Avaya WebRTC Connect Software Development Guide

Release 4.0.6
Issue 1.0
April 2023

AVAYA SOFTWARE DEVELOPMENT KIT LICENSE AGREEMENT

REVISED: October 14, 2019

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.

1.0 DEFINITIONS.

1.1 "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.

1.2 "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.

1.3 "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.

1.4 "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 the Licensee.

1.5 "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.

1.6 "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.

1.7 “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).

1.8 “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.

1.9 “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.

1.10 “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.

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

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

2.0 LICENSE GRANT.

2.1 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.

2.2 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.

2.3 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.

2.4 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: http://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.

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

2.6a 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.

2.6.b 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 (Affero GPL) v.3 or similar licenses; (v) access information that is solely available to root administrators of the Avaya products, systems, and solutions; (vi) develop 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 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.

2.7 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.

2.8 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.

2.9 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.

2.10.1 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.

2.10.2 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.

2.11 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(a) 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.

2.12(b) 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.

2.13 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.

2.14 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.

2.15 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.

2.16 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.

2.17 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.

3. OWNERSHIP.

3.1 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.

3.2 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.

4.0 SUPPORT.

4.1 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.

4.2 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.

5.0 CONFIDENTIALITY.

5.1 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.

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

6.0 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.

7.0 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.

8.0 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.

9.0 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.

10.0 TERM AND TERMINATION.

10.1 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.

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

10.3 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.

10.4 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.

10.5 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.

11.0 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.

12.0 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.

13.0 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.

14.0 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.

15.0 GOVERNING LAW AND DISPUTE RESOLUTION.

15.1 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.

15.2 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.

15.3 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.

15.4 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.

15.5 Injunctive Relief. Nothing in this Agreement will be construed to preclude either party from seeking provisional remedies, including, but not limited to, temporary restraining orders and preliminary injunctions from any court of competent jurisdiction in order to protect its rights, including its rights pending arbitration, at any time. The parties agree that the arbitration provision in Section 15.3 may be enforced by injunction or other equitable order, and no bond or security of any kind will be required with respect to any such injunction or order.

15.6 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.

16.0 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.

17.0 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.

18.0 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.

19. 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

1. **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

Table of Contents

Introduction

This document is the developer guide accompanying the Avaya WebRTC Connect SDK. It contains instructions and information for developers seeking to build and test Web Voice and Video enabled applications on Android, iOS and JavaScript platforms.

- **IMPORTANT**: From Avaya Oceana™ 3.4.0.1 the Web Voice and Video customer client SDKs have been renamed to OceanaCustomerWebVoiceVideo for all three platforms. Please refer to the "Changes from 3.4.0.1" and "Backwards Compatibility" sections for further information and changes.
- **IMPORTANT: This SDK is an update to the existing OceanaSDK and now support for Elite platform has been added as well. As a solution it has been renamed to Avaya WebRTC Connect SDK.**

Target audience

This document is targeted at experienced Android, iOS and JavaScript developers intending to create applications, which utilize the Avaya WebRTC Connect APIs and interact with Avaya Oceana™ and Avaya Workspaces for Elite infrastructure to originate WebRTC Voice and Video interactions with the contact centre agents.

Developers have to build these applications using their own resources and deploy them against their Avaya Oceana™ and Elite environment.

Related resources

Documentation

The following table lists the related documents for Avaya WebRTC Connect. Download the documents from the Avaya Support website at https://support.avaya.com

Title	Description
Avaya Oceana™ Solution Description	Describes Avaya Oceana™ characteristics and capabilities, including feature descriptions and interoperability
Deploying Avaya Oceana™ Solution	Details the specifics of installing and configuring the Avaya Oceana™ solution.
Avaya Workspaces Elite Solution Description	Describes Avaya Workspaces Elite Solution characteristics and capabilities, including feature descriptions and interoperability
Deploying Avaya Workspaces for Elite	Details the specifics of installing and configuring the Avaya Oceana™ solution

Support

Visit the Avaya Support website at 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.

Objectives

The objective of this document is to enable appropriately skilled developers to develop browser and mobile applications similar to the sample reference clients provided as a part of the Avaya WebRTC Connect for Android, iOS and JavaScript.

This document includes an overview of the Avaya WebRTC Connect infrastructure, design as well as pointers and insights into project related details.

Capabilities

The Avaya WebRTC Connect solution allows developers to enable and enrich their applications with voice and video capabilities across multiple platforms.

The most straightforward use case is a simple help button located on a screen or as a menu option, which can start a Voice/Video interaction into Avaya Oceana™ and Elite.

However, it is possible to set up more nuanced use cases for enabling an application with Voice/Video. For example, an attribute set could be built to match a specific pool of contact centre agents based on the page of a website that the customer is viewing.

The customer might be viewing a support page for a smartphone, and so the attributes would route to an agent with the “support” and “smartphone” attributes.

Ultimately developers may analyse how Avaya WebRTC Connect functionality may be integrated with planned or existing web and mobile applications.

Prerequisites and constraints

- Developers must possess the appropriate developer environments and tools and be familiar with their usage.
- For mobile development, developers will require the knowledge and system permissions/provisioning certificates etc. required to install their custom-built applications onto their Android and iOS devices.
- For JavaScript development, developers will require a web application server and the ability to deploy their applications and all associated resources to the web application server.
- The supported web browsers are:
  - Chrome stable versions 77 to 84.
  - Firefox stable versions 77 to 79.
  - Microsoft Edge Chromium version 83.
- The supported iOS versions are 10.X to 16.X
- The supported Android versions are 8.X to 13.X
- High-quality network connections with sufficient bandwidth are required for voice and video calls.
- Some familiarity with Contact Centre concepts is assumed and knowledge of Avaya Oceana™ and Elite components and systems will be beneficial.
- This document does not provide guidance on exception handling, logging, or other crosscutting programming practices, which should be implemented in developed applications according to developer preference and applicable standards.

  IMPORTANT: Avaya doesn't recommend nor support the development of application on CSDK where users of your application will be dependent on your application as their primary means of communication in emergency situation

Reference Clients

A separate Avaya WebRTC Connect application is available for each of the Android, iOS and JavaScript platforms. For each of these platforms, a reference client project is available which demonstrates how to use each function in the SDK. On Android, an Android Studio project is provided. On iOS, the reference client is an Objective-C based XCode project. The JavaScript reference client is provided as a HTML5 project, ready to be hosted on a web server.

Downloading the Reference Clients and SDKs

SDKs and other developer resources are hosted on the Avaya DevConnect portal. Visit the portal and then select Avaya WebRTC Connect from the Product & Resources mega menu. Base-level registered membership and a login is required to download resources. Here is the direct link: Avaya WebRTC Connect.

Using the Reference Clients

The reference clients require a fully operational Web Voice and Video deployment of the Avaya Oceana™ OR Avaya Aura Elite solution to function correctly and provide the full range of capabilities.

- To use the Android reference client, unzip the relevant archive retrieved in the previous section, and install the AvayaWebRTCConnectReferenceClient.apk on your supported Android phone or tablet.
- To use the iOS reference client, unzip the relevant archive retrieved in the previous section on an Apple Mac, and double click on the .xcodeproj file. This will open the XCode project and the reference client can now be built from within XCode.
- To use the JavaScript reference client, unzip the relevant archive retrieved in the previous section, and copy the folder to your web server. The JavaScript reference client should now be reachable on your web server.

More information on how to build your own applications using the Avaya WebRTC Connect is available in the platform-specific sections below.

Backwards Compatibility

It is strongly recommended to align the AvayaWebRTCConnect SDKs version with that of the Oceana™ and Avaya Aura Elite solution version installed in your production environment due to various updates and bug fixes that are applied to the SDKs.
The table below displays the backwards compatibility of the AvayaWebRTCConnect SDKs:

OceanaCustomerWebVoiceVideo SDK Version	**3.5.0.0**	**3.5.0.1**	**3.6.0.0**	**3.7.0.1**
< 3.5.0.0	Incompatible	Incompatible	Incompatible	Incompatible
3.5.0.0	Compatible	Compatible	Compatible	Compatible
3.5.0.1	Compatible	Compatible	Compatible	Compatible
3.6.0.0	Compatible	Compatible	Compatible	Compatible
3.6.1.0	Compatible	Compatible	Compatible	Compatible
3.7.0.1	Compatible	Compatible	Compatible	Compatible
3.8.0.0	Compatible	Compatible	Compatible	Compatible

AvayaWebRTCConnect SDK	**Avaya Oceana 3.7**	**Avaya Oceana 3.8**	**Avaya Aura Elite 3.8**
4.0.6	Incompatible (See Note below)	Compatible	Compatible
4.0.5	Incompatible (See Note below)	Compatible	Compatible
4.0.4	Incompatible (See Note below)	Compatible	Compatible
4.0.1.0	Incompatible (See Note below)	Compatible	Compatible
4.0	Incompatible (See Note below)	Compatible	Compatible

NOTE: For Avaya WebRTC Connect 4.0 to work with Avaya Oceana 3.7, a patch is in preparation. Please see the upcoming patch announcement.

Changes in 4.0.6 release

JavaScript

- Updated lodash library version to 4.17.21

Changes in 4.0.5 release

iOS

- Updated for compatibility with XCode 14 and iOS SDK 16.

Android

- Updated for compatibility with Android SDK 33.

Changes in 4.0.4 release

iOS

- Frameworks are now converted to XCFramework format which support both simulator and device. More details under Supported Build Architectures.
- Logging delegate exposed to access the logs framework logs. More details under Logging.

Changes in 4.0.1.0 release

Android

- The border width and rounded corners for local video are now configurable. videoCornerRadius and videoBorderWidth attributes have been added to VideoSurfaceView to control the corner style and border for local video.
- CallQuality is the new enum added.
- AudioInteractionListener and VideoInteractionListener interface have been updated to add onInteractionQualityChanged(CallQuality quality) method that continuously returns the CallQuality on an active call.

iOS

- The issue for displaying empty video screen until agent answers the call is resolved. Now the remote video view does not render the blank stream.
- AOCallQuality is the new enum added.
- AOAudioInteractionDelegate and AOVideoInteractionDelegate interface have been updated to add onInteractionQualityChanged:(AOCallQuality) quality method that continuously returns the AOCallQuality on an active call.

JS

- CallQuality is the new enum added.

- AudioInteraction interface has been updated to add onAudioInteractionCallQuality(CallQuality quality) and VideoInteraction interface has been updated to add method onVideoInteractionCallQuality(CallQuality quality) that continuously returns the CallQuality on an active call.

Changes in 4.0.0.0

- The SDK has been renamed to AvayaWebRTCConnect.
- Updated the APIs to set video capture resolutions and video capture orientation in iOS and Android. Also added APIs to get available video capture resolutions.
- AvayaWebRTCConnect supports audio/video interactions with Avaya Oceana and Avaya Aura Elite platforms. Using 'setPlatformType' API, the platform can be set before placing audio or video interaction.
- The APIs have been aligned across all three platforms. See below for more information on platform specific changes as a result.
- Sample reference clients for all platforms have been updated with Audio and Video Statistics parameters to be shown during a call.

Android

- audioInteraction.setPlatformType(platformType) is a new function available on the interaction object.
- videoInteraction.setPlatformType(platformType) is a new function available on the interaction object.
- videoDevice.setVideoCaptureResolutionWithCaptureOrientation(videoCapturePreference, videoCaptureOrientation) is a new function available on the com.avaya.ocs.Services.Device.Video.VideoDevice object to set the camera capture resolution with Orientation.
- VideoCapturePreference is a new enum for providing resolution setting to the VideoDevice.
- VideoCaptureOrientation is a new enum for providing orientation setting to the VideoDevice.
- videoDevice.getVideoCapturePreference is new API to get the camera capture resolution as a VideoCapturePreference object.
- videoDevice.getVideoCaptureOrientation is new API to get the camera capture resolution as a VideoCaptureOrientation object.

iOS

- [audioInteraction setPlatformType :]; is new API to set the platform type for Audio interaction.
- [videoInteraction setPlatformType :]; is new API to set the platform type for Video interaction.
- +(NSMutableArray*) getSupportedCameraCaptureResolutionPrefes; is new API which returns a list of Video Resolution preferences that are supported by the device.
- (void setVideoCaptureResolutionWithCaptureOrientation: (AOVideoCapturePreference) resolutionPreference orientationPreference : (AOVideoCaptureOrientation) orientationPreference; is new API to set the camera capture resolution with Orientation.
- AOVideoCapturePreference is a new enum for providing resolution setting to the AOVideoDevice.
- AOVideoCaptureOrientation is a new enum for providing orientation setting to the AOVideoDevice.
- -(AOVideoCapturePreference) getVideoCapturePreference; Is new API to get the camera capture resolution.

- -(AOVideoCaptureOrientation) getVideoCaptureOrientation; Is new API to get the camera capture orientation.

Javascript

- work.createAudioInteraction(OceanaCustomerWebVoiceVideo.Services.Work.PlatformType platformType) is an update to the existing function available on the work object.
- work.createVideoInteraction(OceanaCustomerWebVoiceVideo.Services.Work.PlatformType platformType) is an update to the existing function available on the work object.
- audioInteraction.holdCall(boolean) is a new function available on the interaction object.
- audioInteraction.isCallHeld() is a new function available on the interaction object.
- audioInteraction.setPlatformType() is a new function available on the interaction object.
- videoInteraction.holdCall(boolean) is a new function available on the interaction object.
- videoInteraction.isCallHeld() is a new function available on the interaction object.
- videoInteraction.setPlatformType() is a new function available on the interaction object.

Changes in 3.7.0.1-1 Android Patch

- For Android platform, issue related to setting trusted CA's using subdomain configuration under network-security-config has been resolved.

Changes in 3.6.1.0-1 iOS/Android Patch

- For Android and iOS client platforms, a new interaction callback 'onDiscardComplete' is available on the Audio and Video Interaction objects.
  See platform API documentation provided along with Avaya WebRTC Connect SDK and Android/iOS sections in this document for more detailed information.

Changes in 3.5.0.1

- For Android and iOS client platforms, a new optional property 'webGatewayUrlPath' is available on the WebGatewayConfiguration object. See platform API documentation provided along with Avaya WebRTC Connect SDK for more detailed information.

Changes in 3.5.0.0

- For all three client platforms the SDK has been renamed in 3.5.0.0 to OceanaCustomerWebVoiceVideo.
- The APIs have been aligned across all three platforms. See below for more information on platform specific changes as a result.
- A new property 'Topic' is available on the Work object for all platforms. See 'Topic' section for more information.

