to send a copy of the email message. If you select a complex variable in the **Cc Variable** field, then you must select a complex variable field in the **Cc Variable Field** field. The **Cc Variable** field is unavailable if you type an email address in the **Cc Constant** field.
- **Cc Variable Field**: If you select a complex variable in the **Cc Variable** field, then in the **Cc Variable Field** field, select the complex variable field that contains the email address of the receiver to whom you want to send a copy of the email message. The **Cc Variable Field** field is unavailable if you select a simple variable in the **Cc Variable** field.
- **Bcc Constant**: Type the email address of the receiver to whom you want to send a blind copy of the email message. The **Bcc Constant** field is unavailable if you select a variable in the **Bcc Variable** field. Use a semicolon (;) to separate multiple email addresses.

- **Bcc Variable**: Click the variable that contains the email address of the receiver to whom you want to send a blind copy of the email message. If you select a complex variable in the **Bcc Variable** field, then you must select a complex variable field in the **Bcc Variable Field** field. The **Bcc Variable** field is unavailable if you type an email address in the **Bcc Constant** field.
- **Bcc Variable Field**: If you select a complex variable in the **Bcc Variable** field, then in the **Bcc Variable Field** field, select the complex variable field that contains the email address of the receiver to whom you want to send a blind copy of the email message. The **Bcc Variable Field** field is unavailable if you select a simple variable in the **Bcc Variable** field.
- **Subject Constant**: Type a subject line for the email message. The **Subject Constant** field is unavailable if you select a variable in the **Subject Variable** field.
- **Subject Variable**: Click the variable that contains the subject line that you want to use in the email message. If you select a complex variable in the **Subject Variable** field, then you must select a complex variable field in the **Subject Variable Field** field. The **Subject Variable** field is unavailable if you type a subject line in the **Subject Constant** field.
- **Subject Variable Field**: If you select a complex variable in the **Subject Variable** field, then in the **Subject Variable Field** field, select the complex variable field that contains the subject line that you want to use in the email message. The **Subject Variable Field** field is unavailable if you select a simple variable in the **Subject Variable** field.
- **Reply to Constant**: Type the email address of the receiver to whom you want to send the email message.
- **Reply to Variable**: Click the variable that contains the email address of the receiver who receives the reply if the recipient replies to your email. That is, if you specify an email address in the **Reply To Variable** field, the system sends the reply to the email address specified in the **Reply To Variable** field instead of the email address in the From field. If you select a complex variable in the **Reply to Variable** field, then you must select a complex variable field in the **Reply to Variable Field** field.
- **Reply to Variable Field**: If you select a complex variable in the **Reply to Variable** field, then in the **Reply to Variable Field** field, select the complex variable field that contains the email address of the receiver of the email message.
- **Display Name Constant**: Type the name of the sender of the email message. This name is displayed in the From field instead of the senders email address in the email.
- **Display Name Variable**: Select the variable that contains the name of the sender of the email message. If you select a complex variable in the **Display Name Variable** field, then you must select a complex variable field in the **Display Name Variable Field** field.
- **Display Name Variable Field**: : If you select a complex variable in the **Display Name Variable** field, then in the **Display Name Variable Field** field, select the complex variable field that contains the name of the sender of the email message.
- **Delivery Notification**: Select when you want to receive the notification about the delivery status of the email. The options are: **Never**, **Success**, **Failure**, **Delay**. The default is **Never**.
- **Priority**: The priority that you want to assign to the email message contained in the **Send Email** item. The options are: **None**, **Normal**, **Urgent**, **Very Urgent**, **Emergency**. The default is **None**. The system uses the priority that you set to determine the order in which the system must deliver the email message notification to the destination.

Send SMS

Type

Notification connector item

Available from

The **Send SMS** item is available in the Palette pane of a Data node in a speech application and in a message application only after you enable the **Notification Connector (Email, SMS)** pluggable data connector.

For more information, see Enabling Orchestration Designer project pluggable data connectors on page 503.

Purpose

Use the **Send SMS** item in a speech application or in a message application that is created for a channel other than the SMS channel to send an outbound SMS message also called as a notification.

For example, you receive an inbound call from a customer requesting the phone number of a car rental service. You can use the **Send SMS** item in the speech application to send an SMS message notification that provides the phone number of the car rental service to the customer.

The system uses the platform Web service to further process and deliver the SMS message that is contained within the **Send SMS** item.

You can also use the **Send SMS** item in an SMS channel message application. The difference between using an **SMS** item and a **Send SMS** item in an SMS channel message application is that the **Send SMS** item is specifically designed to send an outbound SMS message from channels other than the SMS channel. Hence, when you use the **Send SMS** item, you must specify the information such as the phone number of the sender and receiver of the SMS message.

For example, you want to send an SMS notification from a speech application. The system stores the data of the voice call. The system does not have the data related to the SMS message such as the phone number of the receiver to whom the system must send the SMS message notification. Hence, you must specify this information in the Properties view of the **Send SMS** item.

When you use the **SMS** item in an SMS channel message application, the system sends an outbound SMS message response by using the data, such as the phone number of the sender and receiver of the SMS message, from the inbound SMS message. For more information, see SMS on page 833.

The **Send SMS** item also provides additional options such as specifying data that you want to pass in the SMS message headers.

You can also use the **Send SMS** item in an SMS channel message application when you want to use the:

- Platform Web service to further process and deliver the SMS message.
- Features of the **Send SMS** item. For example, set the priority to send the outbound SMS message.

Behavior

The **Send SMS** item sends an outbound SMS message, also called as a notification, from a speech application or a message application.

Properties

- **SMS Constant**: Click the SMS file that contains the SMS message that you want to send. The **SMS Constant** field is unavailable if you select a variable in the **SMS Variable** field.
- **SMS Variable**: Click the variable that contains the SMS message that you want to send. If you select a complex variable in the **SMS Variable** field, then you must select a complex variable field in the **SMS Variable Field** field. The **SMS Variable** field is unavailable if you select an SMS file in the **SMS Constant** field.
- **SMS Variable Field**: If you select a complex variable in the **SMS Variable** field, then in the **SMS Variable Field** field, select the complex variable field that contains the SMS message that you want to send. The **SMS Variable Field** field is unavailable if you select a simple variable in the **SMS Variable** field.
- **To Variable**: Click the variable that contains the phone number of the receiver to whom you want to send the SMS message. If you select a complex variable in the **To Variable** field, then you must select a complex variable field in the **To Variable Field** field.
- **To Variable Field**: If you select a complex variable in the **To Variable** field, then in the **To Variable Field** field, select the complex variable field that contains the phone number of the receiver to whom you want to send the SMS message. The **To Variable Field** field is unavailable if you select a simple variable in the **To Variable** field.
- **From Constant**: Type the phone number of the sender of the SMS message. The **From Constant** field is unavailable if you select a variable in the **From Variable** field.
- **From Variable**: Click the variable that contains the phone number of the sender of the SMS message. If you select a complex variable in the **From Variable** field, then you must select a complex variable field in the **From Variable Field** field. The **From Variable** field is unavailable if you type a phone number in the **From Constant** field.
- **From Variable Field**: If you select a complex variable in the **From Variable** field, then in the **From Variable Field** field, select the complex variable field that contains the phone number of the sender of the SMS message. The **From Variable Field** field is unavailable if you select a simple variable in the **From Variable** field.
- **Return ID Variable**: The variable in which you want to store the return ID that the platform Web service returns.

  After the platform Web service receives the outbound SMS message notification that the Orchestration Designer application sends, the platform Web service returns the return ID to the Orchestration Designer application.

  If you select a complex variable in the **Return ID Variable** field, then in the **Return ID Field** field, select a complex variable field.
- **Return ID Field**: If you select a complex variable in the **Return ID Variable** field, then in the **Return ID Field** field, select the complex variable field in which you want to store the return ID that the platform Web service returns.
- **Request Receipt**: The option to specify whether Short Message Service Center (SMSC) must send a delivery status receipt to the SMS channel message application that you specify

for receiving the delivery status receipt on EPMS. SMSC sends the delivery status receipt to the SMS channel message application after delivering the SMS message to the destination.

The options are:

- **true**: If SMSC must send the delivery status receipts.
- **false**: If SMSC must not send the delivery status receipts.

Note:

Ensure that on EPMS, you enable the feature to receive notification messages and delivery status receipts for the outbound SMS messages from SMSC. For more information, see the Avaya Experience Portal documentation.

Tip:

To process the delivery status receipts of SMS notifications that are sent by using the **Send SMS** item, create a separate SMS channel message application.

The telephone number that you configure on the Avaya Experience Portal platform to receive the delivery status receipt messages must be the same as that you specify in the **From Constant**, **From Variable**, or **From Variable Field** field of the **Send SMS** item.

- **Priority**: The priority that you want to assign to the SMS message contained in the **Send SMS** item.

  The options are: **Normal**, **Urgent**, **Very Urgent**, and **Emergency**.

  SMSC uses the priority that you set to determine the order in which SMSC must deliver the SMS message notification to the destination.

- **Validity Period**: The maximum time within which SMSC must deliver the SMS message notification to the destination.

  If SMSC is unable to deliver the SMS message notification within the maximum time, then after the maximum time is reached, SMSC ignores the SMS message notification and does not deliver the SMS message notification.

- **User Reference Data Constant**: The constant data that you want to send in the SMS message notification header.

  The **User Reference Data Constant** field is unavailable if you select a variable in the **User Reference Data Variable** field.

- **User Reference Data Variable**: The variable that contains the data that you want to send in the SMS message notification header.

  If you select a complex variable in the **User Reference Data Variable** field, then in the **User Reference Data Variable Field** field, you must select a complex variable field.

  The **User Reference Data Variable** field is unavailable if you specify constant data in the **User Reference Data Constant** field.

- **User Reference Data Variable Field**: If you select a complex variable in the **User Reference Data Variable** field, then in the **User Reference Data Variable Field** field, you must select the complex variable field that contains the data that you want to send in the SMS message notification header.

Send to AACC

Type

Sends an inbound message to Avaya Aura® Contact Center for assisted care

Available from

The **Send to AACC** item is available in the Palette pane of a Data node editor in a message application only after you enable the **Send AACC Connector** pluggable data connector.

For more information, see Enabling Orchestration Designer project pluggable data connectors on page 503.

Purpose

Use the **Send to AACC** item to forward an inbound SMS or email message to Avaya Aura® Contact Center for agent assistance in case the message application is unable to interpret the inbound message due to reason such as the inbound message did not match the grammars that are configured in the message application.

For example, you have a **Collect** node in the message flow that collects the inbound message. You can set a **Data** node as the next node to execute after the **Collect** node. You can drag the **Send to AACC** item in the Data node editor and assign an AACC message file to the **Send to AACC** item. You can configure the **Collect** node such that if the message application is unable to interpret the inbound message and throws a nomatch event, the message application executes the **Data** node. The message application then forwards the inbound message along with the information contained in the AACC message file to Avaya Aura® Contact Center for agent assistance.

Behavior

The **Send to AACC** item forwards an inbound SMS or email message to Avaya Aura® Contact Center for agent assistance in case the message application is unable to interpret the inbound message.

Properties

- **Name**: The AACC message file that contains the additional information that you want to send to Avaya Aura® Contact Center along with the inbound message for agent assistance.

  For more information, see AACC message file on page 365 and Forward an inbound SMS or email message to Avaya Aura Contact Center on page 364.

Set OD Date/Time

Type

Datetime function

Available from

Data node

Purpose

You can use the **Set OD Date/Time** function to split the input date and time in a Java object, for example, year, month, day of month, day of week, hour, minute, and second, and populate the OD date and time variables.

Behavior

The **Set OD Date/Time** function splits the input date and time in a Java object and populates the date and time values in the OD date and time field variables. The OD date variable must contain at least year, month, and day of month fields. The OD time variable must contain at least hour, minute, and second fields. You can set either one or both the variables in the properties.

If the input date and time consist of time zone values, the OD date and time variable fields are populated provided the fields are already defined in the OD date and time variables.

Properties

- **Function**: Shows the name of the datetime function. If you want to use some other datetime function, select that function from the list.
- **To OD Date Variable**: Select the OD date variable in which you want to populate the date values from the Java date object. The OD date variable is a complex variable that contains at least year, month, and day of month fields.
- **To OD Time Variable**: Select the OD time variable in which you want to populate the time values from the Java date object. The OD time variable is complex variable that contains at least hour, minute, and second fields.

For more information about the supported system variable fields, see System variable fields and properties on page 563.

- **From Variable**: Select the Java date object variable from which you want to extract and split the date and time values and populate the OD date and time variables.
- **From Variable Field**: If you select a complex variable in the **From Variable** field, then select a complex variable field from which you want to extract and split the date and time values and populate the OD date and time variables.

Example usage of the Set OD Date/Time function

About this task

To verify the day of week of a particular date in a Java date object that is contained in the MyDateObjVar variable, and store the date value in the MyODDateVar project variable, perform the following actions:

1. Create the MyODDateVar complex variable with the year, month, day, and dayofweek fields. For more information, see Creating a variable on page 231.
2. From the **Palette** pane, drag **Set OD Date/Time** into the data node editor.
3. In the Properties view, set the following properties:
   a. In the `To OD Date Variable` field, click **MyODDateVar**.
   b. In the **From Variable** field, click **MyDateObjVar**.

To verify the hour and minute of a particular time in a Java date object that is contained in the **MyDateObjVar** variable, and store the time value in the **MyODTimeVar** project variable, perform the following actions:

1. Create the **MyODTimeVar** complex variable with the hour, minute, and second fields. For more information, see Creating a variable on page 231.
2. From the **Palette** pane, drag **Set OD Date/Time** into the data node editor.
3. In the Properties view, set the following properties:
   a. In the **To OD Time Variable** field, click **MyODTimeVar**.
   b. In the **From Variable** field, click **MyODTimeVar**.