Android

- .aar file renamed to oceanacustomerwebvoicevideo.aar.
- libacbjnglpeerconnection_so.so dependency is no longer required.
- Requesting an authorization token has changed. See “Creating an Application: Generic Elements” for more information.
- OceanaCustomerServices class has been renamed to OceanaCustomerWebVoiceVideo.
- OceanaCustomerWebVoiceVideo constructor accepts a new ClientConfiguration object.
- ClientConfiguration object contains the Config object and a new WebGatewayConfiguration object.
- acceptAllCertificates method has been removed.
- In the Work class, createAudioInteraction(context); now accepts Application instead of ApplicationContext.
- In the Work class, createVideoInteraction(context); now accepts Application instead of ApplicationContext.
- New property Topic is available on the Work object.
- Enum DTMFType renamed to DTMFTone.
- videoInteraction.mute(boolean); has been renamed to videoInteraction.muteAudio(boolean);
- videoInteraction.isMuted(); has been renamed to videoInteraction.isAudioMuted();
- videoInteraction.readAudioDetails(AudioDetailsCallback) is a new method available on the interaction.
- videoInteraction.readVideoDetails(VideoDetailsCallback) is a new method available on the interaction.
- VideoSurface and VideoSurfaceImpl have been removed and replaced with VideoSurfaceView object.
- videoDevice.setLocalVideoView(videoSurface); and videoDevice.setRemoteVideoView(videoSurface); has been replaced by videoDevice.setVideoSurface(viewGroup);
- videoDevice.getSupportedCameraCaptureResolutions(cameraType); no longer accepts CameraType as a parameter.
- audioInteraction.mute(boolean); has been renamed to audioInteraction.muteAudio(boolean);
- audioInteraction.isMuted(); has been renamed to audioInteraction.isAudioMuted();
- audiointeraction.readAudioDetails(AudioDetailsCallback) is a new method available on the interaction.

iOS

- .framework file renamed to AvayaWebRTCConnect.framework.
- AvayaClientSDK.framework dependency replaced by AvayaClientServices.framework dependency.
- New dependency required: RVVideoCodec.framework.
- Requesting an authorization token has changed. See “Creating an Application: Generic Elements” for more information.
- AOOceanaCustomerServices class has been renamed to AOOceanaCustomerWebVoiceVideo.
- initWithIsSecure: method in AOOceanaCustomerWebVoiceVideo class has been replaced by: initWithClientConfiguration:clientConfiguration;
- New Configuration objects available – AOClientConfiguration which contains new objects: AOConfiguration and AOWebGatewayConfiguration.
- Optional method urlPath previously in AOOceanaCustomerServicesClass has been moved to AOConfiguration class.
- acceptAllCertificates method has been removed.
- New property Topic is available on the Work object.
- [videoInteraction start:delegate]; has changed to [videoInteraction start];
- [videoInteraction end:delegate]; has changed to [videoInteraction end];
- [videoInteraction sendDTMF:tone andPlayAudio:bool]; has changed to [videoInteraction sendDTMF:tone];
- [videoInteraction readVideoDetailsWithCompletionHandler:handler] is a new method available on the interaction.
- [videoInteraction readAudioDetailsWithCompletionHandler:handler] is a new method available on the interaction.
- [audioInteraction start:delegate]; has changed to [audioInteraction start];
- [audioInteraction end:delegate]; has changed to [audioInteraction end];
- [audioInteraction sendDTMF:tone andPlayAudio:bool]; has changed to [audioInteraction sendDTMF:tone];
- [audioInteraction readAudioDetailsWithCompletionHandler:handler] is a new method available on the interaction.
- [aoDevice getAvailableResolutions]; has changed to: [aoDevice getSupportedCameraCaptureResolutions];
- [aoDevice setResolution:resolution]; has changed to: [aoDevice setCameraCaptureResolution:resolution];
- [aoDevice setCameraType:type]; has changed to: [aoDevice selectCamera:type];
- AOVideoResolution enum has additional values.

JavaScript

- .js file renamed to OceanaCustomerWebVoiceVideo.js
- Requesting an authorization token has changed. See “Creating an Application: Generic Elements” for more information.
- OceanaCustomerServices object has been renamed to OceanaCustomerWebVoiceVideo.
- OceanaCustomerWebVoiceVideo now accepts a new config object which consists of Configuration and WebGatewayConfiguration.
- A new function, Topic, is available on the Work object.
- Enum DTMFDigit has been renamed to DTMFTone.
- videoInteraction.mute(boolean); function has been renamed to videoInteraction.muteAudio(boolean);
- videoInteraction.isMuted(); function has been renamed to videoInteraction.isAudioMuted();
- videoInteraction.readAudioDetails(callback) is a new function available on the interaction object.
- videoInteraction.readVideoDetails(callback) is a new function available on the interaction object.
- videoDevice.setLocalVideoView(view); function has been renamed to videoDevice.setLocalView(view);
- videoDevice.setRemoteVideoView(view); function has been renamed to videoDevice.setRemoteView(view);
- audioInteraction.mute(boolean); function has been renamed to audioInteraction.muteAudio(boolean);
- audioInteraction.isMuted(); function has been renamed to audioInteraction.isAudioMuted();
- audioInteraction.readAudioDetails(callback) is a new function available on the interaction object.

Creating an Application: Generic Elements

Certain objects and concepts are shared across all three SDKs. The shared elements are discussed in this section.

Authorization tokens

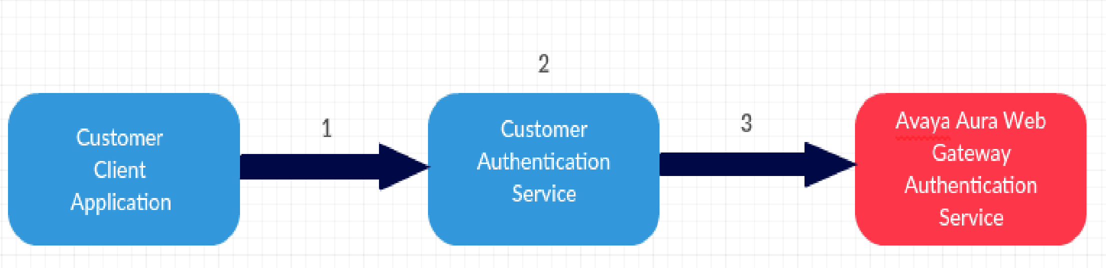
Avaya Aura Web Gateway provides the Web Voice and Video functionality. As such, Avaya Aura Web Gateway authorization token is required for all three client types – Android, iOS and JavaScript.

Avaya Aura Web Gateway Authorization service

The Avaya Aura Web Gateway authorization service is a HTTP-based service that provides authenticated clients with authorization tokens, which are necessary to create a call using the Avaya Aura Web Gateway Server. Clients (be they Android, iOS or JavaScript based) should **never** communicate with the authorization service directly. Avaya Aura Web Gateway will authorize requests only if they are presented with a valid client certificate that is trusted by the Avaya Aura Web Gateway and the client certificate FQDN must match one of those in the trusted hosts list.

These tokens are only valid for a short period of time, by which the client has to connect to the system and make a call. The token does not limit the call duration, it is just required to authorize the client to connect to the system, i.e. to validate a session. Once connected, there will be a default time of 120 seconds to initiate a call (this value can be overridden in the token request). If the session terminates, the client will have to request a new token in order to initiate another call.

The Avaya WebRTC Connect client application should call your own authentication service (Step 1 below), which will authenticate the client (Step 2), and then the authentication service should request the token from the Avaya Aura Web Gateway Server authorization service directly (Step 3) before responding to the client with the token.


Customer Client Application
1
2
Customer Authentication Service
3
Avaya Aura Web Gateway Authentication Service

Custom Web Service Retrieving an Authorization Token

Any web service that needs to create a token should delegate to the Avaya Aura Web Gateway to create the token.

The URL to access the token service is as follows:

```
https://<AAWGSERVER_ADDRESS>/csa/resources/tokens
```

To retrieve a session token, a HTTP POST must be performed on this interface, providing JSON in the following format:

```
{
  "use":””csaGuest”,

  “calledNumber”:<string>,

  “callingNumber”:<string>,

  “displayName”:<string>,

  “expiration”:”20000”
}
```

With the following content type:

```
application/vnd.avaya.csa.tokens.v1+json
```

Where the parameters are defined as follows:

- use – Mandatory field provided when creating the token. It indicates the type of token that will be generated. The only type of token allowed is “csaGuest”.
- calledNumber – Optional field also known as “Destination Address”. This is the number to dial to, the SIP call will be originated to this number. This number will be included in the username of the SIP URI, both in the request URI and the To URI. If not provided it will be encrypted as null. If null, then the token is valid to call any number.
- callingNumber – Optional field, also referred to as “From Address” or “UserName”. This is the number from which the call is made. If this is not provided, a random callingNumber is generated for the call. It is included in the username of the From SIP URI.
- displayName – Optional field. This is the friendly display name of the caller. This value will be included in the display name of the From URI. If this is not provided, then it will be empty.
- expiration – Optional field. This is the length of time the token is valid for in milliseconds, i.e. the time by which the client must connect to the system and make a call after receiving a token, not the call duration. If this is not specified in the request, a default value of 120 seconds will be used.

- **Note**: The callingNumber (also known as From Address) is used as a customers' unique identifier when initiating a Web Voice/Video call into the Contact Centre. Features such as Customer Journey, which use this identifier to link a customer's journey across multiple sessions, will be misrepresented if this value is hardcoded in the application. Therefore, a unique from address per customer is highly recommended. For instance, on mobile devices with cellular capabilities the userName value could use the telephone number associated with a SIM card.

Success Response

Status	Body

200 OK	{
	"encryptedToken" = <string>
	}

Failure Responses

Status	Description
400 Bad Request	"use": "csaGuest" is omitted or incorrect

Getting a Token from the Sample Token Generation Service

A sample token generation service is provided that in turn calls the Avaya Aura Web Gateway authorization service. It can be used by clients to request tokens and can be enabled on any deployed Avaya Aura Web Gateway System. The sample token generation service should **not** be used in a production environment, it is for testing and debug purposes only.

Please refer to the “Deploying Avaya Oceana™ Solution” and “Deploying Avaya Workspaces for Elite” guide for steps required to deploy the sample token generation service on the Avaya Aura Web Gateway.

A sample token-generation-service.war is also included in the DevConnect bundle which can be deployed on any webserver. This is an unsupported reference token service whose purpose is to act as a guide for developing a custom authentication service.
This token-generation-service.war must be deployed on a server which is configured with an FQDN that can be resolved by the Avaya Aura Web Gateway server and must have a valid server certificate for that FQDN.

Furthermore, when starting the application server, Java's network libraries must be configured with a keystore that contains the server certificate, e.g. launched with the following parameters:

```
-Djavax.net.ssl.keyStore=/path/to/server/cert/keystore.jks
-Djavax.net.ssl.keyStorePassword=PasswordForKeystore
```

The FQDN for the token server must be added to the trusted nodes list on Avaya Aura Web Gateway, and the CA of the server's certificate must be added to the Avaya Aura Web Gateway trust store.

As this test authentication service is intended to enable the analysis of integrating Avaya WebRTC Connect with existing or planned enterprise applications, no actual authentication is employed for token generation.

Therefore, the sample token generation service does not perform any validation. It passes the parameters provided onward to the actual authorization web service which generates the token to be returned. In a production scenario, the sample token generation service is replaced with a customer-developed web service which carries out whatever authentication is required by the enterprise’s own business rules. If this authentication succeeds, it continues the process by requesting a token from the authorization web service, then returning the generated token to the client requesting the Web Voice/Video call.

The Avaya WebRTC Connect SDK APIs do not enforce any requirements for the implementation of this token request, it can be sent using whatever code and practices are preferred by the individual developer and appropriate to the platform (examples can be found in the provided reference clients).

> **Note:**
> The reference clients can be configured (‘Token Config’ sections in each reference client) to point to the sample token generation service if it is deployed on the Avaya Aura Gateway server as per the “Deploying Avaya Oceana™ Solution” OR “Deploying Avaya Workspaces for Elite ”guide. It can be also be configured to point to a custom webserver if the sample token-generation-service.war is deployed on a webserver.

The entire returned token should be stored in an in-memory location or variable. It must later be set as a property on an ***interaction*** object (which precedes starting the call) and will also be required to invalidate the token.

When considering integration of Avaya WebRTC Connect with an enterprise application, it is important to understand the capabilities granted by the token. After the token is set as a property on an interaction, the user has the capability to make a call. The start() method can be called once per token. This means a token is valid for one call only. A new token is required to make a subsequent call.

Attributes

Before the API design is explored it is important to explain the concept of an attribute and how the Oceana™ Customer Web Voice/Video SDK uses attributes in the Avaya Oceana™ solution.

Attributes describe characteristics of the customer request. These attributes act on the routing rules when selecting the optimum resource to handle the request. The following are examples of attributes:

Language : English; Service : Sales
Language : Spanish; Service : Support

Attributes in the Avaya Oceana™ solution are used by Work Assignment. The Work Assignment Engine in Oceana™ finds the most suitable resource for a work item. It uses attribute-based Work and Resource matching. For more information on Work Assignment and its attribute-based matching feature consult "Avaya Work Assignment Snap-in Reference" available on Avaya DevConnect.
Please note that Elite platform does not support attributes settings.

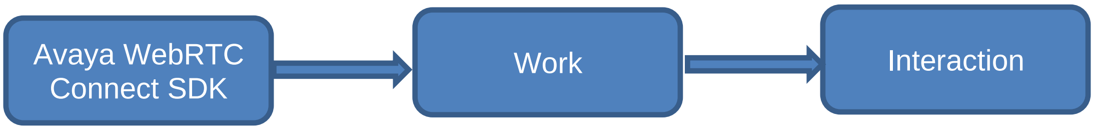
Basic API Design

Although there are minor platform-influenced differences between the three APIs, the basic model or flow is fundamentally the same across them.

1. First, an authorization token must be obtained, as per the previous section.
2. Secondly, **a Avaya WebRTC Connect SDK** object is initialized.
3. Thirdly, a **Work** object is initialized from the **Avaya WebRTC Connect SDK** object. Attributes are set on the Work object after it is initialized.
4. Fourthly, an **Interaction** object is initialized from the **Work** object. The authorization token is set as a property on the interaction object. This is the conceptual "call" object in the API and includes the start() and end() methods necessary for calls.

This basic model is shown below:

- **Note:** the exact terminology for these objects differs across APIs due to varying platform standards.



Avaya WebRTC Connect SDK is the top-level class, which configures the endpoints and gives access to other API elements. **Work** groups interactions under a work request identifier and attaches attributes to the particular work request to match the best contact centre resource for the interactions.

Finally, **Interaction** models the actual calls themselves and has call-related methods such as start and end. The Interaction objects have various callbacks available to notify an application of call events, which are discussed in detail later.

- **Note:** The Avaya WebRTC Connect APIs are designed to allow interactions to originate from applications which may also serve other purposes. For example, an online banking application might also include a “Live Help” button to request help from an agent.

Therefore, the recommended practice is to postpone the process until the user clicks the “Live Help” button. At that point (and not before), applications enabled with the Avaya WebRTC Connect SDK should start execution with the above steps.

Context

The Avaya WebRTC Connect SDK APIs allow for context to be set programmatically when making a call. Context is a string encapsulating some business logic, such as a customer reference number or an Order ID.

For example, if a user is browsing a shopping application, and clicks a call button while looking at a product, the product ID could be set as context to be retrieved when the call is presented to the agent, where a Screen Pop could show the same product page, allowing an agent to immediately see the product a customer is calling about.

The generic syntax is:

```
        work.setContext(context);
```

Please consult the relevant section for the platform you will be developing against for the exact API call.

For more information on configuring Screen Pops, please consult “Appendix A - Configuring Screen Pops”.

Topic

The Avaya WebRTC Connect SDK API allows for a Topic to be set programmatically when making a call. Topic is a string which can encapsulate any subject matter, such as the topic of discussion for the interaction.

For example, if a user is browsing a shopping application and clicks a call button while looking at their billing information, “billing” could be set as the topic to be retrieved when the call is presented to the agent. This is seen in the Customer Journey widget on the Workspaces browser client, if Customer Journey is configured in your Oceana™ or Elite solution.

An agent can select “billing” from the “Select Topic” dropdown in the Customer Journey widget on an active interaction and the agent can see all the previous interactions when that customer contacted the contact center about that specific topic.
The generic syntax is:

```
        work.setTopic(topic);
```

Please consult the relevant section for the platform you will be developing against for the exact API call.

User Interface design

The Avaya WebRTC Connect reference clients typically follow a multi-screen design. A click-to-call screen is followed by an in-call screen.

Developers should be cognisant of the state of the various controls that make up a Web Voice/Video application – for example, “call” buttons should be disabled after a call is initiated, and not enabled again until the call is ended. The following simple matrix shows some sample control states in various app screens for a voice call:

Control / type		Before Call	During Call	After Call
Make Web Voice Call	Button	Y		Y
Call State / Timer	Label		Y	
Mute Audio	Button		Y	
Send DTMF	Button		Y	
Hang Up	Button		Y	

The following simple matrix shows some sample control states in various app screens for a video call:

Control / type		Before Call	During Call	After Call
Make Web Video Call	Button	Y		Y
Call State / Timer	Label		Y	
Mute Audio	Button		Y	
Mute Video	Button		Y	
Enable Video	Button		Y	
Send DTMF	Button		Y	
Hang Up	Button		Y	

Note: This is not a definitive list. Depending on specific requirements, not all controls will apply in all cases. In some cases, other controls will be needed.

Writing an Android application

Setting up the project

Android developers can use their preferred Development Environments, such as Android Studio to build Avaya WebRTC Connect SDK enabled applications. The reference client provided with Avaya WebRTC Connect SDK was created using Android Studio and the steps that follow reflect that.

Please consult the section “Downloading the Reference Clients and SDKs” for information on where to download the relevant resources.

1. Create a new project in Android Studio as shown below.
2. Include the Avaya WebRTC Connect `(oceanacustomerwebvoicevideo.aar)` in the projects libs directory.
3. Open the app/build.gradle file and add a reference to the above aar e.g.

```
repositories {
  dirs 'libs'
  }

dependencies {
  implementation(name: 'oceanacustomerwebvoicevideo', ext: 'aar')
  implementation ‘io.netty:netty-all:4.1.17.Final’
  }
```

4. In order to allow your project to access the required features on Android devices, include the following permissions in the AndroidManifest.xml file

- `android.permission.INTERNET`
- `android.permission.RECORD_AUDIO`
- `android.permission.MODIFY_AUDIO_SETTINGS`
- `android.permission.CAMERA`

Initialize the SDK

When the application has detected the user is attempting to initiate an interaction into the contact centre, the Avaya WebRTC Connect SDK can be instantiated as follows:

OceanaCustomerWebVoiceVideo client = new OceanaCustomerWebVoiceVideo(clientConfiguration);

Where `clientConfiguration` is an instance of the ClientConfiguration class which contains the following configuration objects:

```
ClientConfiguration clientConfiguration = new ClientConfiguration();
```

Config – This object is specific to Avaya Oceana and not applicable for Avaya Aura Elite. It contains configuration properties for connecting to the AvayaMobileCommunications snap-in:

- `setRestUrl` – hostname or IP address of the AvayaMobileCommunications cluster.
- `setPort` – port to connect to the AvayaMobileCommunications cluster with, defaults to 443.
- `setSecure` – set to true for HTTPs; set to false for HTTP protocol to connect to the AvayaMobileCommunications cluster with.
- `setUrlPath` – optional, this allows the URL path to be configured when connecting to the AvayaMobileCommunications cluster. See API documentation provided along with Avaya WebRTC Connect SDK for more information.

```
Config config = new Config();Or Config config = new Config(“amc-cluster.avaya.com”);
config.setRestUrl(“amc-cluster.avaya.com”);
config.setSecure(true);
config.setPort(443);

clientConfiguration.setConfig(config);
```

WebGatewayConfiguration – contains configuration properties for connecting to the Avaya Aura Web Gateway Server:

- `setWebGatewayAddress` – hostname of the Avaya Aura Web Gateway server.
- `setPort` – port to connect to the Avaya Aura Web Gateway server with.
- `setSecure` – the protocol to connect to the Avaya Aura Web Gateway server with, set to true for HTTPs, set to false to use HTTP.
- `webGatewayUrlPath` – optional, this allows the URL path to be configured when connecting to the Avaya Aura Web Gateway server. See API documentation for more information.

Getter methods are also provided for the above setter methods.
For example:

```
WebGatewayConfiguration webGatewayConfiguration = new WebGatewayConfiguration();
webGatewayConfiguration.setWebGatewayAddress(“aawg-fqdn”);
webGatewayConfiguration.setPort(8443);
webGatewayConfiguration.setSecure(true);

clientConfiguration.setWebGatewayConfiguration(webGatewayConfiguration);
```

Creating Work

Next a ***Work*** object can be derived from the Avaya WebRTC Connect SDK object as follows:

```
        Work work = client.createWork();
```

The Work API provides a mechanism to populate the Oceana™ / Elite interaction schema. This interaction schema data is available for use by the Engagement Designer workflow engine (as can be seen in the Web Voice/Video workflow) and is used in requests to Work Assignment for resource selection.
For more information consult the API documentation provided along with Avaya WebRTC Connect SDK.

A Work object must be assigned services or resources for Oceana™ / Elite to be able to handle the work request. If both are provided, Work Assignment will try to assign the work to the provided resources. If the resources are unable to be assigned the work, Work Assignment will use the provided services to assign or queue the work.

Adding Services

```
List<Service> services = new ArrayList<>();
Service service = new Service();
service.setAttributes(attributes);
service.setPriority(5);
services.add(service);

work.setServices(services);
```

This code snippet creates a collection of services. Next a service is created and given attributes and a priority. Finally, the service is added to the services collection and set on the Work object.

Adding Resources

```
List<Resource> resources = new ArrayList<>();
Resource resource = new Resource();
resource.setNativeResourceId(“agent6220”);
resource.setSourceName(“CM”);
resources.add(resource);

work.setResources(resource);
```

This code snippet creates a collection of resources. Next a resource is created and given a resource id and source name. Finally, the resource is added to the resources collection and set on the Work object.

Adding Interaction Context

Optional context may be added to a work request as described in the section “Creating an Application: Generic Elements - Context”
To set a context on the work object, invoke the following method:

```
        work.setContext(context);
```

Where `context` is a string encapsulating some business logic, such as a customer reference number or an Order ID.

Adding an Audio Interaction

A voice call can be created by deriving an `AudioInteraction` from the Work object, as shown here:

```
AudioInteraction interaction = work.createAudioInteraction(application,listener);
```

Where `application` is the current Android application context. `listener is the OnAudioDeviceChangeListener object`

Setting platform type for an Audio Interaction (Oceana/Elite)

The Avaya WebRTC Connect can be used for Oceana and Elite interactions. Once Audio interaction's object is created, set the platform type as follows:

- `setPlatformType(com.avaya.ocs.Services.Work.Enums.PlatformType platformType)` // PlatformType.OCEANA or PlatformType.ELITE

The default platform type is com.avaya.ocs.Services.Work.Enums.PlatformType.OCEANA

Audio Interaction Callbacks

The Audio Interaction class provides `registerListener()` and `unregisterListener()` methods. This offers the ability to register a class which implements the following interaction status callbacks.
These callbacks are available in the `AudioInteractionListener` interface.
These are:

- `onInteractionInitiating()` – Called when the out-going request to initiate an interaction has been sent.
- `onInteractionRemoteAlerting()` – Called when the interaction has received ring back information from the signalling network.
- `onInteractionActive()` – Called when the interaction has been established.
- `onInteractionEnded()` – Called when the call on this interaction is ended. This callback method is called when the interaction is either locally or remotely ended.
- `onInteractionHeld()`–Sent when the interaction is on put hold.
- `onInteractionUnheld()`–Sent when the interaction is unheld
- `onInteractionHeldRemotely()`–Sent when the interaction is put on hold remotely.
- `onInteractionUnheldRemotely()`–Sent when the interaction is unheld remotely.
- `onInteractionAudioMuteStatusChanged(Boolean muted)` – Called when the interaction's audio is muted or unmuted.
- `onInteractionFailed(InteractionError error)` – Called when an error occurred establishing the interaction
- `onDiscardComplete()` – Called when the interaction has finished discarding
- `onInteractionQualityChanged(CallQuality quality) - Called when a interaction's quality state changes. Note that this API callback is called every few seconds to return the current quality.`
-

Starting an Audio Interaction

Before an Audio Interaction can be initiated, an Avaya Aura Web Gateway authorization token must be obtained as detailed in the "Creating an Application: Generic Elements - Authorization Tokens" section. Once a token has been obtained, it must be set on the Audio Interaction object:

```
        interaction.setAuthorizationToken(token);
```

Once the requisite Audio Interaction properties have been set, an Audio Interaction can be initiated by invoking:

```
interaction.start();
```

Optionally, before starting the interaction, `interaction.muteAudio(muted)` can be called. This takes a `Boolean` indicating if the customer's audio stream should be silenced or active when the interaction starts. If this is not set, the audio stream will be active by default.

Optionally, an interaction can be sent to the specified destination rather than the Avaya Oceana™ or Avaya Elite Web Voice/Video default routing number.

For example:

```
interaction.setDestinationAddress(destination);
```

Where `destination` is a string representing a remote SIP address.

During an Audio interaction

There are a number of actions that can take place on the Audio Interaction object during an active interaction.

These include:

- `sendDTMF(digit)` – sends the specified DTMF tone (an enumeration `DTMFDigit` is provided).
- `muteAudio(muted)` – takes a `Boolean` indicating if the audio stream should be silenced. The method `isAudioMuted()` indicates the current mute state.
- `readAudioDetails(AudioDetailsCallback)` – This asynchronous method returns detailed information about the audio channel associated with the call.
- `hold()` – Holds an active call.
- `unhold()` - Unholds an active call.
- `readAudioDetails(audioDetailsCallback)` - This asynchronous method returns detailed information about the audio channel associated with the call.
- getCallType – Returns the `CallType` information associated with the call.

Running interactions can be interrogated for state. The state consists of two elements.

- The first indicates the time the interaction has been in-progress for (in milliseconds) since `start()` was called.

```
interaction.getInteractionTimeElapsed();
```

The second indicates the abstract "state" of the interaction, returning a value from the `InteractionState` enum in Appendix C.

```
interaction.getInteractionState();
```

Ending an Audio Interaction

The `end()` method will terminate the active interaction.

```
interaction.end()
```

The `discard()` method will shutdown the SDK instance after the interaction has ended.

```
interaction.discard()
```

Note:
For more detailed information on the recommended sequence for handling an interaction ending, see 'Appendix D – Recommended Sequence for Ending Interactions'

Adding a Video Interaction

A video call can be created by deriving a `VideoInteraction` from the `Work` object, as shown here:

VideoInteraction `interaction = work.` createVideoInteraction `(application,listener);`
Where `application` is the current Android application context. `listener is the OnAudioDeviceChangeListener object`

Setting platform type for a Video Interaction (Oceana/Elite)

The Avaya WebRTC Connect can be used for Oceana and Elite interactions. Once Video interaction's object is created, set the platform type as follows:

- `setPlatformType(com.avaya.ocs.Services.Work.Enums.PlatformType platformType)` // PlatformType.OCEANA or PlatformType.ELITE

The default platform type is com.avaya.ocs.Services.Work.Enums.PlatformType.OCEANA

Using `VideoDevice` class, one can set/change the following APIs:

- `selectCamera(cameraType)` – Select which camera type to be used for the video interaction.
- `switchCamera()`– Switch from the front to the rear camera or vice-versa.
- `setVideoCaptureResolutionWithCaptureOrientation(videoCapturePreference,videoCaptureOrientation)`– Sets the resolution and orientation for the video call.
- `getVideoCapturePreference` –Returns the current video capture resolution preference.
- `getVideoCaptureOrientation` – Returns the current video capture orientation preference.
- `VideoCaptureOrientation.values()` – Returns the available orientation values.
- `VideoCapturePreference.values()` – Returns the available resolution values.

Setting up local and remote video views.
Local video view : In the XML layout create com.avaya.ocs.Services.Device.Video.VideoSurfaceView view with app:videoDirection="Transmit" app:videoStyle="MovableCorner"
Remote video view : In the XML layout create com.avaya.ocs.Services.Device.Video.VideoSurfaceView view with app:videoDirection=" Receive" app:videoStyle="Fixed"

Laying out the video views

On the Android platform, a `ViewGroup` is created to represent the area available for video views, and two views of the type `com.avaya.ocs.Services.Device.Video.VideoSurfaceView` should be created inside it, one to represent the local video, and one to represent the remote video.

This layout can be created either in your layout/video_view.xml, or through code.

Use the `videoDirection` attribute on the `VideoSurfaceView` object to assign a video direction to the view:

`app:videoDirection="Receive"` for the remote video stream `app:videoDirection="Transmit"` for the local video stream

There are two styles of view surface, Movable Corner and Fixed.

- A Movable Corner view will snap to one of the corners of parent `ViewGroup` which contains the view, and can be dragged into a different corner by the user.
- A Fixed view is stationary and cannot be moved.

Use the videoStyle attribute on the VideoSurfaceView object to assign a style to the view:
`app:videoStyle="Fixed"` for the remote video stream
`app:videoStyle="MovableCorner"` for the local video stream
Use the videoCornerRadius attribute on the VideoSurfaceView (Local) object to assign a rounded corner to the local video stream
`app:videoCornerRadius="32"` for the local video stream. Value should be an integer. For best results set value in multiples of 8. For a rectangular corner set this value to 0.
Use the videoBorderWidth attribute on the VideoSurfaceView (Local) object to assign a border to the local video stream
`app:videoBorderWidth="2"` for the local video stream. Value should be an integer. You can set it as 1,2,3 etc. For no border set this value to 0.