Set VDU Fields

Type

IC Connector item

Available From

Data node on page 653 (IC section)

Purpose

Set VDU Fields allows you to set up to 20 vdu variables in one round trip to the VOX.

Behavior

You must first set all the **vdu_cache** values that you want to send. When this node is invoked, the fields indicated will be pulled from the cache and sent over to the VOX in one round trip.

Properties

- **vdu_field1 - vdu_field20**: Indicates the name of the vdu fields to set. You must first add the field to the vdu complex variable, then set the value of the corresponding fields in the **vdu_cache**.

Set Web Error

Available from

Data node

Purpose

You can use the **Set Web Error** item to set a user error and display it in the web page that follows in the flow execution.

Behavior

When you drag and drop the **Set Web Error** item in the editor, by default, it clears the error message. You can set the error message by configuring the properties of the **Set Web Error** item. The error message gets displayed in the next **Form node** in the flow execution.

Properties

- **Error Name**: Enter a unique name that tells what the error is about.
- **Error Message**: Enter the message that you want to display for the error.
- **Localized Message(Textset ID)**: Enter the **Textset ID** to display the message in different languages.

Sequence

Type

Media item

Available from

Body, Parallel, Sequence

Purpose

The sequence container defines a simple time grouping in which multiple elements can play in sequence.

Behavior

By default, elements within a sequence container play one after another. However, you can set various properties, such as timing offsets, duration, and repeats, on the media itself to control the play back. You can also nest parallel and sequence elements under this container.

Properties

- **Fill**: Determines action to take after active duration. The Fill option can have one of the following values:

  Note:

  If you do not select any value, the Fill behavior defaults to “auto”.

  - **remove**: Specifies that the element will not extend past the end of the last instance of the simple duration.
  - **freeze**: Specifies that the element will extend past the end of the last instance of the simple duration by “freezing” the element state at that point. The parent time container of the element determines how long the element is frozen (as described immediately below).
  - **hold**: Setting this to “hold” has the same effect as setting to “freeze”, except that the element is always frozen to extend to the end of the simple duration of the parent time container of the element (independent of the type of time container). For profiles that support a layered layout model (e.g., SMIL 2.0 Language Profile), held elements (elements with fill=“hold”) will refresh their display area when a layer is added on top then later removed.
  - **transition**: Setting this to “transition” has the same effect as setting to “freeze”, except that the element is removed at the end of the transition. This value is only allowed on elements with media directly associated with them. If specified on any other element (e.g. a time container element in the SMIL language profile), the attribute is ignored. See the SMIL Transitions module.

- **auto**: The fill behavior for this element depends on whether the element specifies any of the attributes that define the simple or active duration. If none of the attributes dur, end, repeatCount or repeatDur are specified on the element, then the element will have a fill behavior identical to that if it were specified as “freeze”. Otherwise, the element will have a fill behavior identical to that if it were specified as “remove”.

Simple Variable

Type

Variable item

Available from

Variable Editor only (see Variable management on page 230)

Purpose

The **Simple Variable** item creates a variable that can contain only one value. This variable can be used as any of the defined variable types. For more information about creating and using simple variables, see Variables on page 226.

Behavior

When you create a simple variable in the Variable Editor, Orchestration Designer automatically assigns it a default name. You can rename the variable. You can also assign it a default value.

After you create the variable, you must save the application before the variable becomes available for use in nodes or node sub-items.

Properties

- **Name**: Enter in this field the name you want to identify the variable.

  Note:

  This name must follow conventions for naming Java components as described in Conventions for naming Java components on page 59.

- **Value**: Enter in this field the default value you want to assign to the variable, if any. If you leave this field blank, the default value is undefined.
- **Private**: (True/False). When you set the **Private** attribute to **true**, the value of this variable is not traced or logged in the framework reporting. However, the value of a private variable is visible only if you add an application trace or application report. If you use a private variable as in/out params for VXML subdialogs, the variable value is encrypted. For example, <var name=“untitledOutput1” expr=“'___AvayaEncrypted_dJklKMkrNmg='”/>. The variable value is unencryted on submission. If your application returns a private variable on exit, the value is not encrypted. However, the value is encrypted on a subdialog return. Private values that are passed to an OSDM are not encrypted. If a prompt contains a private variable, the variable value does not appear in the VXML trace because the page is not written to the trace.log.

SMIL Link

Type

Media item

Available from

Root level of the prompt

Purpose

A link to an existing SMIL file.

Behavior

You can choose to link to either an external file through a URL or you can drag .SMIL files within your <app>/data/<language>/prompts directory and add them as part of the application. The local property searches this directory for .SMIL files.

Properties

- **SMIL Link Variable**: Name of variable containing the link.
- **SMIL Link Field**: Name of field in complex variable containing the link.
- **Local**: SMIL file contained in `<app>/data/<language>/prompts`.
- **External URL**: Direct path to SMIL.

SMS

Type

Prompt Editor Palette item

Available from

Prompt Contents tab of the Prompt File Editor in an SMS channel message application.

For more information about the prompt file editor, see Prompt file editor in a message application on page 207.

Purpose

Use the **SMS** item in an SMS channel message application to send an outbound SMS message.

For example, you want to send an SMS message to your customer for an upcoming offer from an SMS channel message application.

You can specify the SMS message that you want to send in an SMS file. Create a prompt file and add the **SMS** item to the prompt file. Assign the SMS file to the **SMS** item. You can then add the **Prompt** item to the **Announce** node from where you want to send the SMS message and assign the prompt file to the **Prompt** item.

When the system executes the **Announce** node, the system sends the SMS message to the text browser that is located on Avaya Experience Portal Media Processing Platform (MPP) for further processing and delivery.

Behavior

The **SMS** item sends an outbound SMS message from an SMS channel message application.

Properties

- **SMS Constant**: Click the SMS file that contains the SMS message that you want to send. The **SMS Constant** field is unavailable if you select a variable in the **SMS Variable** field.
- **SMS Variable**: Click the variable that contains the SMS message that you want to send. If you select a complex variable in the **SMS Variable** field, then you must select a complex variable field in the **SMS Variable Field** field. The **SMS Variable** field is unavailable if you select an SMS file in the **SMS Constant** field.
- **SMS Variable Field**: If you select a complex variable in the **SMS Variable** field, then in the **SMS Variable Field** field, select the complex variable field that contains the SMS message that you want to send. The **SMS Variable Field** field is unavailable if you select a simple variable in the **SMS Variable** field.

Supervised Transfer

See Consultation Transfer on page 699.

Set UUI for Conference Disconnect (AES)

Type

AES connector item

Available from

Data node on page 653

Purpose

Sets UUI data on a conference call already established.

Behavior

Use the Set UUI for Conference Disconnect option during a conference to pass the UUI data on the call. Either the Orchestration Designer application or a party from a pre-existing conference can initiate this conference.

After you enter the UUI data, the Orchestration Designer application collects the UUI data, issues either an explicit AES `Disconnect` command or a `Remove Conference Party` command and then disconnects the application from the call, sending the UUI data out in the process.

You can change the UUI multiple times using the Set UUI for Conference Disconnect option before the disconnect.

Properties

- **Active Call Info Variable**: A variable that contains the call data for the party that you want to first set UUI and then disconnect from the conference.

- **UUI Variable**: The variable that contains the UUI data to be sent on the call.
- **UUI Variable Field**: The variable field that contains the UUI data if the UUI Variable is a complex variable.
- **UUI as ASCII**: Decides whether to pass the UUI as ASCII or binary.
  - **True**: Pass the UUI as ASCII.
  - **False**: Pass the UUI as binary.

Switch

Type

Condition item

Available from

Data node in speech and message applications

Prompt file editor in speech and message applications

Purpose

The switch statement is a selection control statement that enables a variable or expression to control the flow of application execution. It works in conjunction with Case on page 692 and Default on page 722 items to test conditions and execute the matching case.

Note:

- A switch statement must have at least one **Case** or **Default** statement as a child element.
- You must use a separate **Switch** statement for each condition.

Behavior

The **Switch** statement is similar to a series of **IF** statements for evaluating values of the same condition. Use **Switch** to compare the same variable with multiple values and perform an appropriate operation depending on the value it equals.

The **Switch** statement evaluates all **Case** statements sequentially until it finds a **Case** statement with a value that matches the value of the **Switch** expression. The **Switch** expression is a combination of the values of **Variable** and **Variable Field** properties.

Properties

- **Variable**: Select a variable you want to use to make the comparison.
- **Variable Field**: If you have selected a complex variable in the **Variable** field, select the variable you want to use to make the comparison.

Text

- Text item in a speech application on page 836

- Text item in an SMS channel message application on page 836
- Text item in an email channel message application on page 837

Text item in a speech application 

Type

Media item

Available from

Parallel

Sequence

Purpose

Text item is the text that resides within a text block. It defines the inline text that is played to you and configures playback properties for that text.

Behavior

Text item is included in media to simplify the process of adding text without creating it as a separate media object. You can couple the text element with a region to further define the look of the text as region defines the font characteristics.

Properties

- **Text**: Static text.
- **Text Variable/Field**: Variable/Field pointing to the media object. This field is mutually exclusive with Media property.

Text item in an SMS channel message application 

Type:

Prompt Segment item

Available from:

Prompt Contents tab of the prompt file editor in an SMS channel message application.

Purpose:

If you want to send a simple SMS message only from a single prompt of an SMS channel message application, then you can use the **Text** item instead of creating an SMS file. The **Text** item sends the text that you specify as an SMS message.

Behavior:

If you use the **Text** item in an SMS channel message application, then the **Text** item sends the text that you type in the **Text** property in the form of an SMS message.

The system sends the SMS message to the text browser that is located on Avaya Experience Portal Media Processing Platform (MPP) for further processing and delivery.

Note:

Avaya recommends that you create and use an SMS file to send an SMS message. You can use the SMS file editor to specify the SMS message, insert variables, and format the SMS message. You can also reuse the SMS file in multiple prompts in an application.

For more information, see SMS files on page 347.

Properties:

- **Text**: Type the SMS message that you want to send.

Text item in an email channel message application

Type:

Prompt Segment item

Available from:

Prompt Contents tab of the prompt file editor in an email channel message application.

Purpose:

If you want to send an email message only from a single prompt of an email channel message application, and if the email message does not require attachment files and HTML formatting, then you can use the **Text** item instead of creating an email file.

The **Text** item sends the text that you specify as an email message.

Behavior:

If you use the **Text** item in an email channel message application, then the **Text** item sends the text that you type in the **Text** property in the form of an email message.

The system sends the email message to the text browser that is located on Avaya Experience Portal Media Processing Platform (MPP) for further processing and delivery.

Note:

Avaya recommends that you create and use an email file to send an email message. You can use the email file editor to specify the email message, insert variables, format the email message, and add attachment files to the email message. You can also preview the email message and reuse the email file in multiple prompts in an application.

For more information, see Email files on page 351.

Properties:

- **Text**: Type the email message that you want to send.

HTML Text

Type:

Prompt Segment item

Available from:

Prompt Contents tab of the prompt file editor in a web application.

Purpose:

If you want to display style any static text and display on the form of a web application, then you can use the **HTML Text** item The **HTML Text** item renders the text to the browser in the style that you have defined inside it.

Behavior:

If you use the **HTML Text** item in a web application, then the **HTML Text** item picks the style and the text that you enter in the **Text** property of the **HTML Text** item. Alternatively, you can directly enter the style and the text that you want to display by double clicking the **HTML Text** item in the **Prompt Contents** tab of the prompt file editor. For example, you can put the following inside the **HTML Text** item:

```
<div style =”font-weight:bold”>Hello</div>
```

Properties:

- **Text**: Type the style and the text that you want to display on the screen.

Web Element

Type

Prompt Editor Palette item

Available from

Prompt Contents tab of the Prompt File Editor in a web application.

Purpose

Use the **Web Element** item in an web application to display an input value of different types.

For example, you want to display the name field on your web form.

You can add a label for name field in a textset file. Create a prompt file and add the **Web Element** item to the prompt file. Assign the name **Label** to the **Web Element** item. You can then add the **Prompt** item to the **Announce** node on the web form where you want to display the name and assign the prompt file to the **Prompt** item.

Behavior

The **Web Element** displays various input values on the form of a web application.

Properties

- **Variable**: Click the variable that contains the input that you want to display. If you select a complex variable in the **Variable** field, then you must select a complex variable field in the **Variable Field** property.
- **Variable Field**: If you select a complex variable in the **Variable** property, then in the **Variable Field** property, select the complex variable field that contains the input value that you want to display on the screen. The **Variable Field** property is unavailable if you select a simple variable in the **Variable** field.
- **Type**: Select the type of input field whose value you want to display. For example, Text, Picture, Map and so on.
- **Text ID (in Textset)**: Select the text ID that you have created in a `textset` file for the input field.

Text Block

Type

Media item

Available from

Parallel

Sequence

Purpose

A container for text.

Behavior

A text block can contain 1 to 'n' number of text items. These items are appended to form a series of text items. You can define the font and other properties for the contained text. You can also reference a region.

Note:

The Text Block font settings override any region font settings.

Properties

- **Region**: Name of the region added to the page to map media.
- **Begin**: Time offset in seconds to begin playing the media element.
- **End**: Time offset in seconds to end playing media element.
- **Duration**: Duration in seconds to play media element.
- **Repeat Duration**: Total seconds in duration to repeat media element.
- **Repeat Count**: Total number of repeat media elements.
- **Clip Begin**: Beginning offset in seconds from the start of the media element.
- **Clip End**: Ending offset in seconds from the start of the media element.
- **Fill**: Extends the appearance of the last frame of an element to fill gaps in a presentation.

  For example, a prompt has text and audio elements playing simultaneously and the text is of shorter length than the audio element, if the fill attribute is set to freeze, the last text frame continues to appear until the audio prompt is complete.
- **Font Family**: List of fonts in a drop down box.
- **Font Family Custom**: Option to provide any font which is not listed in Font Family.
- **Font Size**: Size of the font.
- **Font Color**: Foreground color of the font.
- **Font Background Color**: Background color of the font.
- **Text Mode**: This attribute is used for presentation of motion-based text. If you set the value of this attribute to **crawl**, the text moves as a single line as per the direction specified using the Text Writing Mode attribute.