Note:
The local video view currently only supports the Movable Corner style.

Setup the Video Device

Before initiating a Video Interaction, it is necessary to setup the video device. On the Android platform this tells the Avaya Oceana Customer Web Voice/Video SDK where to display the video media stream and which camera and resolution to use.

The device object is created from the interaction object as shown here:

```
VideoDevice device = interaction.getVideoDevice();
```

A `ViewGroup` which contains the local and remote video views can then be set on the device object, for example:

```
device.setVideoSurface(mainLayout);
```

where mainLayout is an object of type `ViewGroup`, as described In the **Laying out the video views** section above.

See Appendix B for information on enumerations.

Video Interaction Callbacks

The Video Interaction class provides `registerListener()` and `unregisterListener()` methods. This offers the ability to register a class which implements the following interaction status callbacks. These callbacks are available in the `VideoInteractionListener` interface.

These are:

- `onInteractionInitiating()` – Called when the out-going request to initiate an interaction has been sent.

- `onInteractionRemoteAlerting()` – Called when the interaction has received ring back information from the signalling network.
- `onInteractionActive()` – Called when the interaction has been established.
- `onInteractionEnded()` – Called when the call on this interaction is ended. This callback method is called when the interaction is either locally or remotely ended.
- `onInteractionHeld()`–Sent when the interaction is on put hold.
- `onInteractionUnheld()`–Sent when the interaction is unheld
- `onInteractionHeldRemotely()`–Sent when the interaction is put on hold remotely.
- `onInteractionUnheldRemotely()`–Sent when the interaction is unheld remotely.
- `onInteractionAudioMuteStatusChanged(Boolean muted)` – Called when the interaction’s audio is muted or unmuted.
- `onInteractionVideoMuteStatusChanged(Boolean muted)` – Called when the interaction’s video is muted or unmuted.
- `onInteractionVideoEnabledStatusChanged(Boolean enabled)` – Called when the interaction’s video is enabled or disabled.
- `onVideoInteractionFailed(InteractionError Error)` – Called when an error occurred establishing the interaction.
- `onDiscardComplete()` – Called when the interaction has finished discarding.
- `onInteractionQualityChanged(CallQuality quality) - Called when a interaction's quality state changes. Note that this API callback is called every few seconds to return the current quality.`
-

Starting a Video Interaction

Before a Video Interaction can be initiated, an Avaya Aura Web Gateway authorization token must be obtained as detailed in the “Creating an Application: Generic Elements - Authorization Tokens” section. Once a token has been obtained it must be set on the Interaction object:

```
interaction.setAuthorizationToken(token);
```

Once the requisite Video Interaction properties have been set, the interaction can be initiated by invoking:

```
interaction.start();
```

Optionally, before starting a video interaction the following methods can be invoked; `muteAudio`, `muteVideo`, `enableVideo`. All the aforementioned methods take a Boolean indicating the state of the customer’s audio or video media stream. Audio and video will be actively streamed by default.

The interaction can be sent to a specified destination rather than the Avaya Oceana™ Web Voice/Video default routing number.

For example:

```
interaction.setDestinationAddress(destination);
```

where `destination` is a string representing a remote SIP address.

During a Video interaction

There are a number of actions that can take place on the Video Interaction object while the interaction is active.

These include:

- `sendDTMF(digit)` – sends the specified DTMF tone (an enumeration `DTMFDigit` is provided).
- `muteAudio(muted)` – takes a `Boolean` indicating if the audio stream should be silenced or active. The method `isAudioMuted()` indicates the current mute state.
- `muteVideo(muted)` – takes a `Boolean` indicating if the video stream should be silenced or active. The method `isVideoMuted()` indicates the current mute state.
- `enableVideo(enabled)` – takes a `Boolean` indicating if the video stream should be enabled or disabled. The method `isVideoEnabled()` indicates the current video state.
- `readAudioDetails(AudioDetailsCallback)` – This asynchronous method returns detailed information about the audio channel associated with the call.
- `readVideoDetails(VideoDetailsCallback)` – This asynchronous method returns detailed information about the video channel associated with the call.
- `hold()` – Holds an active call.
- `unhold()` - Unholds an active call.
- `readAudioDetails(audioDetailsCallback)` - This asynchronous method returns detailed information about the audio channel associated with the call.
- getCallType – Returns the `CallType` information associated with the call.
-

Video Interactions can be interrogated for state. The state consists of two elements.

- The first indicates the time the interaction has been in-progress for (in milliseconds) since `start()` was called.

  ```
  interaction.getInteractionTimeElapsed();
  ```

- The second indicates the abstract “state” of the interaction, returning a value from the `InteractionState` enum in Appendix B.

  ```
  interaction.getInteractionState();
  ```

Ending a Video Interaction

The `end()` method will terminate the active interaction.

```
interaction.end();
```

The `discard()` method will shutdown the SDK instance after the interaction has ended.

```
interaction.discard()
```

Note:
For more detailed information on the recommended sequence for handling an interaction ending, see ‘Appendix D – Recommended Sequence for Ending Interactions’

Connection Listener Callbacks

The audio and video interaction classes provide `registerConnectionListener()` and `unregisterConnectionListener()` methods. This offers the ability to register a class which implements the following interaction connectivity callbacks. These callbacks are available in the `ConnectionListener` interface.

- `interactionServiceConnected()` – Called when the signalling path for an interaction is available.
- `onInteractionServiceConnecting()` – Called when the signalling path for an interaction is attempting to reconnect after an outage, expect limited call capabilities and the media path to remain active.

- `onInteractionServiceDisconnectedWithError(InteractionError interactionError)` – Called when the signalling path for an interaction has failed or is unavailable.

Application background mode behaviour

When the user presses the Home button, presses the power button, or the system launches another application, the foreground application transitions first to an inactive state, and then to a background state.

If you are currently streaming video from your application, this will stop when the application goes into background mode.

It is an application developer's responsibility to consider both functional and privacy implications when transitioning to background mode.

Note: The application background behaviour of Android is different to that of iOS, please see the iOS section for more details.

Note: Please consult the API documentation provided with the Android Avaya WebRTC Connect SDK for a complete list of available methods.

Writing an iOS application

iOS Version Specifics and IPv6

iOS Version Changes

For information regarding new requirements introduced for iOS version updates, please refer to: Appendix C – iOS Version Updates.

Testing IPv6

Apple require that apps submitted to the Apple store support IPv6-only networks, and this should be tested during development; see:

> https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/NetworkingOverview/UnderstandingandPreparingfortheIPv6Transition/UnderstandingandPreparingfortheIPv6Transition.html

Apple laptops and desktops support providing a NAT64 Wi-Fi hotspot, as long as you are able to connect to your network through another interface such as an Ethernet cable - for details on enabling this, see the *Test for IPv6 DNS64/NAT64 Compatibility Regularly* section in the above link.

> **Note:**
> Apple sometimes requires testing an app in full during submission, in which case a public NAT64 is required.

For more information on setting up your Oceana™ solution for IPv6, please refer to “Support for IPv6 addresses” section in “Avaya Session Border Controller for Enterprise Overview and Specification” available on DevConnect.

Supported Build Architectures

The Avaya WebRTC Connect SDK supports being compiled and run with both an iOS simulator and iOS physical devices.

The OceanaCustomerWebVoiceVideo.xcframework, AvayaClientServicesLite.xcframework, AvayaClientMedia.xcframework and RVVideoCodec.xcframework are universal frameworks and can be used for running the application on both simulator and device. These frameworks can be found in the AvayaWebRTCConnectSDK.zip of the OceanaWebVoiceVideo SDK bundle obtained from DevConnect.

See “Setting up the project” section for details on how to add these dependencies to a project.
If you are using Xcode 13, we suggest using the iPhone 13 pro max option as the targeted simulator.

The supported build architectures for the iphonesimulator AvayaCustomerWebVoiceVideo.framework are x86_64 and Apple Silicone arm64.

For running on actual device, use the option “<connected_iPhone>” as the targeted physical device.

As recommended by Apple, the underlined CSDK and the OceanaCustomerWebVoiceVideo SDK are supports only 64bit architectures. Use arm64 and arm64e as Valid Architectures in build settings of the project.

> **Note:**

Due to audio and video requirements for OceanaCustomerWebVoiceVideo, developers must use a physical device rather than an emulator for testing during development.

Note:
The iOS Reference Client included with the SDK bundle should build out of the box with both the iPhone simulator and actual device.

iOS 13

As recommended by Apple, the underlined CSDK and the AvayaWebRTCConnect SDK are supports only 64bit architectures. Use arm64 and arm64e as Valid Architectures in build settings of the project.

Swift Support

The current AvayaWebRTCConnect SDK release has been tested for support with the Swift programming language (Swift Version 5.1). You can make Audio/Video interactions in the application developed in swift.

XCode 11

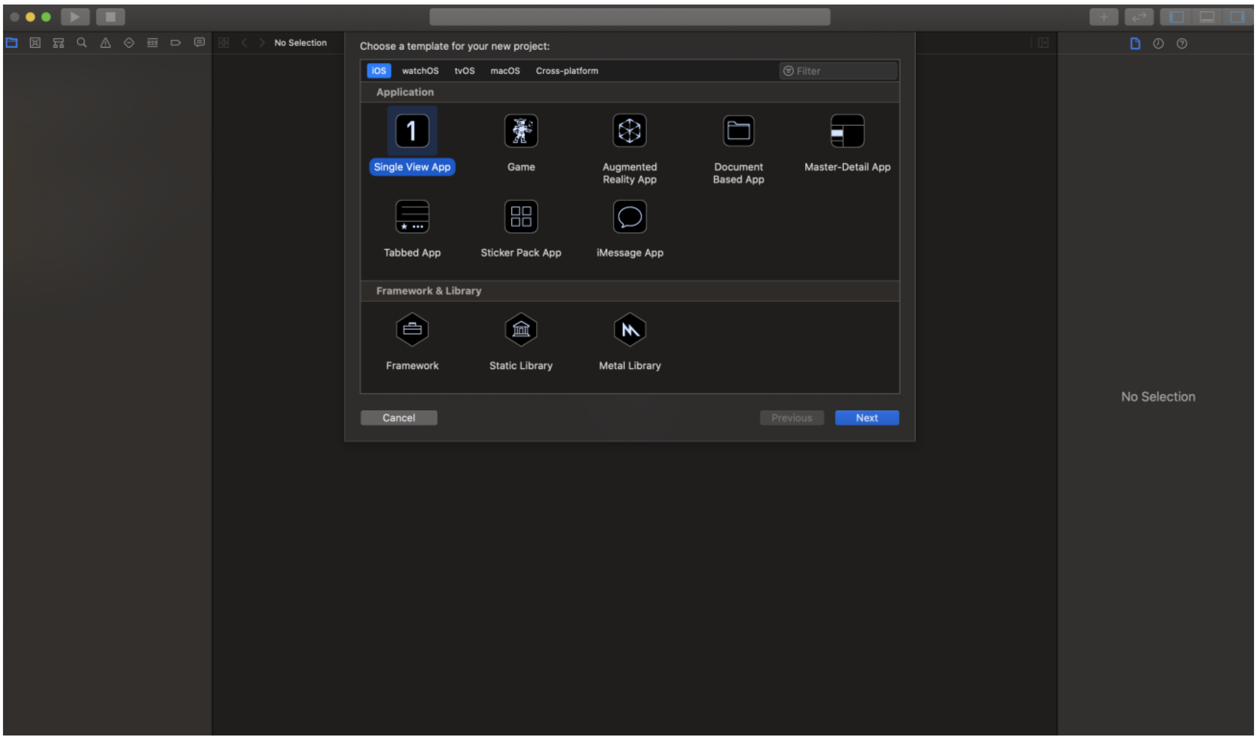
AvayaWebRTCConnect SDK does not support bitcode. You must disable bitcode support in your project or the compiler will throw an error. To disable bitcode, navigate to your build settings, and in the search, box enter “bitcode”. There will be a setting labelled “Enable Bitcode”, and this should be set to NO.

Setting up the project

Developers of iOS apps can use Apple XCode versions 11 and above.
Please consult the section “Downloading the Reference Clients and SDKs” for information on where to download the relevant resources.

To set up a project including the Avaya WebRTC Connect SDK, you first need to create a new project and add iOS native frameworks to it:

1. Open XCode and choose to create a Single View Application, giving your project an appropriate name.



2. Click General tab and expand the ‘Frameworks, Libraries and embedded content’ section by clicking on the title.
3. Click the + button, the file explorer displays.
4. Select the following iOS native dependencies from the iOS folder:

   - Foundation.framework
   - MessageUI.framework

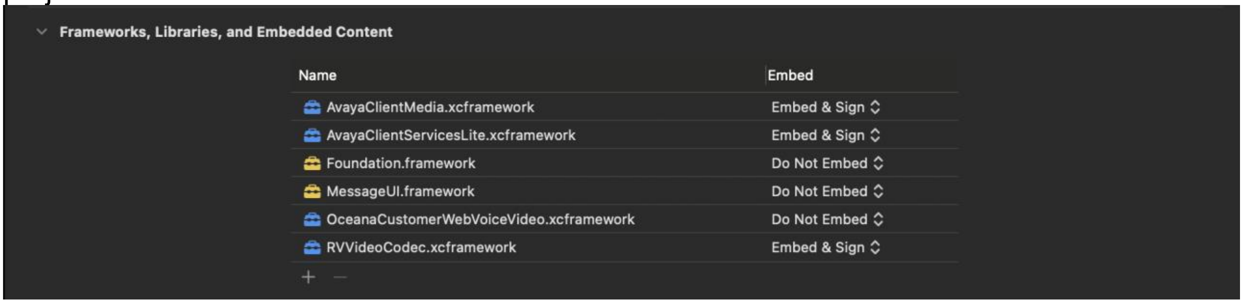
These are system frameworks so do not embed them. The dependencies you selected are now displayed in the Linked Frameworks and Libraries section. Please refer the image below showing all added frameworks.

Now, you need to add the Avaya WebRTC Connect SDK framework to your project.

1. Select your project and click the General tab.
2. Expand the ‘Frameworks, Libraries and embedded content’ section by clicking on the title.
3. Click the + button. When the file explorer displays, click Add Other.
4. Navigate to the OceanaCustomerWebVoiceVideo.xcframework in the downloaded sdk bundle, select it and click OK.
5. Make sure ‘Do not embed’ option is selected for it otherwise Xcode will complain for embedding a static library.

As the Avaya WebRTC Connect SDK framework has a dependency on the AvayaClient SDK, which is packaged as part of the Avaya WebRTC Connect SDK, this also needs to be added to the Linked Frameworks and Libraries section.
To do this:

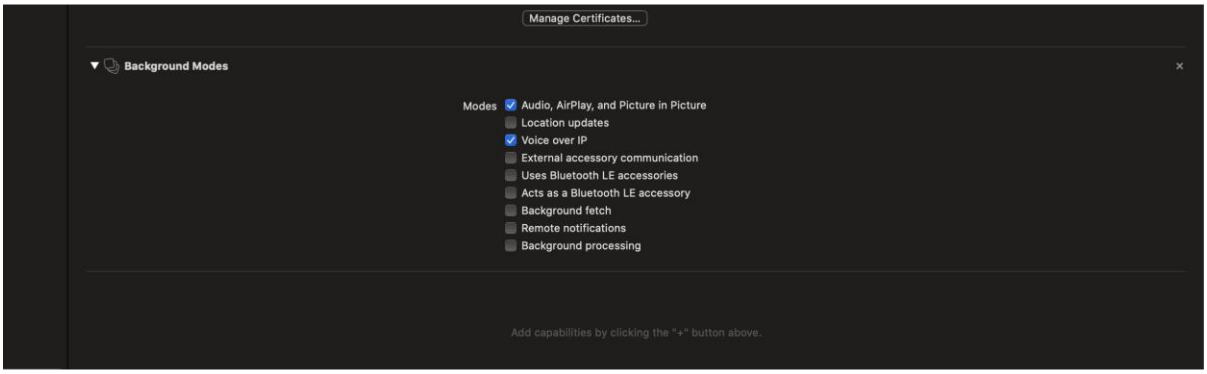
1. Select your project and click the General tab.
2. Expand the ‘Frameworks, Libraries and embedded content’ section by clicking on the title.
3. Click the + button. When the file explorer displays, click Add Other.
4. Navigate to the Avaya WebRTC Connect SDK folder.
5. Navigate to the Frameworks folder and select AvayaClientMedia.xcframework, RVVideoCodec.xcframework, and AvayaClientMedia.xcframework.
6. Click open.
7. These 3 frameworks need to be embedded and signed.
8. The OceanaCustomerWebVoiceVideo and other SDKs has now been added to the project.



Frameworks, Libraries, and Embedded Content

Name	Embed
AvayaClientMedia.xcframework	Embed & Sign
AvayaClientServicesLite.xcframework	Embed & Sign
Foundation.framework	Do Not Embed
MessageUI.framework	Do Not Embed
OceanaCustomerWebVoiceVideo.xcframework	Do Not Embed
RVVideoCodec.xcframework	Embed & Sign

9. To add background mode, go to Project -> Signing & Capabilities -> Background Modes, select following option: Audio, AirPlay, and Picture in Picture & Voice over IP.



Background Modes

Modes
☑ Audio, AirPlay, and Picture in Picture
☐ Location updates
☑ Voice over IP
☐ External accessory communication
☐ Uses Bluetooth LE accessories
☐ Acts as a Bluetooth LE accessory
☐ Background fetch
☐ Remote notifications
☐ Background processing

Add capabilities by clicking the "+" button above.

Initialize the SDK

The Avaya WebRTC Connect SDK can be instantiated as follows:

```
oceanaCustomerWebVoiceVideo = [[AOOceanaCustomerWebVoiceVideo alloc]
       initWithClientConfiguration:clientConfiguration]];
```

Where `clientConfiguration` is an instance of `AOClientConfiguration` class which contains the following configuration objects: The `AOClientConfiguration` object is required for Oceana platform. Set `AOClientConfiguration’s` object only if the interaction would be for Oceana platform.

AOConfiguation – This is specific to Avaya Oceana and not applicable to Avaya Aura Elite. This object contains configuration properties for connecting to the AvayaMobileCommunications snap-in:

- `restUrl` – hostname or IP address of the AvayaMobileCommunications cluster.
- `port` – port to connect to the AvayaMobileCommunications cluster with, defaults to 443.
- `isSecure` – the protocol to connect to the AvayaMobileCommunications cluster with, set to true for HTTPs, set to false to use HTTP.
- `urlPath` – optional, this allows the URL path to be configured when connecting to the AvayaMobileCommunications cluster. See API documentation for more information.

AOWebGatewayConfiguration – contains configuration properties for connecting to the Avaya Aura Web Gateway server:

- `webGatewayAddress` – hostname of the Avaya Aura Web Gateway server.
- `port` – port to connect to the Avaya Aura Web Gateway server with.
- `isSecure` – the protocol to connect to the Avaya Aura Web Gateway server with, set to true for HTTPs, set to false to use HTTP.
- `webGatewayUrlPath` – optional, this allows the URL path to be configured when connecting to the Avaya Aura Web Gateway server. See API documentation for more information.

The `AOCustomerWebVoiceVideo` object then grants access to the following:

- Getting the current version of the Avaya WebRTC Connect SDK:

  oceanaCustomerWebVoiceVideo.versionNumber;

Creating Work

Next a Work object can be derived from the Oceana™ / Elite Customers Web Voice/Video object as follows:

```
      work = oceanaCustomerWebVoiceVideo.createWork;
```

The Work API provides a mechanism to populate the Oceana™ / Elite interaction schema. This interaction schema data is available for use by the Engagement Designer workflow engine (as can be seen in the Web Voice/Video workflow) and is used in requests to Work Assignment for resource selection.
For more information consult the API documentation.

A Work object must be assigned services or resources for Oceana™ / Elite to be able to handle the work request. If both are provided, Work Assignment will try to assign the work to the provided resources. If the resources are unable to be assigned the work, Work Assignment will use the provided services to assign or queue the work.

Adding Services

```
NSMutableArray *services = [[NSMutableArray alloc] init];
AOService *service = [[AOService alloc] init];
service.attributes = attributes;
service.priority = @"5";
[services addObject:service];

work.serviceMap = serviceMap;
```

This code snippet creates an array of services. Next a service is created and given attributes and a priority. Finally, the service is added to the services array and set on the Work object.

Adding Resources

```
NSMutableArray *resources = [[NSMutableArray alloc] init];
AOResource *resource = [[AOResource alloc] init];
resource.nativeResourceId = @"12345";
resource.sourceName =  @"CM";
[resources.addObject:resource];

work.resourceMap = resources;
```

This code snippet creates an array of resources. Next a resource is created and given a resource id and source name. Finally, the resource is added to the resources array and set on the Work object.

Adding Interaction Context

Optional context may be added to a work request as described in the section "Creating an Application: Generic Elements - Context"
To set context on the work object, invoke the following method:

```
      work.context = contextID
```

Where contextID is a string encapsulating some business logic, such as a customer reference number or an Order ID.

Adding an Audio Interaction

The last object required is the ***AOAudioInteraction***, which encapsulates the WebRTC voice call itself. Interactions are derived from the Work object and after providing all the required params to the Audio Interaction object, you can start the Audio interactions as follows:

```
aoAudioInteraction = aoWork.createAudioInteraction;

aoAudioInteraction.authorizationToken = token;
aoAudioInteraction.delegate = audioDelegate;
aoAudioInteraction.connectionListenerDelegate = connectionDelegate;
```

Setting platform type for an Audio Interaction (Oceana/Elite)

The Avaya WebRTC Connect can be used for Oceana and Elite interactions. Once Audio interaction's object is created, set the platform type as follows:

```
    [aoAudioInteraction       setPlatforType:<AOPlatformType>];       //
AOPlatformType_OCEANA or AOPlatformType_ELITE
```

The default platform type is `AOPlatformType_OCEANA`

Audio Interaction Delegate Methods

There is a delegate available on the AOAudioInteraction object that can be used to register for callbacks which provide status information about the AOAudioInteraction that is attempting to be established, or has already been established.
A list of all AOAudioInteraction callbacks that can be registered are:

- interactionInitiating— Sent when the interaction is requesting a session from the Avaya Mobile Communications snap-in.
- interactionRemoteAlerting— Sent when the interaction begins alerting at the remote end..
- interactionActive - Sent when the interaction has been established.
- interactionEnded— Sent when the interaction has ended.
- interactionFailed:NSError error— Sent when the interaction has failed. The error parameter is used to indicate the type of failure that occurred.
- interactionAudioMuteStatusChanged:BOOL muted—Called when the session's audio is muted or unmuted, as indicated by the BOOL.
- discardComplete— Sent when the interaction has finished discarding.
- holdComplete— Sent when the interaction is on hold.
- unholdComplete— Sent when the interaction has unheld
- remoteHoldComplete — Sent when the interaction is on remotely hold.
- remoteUnholdComplete — Sent when the interaction has remotely unheld.
- onInteractionQualityChanged:(AOCallQuality) : Called when a interaction's quality state changes. Note that this API callback is called every few seconds to return the current quality.

Starting an Audio Interaction

Once the requisite Audio Interaction properties have been set, an Audio Interaction can be initiated by invoking:

```
 [aoAudioInteraction start];
```

Optionally, an interaction can be sent to the specified destination rather than the Avaya Oceana™ WebRTC voice routing number. For example:

```
          aoAudioInteraction.destination = @”1234”;
```

Where `destination` is a string representing a remote SIP address.

During an Audio Interaction

There are a number of actions that can take place on the Audio Interaction object during an active interaction.

- - (void) start; // Initiate an audio interaction.
- - (void) end; // End the active audio interaction.
- - (void) discard; // Shutdown the SDK after the interaction has ended.
- - (void)muteAudio:(BOOL)mute; // Mutes or unmutes the interaction's audio.
- - (void)holdWithCompletionHandler:(void (^)(NSError *))handler; //Hold active call
- - (void)unholdWithCompletionHandler:(void (^)(NSError *))handler; //Unhold active call
- - (long)getInteractionTimeElapsed; //Gets time elapsed of the current interaction.
- - (void)sendDTMF:(AODTMFTone)tone; // Sends the specified DTMF tone. @param tone The tone to transmit.
- - (AOInteractionState)getInteractionState; // Returns the current interaction state
- - (void)readAudioDetailsWithCompletionHandler:(void (^)(AOAudioDetails *audioDetails))handler; // Reads detailed information about the audio channel associated with the call. @param handler A block to be called to with audio details in AOAudioDetails.
- - (void)readCallTypeWithCompletionHandler:(void (^)(NSString *csCallTypeString))handler; // Reads Call Type information associated with the call. param handler A block to be called to with Call Type in NSString.
- - (AOInteractionState)getInteractionState; //Returns the current interaction state

For more information, please check the APIDoc and the Reference Client Source code.

Ending an Audio Interaction

To end an interaction in progress, the end method should be called on the Audio Interaction object, for example:

```
[audioInteraction end];
```

The `discard()` method will shutdown the SDK instance after the interaction has ended.

```
[audioInteraction discard];
```

Note:
For more detailed information on the recommended sequence for handling an interaction ending, see ‘Appendix D – Recommended Sequence for Ending Interactions’

Adding a Video Interaction

The last object required is the ***AOVideoInteraction***, which encapsulates the WebRTC video call itself. Interactions are derived from the Work object and after providing all the required params to the Audio Interaction object, you can start the Audio interactions as follows:

```
aoVideoInteraction = aoWork.createVideoInteraction;

aoVideoInteraction.authorizationToken = token;
aoVideoInteraction.delegate = audioDelegate;
aoVideoInteraction     .connectionListenerDelegate = connectionDelegate;
```

AOVideoDevice* device = video.videoDevice;

Setting platform type for a Video Interaction (Oceana/Elite)

The Avaya WebRTC Connect can be used for Oceana and Elite interactions. Once Audio interaction's object is created, set the platform type as follows:

```
    [aoVideoInteraction      setPlatforType:<AOPlatformType>];      //
AOPlatformType_OCEANA or AOPlatformType_ELITE
```

The default platform type is AOPlatformType_OCEANA

Using AOVideoDevice class, one can set/change the following APIs:

- (void) selectCamera: (AOCameraType) cameraType; // Select which camera type to be used for the video interaction.
- -(void) switchCamera;// Switch from the front to the rear camera or vice-versa.
- -(void) setRemoteView:(UIView*) remote; //Set the VideoSurface to be used to render the remote video stream.
- -(void) setLocalView:(UIView*) local; //Set the VideoSurface to be used to render the local video stream.
- +(NSMutableArray*) getSupportedCameraCaptureResolutionPrefes; // Returns a list of Video Resolution preferences that are supported by the device.
- (void) setVideoCaptureResolutionWithCaptureOrientation : (AOVideoCapturePreference) resolutionPreference orientationPreference : (AOVideoCaptureOrientation) orientationPreference; //Set the camera capture resolution with Orientation.
- -(AOVideoCapturePreference) getVideoCapturePreference;// Get the camera capture resolution.
- - (AOVideoCaptureOrientation) getVideoCaptureOrientation; // Get the camera capture Orientation.

Video Interaction Delegate Methods

There is a delegate available on the AOAudioInteraction object that can be used to register for callbacks which provide status information about the AOAudioInteraction that is attempting to be established or has already been established.
A list of all AOAudioInteraction callbacks that can be registered are:

- interactionInitiating— Sent when the interaction is requesting a session from the Avaya Mobile Communications snap-in.
- interactionRemoteAlerting— Sent when the interaction begins alerting at the remote end..
- interactionActive - Sent when the interaction has been established.
- interactionEnded— Sent when the interaction has ended.
- interactionFailed:NSError error— Sent when the interaction has failed. The error parameter is used to indicate the type of failure that occurred.
- interactionAudioMuteStatusChanged:BOOL muted—Called when the session's audio is muted or unmuted, as indicated by the BOOL.

- -(void)interactionVideoMuteStatusChanged:(BOOL)isMuted; //Sent when the interaction's video is muted or unmuted. @param isMuted The new video mute state.
- - (void)interactionVideoEnabledStatusChanged:(BOOL)isEnabled; //Sent when the interaction's video is enabled or disabled. @param isEnabled The new video state.
- discardComplete— Sent when the interaction has finished discarding.
- holdComplete— Sent when the interaction is on hold.
- unholdComplete— Sent when the interaction has unheld
- remoteHoldComplete — Sent when the interaction is on remotely hold.
- remoteUnholdComplete — Sent when the interaction has remotely unheld
- onInteractionQualityChanged:(AOCallQuality) : Called when a interaction's quality state changes. Note that this API callback is called every few seconds to return the current quality.
- .

Starting a Video Interaction

Once the requisite Video Interaction properties have been set, an Video Interaction can be initiated by invoking:

[aoVideoInteraction start];

Optionally, an interaction can be sent to the specified destination rather than the Avaya Oceana™ WebRTC video routing number. For example:

aoVideoInteraction.destination = @”1234”;

Where destination is a string representing a remote SIP address.

During a Video Interaction

There are several actions that can take place on the Video Interaction object during an active interaction.