- **Text Writing Mode**: This attribute sets the primary and secondary writing directions of text added to a region. You can set one of the following directions:
  - **lr**: (Default) Sets the primary writing direction of text from left-to-right and the secondary writing direction from top-to-bottom.
  - **rl**: Sets the primary writing direction of text from right-to-left and the secondary writing direction from top-to-bottom.

Text Variable

Text Variable item in a speech application

Type

Prompt Segment item

Available from

Any level tab in the prompt file editor. For more information about the prompt file editor, see Prompt file editor in a speech application on page 207.

Purpose

Text Variable item is used for:

- Rendering the value of a variable or variable field as a prompt segment using speech synthesis (TTS).
- Play a designated audio file as a prompt segment in the case of prerecorded audio files.

Behavior

When you drag a **Text Variable** item into a prompt, it takes the value of the designated variable (or variable field, if it is a complex variable). If the variable or variable field contains text, the text variable renders the text as spoken output using TTS.

Note:

Instead of using the text variable with a URL to an audio file, Avaya recommends that you use the Phrase Variable on page 794 type because it includes additional functionality for playing recorded audio files. The audio file must be of a supported format: G.711 or *.wav formats only.

For more information about how to use text variables effectively in applications, see Employ variables in a speech application using the prompt file editor on page 235.

Properties

- **Variable**: Select the variable you want to render. If the variable you select is a complex variable, you must also select a **Variable Field**.

- **Variable Field**: If the variable in the **Variable Field** is a complex variable, you must select the variable field you want to use.

  The contents of this list depend on which variable you selected in the **Variable** property: This list contains only the fields associated with the selected variable.

  Note:

  If you have selected a simple variable in the **Variable** field, this field is inactive.

- **Format**: Select the format you want to use for the variable. Use the following table to help you determine the correct format.

  Note:

  If your system supports SSML, Avaya recommends that you use the SSML Say As option, as it gives you greater control over the TTS output. For more information, see Say As on page 821.

Format	Description/Comments
text	Reads the contents of the variable or variable field as normal text. If the contents include numbers, it reads them as numbers. The exact interpretation of numbers depends on the TTS engine.
filename	Renders the contents of the file. If the file is: - A text file, the system renders it to audible speech using TTS. The text file must have a .txt extension. - An audio file, the system plays the file. The audio file must have a .wav extension. Note that these are the only two file formats supported for this option. Also the file must reside in the `/data/temp` directory for the application.
digits	Reads the contents of the variable or variable field as separate digits. For example, if the value of the variable is **H14295**, the system reads the value as “H one four two nine five.”
url	Retrieves the file specified in the URL and plays it back. If the file is: - A text file with a .txt extension, the system renders it to audible speech using TTS. - Any other file type, the system attempts to play the file as an audio file. This format requires that the value of the variable or variable field be a valid URL.

Text Variable item in an SMS channel message application

Type:

Prompt Segment item

Available from:

Prompt Contents tab of the prompt file editor in an SMS channel message application

Purpose:

If you want to send the contents of a variable in the form of an SMS message only from a single prompt of an SMS channel message application, then you can use the **Text Variable** item instead of creating an SMS file.

Behavior:

If you use the **Text Variable** item in an SMS channel message application, then the **Text Variable** item sends the contents of the variable that you specify in the form of an SMS message.

The system sends the SMS message to the text browser that is located on Avaya Experience Portal Media Processing Platform (MPP) for further processing and delivery.

Note:

Avaya recommends that you create and use an SMS file to send an SMS message. You can use the SMS file editor to specify the SMS message, insert variables, and format the SMS message. You can also reuse the SMS file in multiple prompts in an application.

For more information, see SMS files on page 347.

Properties:

- **Variable**: Select the variable. The system sends the contents of the variable that you select as an SMS message.
- **Variable Field**: If you select a complex variable in the **Variable** field, then in the **Variable Field** field, select the complex variable field. The system sends the contents of the variable field that you select as an SMS message.
- **Format**: Select the format that is relevant to the contents of the variable.

The following table shows the options that are available in the **Format** property:

Name	Description
text	If the variable that you specify contains text, then the **text** option sends the text in the form of an SMS message.
filename	If the variable that you specify contains the path of a file, then the **filename** option sends the contents of the file in the form of an SMS message.
url	If the variable that you specify contains a URL to a file, then the **url** option sends the contents of the file in the form of an SMS message.

Text Variable item in an email channel message application

Type:

Prompt Segment item

Available from:

Prompt Contents tab of the prompt file editor in an email channel message application

Purpose:

If you want to send the contents of a variable in the form of an email message only from a single prompt of an email channel message application, and if the email message does not require

attachment files and HTML formatting, then you can use the **Text Variable** item instead of creating an email file.

Behavior:

If you use the **Text Variable** item in an email channel message application, then the **Text Variable** item sends the contents of the variable that you specify in the form of an email message.

The system sends the email message to the text browser that is located on Avaya Experience Portal Media Processing Platform (MPP) for further processing and delivery.

Note:

Avaya recommends that you create and use an email file to send an email message. You can use the email file editor to specify the email message, insert variables, format the email message, and add attachment files to the email message. You can also preview the email message and reuse the email file in multiple prompts in the application.

For more information, see Email files on page 351.

Properties:

- **Variable**: Select the variable. The system sends the contents of the variable that you select as an email message.
- **Variable Field**: If you select a complex variable in the **Variable** field, then in the **Variable Field** field, select the complex variable field. The system sends the contents of the variable field that you select as an email message.
- **Format**: Select the format that is relevant to the contents of the variable.

The following table shows the options that are available in the **Format** property:

Name	Description
text	If the variable that you specify contains text, then the **text** option sends the text in the form of an email message.
filename	If the variable that you specify contains the path of a file, then the **filename** option sends the contents of the file in the form of an email message.
url	If the variable that you specify contains a URL to a file, then the **url** option sends the contents of the file in the form of an email message.

Text Variable item in a web application

Type:

Prompt Segment item

Available from:

Prompt Contents tab of the prompt file editor in a web application

Purpose:

If you want to display the value of a variable on the form of a web application, then you can use the **Text Variable**.

Behavior:

A **Text Variable** item displays the value of a variable as plain text. You can wrap the **HTML Text** item around this item to add style to the text.

Properties:

- **Variable**: Select the variable from the list of variables created. The system displays on the screen, the contents of the **Variable** selected
- **Variable Field**: If you select a complex variable in the **Variable** property, then in the **Variable Field** field, select the complex variable field. The system displays on screen, the contents of the variable field that you select.
- **Format**: Select the format that is relevant to the contents of the variable.

The following table shows the options that are available in the **Format** property:

Name	Description
text	If the variable that you specify contains text, then the **text** option displays text on the screen.
filename	If the variable that you specify contains the path of a file, then the **filename** option displays the contents of the file on the screen.
digits	If the variable that you specify contains digits, then the **digits** option displays digits on the screen.
url	If the variable that you specify contains a URL to a file, then the **url** option displays the contents of the file on the screen.

Throw 

Type

Form Item

Available from

- **AppRoot** node (see AppRoot node on page 132)
- Announce node on page 634
- Prompt and Collect node on page 659
- Collect node on page 651
- Blind Transfer node on page 646
- Bridged Transfer node on page 647
- Record node on page 660
- Menu node on page 656
- Form node on page 655

Purpose

In Orchestration Designer, custom events can be created for situations where they are needed. For example, perhaps a custom event is needed to handle users requests to be transferred to an operator. Or custom events can be used to handle other out-of-the-ordinary situations.

Use the **Throw** item to throw events created using the Event Editor. For more information about creating and using custom events, see Events on page 314.

Note:

The default VoiceXML or TextXML events are thrown automatically, as the system responds to various predefined situations. Avaya designed the **Throw** event to only throw those custom events that have been created.

Behavior

Before using the **Throw** item, an event must be created for it to throw. Also, somewhere in the application, within the scope of this item, a Catch on page 693 item must be used.

In most cases, a **Throw** item is used to respond to actions or inputs from users that are outside the scope of what is expected. For example, perhaps a custom event is created in which the user can request to transfer to an operator from anywhere in the call flow or the message flow. Create an event called **Xfer2Operator**, for example. For more information about custom events, see Custom events on page 316.

A Link on page 762 item can be used in the AppRoot node, to monitor and respond to a user request from anywhere in the call flow or the message flow. When the utterance of the word or phrase that activates the **Link** item, the **Link** item throws the custom event. At this point, set up a Catch on page 693 event handler to instruct the application on what to do, for example, transfer the caller to an operator, in this case.

Throw items can also be used to handle situations where a certain threshold of other events has been reached and you want the system to respond differently after that point. For example, you want to give users only three chances to make an acceptable response to a Prompt and Collect node or a Collect node. However, if the No Match on page 770 and No Input on page 769 event handlers were treated independently and separately, the user can have a total of five chances before the appropriate event is thrown.

A **Throw** item can be used in each of the **No Match** and **No Input** event handlers, and then catch it within the same node. By setting the **Threshold** for the **Throw** item to 3, then it does not matter whether the user has a No Match or No Input situation. After the third event, the **Throw** item is activated, and the call flow or the message flow moves on to whatever you set the **Catch** for this **Throw** item to do.

When an event is thrown, it is caught by the next higher level element in the application. In an input form, event throws at the field level can be caught at the form level, and an event thrown at the form level can be caught at the application root. You cannot catch an event at the same level that it is thrown. An event thrown at the menu level is caught at the application root level.

Tip:

For an example the **Throw** item being used as described, see the sample application named *Events* included with the Orchestration Designer software package.

Properties

- **Event**: Select the event that you want to throw at this point in the application.

Note:

If you do not have any custom events defined, this list is populated only with the built-in events for **cancel**, **exit**, and **help**.

- **Threshold**: The number of times the event must be caught before throwing an event. The Threshold value must be a positive integer.

  This property is used to disable the **Throw** item until the threshold value is met. For example, if the **Threshold** property is set to 3, the system ignores the Throw item the first two times the specified event is encountered. Only the third time that the event is encountered does the system actually throw the event.
- **Threshold Variable**: The variable that contains the number of time the event must be caught before throwing the event.
- **Threshold Variable Field**: If **Threshold Variable** is a complex variable, this field contains the number of times the event must be caught before throwing the event.

Note:

The **Threshold**, **Threshold Variable**, and **Threshold Variable Field** fields are available only when the **Throw** item is dragged into an event handler or a node. These fields are not available when the **Throw** item is dragged into a **Link** item.

- **Message**: The message that is displayed when an event is thrown. This is an additional information about the event.
- **Message Variable**: A variable that contains the message used in a throw rather than a hardcoded message string.
- **Message Field**: If the Message Variable is a complex variable, this field contains the message variable rather than using the hard coded message string.

Trace

Type

Tracking item

Available from

- Announce node on page 634
- Prompt and Collect node on page 659
- Collect node on page 651
- Blind Transfer node on page 646
- Bridged Transfer node on page 647
- Record node on page 660
- Disconnect node on page 654
- Menu node on page 656
- Form node on page 655

- Data node on page 653
- Tracking node on page 669

Purpose

The primary use of the **Trace** item is as a debugging tool during application development. You can use the **Trace** item to:

- Check the value of variables or variable fields at different points during application simulation and execution.
- Track application progress, to make sure it is reaching certain points at run time.
- Define different levels of criticality when unexpected or error-type events occur.

Behavior

During simulation or run time, the **Trace** item takes the value of the **Text** field and any selected **Variable** or **Variable Field** and writes the value to a log file. The application locates this log file in the *ProjectName*`/data/log` directory, where *ProjectName* represents the name of the speech application or message application project.

To view the log file contents in Orchestration Designer, locate the appropriate log file in this directory and double-click it. To view the log file contents in a run-time environment, locate the appropriate log file on the application server and view it with your text viewer or word processor of choice.

Tip:

The name of the log file for the current day is always `trace.log`. Each time you run a simulation, Orchestration Designer appends the log data to the end of this file. At the end of each day, Orchestration Designer renames the log file to: `trace.log.`*yyyy-mm-dd*, where *yyyy-mm-dd* represents the date on which the log file was compiled.

During simulation, Orchestration Designer also displays tracing data in the Console view display, and you can review it there.

Note:

Trace log data is written using log4j logging technology. Because this is a new technology and information available is constantly changing, for more information about log4j technology, Avaya recommends that you perform an Internet search with the search term “log4j”.

The **Trace** item requires that you enable it before the system recognizes it during simulation or run time. For this reason, if you use a **Trace** item in a call flow or a message flow, and if you do not enable tracing output during simulation or run time, the system ignores it and does not write any trace data to the log file.

For information about enabling tracing for simulation, see Enabling tracing for simulation on page 392.

Because the **Trace** item is primarily a development and debugging tool, it is not normally used with a deployed application during run time. The log files that are generated during run time can consume large amounts of drive space. However, the log files can be useful when testing or troubleshooting an application deployment.

For information about enabling or disabling tracing in a run-time environment, see Enabling or disabling tracing in a run-time environment on page 473.

Properties

- **Priority**: Select the priority level to assign to this **Trace** item.

  The priority level you select is written to the log file as part of the tracing data. This makes it easy to search the log file for specific problems based on the priority of the trace entry.

  The following table lists the priority levels for **Trace** items, shows how **Trace** items appear in the log file, and provides a brief description of what each level means.

Priority	Log Entry	Description
Fatal	FATAL	A situation in which the application would crash.
Error	ERROR	A situation that would cause the application to generate an error.
Warning	WARN	A situation that would cause the application to generate a warning.
Info	INFO	A situation that would cause the application to generate an informational message.
Debug	DEBUG	A situation that would cause the application to generate debugging data.