- -(void) start; //Initiate a video interaction.
- - (void)end; //End the active video interaction.
- - (void) discard; //Shutdown the SDK after the interaction has ended.
- - (void)muteAudio:(BOOL)mute; //Mutes or unmutes the interaction's audio. @param mute The new audio mute state.
- - (void)muteVideo:(BOOL)mute; //Mutes or unmutes the interaction's video. @param mute The new video mute state.
- - (void)holdWithCompletionHandler:(void (^)(NSError *))handler; //Hold active call
- - (void)unholdWithCompletionHandler:(void (^)(NSError *))handler; //Unhold active call
- - (void) enableVideo:(BOOL)enable; //Enables or disables the interaction's video. @param enable The new video state.
- - (long)getInteractionTimeElapsed; //Gets time elapsed of the current interaction.
- - (void)sendDTMF:(AODTMFTone)tone; //Sends the specified DTMF tone. @param tone The tone to transmit.
- - (AOInteractionState)getInteractionState; //Returns the current interaction state.

- - (void)readVideoDetailsWithCompletionHandler:(void (^)(AOVideoDetails *videoDetails))handler; // Reads list of AOVideoDetails objects representing details of all video streams in the interaction. @param handler A block to be called to with video details in AOVideoDetails.
- `- (void)readAudioDetailsWithCompletionHandler:(void (^)(AOAudioDetails *audioDetails))handler;` // Reads detailed information about the audio channel associated with the call. @param handler A block to be called to with audio details in AOAudioDetails.

- (void)readCallTypeWithCompletionHandler:(void (^)(NSString *csCallTypeString))handler; //Reads Call Type information associated with the call. @param handler A block to be called to with Call Type in NSString.

Ending a Video Interaction

To end an interaction in progress, the end method should be called on the Video Interaction object, for example:

```
[videoInteraction end:videoDelegate];
```

The `discard()` method will shutdown the SDK instance after the interaction has ended.

```
[videoInteraction discard];
```

Note:
For more detailed information on the recommended sequence for handling an interaction ending, see 'Appendix D – Recommended Sequence for Ending Interactions'

Connection Listener Delegate

There is a delegate available for both the `AudioInteraction` and `VideoInteraction` classes that can be used to register for callbacks which provide connectivity information on the interaction. A list of all `ConnectionListener` callbacks that can be registered are:

- `interactionServiceConnected` – Called when the signalling path for an interaction is available.
- `onInteractionServiceConnecting` – Called when the signalling path for an interaction is attempting to reconnect after an outage, expect limited call capabilities and the media path to remain active.
- `onInteractionServiceDisconnectedWithError:NSError error` – Called when the signalling path for an interaction has failed or is unavailable.

Note:
Please consult the API documentation provided with the iOS Avaya WebRTC Connect SDK for a complete list of available methods.

Logging

The AvayaWebRTC sdk exposes a AOLoggerDelegate which has below method.

```
- (void)didPrintLog:(CSLogLevel)level withTag:(NSString * _Nullable)tag message:(NSString * _Nonnull)message ;
```

In this method you get access to the log messages printed from the sdk. You can implement this delegate method in your app and use these log messages for saving them to the file in addition to their getting to the console.

Application background mode behaviour

When the user presses the Home button, presses the Sleep/Wake button, or the system launches another application, the foreground application transitions to an inactive state and then to a background state.

If you are currently streaming video from your application, this is suspended when the application goes into background mode.

The video streaming is automatically resumed when the application returns to the foreground.

Audio continues to be streamed when an application goes into background mode.

It is an application developer's responsibility to consider both functional and privacy implications, and to decide whether their application should mute audio and video when transitioning to background mode.

If you mute the video when in background mode, you must unmute in order to resume capture/streaming.

Note: The application background behaviour of iOS is different to that of Android, see the Android section for specific Android details.

Writing a JavaScript application

Setting up the project

Developers can use their preferred Development Environments or HTML editors to build JavaScript apps. One consideration unique to the JavaScript application (as opposed to the mobile apps) is that it must be deployed on a web server. The sample app provided with the SDK was developed using a NodeJS express webserver and tested on a JBoss Application Server.

Please consult the section “Downloading the Reference Clients and SDKs” for information on where to download the relevant resources.

To commence writing a Web Voice and Video enabled JavaScript application, the Avaya WebRTC Connect SDK must be made available via HTML script tags in the code of the application:

```
<script src=”OceanaCustomerWebVoiceVideo.js"></script>
```

```
Note: Please note that minimum version for jQuery is 3.3.1 for the
application to work.
```

Initialize the SDK

When the application has detected the user is attempting to initiate an interaction into the contact centre. The Avaya WebRTC Connect SDK can be instantiated as follows:

```
var client = new OceanaCustomerWebVoiceVideo(clientConfiguration);
```

Where `clientConfiguration` is a JavaScript object literal with the following properties:

Configuration – This is specific to Avaya Oceana and not applicable for Avaya Aura Elite. This object contains configuration properties for connecting to the AvayaMobileCommunications snap-in:

- `restUrl` – hostname or IP address of the AvayaMobileCommunications cluster.
- `port` – port to connect to the AvayaMobileCommunications cluster with, defaults to 443.
- `secure` – the protocol to connect to the AvayaMobileCommunications cluster with, set to `true` for HTTPs, set to `false` to use HTTP.
- `urlPath` – optional, this allows the URL path to be configured when connecting to the AvayaMobileCommunications cluster. See API documentation for more information.

WebGatewayConfiguration – contains configuration properties for connecting to the Avaya Aura Web Gateway server:

- `webGatewayAddress` – hostname of the Avaya Aura Web Gateway server.
- `port` – port to connect to the Avaya Aura Web Gateway server with.
- `secure` – the protocol to connect to the Avaya Aura Web Gateway server with, set to `true` for HTTPs, set to `false` to use HTTP.

Creating Work

Next a Work object can be derived from the Customer object as follows:

```
      var work = client.createWork();
```

The Work API provides a mechanism to population the Oceana / Elite interaction schema. This interaction schema data is available for use by the Engagement Designer workflow engine (as can be seen in the Web Voice/Video workflow) and is used in requests to Work Assignment for resource selection.
For more information consult the API documentation.

A `Work` object must be assigned services or resources for Oceana / Elite to be able to handle the work request. If both are provided, Work Assignment will try to assign the work to the provided resources. If the resources are unable to be assigned the work, Work Assignment will use the provided services to assign or queue the work.

Adding Services

```
var service = new OceanaCustomerWebVoiceVideo.Services.Work.Schema.Service();
service.setAttributes(attributes);
service.setPriority(5);

work.setServices([service]);
```

This code snippet creates a service and given attributes and a priority. The service is added to the services collection and set on the Work object.

Adding Resources

```
var resource = new OceanaCustomerWebVoiceVideo.Services.Work.Schema.Resource();
resource.setNativeResourceId(“agent6220”);
resource.setSourceName(“CM”);

work.setResources([resource]);
```

This code snippet creates a resource, given a resource id and source name. The resource is added to the resources collection and set on the Work object.

Adding Interaction Context

Optional context may be added to a work request as described in the section “Creating an Application: Generic Elements - Context”
To set a context on the work object, invoke the following method:

```
      work.setContext(context);
```

Where `context` is a string encapsulating some business logic, such as a customer reference number or an Order ID.

The last object required is the interaction which encapsulates the Web Voice/Video call itself.

Adding an Audio Interaction

A voice call can be created by deriving an `AudioInteraction` from the `Work` object, as shown here:

```
      var interaction =
work.createAudioInteraction(OceanaCustomerWebVoiceVideo.Services.Work.PlatformTyp
e platformType) // PlatformType.OCEANA or PlatformType.ELITE
```

Setting platform type for an Audio Interaction (Oceana/Elite)

The Avaya WebRTC Connect can be used for Oceana and Elite interactions. Instead of setting the platformType in createAudioInteraction() method, you can also pass the PlatformType once Audio interaction's object is created as follows:

```
      setPlatformType(OceanaCustomerWebVoiceVideo.Services.Work.PlatformType
platformType) // PlatformType.OCEANA or PlatformType.ELITE
```

The default platform type is com.avaya.ocs.Services.Work.Enums.PlatformType.OCEANA
Please note that you can skip this API call if you already passed the PlatformType in `createAudioInteraction()`

Audio Interaction Callbacks

The Audio Interaction class provides callbacks that can be used to register for interaction status information.

These are:

- `onAudioInteractionServiceConnected()` – Called when the signalling path for an interaction is available.
- `onAudioInteractionServiceConnecting()` – Called when the signalling path for an interaction is attempting to reconnect after an outage, expect limited call capabilities and the media path to remain active.
- `onAudioInteractionServiceDisconnected()` – Called when the signalling path for an interaction has failed or is unavailable.
- `onAudioInteractionInitiating()` – Called when the out-going request to initiate an interaction has been sent.
- `onAudioInteractionRemoteAlerting()` – Called when the interaction has received ring back information from the signalling network.
- `onAudioInteractionEstablished()` – Called when the interaction has been established.
- `onAudioInteractionEnded()` – Called when the call on this interaction is ended. This callback method is called when the interaction is either locally or remotely ended.
- `onAudioInteractionAudioMuteStatusChanged(Boolean muted)` – Called when the interaction's audio is muted or unmuted.
- `onAudioInteractionFailed(InteractionError Error)` – Called when an error occurred establishing the interaction.
- `onAudioInteractionHoldStatusChanged(Boolean isHeld)` – Called when the interaction is held or unheld.
- onAudioInteractionCallQuality:(CallQuality callQuality) : `Called when a interaction's quality state changes. Note that this API callback is called every few seconds to return the current quality.`
-

Starting an Audio Interaction

Before an Audio Interaction can be initiated, an Avaya Aura Web Gateway authorization token must be obtained as detailed in the "Creating an Application: Generic Elements - Authorization Tokens" section. Once a token has been obtained it must be set on the Interaction object:

```
interaction.setAuthorizationToken(token);
```

Once the requisite Audio Interaction properties have been set, the Interaction can be initiated by invoking:

```
interaction.start();
```

Optionally, before starting an interaction, `muteAudio` can be called.
The aforementioned method takes a `Boolean` indicating the state of the customer's audio. Audio will be streamed by default.

The interaction can be sent to a specified destination rather than the Avaya Oceana™ / Elite Web Voice/Video default routing number.
For example:

```
interaction.setDestinationAddress(destination);
```

where `destination` is a string representing a remote SIP address.

During an Audio interaction

There are a number of actions that can take place on the Audio Interaction object while the interaction is active.

These include:

- `sendDTMF(digit)` – sends the specified DTMF tone (an enumeration `DTMFDigit` is provided).
- `muteAudio(muted)` – takes a `Boolean` indicating if the audio stream should be silenced or active. The method `isAudioMuted()` indicates the current mute state. This method can also be invoked before calling `start()`.
- `readAudioDetails(callback)` - This asynchronous function returns detailed information about the audio channel associated with the call
- `holdCall(isHeld)` – Takes a `Boolean` indicating if the call is to be held or unheld The method `isCallHeld()` indicates the current hold state.

Audio Interactions can be interrogated for state. The state consists of two elements.

- The first indicates the time the interaction has been in-progress for (in milliseconds) since `start()` was called.

  ```
  interaction.getInteractionTimeElapsed();
  ```

- The second indicates the abstract “state” of the interaction, returning a value from the `InteractionState` enum in Appendix B.

  ```
  interaction.getInteractionState();
  ```

Ending an Audio Interaction

The `end()` method will terminate the active interaction.

```
interaction.end();
```

Adding a Video Interaction

A video call can be created by deriving a `VideoInteraction` from the `Work` object, as shown here:

```
var interaction = work.createVideoInteraction(OceanaCustomerWebVoiceVideo.Services.Work.PlatformType platformType) // PlatformType.OCEANA or PlatformType.ELITE
```

Setting platform type for a Video Interaction (Oceana/Elite)

The Avaya WebRTC Connect can be used for Oceana and Elite interactions. Instead of setting the platformType in createVideoInteraction() method, you can also pass the PlatformType once Video interaction's object is created as follows:

```
setPlatformType(OceanaCustomerWebVoiceVideo.Services.Work.PlatformType platformType) // PlatformType.OCEANA or PlatformType.ELITE
```

The default platform type is com.avaya.ocs.Services.Work.Enums.PlatformType.OCEANA. Please note that you can skip this API call if you already passed the PlatformType in `createVideoInteraction()`

Setup the Video Device

Before initiating a Video Interaction, it is necessary to setup the video device. On the JavaScript platform this tells the browser where to display the video media stream and which camera resolution to use.

The device object is created from the interaction object as shown here:

```
var device = interaction.getVideoDevice();
```

The views representing the local and remote video steams can then be set on the device object, for example:

```
device.setLocalView(localVideoId);
device.setRemoteView(remoteVideoId);
```

Note: If you are using a VIDEO tag to display the local video view, you must remember to set the attribute "muted" on the video if you do not want the user to hear the output of their own microphone, for example:
`<video id="local-video-view" muted></video>`

The following method on the device object can be used to set the cameras capture resolution:

```
device.setCameraCaptureResolution(CameraResolution.RESOLUTION_640x480);
```

For a full list of available resolutions, please refer to Appendix B: Avaya WebRTC Connect Enumerations Enumerations.

Video Interaction Callbacks

The Video Interaction class provides callbacks that can be used to register for interaction status information.

These are:

- `onVideoInteractionServiceConnected()` – Called when the signalling path for an interaction is available.

- `onVideoInteractionServiceConnecting()` – Called when the signalling path for an interaction is attempting to reconnect after an outage, expect limited call capabilities and the media path to remain active.
- `onVideoInteractionServiceDisconnected()` – Called when the signalling path for an interaction has failed or is unavailable.
- `onVideoInteractionInitiating()` – Called when the out-going request to initiate an interaction has been sent.
- `onVideoInteractionRemoteAlerting()` – Called when the interaction has received ring back information from the signalling network.
- `onVideoInteractionEstablished()` – Called when the interaction has been established.
- `onVideoInteractionEnded()` – Called when the call on this interaction is ended. This callback method is called when the interaction is either locally or remotely ended.
- `onVideoInteractionAudioMuteStatusChanged(Boolean muted)` – Called when the interaction's audio is muted or unmuted.
- `onVideoInteractionVideoMuteStatusChanged(Boolean muted)` – Called when the interaction's video is muted or unmuted.
- `onVideoInteractionVideoEnabledStatusChanged(Boolean enabled)` – Called when the interaction's video is enabled or disabled.
- `onVideoInteractionFailed(InteractionError Error)` – Called when an error occurred establishing the interaction.
- `onVideoInteractionHoldStatusChanged(Boolean isHeld)` – Called when the interaction is held or unheld.
- onVideoInteractionCallQuality:(CallQuality callQuality) : `Called when a interaction's quality state changes. Note that this API callback is called every few seconds to return the current quality.`
-

Starting a Video Interaction

Before a Video Interaction can be initiated, an Avaya Mobile Video authorization token must be obtained as detailed in the "Creating an Application: Generic Elements - Authorization Tokens" section. Once a token has been obtained it must be set on the Interaction object:

```
interaction.setAuthorizationToken(token);
```

Once the requisite Video Interaction properties have been set, the interaction can be initiated by invoking:

```
interaction.start();
```

Optionally, before starting a interaction, the following methods can be called; `muteAudio, muteVideo, enableVideo.`
All the aforementioned methods take a `Boolean` indicating the state of the customers audio or video media stream. Audio and video will be streamed by default.