- **Text**: Enter in this field any text that you want the system to record as part of the **Trace** item entry in the log file. The system enters the text in the log file verbatim.
- **Variable**: Select the variable whose value you want to record to the log file as part of the **Trace** item entry.

  If this variable is a complex variable, you must also select a **Variable Field**.
- **Variable Field**.: If the **Variable** you selected is a complex variable, select the variable field whose value you want to record to the log file as part of the **Trace** item entry.

Transfer

Type

Form item.

Available From

- **AppRoot** node (see AppRoot node on page 132)
- Module nodes on page 143

Purpose

The purpose of the Transfer item is to be able to transfer the caller to another number (For example, an agent) in the event that there are errors from the application. This is intended to handle “fatal error” type cases where the application cannot recover (that is when the application server is down) so the application can direct the platform to transfer the caller to another number.

Behavior

This item can be added as a child of a Catch item. At run time, the application will generate a VXML event handler that will perform a blind transfer to the destination number. A good example of this is using a Transfer on Event item as a child of a Catch “error.badfetch.http” event handler

with a threshold set to 3. This means after 3 consecutive HTTP bad fetch errors, the caller will be transferred from the application to another number, such as an operator or an agent.

Properties

- **Destination number**: The (hardcoded) number that is used for the transfer destination.
- **Destination Variable**: The variable that stores the number used for the transfer destination
- **Destination Variable Field**: If **Destination Variable** is a complex variable, the field that stores the transfer destination.
- **AAI Data**: The (hardcoded) value that is passed as application-to-application information (AAI) to the receiving end of the call transfer.
- **AAI Variable**: The variable that contains AAI data.
- **AAI Variable Field**: If **AAI Variable** is a complex variable, the field that contains the AAI data.
- **Threshold**: The (hardcoded) number of times the event must be caught before performing the transfer.
- **Threshold Variable**: The variable that contains the number of time the event must be caught before performing the transfer.
- **Threshold Variable Field**: If **Threshold Variable** is a complex variable, the field that contains the number of times the event must be caught before performing the transfer.

Transfer Call 

Type

IC connector item

Available from

Data node on page 653

Purpose

The **Transfer Call** item makes it possible to have the IC system transfer a caller to a destination.

Behavior

Before you can use this or any other IC item, you must enable IC for Orchestration Designer. For more information about the ability of Orchestration Designer to interact with IC, and to enable IC in your Orchestration Designer applications, see About the IC connector on page 622.

Note:

Support for call transfer functions can vary from one IVR system to another. For more information and details about support for transfer functions on your system, see the documentation for your IVR system.

When the speech application encounters the **Transfer Call** item, the application directs the IC system to initiate a call transfer to the destination as assigned in the **Destination** property. See the “Properties” section that follows.

Properties

- **Destination**: Enter in this field the destination to which the IC system is to transfer the call. Valid entry types include:
- A number that represents:
  - A specific agent's extension number.
  - A telephone number that is configured to accept and route such incoming calls.
- An alphanumeric string that represents an agent's login ID.

  This type of entry transfers the call to the agent associated with this login ID.

Try 

Type

Data item

Available from

Data node on page 653

Purpose

To try to execute one or more operation items. If any of the operations beneath the Try throw an exception, the Java exceptions can be caught at run time.

Behavior

Try/Catch items in the Data node on page 653 will *try* to execute items under the Try on page 850 item. To handle exceptions, add Catch (Exception) on page 692 items (under the parent Try on page 850). For every Try item, you need at least one Catch (Exception) on page 692.

By default, Catch (Exception) on page 692 will catch *all* exceptions. Catch (Exception) on page 692 can also be used to catch specific exception types for handling specific errors differently. Comma separated exception types can also be specified for catching different exception types, but handling them in the same way.

List of common exceptions are provided in the Avaya Properties view. Currently this list contains:

- SQLException
- IOException
- SCERuntimeException

A new variable called **ddLastException** automatically captures the exception caught by a Catch (Exception) on page 692 item (includes the following data members: errorcode, message, object, stacktrace, and type).

Following is an example of the Try/Catch mechanism.

TTS

Type

Prompt Segment item

Available from

Any level tab in the prompt file editor. For more information about the prompt file editor, see Prompt file editor in a speech application on page 207.

Purpose

The **TTS** item renders text input as synthesized speech output. This can be valuable when it is impossible or impractical to use prerecorded audio files for prompts.

Behavior

The TTS item takes text entered into the TTS Text field and passes it to the TTS speech synthesis engine. The TTS engine then renders the text input into audible synthesized speech.

To control the pitch, rate, volume, and other aspects of the TTS rendering, use the SSML options. The SSML options include:

- Break on page 682
- Emphasis on page 726
- Prosody on page 804
- Say As on page 821
- Voice on page 853

Properties

- **TTS Text**: Enter in this field the text that you want rendered as synthesized speech.

Value Operand

Type

Database Operation Condition Operand

Available From

Database Operation Editor Predicate Tab

Purpose

It is used by the Simple condition item to set up search criteria with a constant value.

Behavior

A Simple condition requires having one Value operand or one Variable operand. After adding a Simple condition item, add the Value operand if you want to search for a pre-determined value.

Properties

- **Value**: Specify any constant value in string format.

Variable Field 

Type

Variable item

Available from

Data node, with complex variables only

Purpose

A complex variable is a variable that has several attributes called variable fields. Complex variable makes it possible to use a single variable to pass multiple values. To make a complex variable functional, at least one variable field must be assigned to the complex variable.

Behavior

The Variable Field item acts like a simple variable within a complex variable. In other words, each Variable Field item contains a single variable value.

When you create a complex variable, the system automatically creates a variable field and assigns it to the complex variable. You can add as many additional Variable Field items as you want to a complex variable.

Properties

- **Name**: Type a name for the variable field. The variable field name must follow conventions for naming Java components. For more information, see Conventions for naming Java components on page 59.
- **Value**: Type a default value that you want to assign to the variable field. If you do not specify a value in this field, the default value is set to undefined.

Variable Operand

Type

Database Operation Condition Operand

Available From

Database Operation Editor Predicate Tab

Purpose

It is used by the Simple condition item to set up search criteria with a value that comes from a variable at run time.

Behavior

A Simple condition requires having one Value operand or one Variable operand. After adding a Simple condition item, add the Variable operand if the value you are searching for comes from a variable in run time.

Properties

- **Name**: Select a variable from the list
- **Field**: Select a field from the variable selected.

Voice 

Type

SSML item

Available from

Any level tab in the prompt file editor. For more information about the prompt file editor, see Prompt file editor in a speech application on page 207.

Purpose

Because of cultural or other preferences, you want to change the type of voice the speech synthesis engine uses to speak text. The **Voice** item makes it possible to control various aspects of the voice used to render text, including gender and age.

Behavior

Based on the properties you set, the **Voice** item controls the type of voice used to render text. All properties are optional, but if you use the **Voice** item, you must use at least one property. The **Voice** item has four properties:

Note:

The specific effects of these properties can vary from one TTS engine to another.

Properties

- **Gender**: Select the gender you want the voice to have. The choices include: **Male**, **Female**, and **Neutral**.
- **Age**: Enter in this field a non-negative integer.

  This number represents the supposed age of the speaker. So, if you want a childlike voice to render the text, you can set this to **6** or **7**. If you want the voice to sound like an older person, you can set this to **60** or **70**.

- **Variant**: Enter in this field a positive integer.

  This number represents an alternate voice as defined by the speech synthesis engine. Valid values are determined by the speech synthesis engine. So, to use this property, consult the documentation for your speech synthesis engine.

- **Name**: Enter in this field the name that you want to identify the voice.

  This name is, usually, a name of a person. So, for example, a male English voice can be named “John,” and a female Spanish voice can be named “Juanita.”

  Note:

  Even though Orchestration Designer does not enforce it, this name should follow conventions for naming Java components as described in Conventions for naming Java components on page 59.

Configurable Complex Variable

Type

Variable

Available from

Variable Editor only (see Variable management on page 230.

Be sure that the **AVP/AEP Configurable Application Variables** pluggable data connector is enabled. For more information, see Enabling Orchestration Designer project pluggable data connectors on page 503.

Purpose

The value of the Configurable Complex Variable is administered through the EPM interface. The idea is that the same application that uses the variable can be deployed on different sites in an enterprise. The behavior of the application can be changed administratively to accommodate the local specifics.

Behavior

The Configurable Complex Variable is an extension to the Complex Variable, and it inherits the same behavior in the Variable Editor. When used in the application, you can only access its value, but not modify it. The value of the variable is being administered in the EPM interface. The value gets retrieved from EPM in runtime everytime it is updated.

Properties

- **Name**: The name you want to use to identify the variable.

  Note:

  This name must follow conventions for naming Java components as described in Conventions for naming Java components on page 59.

- **Display Name**: This name will show as a Field label in the EPM administrative screen. If it is empty, the variable name will be used instead.

- **Data Type**: EPM uses this to validate input values. This has the enumeration of String, Integer, Float, Date, Time, Date/Time, File, Boolean, and List. The default value is String.
- **Min. Length**, **Max. Length**: Dynamic properties that show up when String is selected for the Data Type. They are used for validating value inputs in EPM.
- **Min. Value**, **Max. Value**: Dynamic properties that show up when Integer or Flat is selected for Data Type. They are used for validating inputs in EPM.
- **Date Format**, **Time Format**: Dynamic properties that show up when Date, Time, or Date/Time is selected for Data Type. They are used for validating inputs in EPM.
  - If you select **Date** as a **Data Type**, the **Date Format** field is displayed. You must specify the date format using standard pattern characters, M for month, d for day of month, and y for year. You can use any separator. For example, you can specify MM/dd/yyyy or dd-MM-yyyy and so on. For details on pattern characters, refer http://java.sun.com/javase/6/docs/api/java/text/SimpleDateFormat.html.
  - If you select **Time** as Data Type, the **Time Format** field is displayed. The format for specifying time is hh:mm:ss.
  - If you select **Date/Time** as Data Type, both the **Date Format** and **Time Format** fields are displayed.
- **File Extension**: Dynamic properties that show up when File is selected for Data Type. You can upload files only with this file extension.
- **Default Value (URL)**: Dynamic properties that show up when File is selected for Data Type. Enter the URL of an existing file that you want to use as default.
- **Comma-delimited values**: Dynamic property that shows up for the list data type. Enter values separated by ",". These values are displayed as a list in EPM for users to select.
- **Default Value**: The value used by the application in runtime when it has not been changed in EPM. You can also use this field to verify if the specified value matches the dynamic properties for a particular data type.
- **Description**: The description that shows up as a tooltip for the variable in EPM.
- **Order**: The vertical order of the input field in EPM.
- **Mandatory**: Set this value to make it a mandatory field. If you set this value to true, you must specify a value for this field in EPM.

Configurable Variable

Type

Variable

Available from

Variable Editor only (see Variable management on page 230.

Be sure that **AVP/AEP Configurable Application Variables** pluggable data connector is enabled. For more information, see Enabling Orchestration Designer project pluggable data connectors on page 503.

Purpose

The value of the Configurable Variable is administered through the EPM interface. The idea is that the same application that uses the variable can be deployed on different sites in an enterprise. The behavior of the application can be changed administratively to accommodate the local specifics.

Behavior

The Configurable Variable is an extension to the Simple Variable, and it inherits the same behavior in the Variable Editor. When used in the application, you can only access its value, but not modify it. The value of the variable is being administered in the EPM interface. The value get retrieved from EPM in runtime everytime it is updated.

Properties

- **Name**: The name you want to use to identify the variable.

  Note:

  This name must follow conventions for naming Java components as described in Conventions for naming Java components on page 59.

- **Display Name**: This name will show as a Field label in the EPM administrative screen. If it is empty, the variable name will be used instead.
- **Data Type**: EPM uses this to validate input values. This has the enumeration of String, Integer, Float, Date, Time, Date/Time, File, Boolean, and List. The default value is String.
- **Min. Length**, **Max. Length**: Dynamic properties that show up when String is selected for the Data Type. They are used for validating value inputs in EPM.
- **Min. Value**, **Max. Value**: Dynamic properties that show up when Integer or Flat is selected for Data Type. They are used for validating inputs in EPM.
- **Date Format**, **Time Format**: Dynamic properties that show up when Date, Time, or Date/Time is selected for Data Type. They are used for validating inputs in EPM.
  - If you select **Date** as a **Data Type**, the **Date Format** field is displayed. You must specify the date format using standard pattern characters, M for month, d for day of month, and y for year. You can use any separator. For example, you can specify MM/dd/yyyy or dd-MM-yyyy and so on. For details on pattern characters, refer http://java.sun.com/javase/6/docs/api/java/text/SimpleDateFormat.html.
  - If you select **Time** as Data Type, the **Time Format** field is displayed. The format for specifying time is hh:mm:ss.
  - If you select **Date/Time** as Data Type, both the **Date Format** and **Time Format** fields are displayed.
- **File Extension**: Dynamic properties that show up when File is selected for Data Type. You can upload files only with this file extension.
- **Default Value (URL)**: Dynamic properties that show up when File is selected for Data Type. Enter the URL of an existing file that you want to use as default.
- **Comma-delimited values**: Dynamic property that shows up for the list data type. Enter values separated by “,”. These values are displayed as a list in EPM for users to select.

- **Default Value**: The value used by the application in runtime when it has not been changed in EPM. You can also use this field to verify if the specified value matches the dynamic properties for a particular data type.
- **Description**: The description that shows up as a tooltip for the variable in EPM.
- **Order**: The vertical order of the input field in EPM.
- **Mandatory**: Set this value to make it a mandatory field. If you set this value to true, you must specify a value for this field in EPM.

Web Service (Operation)

Type

Node item

Available from

Data node on page 653

Purpose

The **WebService** item makes it possible to use a predefined Web service operation. The **WebService** item in itself only invokes the Web service operation and tells the call flow or the message flow where to use it.

Behavior

Before you can use the **WebService** item in the Data node, you must have a Web service operation defined for the application. For more information about defining and using Web service operations, see Web Services on page 320.

The way the Web service operation actually behaves in the Data node depends on how the Web service operation is defined. When you drag the **WebService** item into the Data node flow, all Orchestration Designer does is invoke the selected Web service operation.

Properties

- **Name**: Select the Web service operation you want to invoke.

Web Service (REST)

Type

Node item

Available from

The **Web Service (REST)** item is available in the Palette pane of a Data node editor only after you enable the **Web Services (REST)** pluggable data connector.

For more information, see Enabling Orchestration Designer project pluggable data connectors on page 503.

Purpose

Use the **Web Service (REST)** item to invoke a predefined REST Web service operation in an Orchestration Designer speech application or message application.

Behavior

Before you use the **Web Service (REST)** item in a **Data** node, you must create and configure a REST Web service operation file in the Orchestration Designer project.

For more information, see Creating a REST Web service operation file on page 334 and Configuring a REST Web service operation file on page 335.

The way the REST Web service operation is processed in the **Data** node depends on how the Web service operation file is configured. Orchestration Designer invokes the REST Web service operation file that you select in the **Name** property of the **Web Service (REST)** item.

Properties

- **Name**: The REST Web service operation file that you want to invoke.

Detailed Web Node Descriptions

Announce node 

Type

Template (composite) node

Purpose

The purpose of an **Announce** node is to display information to the user. It is a **Form** node with a default **Prompt** element.

An **Announce** node generates a JSP page in the `jsp` directory. This JSP is processed by the application server to render HTML for the web page at runtime.

Behavior

When you drag an **Announce** node into a web flow, you must define a **Prompt** to display information to the user. **Announce** node, then, uses that **Prompt** to display information on the web page when the node is executed.

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that

has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to IQ for reporting.

- **Secure Fetch**: Set this property to true if you want the next node in the web flow to get and post data by using HTTPS. If you set this property to true, the system generates the URL for the next node in the web flow as HTTPS and uses the configured fetch secure port.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.
- **Graphic** : Enter the path and name of the graphic file that you want to display on top of the web page. For example, `css/images/avaya-logo.png`.
- **Graphic URI Variable**: Select the variable that contains the path of the graphic file. Ensure that you copy the image files in the directory that is accessible using url or uri, and that the value of the variable assigned to the **Graphic URI Variable** property matches the url or uri.
- **Graphic URI Variable Field**: Select the field of the variable that contains the graphic file path.
- **Graphic Style** : Enter any CSS style property or properties for the way you want the image to be displayed. For example, to make the image as wide as the browser window, you can enter the property “width:100%”.
- **Title Message** : Enter the message that you want to display on top of the web page, below the graphic.
- **Message Style** : Enter any CSS style property or properties for the way you want the message to be displayed. For example, to make the message bold, you can enter the property “font-weight:bold”.

Default node structure (Node Detail Editor Flow)

- **Prompt** item: Indicates what prompt Orchestration Designer must use to display the information for the web page.
- **Next** item: Indicates which node comes next in the web flow.

Collect node

Type

Template (composite) node

Purpose

You can collect information from the user on a web page using a collect node.

The main difference between the collect node in a web project and any other project such as a speech project is that, in a web project, you can set up multiple inputs in one collect node whereas in other projects, you can set up only single input.

Behavior

When you drag this node into the workspace, Orchestration Designer automatically creates an input item.

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya IQ for reporting.
- **Secure Fetch**: Set this property to true if you want the next node in the web flow to get and post data by using HTTPS. If you set this property to true, the system generates the URL for the next node in the web flow as HTTPS and uses the configured fetch secure port.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.
- **Graphic** : Enter the path and name of the graphic file that you want to display on top of the web page. For example, `css/images/avaya-logo.png`.
- **Graphic URI Variable**: Select the variable that contains the path of the graphic file. Ensure that you copy the image files in the directory that is accessible using url or uri, and that the value of the variable assigned to the **Graphic URI Variable** property matches the url or uri.
- **Graphic URI Variable Field**: Select the field of the variable that contains the graphic file path.
- **Graphic Style** : Enter any CSS style property or properties for the way you want the image to be displayed. For example, to make the image as wide as the browser window, you can enter the property “width:100%”.
- **Title Message** : Enter the message that you want to display on top of the web page, below the graphic. To display a multi-language title message, use a prompt instead of a **Title Message**.
- **Message Style** : Enter any CSS style property or properties for the way you want the message to be displayed. For example, to make the message bold, you can enter the property “**font-weight:bold**”.

Default node structure (Node Detail Editor Flow)

- **Text Input** item: Allows the user to enter information in the form of text from a web page
- **Next** item: Indicates which node comes next in the web flow.

Form node

Type

Application Item (basic) node

Purpose

Use this multipurpose node to perform a wide variety of functions, from collecting and processing user input to displaying and confirming user information. This node is the most generic of all the nodes and is actually the basis of many of the Templates (Composite nodes).

You can also use this node to specify whether you want the next node in the web flow to get and post data by using HTTPS.

A **Form** node has following palette items:

- **Text Input**: The **Text Input** is used to collect user information in form of text in a web application. For more information, see Text Input on page 876.
- **Choice Input**: The **Choice Input** allows the user to select a value from the set of available options. For more information, see Choice Input on page 863.
- **Location Input**: The **Location Input** allows the user to provide the location co-ordinates. For more information, see Location Input on page 870.
- **Text Area Input**: The **Text Area Input** is used to collect user information in the form of multi-line text in a web application. For more information, see TextArea Input on page 877.
- **Picture Input**: The **Picture Input** allows the user to take a picture and upload it. For more information, see Picture Input on page 872.
- **Voice Input**: The **Voice Input** allows the user to record an audio and provide it as an input. For more information, see Voice Input on page 879.
- **Video Input**: The **Video Input** allows the user to record a video and provide it as an input. For more information, see Video Input on page 878.

Behavior

When you use this node into the web flow, this node sets up a generic form that you can use for a wide variety of purposes.

A **Collect** node and an **Announce** node, both based on **Form** node, generates a JSP page in the `jsp` directory which, at runtime, renders an HTML page to the web browser.

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya IQ for reporting.
- **Secure Fetch**: Set this property to true if you want the next node in the web flow to get and post data by using HTTPS. If you set this property to true, the system generates the URL for the next node in the web flow as HTTPS and uses the configured fetch secure port.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.
- **Graphic** : Enter the path and name of the graphic file that you want to display on top of the web page. For example, `css/images/avaya-logo.png`.
- **Graphic URI Variable**: Select the variable that contains the path of the graphic file. Ensure that you copy the image files in the directory that is accessible using url or uri, and that the value of the variable assigned to the **Graphic URI Variable** property matches the url or uri.

- **Graphic URI Variable Field**: Select the field of the variable that contains the graphic file path.
- **Graphic Style** : Enter any CSS style property or properties for the way you want the image to be displayed. For example, to make the image as wide as the browser window, you can enter the property “width:100%”.
- **Title Message** : Enter the message that you want to display on top of the web page, below the graphic.
- **Message Style** : Enter any CSS style property or properties for the way you want the message to be displayed. For example, to make the message bold, you can enter the property “**font-weight:bold**”.

Default node structure (Node Detail Editor Flow)

There is no default flow in the node editor for this node.

Menu node 

Type

Application Item (basic) node

Purpose

The purpose of this node is to present the user with a limited series, or menu of options. The user can use these menu options to navigate through the web application and provide responses wherever required.

Behavior

When you drag this node into the web flow, this node sets up a framework to construct a menu of options for users, mostly through the options offered on the node editor palette. One palette option, in particular, is optimized for use with this node: the **Choice** item. Use this item to create a menu, which is a link to the **Next** node.

When you drag a **Menu** item into the web flow, Orchestration Designer automatically creates a complex variable with the same name. A complex variable contains the following field, which is populated upon completion of the node:

- **value**: Once the user selects the menu option, the **Value** field of the variable gets updated by the name of the choice that the user selects. You can process the value of the **Value** field and decide next action in the data node.

Properties

- **Name**: Enter a name that reflects the central purpose of the node. Use naming conventions for Java components.
- **Preferred Path**: Set this property to true to use the preferred path at the time of application execution. The default value is true. If you set this property to false and execute the node, the value in the session:exitPreferredPath session object is set to 0. After session:exitPreferredPath is set to 0, the system does not reset it when executing a node that has **Preferred Path** set to true. The exitPreferredPath value is stored in the session data record (SDR) in Avaya Experience Portal, and then sent to Avaya IQ for reporting.

- **Secure Fetch**: Set this property to true if you want the next node in the web flow to get and post data by using HTTPS. If you set this property to true, the system generates the URL for the next node in the web flow as HTTPS and uses the configured fetch secure port.
- **Comments**: Enter any comment you want to add as a reminder or description of the purpose or content of the node. You can leave this field blank.
- **Graphic** : Enter the path and name of the graphic file that you want to display on top of the web page. For example, `css/images/avaya-logo.png`.
- **Graphic URI Variable**: Select the variable that contains the path of the graphic file. Ensure that you copy the image files in the directory that is accessible using url or uri, and that the value of the variable assigned to the **Graphic URI Variable** property matches the url or uri.
- **Graphic URI Variable Field**: Select the field of the variable that contains the graphic file path.
- **Graphic Style** : Enter any CSS style property or properties for the way you want the image to be displayed. For example, to make the image as wide as the browser window, you can enter the property “width:100%”.
- **Title Message** : Enter the message that you want to display on top of the web page, below the graphic.
- **Message Style** : Enter any CSS style property or properties for the way you want the message to be displayed. For example, to make the message bold, you can enter the property “**font-weight:bold**”.

Default node structure (Node Detail Editor Flow)

There is no default flow in the node editor for this node.

Detailed Web Palette Option Descriptions

Choice Input

Type:

Form item

Available from:

Announce node

Collect node

Purpose:

The **Choice Input** allows the user to select a value from the set of available options.

There are two types of **Choice Input** from which user can provide the input – **Radio** and **Dropdown**.

Behavior:

Use the **Choice Input** item when you want the user to provide input from the available set of options. For the **Choice Input**, each choice that is displayed on the web page has a corresponding value associated with it. Associating a value to the text is important in case of multi language support, where the runtime logic deals with only the value, while the corresponding text may vary.

To give the values for the text, set the **Choice Values (separated by ‘,’)** property.

Note:

You can have more than one **Choice Input** items in a single **Collect** or **Announce** node.

For each **Choice Input** that you drag into the **Collect** or **Announce** node, Orchestration Designer automatically creates a simple variable with the same name. You can use these variables the same way as you would use any other variable. If a **Label ID** is not assigned to a **Choice Input**, Orchestration Designer generates an error.

Properties:

- **Name** : Enter the name you want to assign to the **Choice Input** and its corresponding variable.

  If you change the name of this item in the Avaya Properties view, Orchestration Designer automatically changes the name of the associated variable as well.

  Note:

  This name must follow conventions for naming Java components as described in Conventions for naming Java components on page 59.

- **Choice Values (separated by ‘,’)**: Specify the choice values corresponding to the text values that are displayed in the **Radio** or **Dropdown** for the **Choice Input** item. For example, specify values `1,2,3` for text **Hot, Warm, Cold** configured in the text set ID of **Choices Texstset ID**.
- **Label ID** : Select a label to describe the **Choice Input**. Labels can be created in a `textset` file. For more information on textsets, see About textset files on page 125.

  Important:

  If you do not give a **Label ID** for a **Choice Input**, Orchestration Designer generates an error.

- **Type** : Select the type of **Choice Input**. It can be either **Radio** or **Dropdown**.
- **Required** : Enter if the **Choice Input** is mandatory or not.
- **Variable for Choices** : Select the variable that has a list of choices in its collection. Ensure you set either **Variable for Choices**, or **Choices Texstset ID** to provide the list of choices.
- **Variable Field** : If you select a complex variable in **Variable for Choices** field, then in **Variable Field** you must select the complex variable in which you want to store the choice values. The **Variable Field** property is unavailable if you select a simple variable in the **Variable for Choices** field.
- **Choices Texstset ID** : Select the textset entry created in a `textset` file for the **Choice Input**. The textset entry must have a value that specifies a comma delimited list of items. For example, “Male, Female” must create two choices: “Male” and “Female” respectively.

Click To Call

Type

Prompt Editor Palette item

Available from

Prompt Contents tab of the Prompt File Editor in a web application.

Purpose

Use the **Click To Call** item to add a call button to a webpage.

Behavior

Using the **Click To Call** button, you can call on a constant number or a number that is dynamically determined at runtime. The **Click To Call** item is available in the palette of a **Prompt** associated with an **Announce** or **Collect** node. If you want to call a constant number, then configure the **Phone Number** property, else specify the **Variable** to determine the number dynamically.

Properties

- **Label**: Specify the label that you want to display on the **Click To Call** button.
- **Phone Number**: Specify the constant number that you want to dial when you click **Click To Call**.
- **Variable**: Specify the variable that contains the dynamic phone number that is generated at runtime. If you select a complex variable in the **Variable** field, then you must select the complex variable field in the **Variable Field** property.
- **Variable Field**: Select the complex variable field that contains the dynamic phone number that is generated at runtime if you selected a complex variable in the **Variable** field. The **Variable Field** property is unavailable if you select a simple variable in the **Variable** field.

Create Context

Type

Workflow Palette item

Available from

Workflow palette of a **Data** node in a web application.

Purpose

Use the **Create Context** item to create a new context in the Context Store.

Behavior

Using **Create Context**, you can place the data for an Orchestration Designer application into a context in the context store. The ID of this context is passed in the shared UUI. Orchestration Designer retrieves data from the context store and updates and/or adds new data in the context.