The interaction can be sent to a specified destination rather than the Avaya Oceana™ Web Voice/Video default routing number.

For example:

```
interaction.setDestinationAddress(destination);
```

where `destination` is a string representing a remote SIP address.

During a Video interaction

There are a number of actions that can take place on the Video Interaction object while the interaction is active.

These include:

- `sendDTMF(digit)` – sends the specified DTMF tone (an enumeration `DTMFDigit` is provided).
- `muteAudio(muted)` – takes a `Boolean` indicating if the audio stream should be silenced. The method `isAudioMuted()` indicates the current mute state. This method can also be invoked before calling `start()`.
- `muteVideo(muted)` – takes a `Boolean` indicating if the video stream should be silenced or active. The method `isVideoMuted()` indicates the current mute state. This method can also be invoked before calling `start()`.
- `enableVideo(enabled)` – takes a Boolean indicating if the video stream should be enabled or disabled. The method `isVideoEnabled()` indicates the current video state. This method can also be invoked before calling `start()`.
- `readAudioDetails(callback)` - This asynchronous function returns detailed information about the audio channel associated with the call.
- `readVideoDetails(callback)` - This asynchronous function returns detailed information about the video channel associated with the call.
- `holdCall(isHeld)` – Takes a `Boolean` indicating if the call is to be held or unheld The method `isCallHeld()` indicates the current hold state.

Video Interactions can be interrogated for state. The state consists of two elements.

- The first indicates the time the interaction has been in-progress for (in milliseconds) since `start()` was called.

  ```
  interaction.getInteractionTimeElapsed();
  ```

- The second indicates the abstract “state” of the interaction, returning a value from the `InteractionState` enum in Appendix B.

  ```
  interaction.getInteractionState();
  ```

Ending a Video Interaction

The `end()` method will terminate the active interaction.

```
interaction.end();
```

Note: Please consult the API documentation provided with the JavaScript Avaya WebRTC Connect SDK for a complete list of available methods.

Known Issues and limitations

HTTPS and certificates

In order to use HTTPS to connect to Avaya Mobile Communications and Avaya Aura Web Gateway, client PCs and mobile devices need certificates from the target servers to be installed and trusted. In general, self-signed certificates are used in a test/lab scenario, but external devices will not trust self-signed certificates by default.

Certificates should be signed by a reputable third part Certificate Authority (CA), and these certificates will not need to be trusted manually.

For more information consult the Avaya Mobile Communications chapters in the “Deploying Avaya Oceana™ Solution” available on . Navigate to “Support by Product” > “Documents”. Search “Avaya Oceana Solution” with the “Installation, Upgrades & Config” filter applied.

Edge DTMF Function

JavaScript sendDTMF(tone) function does not work on Microsoft Edge, so it should be disabled if the customer is using the Edge browser.

JavaScript readAudio/readVideoDetails

JavaScript `readAudioDetails(callback)` and `readVideoDetails(callback)` functions return a subset of the results in comparison to their Android and iOS equivalent.

JavaScript webGatewayUrlPath

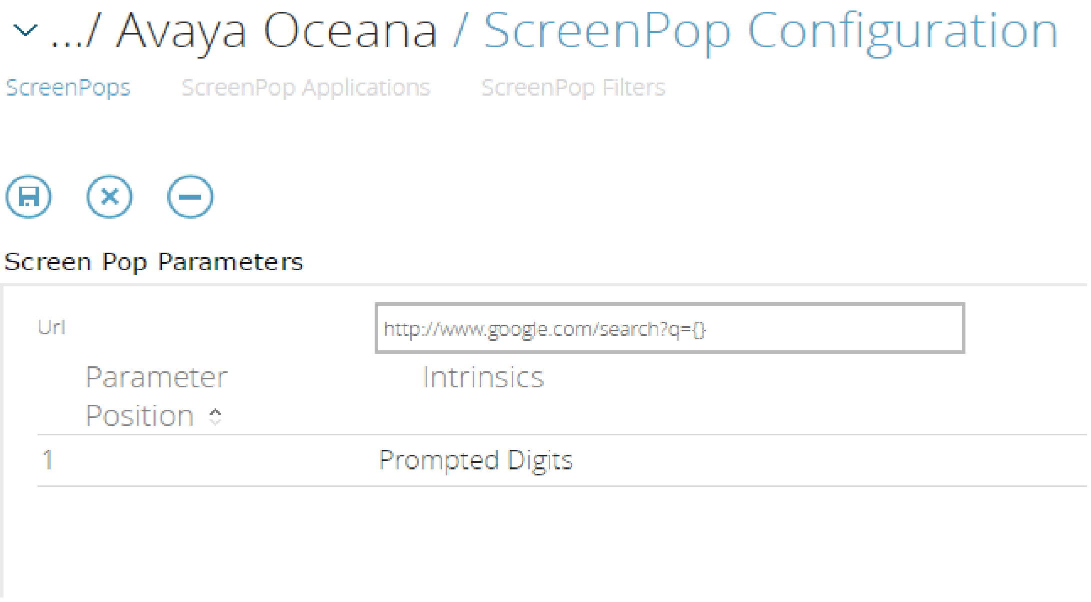
In the JavaScript WebGatewayConfiguration object, it does not offer the ability to configure the webGatewayUrlPath which is a property present in Android and iOS.

Appendix A – Configuring Screen Pops

For more detailed information on Screen Pop configuration, please refer to the “Configuring screen pop parameters” section found in the “Configuring Avaya Control Manager” document, available on . Navigate to “Support by Product” > “Documents”. Search component “Contact Center Control Manager” and apply the filter “Installation, Upgrades & Config”.

The required configuration to Screen Pop on the Context value is as follows:

1. Browse to Avaya Control Manager (ACM), url: https://<ACM_FQDN>/ACCCMPortal/.
2. From the ACM home page, browse to: Configuration / Avaya Oceana / ScreenPop Configuration.
3. Select a Screen Pop, edit and ensure the intrinsic selected is Prompted Digits when configuring the Screen Pop parameters in ACM. Prompted Digits is populated with the value set via the setContext() method call on the Work API.

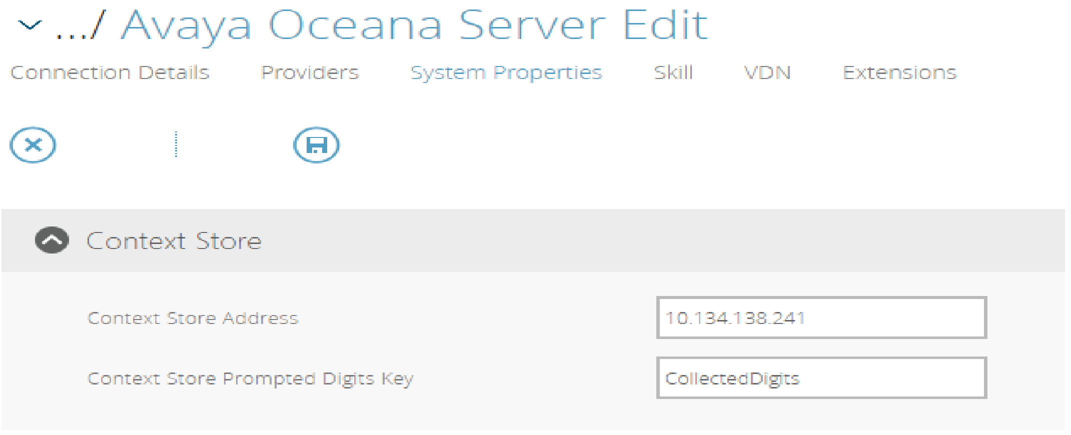


Screen Pop Parameters

Url: http://www.google.com/search?q={}

Parameter Position	Intrinsics
1	Prompted Digits

4. In ACM, browse to: Configuration / Avaya Oceana / Server Details. Select a server, edit and select the System Properties tab in the edit menu.
5. Set the Context Store Prompted Digits Key field to CollectedDigits as this is the name of the key within Context Store where Prompted Digits are stored.



Context Store

Context Store Address: 10.134.138.241

Context Store Prompted Digits Key: CollectedDigits

Appendix B – Avaya WebRTC Connect Enumerations

Interaction State

The `InteractionState` in JavaScript and Android or `AOInteractionState` in iOS is an enumeration indicating the current state of an interaction available by calling `getInteractionState()` on an interaction object.

State	Description
`IDLE`	Indicates the interaction is uninitialized.
`INITIATING`	Indicates the outgoing interaction is being created (no response yet).
`REMOTE_ALERTING`	Indicates the interaction is ringing on the remote side.
`ESTABLISHED`	Indicates the interaction is established and media is active.
`ENDING`	Indicates the interaction is shutting down.
`ENDED`	Indicates the interaction has shutdown.
`FAILED`	Indicates the interaction failed to initialize or if a failure was encountered mid-call.

Video resolution preferences for Javascript

The `CameraResolution` in Javascript is an enumeration indicating the supported camera capture resolution preference for that device by calling a getter method on a VideoDevice object and `VideoCaptureOrientation` is an enumeration indicating the supported camera capture orientation.

Resolution Preferences
`RESOLUTION_416x240`
`RESOLUTION_640x360`
`RESOLUTION_848x480`
`RESOLUTION_1280x720`
`RESOLUTION_1920x1080`

Video resolution preferences and orientations for Android

The `VideoCapturePreference` in Android is an enumeration indicating the supported camera capture resolution preference for that device and `VideoCaptureOrientation` is an enumeration indicating the supported camera capture orientation.

Resolution Preferences
`VideoCapturePreference_Min`
`VideoCapturePreference_270p`
`VideoCapturePreference_360p`
`VideoCapturePreference_540p`
`VideoCapturePreference_Max`

Video Capture Orientation
`VideoCaptureOrientation_LandscapeOnly`
`VideoCaptureOrientation_LandscapeOrPortrait`

Video resolution preferences and orientations for iOS

The `AOVideoCapturePreference` in iOS is an enumeration indicating the supported camera capture resolution resolutions for that device by calling a getter method on a Video Device object and `AOVideoCaptureOrientation` is an enumeration indicating the supported camera capture orientation.

Resolution Preferences
`AOVideoCapturePreference_Min`
`AOVideoCapturePreference_270p`
`AOVideoCapturePreference_360p`
`AOVideoCapturePreference_540p`
`AOVideoCapturePreference_Max`

Video Capture Orientation
`AOVideoCaptureOrientation_LandscapeOnly`
`AOVideoCaptureOrientation_LandscapeOrPortrait`

Camera Type

The `CameraType` in JavaScript and Android or `AOCameraType` in iOS is an enumeration indicating the camera types that can be selected for that device by calling a getter method on a Video Device object.

Camera Type only applies on the Android and iOS platforms.

Camera Type
`FRONT`
`BACK`

CallQuality

The `CallQuality` in JavaScript, Android iOS is an enumeration indicating the ongoing audio/video call quality. The call quality is continuously calculated in the background based on the Call Quality parameters and emitted via the SDKs InteractionListener interface.

CallQuality
EXCELLENT
GOOD
FAIR
POOR
BAD

Appendix C – iOS Version Updates

iOS 10 and above

In iOS 10, Apple extended the scope of their privacy control. You are now required to ask to use permission to access user private data and, in the case of an Avaya WebRTC Connect application, permission to use the Microphone and Camera. Go to your application's Info.plist file and add the following keys:

Privacy – Microphone Usage Description, along with a suitable description.
Privacy – Camera Usage Description, along with a suitable description.

Now the user will get prompted to allow their microphone permissions. If this addition is not made when running iOS 10 and later, the application will crash when requesting to use the microphone or camera.

iOS 11 and above

In iOS 11, Apple has stopped supporting 32-bit applications. Therefore, Apple recommends setting the Architectures in Build Settings to:
Standard architectures (armv7, arm64) - $(ARCHS_STANDARD)

The Avaya WebRTC Connect SDK supports 64-bit applications.

Appendix D – Recommended Sequence for Ending Interactions

On Android and iOS, the following sequence is recommended to end the interaction and terminate the SDK cleanly after each interaction:

When an interaction is ended, either by the agent ending the call, the user ending the call, or because an error is encountered. It is recommended to call the `discard()` method available on the `AudioInteraction` and `VideoInteraction` objects from the following callbacks:

- `onInteractionEnded()`
- `onInteractionFailed(InteractionError error)`

The `discard()` function has a corresponding `onDiscardComplete()` callback which fires when the SDK has finished terminating successfully.

It is recommended to leverage this `onDiscardComplete()` callback to transition from the active 'in-call' screen to the previous 'Make Call' screen.

So only when the `onDiscardComplete()` is fired, the user is then transitioned away from the active 'in-call' screen.

This sequence of behaviour can be observed in the corresponding OceanaReferenceClients supplied for Android and iOS:

Interaction Ends -> onInteractionEnded() / OnInteractionFailed(error) is fired -> discard() is called -> onDiscardComplete() is fired -> Application transitions from active in call screen

Appendix E – Migration plan from AMV 3.x to Avaya WebRTC Connect 4.0

This document will help the developer to upgrade from AMV 3.X to Avaya WebRTC Connect and understand the Avaya WebRTC Connect APIs.

This document describes the migration from AMV 3.x to Avaya WebRTC Connect at the API level. The developer can understand the counter APIs from Avaya's WebRTC 4.0 and can get a clear understanding of how to update the APIs to achieve the same feature or functionality.

- **Token Generation Service:-**

The Authentication Token is used to register the user anonymously. It is the Application layer functionality as the authentication service would be implemented by a consumer application. In AMV 3.X, a GET method is used to get the authentication token and in Avaya WebRTC 4.0, the POST method is used. As this is an HTTPs web service, an iOS developer can use iOS APIs like NSMutableURLRequest or NSURLSession or third party libraries like AFNetworking and Alamofire.
Here is the web service used to get the Authentication token. The web services can be different as per the implementations and lab configurations.

AMV 3.X	Avaya WebRTC Connect
<http/https>://<SERVER>:<PORT>/avayatest/auth ?displayName=<DISPLAY_NAME>&userName=USER_NAME GET	<http/https>://<Token-SERVER>:<PORT>/token-generation-service/token/getEncryptedToken POST

- **Audio Call:**
 Before making any audio call, in AMV 3.X an Audio only session needs to be created with the help of CPUser class. In WebRTC 4.0, all these tasks can be done by AOVAudioInteraction Class.
This is how the Audio-only session is created in AMV 3.x.