Properties

- **Context ID Variable**: Select the variable that contains the context ID of the context. Context IDs must be unique. If you keep the **Context ID Variable** as blank, a unique context ID is generated and returned in the **Context ID Variable**.
- **Context ID Variable Field**: Select the field of the variable that contains the context ID.
- **Context ID Constant**: Enter a constant value for the context ID.
- **Group ID Variable**: Select the variable that contains the group ID to which the context belongs. You can relate several contexts together using a group ID. **Group ID Variable** is an optional property.
- **Group ID Variable Field**: Select the field of the variable that contains the group ID to which the context belongs.
- **Group ID Constant**: Enter a unique constant value for the group ID.
- **ContextData Key Variable**: Select the variable that contains the key or attribute name for the initial value of the context. **ContextData Key Variable** is an optional property.
- **ContextData Key Variable Field**: Select the field of the variable that contains the key or attribute name for the initial value of the context.
- **ContextData Key Constant**: Enter a unique constant value as the key or attribute name for the initial value of the context.
- **ContextData Value Variable**: Select the variable that contains the data to be stored in the context when created. If the key is empty, then Orchestration Designer uses the variable name as the key and stores the entire variable into the context. If the variable is a complex variable, then Orchestration Designer stores all fields of the variable into the context.
- **ContextData Value Variable Field**: Select the field of the variable that contains the data to be stored in the context when created.
- **ContextData Value Constant**: Enter the constant value that you want to store in the context.
- **LeaseTime Variable**: Select the variable that contains the time for which the context will remain active. When the lease time expires, the context is removed from the Context Store. The default is 3600 seconds.
- **LeaseTime Variable Field**: Select the field of the variable that contains the time for which the context will remain active.
- **LeaseTime Constant**: Enter the constant value as the lease time for the context.
- **Journey Element Variable**: Select the variable that contains the value that will be stored in the audit trail of the context. **Journey Element Variable** is an optional property.
- **Journey Element Variable Field**: Select field of the variable that contains the value that will be stored in the audit trail of the context.
- **Journey Element Constant**: Enter the value that you want to store in the audit trail of the context.
- **Persist Context to External Database**: If the Context Store is configured with an External Data Mart (EDM) , then set **Persist Context to External Database** to `true` if you want to write the context to an external database. The default value is false.

Delete Context 

Type

Workflow Palette item

Available from

Workflow palette of a **Data** node in a web application.

Purpose

Use the **Delete Context** a context from the Context Store.

Behavior

Delete Context deletes entire context from the Context Store.

Properties

- **Context ID Variable**: Select the variable that contains the context ID of the context that you want to delete from the Context Store.
- **Context ID Variable Field**: Select the field of the variable that contains the context ID of the context that you want to delete.
- **Context ID Constant**: A unique constant value for the context ID of the context that you want to delete.
- **Journey Element Variable**: Select the variable that contains the value that you want to delete from the audit trail of the context. **Journey Element Variable** is an optional property.
- **Journey Element Variable Field**: Select field of the variable that contains the value that you want to delete from the audit trail of the context.
- **Journey Element Constant**: A constant value that you want to delete from the audit trail of the context.

Delete Context Item 

Type

Workflow Palette item

Available from

Workflow palette of a **Data** node in a web application.

Purpose

Use the **Delete Context Item** to delete a single item from a context in the Context Store.

Behavior

Delete Context Item deletes a single item from the context.

Properties

- **Context ID Variable**: Select the variable that contains the context ID of the context from which you want to delete a data item.
- **Context ID Variable Field**: Select the field of the variable that contains the context ID of the context from which you want to delete a data item.
- **Context ID Constant**: A unique constant value for the context ID of the context from which you want to delete a data item.
- **Key Variable**: Select the variable that contains the key that you want to delete from the context. When you delete a key, Orchestration Designer also deletes the data associated with that key.
- **Key Variable Field**: Select the field of the variable that contains the key that you want to delete from the context.
- **Key Constant**: A unique constant value for the key that you want to delete from the context.
- **Journey Element Variable**: Select the variable that contains the data item that you want to delete from the audit trail of the context. **Journey Element Variable** is an optional property.
- **Journey Element Variable Field**: Select field of the variable that contains the data item that you want to delete from the audit trail of the context.
- **Journey Element Constant**: A constant data item that you want to delete from the audit trail of the context.

Get Context Data 

Type

Workflow Palette item

Available from

Workflow palette of a **Data** node in a web application.

Purpose

Use the **Get Context Data** item to retrieve context data from a context in the Context Store.

Behavior

Get Context Data retrieves a single value from a context in the Context Store.

Properties

- **Result Variable**: Select the variable in which you want to store the value retrieved from a context in the Context Store.
- **Result Variable Field**: Select the field of the variable in which you want to store the value retrieved from a context in the Context Store.
- **Context ID Variable**: Select the variable that contains the context ID of the context that you want to access.
- **Context ID Variable Field**: Select the field of the variable that contains the context ID of the context that you want to access.

- **Context ID Constant**: A unique constant value for the context ID of the context that you want to access.
- **Key Variable**: Select the variable that contains the key or attribute name of the data item that you want to retrieve from the context.
- **Key Variable Field**: Select the field of the variable that contains the key or attribute name of the data item that you want to retrieve from the context.
- **Key Constant**: A constant value of the key or attribute name of the data item that you want to retrieve from the context.
- **Journey Element Variable**: Select the variable that contains the value stored in the audit trail of the context. **Journey Element Variable** is an optional property.
- **Journey Element Variable Field**: Select field of the variable that contains the value stored in the audit trail of the context.
- **Journey Element Constant**: A constant value that is stored in the audit trail of the context.

Get Variable from Context 

Type

Workflow Palette item

Available from

Workflow palette of a **Data** node in a web application.

Purpose

Use the **Get Variable From Context** item to retrieve a variable from a context in the Context Store.

Behavior

Get Variable From Context retrieves a simple variable, complex variable, simple collection, or complex collection from the Context Store.

Properties

- **Context ID Variable**: Select the variable that contains the context ID of the context that you want to access.
- **Context ID Variable Field**: Select the field of the variable that contains the context ID of the context that you want to access.
- **Context ID Constant**: A unique constant value for the context ID of the context that you want to access.
- **Variable Name Variable**: Select the variable that will receive the data from the context. If you provide the variable name and variable fields, then they are used as keys to access the context.
- **Journey Element Variable**: Select the variable that contains the value stored in the audit trail of the context. **Journey Element Variable** is an optional property.

- **Journey Element Variable Field**: Select field of the variable that contains the value stored in the audit trail of the context.
- **Journey Element Constant**: A constant value that is stored in the audit trail of the context.

Location Input

Type:

Form item

Available from:

Announce node

Collect node

Purpose:

The **Location Input** allows developers to create a widget to capture the end user's location coordinates.

Behavior:

The widget that the **Location Input** creates is composed of a checkbox, a textbox and a map. When the end-user clicks on the checkbox, the map graphic shows the location of the end-user. When the user clicks **Next** on the web page, Orchestration Designer submits the geographic coordinates to the application server and stores them in the input variable.

For each **Location Input** that you drag into the **Collect** or **Announce** node, Orchestration Designer automatically creates a simple variable with the same name. The variable stores the geographic coordinates once the user clicks **Next** on the web page.

You can configure the variable in a **Web Element** of a **Prompt** to show the map graphic with the location marker in the subsequent node in the flow.

Properties:

- **Name** : Enter the name you want to assign to the **Location Input** and its corresponding variable.

  If you change the name of this item in the Avaya Properties view, Orchestration Designer automatically changes the name of the associated variable as well.

  Note:

  This name must follow conventions for naming Java components as described in Conventions for naming Java components on page 59.

- **Text in textset** : Select text to be displayed against the **Location Input**. This text to be displayed can be created in a `textset` file. For more information on textsets, see About textset files on page 125.

  Important:

  If you do not give a **Text in textset** value for a **Location Input**, Orchestration Designer generates an error.

Selecting maps for the Location input

About this task

If you use a **Location Input** item, then you can switch between different maps, such as a Google map or a Baidu map. You can switch between the maps from the properties of the project.

Procedure

1. Right-click your project.
2. In the context menu, click **Properties** .
3. In the Properties window, click Orchestration Designer.
4. Click the JS/CSS/Theme tab.
5. In the **Select Map Type** section, select the map as :
   - **Baidu Map**: For **Baidu Map**, add the API key in the `js/avaya.js` file. In `js/avaya.js` file, set the value of the *location1_MAP_BAIDU_APIKEY* constant to "`ak=<API key given to you by Baidu>`". For example,
     ```
     var location1_MAP_BAIDU_APIKEY = "ak=xP6y12Uc5lelUZkt40T9N952AOtGVvqt";
     ```
   - **Google Map**: For **Google Map**, set the value of *location1_MAP_GOOGLE_APIKEY* to "`key=xxxxxxxxxxxxxxxxxx`"
   - **Other Map**: For **Other Map**, customize *location1_OTHER_MAP_PARAMS* to set the respective information to access the map, such as the URL. For example,
     ```
     var location1_MAP_TYPE_OTHER = "OTHER";
     var location1_MAP_OTHER_APIKEY = "";
     var location1_OTHER_MAP_PARAMS = {"url":"http://<domain>/<api
     url>", "zoom":"14", "size":"400x300", "markerStyle":"color:red%7C"};
     ```

   If there is any prompt file in the project that uses the **Web Element** to display the map for geo location coordinates, then you must set the API key in the `web.xml` descriptor. You can set the API key as a parameter either through the Web Descriptor tab of the project **Properties** using the key mapAPIKey, or by manually editing the `web.xml` file.

Loop Variable Collection 

Available from

Prompt Contents tab of the Prompt File Editor in a web application.

Purpose

You can use a **Loop Variable Collection** to loop through a collection in a variable.

Behavior

The **Loop Collection** item creates a loop that iterates through a variable collection. You can add **HTML Text** and **Text Variable** items within a loop to create a list or a table in HTML that displays values in the collection of a variable.

Properties

- **Variable**: Click the variable that contains the input that you want to display. If you select a complex variable in the **Variable** field, then you must select a complex variable field in the **Variable Field** property.
- **Variable Field**: If you select a complex variable in the **Variable** property, then in the **Variable Field** property, select the complex variable field that contains the input value that you want to display on the screen. The **Variable Field** property is unavailable if you select a simple variable in the **Variable** field.

Picture Input

Type:

Form item

Available from:

Announce node

Collect node

Purpose:

The **Picture Input** allows the user to take a picture and upload it. It is especially useful in mobile devices.

Behavior:

For each **Picture Input** that you drag into the **Collect** or **Announce** node, Orchestration Designer automatically creates a simple variable with the same name.

When an end-user clicks **Next** on a web page, the picture gets uploaded to the application server and is stored in the project's `data` or `temp` directory. The path of the file is assigned to the input variable. The application developer may save the file as the `temp` directory gets cleaned out by default at the end of the session. To display the picture in the subsequent node of the flow, you can use the **Web Element** in the **Prompt** editor. The same process happens in simulation testing on a desktop web browser. However, when running an application in simulation, no file is created; and everything happens on the web browser.

Properties:

- **Name**: Enter the name you want to assign to the **Picture Input** and its corresponding variable.

  If you change the name of this item in the Avaya Properties view, Orchestration Designer automatically changes the name of the associated variable as well.

  Note:

  This name must follow conventions for naming Java components as described in Conventions for naming Java components on page 59.

- **Label ID** : Select a label to describe the **Picture Input**. Labels can be created in a `textset` file. For more information on textsets, see About textset files on page 125.

Important:

If you do not give a **Label ID** for a **Picture Input**, Orchestration Designer generates an error.

Send Event 

Type

Workflow Palette item

Available from

Workflow palette of a **Data** node in a web application.

Purpose

Using the **Send Event** item, you can send an event to a specific instance of a workflow on Avaya Breeze® platform.

Behavior

The event that you send using the **Send Event** item either has the collected data or the context ID where the data is stored. After receiving this event Avaya Breeze® platform understands that Orchestration Designer has finished collecting the data, and then uses this data for further processing.

Properties

- **Event Type**: Enter the name of the event that you want to send.
- **InstanceId Variable**: Select the variable for the ID of the workflow instance to which you want to send the event.
- **InstanceId Variable Field**: Select the field of the variable for the ID of the workflow instance to which you want to send the event.
- **Parameter (N) Name**: Enter the name of the parameter that you want to pass to the event. The default name of a parameter is param(N). You can change the name of the parameter. Ensure that the parameter names match the names of the event defined on Avaya Breeze® platform.
- **Parameter (N) Variable**: Select the variable having the parameter that you want to pass to the event. You can pass up to 8 parameters.
- **Parameter (N) Variable Field**: Select the field of the variable having the parameter that you want to pass to the event.
- **Parameter (N) Constant**: Enter the constant value that you want to pass as a parameter to the event.

Set Context Item 

Type

Workflow Palette item

Available from

Workflow palette of a **Data** node in a web application.

Purpose

Use the **Set Context Item** item to store data into a context in the Context Store.

Behavior

Set Context Item stores a single value in a context of the Context Store.

Properties

- **Context ID Variable**: Select the variable that contains the context ID of the context that you want to access.
- **Context ID Variable Field**: Select the field of the variable that contains the context ID of the context that you want to access.
- **Context ID Constant**: A unique constant value for the context ID of the context that you want to access.
- **Key Variable**: Select the variable that contains the key or attribute name of the data item that you want to update or create.
- **Key Variable Field**: Select the field of the variable that contains the key or attribute name of the data item that you want to update or create.
- **Key Constant**: A constant value of the key or attribute name of the data item that you want to update or create.
- **Value Variable**: Select the variable that contains the value to be stored with the key in the context.
- **Value Variable Field**: Select the field of the variable that contains the value to be stored with the key in the context..
- **Value Constant**: Enter the constant value that you want to store with the key in the context.
- **LeaseTime Variable**: Select the variable that contains the time for which the context will remain active. When the lease time expires, the context is removed from the Context Store. The default is 3600 seconds.
- **LeaseTime Variable Field**: Select the field of the variable that contains the time for which the context will remain active.
- **LeaseTime Constant**: Enter the constant value as the lease time for the context.
- **Journey Element Variable**: Select the variable that contains the value stored in the audit trail of the context. **Journey Element Variable** is an optional property.
- **Journey Element Variable Field**: Select field of the variable that contains the value stored in the audit trail of the context.

- **Journey Element Constant**: A constant value that is stored in the audit trail of the context.

Set Context from Variable 

Type

Workflow Palette item

Available from

Workflow palette of a **Data** node in a web application.

Purpose

Use the **Set Context From Variable** item to store a variable into a context in the Context Store.

Behavior

Set Context From Variable stores a simple variable, complex variable, simple collection, or complex collection into the Context Store. It stores the entire variable (all fields and all collection elements) into the context store.

Properties

- **Context ID Variable**: Select the variable that contains the context ID of the context that you want to access.
- **Context ID Variable Field**: Select the field of the variable that contains the context ID of the context that you want to access.
- **Context ID Constant**: A unique constant value for the context ID of the context that you want to access.
- **Variable Name Variable**: Select the variable that you want to store into the context. If you provide the variable name and variable fields, then they are used as keys to access the context.
- **LeaseTime Variable**: Select the variable that contains the time for which the context will remain active. When the lease time expires, the context is removed from the Context Store. The default is 3600 seconds.
- **LeaseTime Variable Field**: Select the field of the variable that contains the time for which the context will remain active.
- **LeaseTime Constant**: Enter the constant value as the lease time for the context.
- **Journey Element Variable**: Select the variable using which you want to store the value in the audit trail of the context. **Journey Element Variable** is an optional property.
- **Journey Element Variable Field**: Select field of the variable using which you want to store the value in the audit trail of the context.
- **Journey Element Constant**: A constant value that is stored in the audit trail of the context.

Start Workflow

Type

Workflow Palette item

Available from

Workflow palette of a **Data** node in a web application.

Purpose

Using the **Start Workflow** item, you send an event to Avaya Breeze® platform to start a workflow.

Behavior

When you send an event to Avaya Breeze® platform using the **Start Workflow** item, then any workflow on Avaya Breeze® platform having this event as the “start” event receives this event.

Properties

- **Event Type**: Enter the name of the event that you want to send.
- **Workflow Name Variable**: Select the variable for the name of the workflow that you want to start.
- **Workflow Name Variable Field**: Select the field of the variable that contains the name of the workflow that you want to start.
- **Workflow Name Constant**: Enter the constant name of the workflow that you want to start.
- **Parameter (N) Name**: Enter the name of the parameter that you want to pass to the event. The default name of a parameter is param(N). You can change the name of the parameter. Ensure that the parameter names match the names of the event defined on Avaya Breeze® platform.
- **Parameter (N) Variable**: Select the variable having the parameter that you want to pass to the event. You can pass up to 8 parameters.
- **Parameter (N) Variable Field**: Select the field of the variable having the parameter that you want to pass to the event.
- **Parameter (N) Constant**: Enter the constant value that you want to pass as a parameter to the event.

Text Input

Type:

Form item

Available from:

Announce node

Collect node

Purpose:

The **Text Input** is used to collect user information in form of text in a web application. The collected input is then used in some way to direct the flow of the web application. It can also be used as data that can be used by the web application in a variety of ways.

Behavior:

Use the **Text Input** item when you want to collect user input for the node in the flow of a web project. The **Text Input** type property determines the data type of the text.

You must drag a **Text Input** item into the **Collect** or **Announce** node for each text input field that you want to show on a web page.

Note:

You can have more than one Text Input items in a single **Collect** or **Announce** node.

For each **Text Input** that you drag into the **Collect** or **Announce** node, Orchestration Designer automatically creates a simple variable with the same name. You can use these variables the same way as you would use any other variable.

Properties:

- **Name** : Enter the name you want to assign to the **Text Input** and its corresponding variable.

  If you change the name of this item in the Avaya Properties view, Orchestration Designer automatically changes the name of the associated variable as well.

  Note:

  This name must follow conventions for naming Java components as described in Conventions for naming Java components on page 59.

- **Label ID** : Select a label to describe the **Text Input**. Labels can be created in a `textset` file. For more information on textsets, see About textset files on page 125.

  Important:

  If you do not give a **Label ID** for a **Text Input**, Orchestration Designer generates an error.

- **Type** : Select the type of input user can enter in the **Text Input**. The type would determine the type of widget used for user input. For example, if date is selected, the input widget would popup a calendar for date selection on the web page.
- **Required** : Enter if the **Text Input** is mandatory or not.
- **Min Length** : Enter the minimum length of input the user should provide.
- **Additional Properties** : Specify additional properties if any in the form of name=value pair and separated by ‘,’ in case of multiple properties.

TextArea Input

Type:

Form item

Available from:

Announce node

Collect node

Purpose:

The **TextArea Input** is used to collect user information in the form of multi-line text in a web application. The collected input is then used in some way to direct the flow of the web application. It can also be used as data that can be used by the web application in a variety of ways.

Behavior:

Use the **TextArea Input** item when you want to collect user input for the node in the flow of a web project.

You must drag a **TextArea Input** item into the **Collect** or **Announce** node for each text input field that you want to show on a web page.

Note:

You can have more than one **TextArea Input** items in a single **Collect** or **Announce** node.

For each **TextArea Input** that you drag into the **Collect** or **Announce** node, Orchestration Designer automatically creates a simple variable with the same name. You can use these variables the same way as you would use any other variable.

Properties:

- **Name** : Enter the name you want to assign to the **TextArea Input** and its corresponding variable.

  If you change the name of this item in the Avaya Properties view, Orchestration Designer automatically changes the name of the associated variable as well.

  Note:

  This name must follow conventions for naming Java components as described in Conventions for naming Java components on page 59.
- **Label ID** : Select a label to describe the **TextArea Input**. Labels can be created in a `textset` file. For more information on textsets, see About textset files on page 125.

  Important:

  If you do not give a **Label ID** for a **TextArea Input**, Orchestration Designer generates an error.
- **Required** : Enter if the **TextArea Input** is mandatory or not.
- **Number of Rows** : Enter the number of rows that will be available in the **TextArea Input**.
- **Number of COLS** : Enter the number of rows that will be available in the **TextArea Input**.

Video Input

Type:

Form item

Available from:

Announce node

Collect node

Purpose:

The **Video Input** allows the user to record a video and provide it as an input. When you submit the input, the video file is sent to the application server for temporary storage.

When an end-user clicks **Next** on a web page, the video gets uploaded to the application server and is stored in the project's `data` or `temp` directory. The path of the file is assigned to the input variable. The application developer may save the video as the `temp` directory gets cleaned out by default at the end of the session. To play back the video in the subsequent node of the flow, you can use the **Web Element** in the **Prompt** editor. The same process happens in simulation testing on a desktop web browser. However, when running an application in simulation, no file is created; and everything happens on the web browser.

Behavior:

Note:

You can have more than one **Video Input** items in a single **Collect** or **Announce** node.

For each **Video Input** that you drag into the **Collect** or **Announce** node, Orchestration Designer automatically creates a simple variable with the same name. You can use these variables the same way as you would use any other variable.

Properties:

- **Name** : Enter the name you want to assign to the **Video Input** and its corresponding variable.

  If you change the name of this item in the Avaya Properties view, Orchestration Designer automatically changes the name of the associated variable as well.

  Note:

  This name must follow conventions for naming Java components as described in Conventions for naming Java components on page 59.

- **Label ID** : Select a label to describe the **Video Input**. Labels can be created in a `textset` file. For more information on textsets, see About textset files on page 125.

  Important:

  If you do not give a **Label ID** for a **Video Input**, Orchestration Designer generates an error.

Voice Input

Type:

Form item

Available from:

Announce node

Collect node

Purpose:

The **Voice Input** allows the user to record an audio and provide it as an input. When you submit the input, the audio file is sent to the application server for temporary storage.

Behavior:

Note:

You can have more than one **Voice Input** items in a single **Collect** or **Announce** node.

For each **Voice Input** that you drag into the **Collect** or **Announce** node, Orchestration Designer automatically creates a simple variable with the same name.

When an end-user clicks **Next** on a web page, the audio file gets uploaded to the application server and is stored in the project's `data` or `temp` directory. The path of the file is assigned to the input variable. The application developer may save the audio file as the `temp` directory gets cleaned out by default at the end of the session. To play back the audio in the subsequent node of the flow, you can use the **Web Element** in the **Prompt** editor. The same process happens in simulation testing on a desktop web browser. However, when running an application in simulation, no file is created; and everything happens on the web browser.

There is also a variation of experience between devices such as iPhone and Samsung. An iPhone just activates the same camera and audio application which it uses for video capturing, while Sumsung uses a specific application for voice recording.

Properties:

- **Name** : Enter the name you want to assign to the **Voice Input** and its corresponding variable.

  If you change the name of this item in the Avaya Properties view, Orchestration Designer automatically changes the name of the associated variable as well.

  Note:

  This name must follow conventions for naming Java components as described in Conventions for naming Java components on page 59.

- **Label ID** : Select a label to describe the **Voice Input**. Labels can be created in a `textset` file. For more information on textsets, see About textset files on page 125.

  Important:

  If you do not give a **Label ID** for a **Voice Input**, Orchestration Designer generates an error.

Choice

Type

Menu item

Available from

Menu node

Purpose

Use **Choice** item inside a **Menu** node of a web application. The **Choice** items gives users the menu options to choose from.

Double click the **Choice** item to open the node specified in the next form attribute in the node editor.

Behavior

Each **Choice** item in a **Menu** node represents a single option available to the user. So, if you want to offer users a menu with four options, you must use four **Choice** items, one for each option.

The text for the **Choice** item can be either a static text coming from a textset ID of a textset, or it can be determined dynamically at runtime using a variable. To generate the text for the **Choice** item dynamically, you must set the variable that will contain the text of the textset ID in the **Text Variable for Choice** property of the **Choice** item.

Properties

Enter values for the following properties:

- **Name**: Enter a descriptive name to indicate what this choice represents.
- **Name Text ID**: Select the textset ID that contains the text that you want to display for the **Choice** item.
- **Description Text ID**: Select the textset ID that contains the description for the **Choice** item.
- **Icon**: Specify the icon, if any, that you want to display with the **Choice** item.
- **Next Form**: Specify the next node this choice goes to when selected.
- **Text Variable for Choice**: Specify the variable that will contain the text of the textset ID.
- **Variable Field**: If you select a complex variable in **Text Variable for Choice** field, then in **Variable Field** you must select the complex variable in which you want to store the choice values. The **Variable Field** property is unavailable if you select a simple variable in the **Text Variable for Choice** field.

Chapter 42: Resources

Related resources

Documentation

The following table lists the documents related to this product. Download the documents from the Avaya Support website at http://support.avaya.com

Title	Description	Audience
Getting Started with Avaya Orchestration Designer	This PDF document contains the information needed to install and configure Orchestration Designer for initial use, and to understand the basics of Orchestration Designer graphical user interface (GUI).	• Application Developers • Implementation Engineers
Avaya Orchestration Designer Developer's Guide	This PDF document contains the same information as available in the online Help, but you can view or print the document using Adobe Acrobat Reader.	Application Developers
Avaya Orchestration Designer online Help	The online Help provides detailed information and procedures for using Orchestration Designer features and options to create speech, message, and call control applications. When installing Orchestration Designer, the system installs the online Help as an additional Eclipse plug-in.	• Application Developers • Implementation Engineers
Programmer Reference Guide	This online documentation is designed for the programmers of Orchestration Designer. This documentation includes: • A Constants (Quick reference) guide. • A Class Hierarchy reference guide. • An API Reference guide.	Application Developers

Table continues…

Title	Description	Audience
Deploying Avaya Breeze® platform	This document provides information and procedures on deploying the Avaya Breeze® platform services.	• System administrators • Services and Support personnel • Avaya Professional Services • Implementation engineers
Avaya Context Store Snap-in Reference	This document is intended for anyone who wants to install, configure, and administer Context Store.	• Application Developers • Implementation Engineers
Deploying Avaya Analytics™	This document is intended for anyone who wants to deploy Avaya Analytics™.	• Application Developers • Implementation Engineers

For information about viewing the Orchestration Designer documentation, see Viewing the Orchestration Designer documentation on page 33.

Finding documents on the Avaya Support website

Procedure

1. Go to https://support.avaya.com.
2. At the top of the screen, type your username and password and click **Login**.
3. Click **Support by Product** > **Documents**.
4. In **Enter your Product Here**, type the product name and then select the product from the list.
5. In **Choose Release**, select the appropriate release number.

   The **Choose Release** field is not available if there is only one release for the product.
6. In the **Content Type** filter, click a document type, or click **Select All** to see a list of all available documents.

   For example, for user guides, click **User Guides** in the **Content Type** filter. The list only displays the documents for the selected category.
7. Click **Enter**.

Avaya Documentation Center navigation

The latest customer documentation for some programs is now available on the Avaya Documentation Center website at https://documentation.avaya.com.

Important:

For documents that are not available on Avaya Documentation Center, click **More Sites** > **Support** on the top menu to open https://support.avaya.com.

Using the Avaya Documentation Center, you can:

- Search for content by doing one of the following:
  - Click **Filters** to select a product and then type key words in **Search**.
  - From **Products & Solutions**, select a solution category and product, and then select the appropriate document from the list.
- Sort documents on the search results page.
- Click **Languages** ( ) to change the display language and view localized documents.
- Publish a PDF of the current section in a document, the section and its subsections, or the entire document.
- Add content to your collection by using **My Docs** ( ).

  Navigate to the **Manage Content** > **My Docs** menu, and do any of the following:
  - Create, rename, and delete a collection.
  - Add topics from various documents to a collection.
  - Save a PDF of selected content in a collection and download it to your computer.
  - Share content in a collection with others through email.
  - Receive collection that others have shared with you.
- Add yourself as a watcher using the **Watch** icon ( ).

  Navigate to the **Manage Content** > **Watchlist** menu, and do the following:
  - Enable **Include in email notification** to receive email alerts.
  - Unwatch selected content, all content in a document, or all content on the Watch list page.

  As a watcher, you are notified when content is updated or deleted from a document, or the document is removed from the website.
- Share a section on social media platforms, such as Facebook, LinkedIn, and Twitter.
- Send feedback on a section and rate the content.

Note:

Some functionality is only available when you log on to the website. The available functionality depends on the role with which you are logged in.

Training

The following courses are available on the Avaya Learning website at www.avaya-learning.com. After logging into the website, enter the course code or the course title in the **Search** field and click **Go** to search for the course.

Course Code	Course Title
4C00095W	Avaya Orchestration Designer for Developers
5C00092V	Avaya Experience Portal, Avaya Orchestration Designer, and Avaya Proactive Outreach Manager Installation, Maintenance and Troubleshooting Essentials
3610C	Avaya Aura Contact Center - Orchestration Designer Scripting
2C00081O	Selling Avaya Orchestration Designer
5C00080E	Knowledge Access: Avaya Aura Contact Center Orchestration Designer Scripting Administration
W: Web (online) course V: Virtual E: Self-paced in virtual campus O: On Demand	

Viewing Avaya Mentor videos

Avaya Mentor videos provide technical content on how to install, configure, and troubleshoot Avaya products.

About this task

Videos are available on the Avaya Support website, listed under the video document type, and on the Avaya-run channel on YouTube.

- To find videos on the Avaya Support website, go to https://support.avaya.com/ and do one of the following:
  - In **Search**, type `Avaya Mentor Videos`, click **Clear All** and select **Video** in the **Content Type**.
  - In **Search**, type the product name. On the Search Results page, click **Clear All** and select **Video** in the **Content Type**.

    The **Video** content type is displayed only when videos are available for that product.

  In the right pane, the page displays a list of available videos.

- To find the Avaya Mentor videos on YouTube, go to www.youtube.com/AvayaMentor and do one of the following:
  - Enter a key word or key words in the **Search Channel** to search for a specific product or topic.
  - Scroll down Playlists, and click a topic name to see the list of videos available for the topic. For example, Contact Centers.

Note:

Videos are not available for all products.

Support

Go to the Avaya Support website at https://support.avaya.com for the most up-to-date documentation, product notices, and knowledge articles. You can also search for release notes, downloads, and resolutions to issues. Use the online service request system to create a service request. Chat with live agents to get answers to questions, or request an agent to connect you to a support team if an issue requires additional expertise.

Using the Avaya InSite Knowledge Base

The Avaya InSite Knowledge Base is a web-based search engine that provides:

- Up-to-date troubleshooting procedures and technical tips
- Information about service packs
- Access to customer and technical documentation
- Information about training and certification programs
- Links to other pertinent information

If you are an authorized Avaya Partner or a current Avaya customer with a support contract, you can access the Knowledge Base without extra cost. You must have a login account and a valid Sold-To number.

Use the Avaya InSite Knowledge Base for any potential solutions to problems.

1. Go to http://www.avaya.com/support.
2. Log on to the Avaya website with a valid Avaya user ID and password.

   The system displays the Avaya Support page.
3. Click **Support by Product** > **Product-specific Support**.
4. In **Enter Product Name**, enter the product, and press `Enter`.
5. Select the product from the list, and select a release.

6. Click the **Technical Solutions** tab to see articles.
7. Select relevant articles.

Glossary

AACC	Avaya Aura® Contact Center.
AAEP	Avaya Experience Portal
AAS	Avaya Application Simulator.
ADR	See application detail record (ADR) on page 888.
AMS	Avaya Aura® Media Server.
ANI	See automatic number identification (ANI) on page 888.
API	See application program interface (API) on page 888.
application detail record (ADR)	Data records which contain historical information about an application used as part of a session. These records include information such as the session ID number, a timestamp, and a “friendly name” string determined by the developer who created the application.
application program interface (API)	A set of routines, protocols, and tools for building software applications. A good API makes it easier to develop a program by providing all the building blocks.
application server	A computer on which the Orchestration Designer speech application resides and runs. This computer is also where the Orchestration Designer run-time libraries are installed, thus making it possible to run Orchestration Designer applications on that server. The IVR system must be configured to start the speech application from this location.
ASR	See automated speech recognition (ASR) on page 888.
automated speech recognition (ASR)	Technology that employs a computer to recognize spoken words and respond appropriately.
automatic number identification (ANI)	A service that provides the originating telephone number of a call coming in to the system.
call flow	As implemented in speech applications, the call flow determines how a call is handled when it enters an interactive voice response system, based on options offered to callers and their responses to those options.

CCXML

Call Control eXtensible Markup Language.

An emerging XML specification, being developed to work in conjunction with VoiceXML and which addresses some of the technical limitations of VoiceXML. It enables the processing of asynchronous events, filtering and routing of incoming calls, and placement of outbound calls. Note that it is not intended to replace VoiceXML but rather to supplement it. See Ian Moraes's excellent article, "VoiceXML, CCXML, SALT: Architectural Tools for Enabling Speech Applications," on the Internet.

Computer Telephony Integration (CTI)

Software technology that integrates the use of telephones and computers without the need for special telephones, connectors, computer circuit packs, or other specialized hardware.

CTI

See Computer Telephony Integration (CTI) on page 889.

dialed number identification service (DNIS)

A service that identifies for the receiving system what telephone number was dialed by the caller. In theAvaya Experience Portal system this is often used to direct the call to a particular speech application, which is identified with that dialed number.

DNIS

See dialed number identification service (DNIS) on page 889.

DTMF

See dual tone multi-frequency (DTMF) on page 889.

dual tone multi-frequency (DTMF)

The system used by touchtone telephones, DTMF assigns a specific frequency (consisting of two separate tones) to each telephone key on the keypad, so that it can easily be identified by a microprocessor.

Eclipse

A Java-based open-source extensible IDE (integrated development environment) that provides application developers a feature-rich interface to develop their applications. Orchestration Designer is designed as a set of Eclipse plug-in modules that make it possible for application developers to design and build speech applications without having to write the code manually.

gateway

A network point that acts as an entry point to another network. In the context of Orchestration Designer and VoIP applications, a gateway is the entry point, often associated with one or more switches, to the interactive voice response (IVR) system application environment. See interactive voice response (IVR) system on page 890.

grammar

Elements that recognize the input received through inbound voice calls and messages.

In the context of IVR or speech applications, a grammar is a predefined set of words or DTMF tones that a speech application uses in conjunction with automated speech recognition (ASR) technology to interpret and respond to caller inputs. That is, grammars are lists of possible responses that callers make when responding to prompts by using spoken replies.

	Grammars define which words or phrases the ASR engine can recognize and respond to. In the context of text-based applications, a grammar is a predefined set of words in a message application that a text-processing system can use to interpret and respond to an inbound SMS or email message. The text-processing system collects and recognizes the input from inbound SMS and email messages and uses this input to direct the flow of a message application.
H.323	A hierarchical, IP-based telephony standard for connecting IP telephones and speech applications to switches.
IC	See Interaction Center (IC) on page 890.
IDE	See integrated development environment (IDE) on page 890.
integrated development environment (IDE)	A software application that usually provides a GUI environment, a text and/or code editor, a compiler and/or interpreter, and a debugger. This environment means that application or web developers can develop, test, and build their applications or Web sites within a single integrated space.
Interaction Center (IC)	A multichannel contact management platform that enables businesses to align real-time contact center operations with business objectives.
interactive voice response (IVR) system	A system, such as Avaya Experience Portal or Avaya IR, in which callers interact with a self-service application to get information, conduct transactions, or help with problems.
IVR system	See interactive voice response (IVR) system on page 890.
JDBC	An application program interface (API) on page 888 specification in which programs written in Java connect with and access data contained in database programs using Structured Query Language (SQL) on page 892.
localization	The process of modifying an application to operate and be understood in a different language, or locale. This usually involves modifying any phrases, prompts, and grammars associated with an application.
MPS	Media Processing Server.
MRCP	Media Resource Control Protocol.
NDM	See Nuance Dialog Module (NDM) on page 891.
notebook	(Also known as a tabbed or stacked notebook) In the Eclipse context, a notebook is a set of views “stacked” on top of one another as a space saving measure. The views in the notebook are accessible by clicking

	tabs arranged along the top of the notebook. See the Eclipse documentation.
Nuance Dialog Module (NDM)	Speech application modules produced by Nuance software products, similar to speech application modules created by using Orchestration Designer. You can use NDMs in the Orchestration Designer speech applications. Orchestration Designer supports NDM version 5.0 and later. Before version 5.0, Nuance Dialog Module (NDM) was known as Open Speech Dialog Module (ODSM). See Open Speech Dialog Module (OSDM) on page 891.
Open Speech Dialog Module (OSDM)	Speech application modules produced by Nuance software products, similar to application modules created with Orchestration Designer. OSDMs can be used in Orchestration Designer applications. (Orchestration Designer supports the following OSDM versions: Address OSDM 2.0.3, Core OSDM 2.0.4, and Name OSDM 2.0.1.) OSDM is renamed to Nuance Dialog Module (NDM) from version 5.0 and later.
OSDM	See Open Speech Dialog Module (OSDM) on page 891.
palette	In the Orchestration Designer Editor views, this is the pane to the left of the view, in which editor options are displayed and selected.
Real-time Transfer Protocol (RTP)	A protocol for transmitting “real-time” data, such as audio or video data, across the Internet. This protocol does not guarantee “real-time” delivery of such data, but it does provide mechanisms to support data “streaming.”
RTP	See Real-time Transfer Protocol (RTP) on page 891.
RTSP	The Real Time Streaming Protocol, serves as a control protocol, and as a jumping off point for negotiating transports, such as RTP, multicast and unicast, and negotiating codecs off of servers in a file format independent way.
SCE	See service creation environment (SCE) on page 891.
service creation environment (SCE)	A set of software tools used to develop, test, and debug speech applications. Orchestration Designer is an SCE.
servlet	A small program that runs on a server, often Java-based.
servlet engine	A program that coordinates the overall operation and integration of a number of servlets. In the context of Orchestration Designer, the supported servlet engines are Apache Jakarta Tomcat, IBM WebSphere/ WebSphere Express, Oracle WebLogic, and JBoss EAP/Wildfly.

Session Initiation Protocol (SIP)	A signaling protocol for the Internet that makes it possible to set up conferencing, telephony, events notification, and instant messaging. Within a VoIP framework, it initiates call setup, routing, authentication, to endpoints within an IP domain.
SIP	See Session Initiation Protocol (SIP) on page 892.
speech recognition	See automated speech recognition (ASR) on page 888.
speech synthesis	See text-to-speech (TTS) on page 893.
speech user interface (SUI)	Any software interface in which the user interacts with the system using speech commands and audio prompts.
SQL	See Structured Query Language (SQL) on page 892.
SSL	Secure Sockets Layer. A protocol for transmitting private data securely over the Internet. By convention, URLs that use SSL require a connection using the HTTPS protocol, rather than just HTTP.
SSML	Speech Synthesis Markup Language. A W3C standard designed to provide an XML-based markup language for assisting with the generation of synthetic speech in Web and other applications. The essential role of the markup language is to provide authors of synthesizable content a standard way to control aspects of speech such as pronunciation, volume, pitch, rate, and so forth, across different synthesis-capable platforms.
stacked notebook	See notebook on page 890.
Structured Query Language (SQL)	A standard interactive and programming language for getting data to and from a database.
SUI	See speech user interface (SUI) on page 892.
tabbed notebook	See notebook on page 890.
TDD	See Telecommunications Display Device (TDD) on page 892.
Telecommunications Display Device (TDD)	Sometimes designated as a teletypewriter (TTY) device, a telephone equipped with a keyboard and display, used by hearing- or speech-impaired callers to send and receive typed messages.
telephone user interface (TUI)	Any software interface in which the user interacts with the system using a telephone or similar device.
teletypewriter (TTY) device	See Telecommunications Display Device (TDD) on page 892.

text-to-speech (TTS) Technology by which information in text format is rendered as audio output using a speech synthesis engine to simulate human speech.

TTS See text-to-speech (TTS) on page 893.

TTY See Telecommunications Display Device (TDD) on page 892.

TUI See telephone user interface (TUI) on page 892.

vector A user-defined sequence of functions that may be performed, such as routing the call to a destination, giving a busy signal, or playing a recorded message.

VoiceXML (Sometimes presented as VXML) Voice eXtensible Markup Language.

A specification which provides for a user to interact with Internet-based resources using voice recognition technology. Instead of a typical Web browser that requires a combination of HTML, keyboard, and mouse device, VoiceXML relies on an Internet voice browser and/or telephone. Using VoiceXML, the user interacts with the Web “page” by listening to audio outputs (either pre-recorded or using a technology such as TTS) and by submitting input in the form of the user’s natural speaking voice and/or manual responses, such as telephone key presses.

Web service A standardized way of offering Web-based applications or services. Because Web services are Web-based and standards-based applications, delivered over the Internet, Web services make it possible for organizations to communicate and share data that use different file formats and programming languages.

workspace In Orchestration Designer, the area within the Editor view used to build the functionality for the selected editor. For example, in the Call Flow Editor, this is the space to the right of the palette, in which you drag the nodes that represent application functions.

See palette on page 891.

WSDL Web services Description Language.

An XML-formatted language used to describe a Web service’s capabilities.

XML eXtensible Markup Language.

A specification for the presentation of Internet documents, one which expands on the capabilities of HTML. A pared-down version of SGML (Standard Generalized Markup Language), XML makes it possible for designers to create their own customized tags, which in turn makes it possible to do things over the Internet that cannot be done using simple HTML.

Index

Special Characters

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

Z