Steps to initiate the Audio Only Session	Objective	AMV 3.X APIs
Create CPUser Object	To handle user-level features like create a session, terminate a session, etc	CPAudioOnlyUser *cpUser = [[CPAudioOnlyClientPlatform clientPlatform] user];
Create CPDevice Object	To handle the AudioPath (phone or scpeaker)	CPDevice *cpDevice = [[CPAudioOnlyClientPlatform clientPlatform] device];

Set the Authorization Token to the user	This API is used to set the authorization token to the user	cpUser.authorizationToken = token;
Set the delegate to the CPUser object	To achieve forward message passing mechanism for lately created objects. (an CPAudioOnlyUserDelegate object)	cpUser.delegate = userDelegate;
Create CPAudioOnlySession Object	This object is used to start/end call, mute/unmute audio, send DTMF tone, etc	cpSession = [cpUser createSession];
Set User to User information/ ContextID	To pass user to user information	[cpSession setContextId:uui];
Set a remote address.	Set the remote address to the Audio Only Session.	[cpSession setRemoteAddress:remoteAddress];
Set Audio Only Session Delegate	To achieve forward message passing mechanism for lately created objects. (an CPAudioOnlySessionDelegate object)	[cpSession setDelegate:delegate];

In Avaya WebRTC Connect 4.0 following steps need to be followed to create an audio interaction.

Steps to create Audio Interaction	Objective	Avaya WebRTC Connect 4.0 APIs

Create AOClientConfiguration Object	AOClientConfiguration's object is used to instantiate the AOOceanaCustomerWebVoiceVideo. AOClientConfiguration has two properties, AOWebGatewayConfiguration's webGatewayConfiguration & AOConfiguration's configuration.
	AOWebGatewayConfiguration is used to set the AAWG server details and AOConfiguration is used to set AMC (Oceana Platform server details)
		AOClientConfiguration *clientConfig = [[AOClientConfiguration alloc]init];
		AOWebGatewayConfiguration* webGatewayConfig = [[AOWebGatewayConfiguration alloc]init];
		webGatewayConfig.webGatewayAddress = [AppSettings getAAWGServerAddress];
		webGatewayConfig.port = [AppSettings getAAWGServerPort];
		webGatewayConfig.isSecure =

Steps to create Audio Interaction	Objective	Avaya WebRTC Connect 4.0 APIs

		[AppSettings isAawgSecure]; webGatewayConfig.webGatewayUrlPath = [AppSettings getAAWGRestUrl]; clientConfig.webGatewayConfiguration = webGatewayConfig;
		AOConfiguration* config = [[AOConfiguration alloc]init];
		config.restUrl = [AppSettings getServer];
		config.port = [AppSettings getPort];
		config.isSecure = [AppSettings useSecureLogin];
		config.urlPath = [AppSettings getAMCRestUrl];
		clientConfig.configuration = config;

Create AOOceanaCustomerWebVoiceVideo's Object	This is the entry point of the Avaya WebRTC SDK 4.0 To instantiate Audio/Video interactions, the AOOceanaCustomerWebVoiceVideo's object needs to be created.	AOOceanaCustomerWebVoiceVideo *customerWebVoice = [[AOOceanaCustomerWebVoiceVideo alloc]initWithClientConfiguration:clientConfig];

Create and setup AOWork's Object.	To set the routing strategy, local, services, topic, etc the AOWork's object needs to be created. These are the routing strategies used to route the call.
	These parameters can be provided with the Lab details.
		AOWork* work = customerWebVoice.createWork ;
		work.context = <context_string>;
		work.topic = <topic>
		work.locale = <local>;
		work.routingStrategy = <routing_strategy>;
		work.services = <services>;
		work.resources = <resources>; //If available

Create AOAudioInteraction's object	To make an audio call, AOAudioInteraction's object needs to be created.	AOAudioInteraction *audioInteraction = work.createAudioInteraction;

Steps to create Audio Interaction	Objective	Avaya WebRTC Connect 4.0 APIs
Set up AOAudioInteraction's object	The properties like setPlatformType(Oceana/Elite), delegate, connectionListenerDelegate, and destination address.	audioInteraction.delegate = audioDelegate; audioInteraction.connectionListenerDelegate = connectionDelegate; audioInteraction.destinationAddress = <remote_address>

At this point, the audio interaction object is created. Now to place the audio call, we need to start the Audio interaction and before that, some properties need to be set like Platform Type(Oceana/Elite), delegate, connectionListenerDelegate, and destination address.

This is how the Audio call/interaction is started in AMV 3.X and Avaya WebRTC Connect with all the Audio Call Features.

-

Audio Call Features	AMV 3.X	Avaya WebRTC 4.0
Start Call/Interaction	[cpSession start];	[audioInteraction start];
End Call	[cpSession end];	[audioInteraction end];
Discard the call	[cpUser terminate];	[audioInteraction discard];
Mute/Un-mute Audio	[cpSession muteAudio: bool_value];	[audioInteraction muteAudio: bool_value];
Get if the active call is mute or not		
Hold Call	[cpSession hold];	[audioInteraction holdWithCompletionHandler:^(NSError *error){}];
Resume Call	[cpSession resume];	[audioInteraction unholdWithCompletionHandler:^(NSError *error){}];
Get Time Elapsed	[cpSession getCallTimeElapsed];	[audioInteraction getInteractionTimeElapsed];
Get Current Call Sate	CPSessionState state = cpSession.state;	[audioInteraction getInteractionState];

Audio Call Features	AMV 3.X	Avaya WebRTC 4.0
Send DTMF tone	[cpSession sendDTMF:tone andPlayAudio:YES]; // And [cpSession sendDTMF:tone];	[audioInteraction sendDTMF:tone];
Send hands-free Audio (phone-speaker)	[cpDevice setHandsfreeAudio:<bool_value>];	This can be handled at the Application layer. Use iOS' ***AVRoutePickerView*** to change the Audio routes.
Get Call Quality. In AMV 3.X, the CPBaseSession's didChangeQuality API gives the current call quality whereas, in Avaya WebRTC 4.0, the call quality can be determined by ***readAudioDetailsWithCompletionHandler*** API which returns **AOAudioDetails**'s object.	Implement CPBaseSession's - (void)session:(CPBaseSession *)session didChangeQuality: (int)quality;	[audioInteraction readAudioDetailsWithCompletionHandler:^(AOAudioDetails *audioDetails) {}];

- **Video Call:**
 Before making any video call, in AMV 3.X a session needs to be created with the help of CPUser class. In Avaya WebRTC Connect , all these tasks can be done by AOVideoInteraction Class. This is how the Audio-only session is created in AMV 3.x.

Steps to initiate the Video Session	Objective	AMV 3.X APIs
Create CPDevice Object	To handle the AudioPath (phone or scpeaker), Video mute/unmute, set remote and local video rendering view etc	CPDevice *cpDevice = [[CPClientPlatform clientPlatform] device];
Create CPSession's Object	This object is used to start/end call, mute/unmute audio, send DTMF tone, etc	CPSession *cpSession = [cpUser createSession];
Create CPUser Object	To handle user-level features like create a session, terminate a session, etc	CPUser *cpUser = [[CPClientPlatform clientPlatform] user];

Steps to initiate the Video Session	Objective	AMV 3.X APIs
Set a remote address.	Set the remote address to the Audio Only Session.	[cpSession setRemoteAddress:remoteAddress];
Set Session Delegate	To achieve forward message passing mechanism for lately created objects. (a CPSessionDelegate object)	[cpSession setDelegate:delegate];
Set the Authorization Token to the user	This API is used to set the authorization token to the user	cpUser.authorizationToken = token;
Set the delegate to the CPUser object	To achieve forward message passing mechanism for lately created objects. (an CPAudioOnlyUserDelegate object)	cpUser.delegate = userDelegate;

Set up CPDevice Object	Set camera, orientation, local and remote rendering views etc.	device.localVideoView = localVideoView; device.remoteVideoView = remoteVideoView;
		device.cameraType = CPCameraTypeFront;
		device.cameraOrientation = CPCameraOrientation_<value>

Set User to User information/ ContextID	To pass user to user information	[cpSession setContextId:uui];

In Avaya WerbRTC 4.0 following steps need to be followed to create a video interaction.

Steps to create Video Interaction	Objective	Avaya WebRTC Connect APIs
Create AOClientConfiguration Object	AOClientConfiguration's object is used to instantiate the AOOceanaCustomerWebVoiceVideo. AOClientConfiguration has two properties,	AOClientConfiguration *clientConfig = [[AOClientConfiguration alloc]init]; AOWebGatewayConfiguration* webGatewayConfig = [[AOWebGatewayConfiguration

Steps to create Video Interaction	Objective	Avaya WebRTC Connect APIs

	AOWebGatewayConfiguration's webGatewayConfiguration & AOConfiguration's configuration.
	AOWebGatewayConfiguration is used to set the AAWG server details and AOConfiguration is used to set AMC (Oceana Platform server details)
		alloc]init];
		webGatewayConfig.webGatewayAddress = [AppSettings getAAWGServerAddress];
		webGatewayConfig.port = [AppSettings getAAWGServerPort];
		webGatewayConfig.isSecure = [AppSettings isAawgSecure];
		webGatewayConfig.webGatewayUrlPath = [AppSettings getAAWGRestUrl];
		clientConfig.webGatewayConfiguration = webGatewayConfig;
		AOConfiguration* config = [[AOConfiguration alloc]init];
		config.restUrl = [AppSettings getServer];
		config.port = [AppSettings getPort];
		config.isSecure = [AppSettings useSecureLogin];
		config.urlPath = [AppSettings getAMCRestUrl];
		clientConfig.configuration = config;

Create AOOceanaCustomerWebVoiceVideo's Object	This is the entry point of the Avaya WebRTC SDK 4.0 To instantiate Audio/Video interactions, the AOOceanaCustomerWebVoiceVideo's object needs to be created.	AOOceanaCustomerWebVoiceVideo *customerWebVoice = [[AOOceanaCustomerWebVoiceVideo alloc]initWithClientConfiguration:clientConfig];

Create and setup AOWork's Object.	To set the routing strategy, local, services, topic, etc the AOWork's object needs to be created. These are the routing strategies used to route the call.	AOWork* work = customerWebVoice.createWork;
		work.context = <context_string>;
		work.topic = <topic>
		work.locale = <local>;
		work.routingStrategy = <routing_strategy>;
		work.services = <services>;
		work.resources = <resources>; //If available

Steps to create Video Interaction	Objective	Avaya WebRTC Connect APIs
	These parameters can be provided with the Lab details.	
Create AOVideoInteraction's object	To make an audio call, AOVideoInteraction's object needs to be created.	AOVideoInteraction *videoInteraction = work.createVideoInteraction;
Set up AOVideoInteraction's object	The properties like setPlatformType(Oceana/Elite), delegate, connectionListenerDelegate, and destination address.	videoInteraction.delegate = videoDelegate; audioInteravideoInteractionction.connectionListenerDelegate = connectionDelegate; videoInteraction. destinationAddress = <remote_address>
Create AOVideoDevice's Object	To change the video capture resolution, orientation, camera type(front-back), set remote and local rendering views.	AOVideoDevice *device = videoInteraction.videoDevice;

At this point, the video interaction object is created. Now to place the video call, we need to start the video interaction and before that, some properties need to be set like PlatformType(Oceana/Elite), delegate, connectionListenerDelegate, and destination address.

This is how the video call/interaction is started in AMV 3.X and Avaya WebRTC Connect with all the video Call Features.

Video Call Features	AMV 3.X	Avaya WebRTC 4.0
Start Call/Interaction	[cpSession start];	[videoInteraction start];
End Call	[cpSession end];	[videoInteraction end];
Discard the call	[cpUser terminate];	[videoInteraction discard];
Mute/Un-mute Audio	[cpSession muteAudio: bool_value];	[videoInteraction muteAudio: bool_value];

Video Call Features	AMV 3.X	Avaya WebRTC 4.0
Mute/Un-mute Video	[session muteVideo: bool_value];	[videoInteraction muteVideo: bool_value];
Enable/Disable Video	[session enableVideo:bool_value];	[videoInteraction enableVideo:bool_value];
Set Camera type (Front/Back)	device.cameraType = CPCameraType_<type>;	[device selectCamera:<camera_type>];
Switch Camera.	device.cameraType = CPCameraType_<new_type>;	[device switchCamera];
Set video capture resolution.	device.cameraCaptureResolution = <resolution>;	[device setVideoCaptureResolutionWithCaptureOrientation:<video_record_preference> orientationPreference:<video_orientation_preference>];
Set video capture orientation.	device.cameraOrientation = CPCameraOrientation_<value>	[device setVideoCaptureResolutionWithCaptureOrientation:<video_record_preference> orientationPreference:<video_orientation_preference>];
Get video capture resolution.	CPVideoResolution object = device.cameraCaptureResolution;	AOVideoCapturePreference object = getVideoCapturePreference;
Get video capture orientation.	CPCameraOrientation object = device.cameraOrientation;;	AOVideoCaptureOrientation object = getVideoCaptureOrientation;
Hold Call	[cpSession hold];	[videoInteraction holdWithCompletionHandler:^(NSError *error){}];
Resume Call	[cpSession resume];	[videoInteraction unholdWithCompletionHandler:^(NSError *error){}];

Video Call Features	AMV 3.X	Avaya WebRTC 4.0
Get Time Elapsed	[cpSession getCallTimeElapsed];	[videoInteraction getInteractionTimeElapsed];
Get Current Call Sate	CPSessionState state = cpSession.state;	[videoInteraction getInteractionState];
Send DTMF tone	[cpSession sendDTMF:tone andPlayAudio:YES]; // And [cpSession sendDTMF:tone];	[videoInteraction sendDTMF:tone];
Send hands-free Audio (phone-speaker)	[cpDevice setHandsfreeAudio:<bool_value>];	This can be handled at the Application layer. Use iOS' ***AVRoutePickerView*** to change the Audio routes.

Get Call Quality. In AMV 3.X, the CPBaseSession's didChangeQuality API gives the current call quality whereas, in Avaya WebRTC 4.0, the call quality can be determined by ***readAudioDetailsWithCompletionHandler*** API which returns **AOAudioDetails** object & ***readVideoDetailsWithCompletionHandler*** API which returns ***AOVideoDetails*** object.	Implement CPBaseSession's -(void)session:(CPBaseSession *)session didChangeQuality: (int)quality;	[videoInteraction readAudioDetailsWithCompletionHandler:^(AOAudioDetails *audioDetails) {}];
		[video readVideoDetailsWithCompletionHandler:^(AOVideoDetails *videoDetails) {}];

Notes

- The `sendDTMF` operation takes an enumeration as an argument. This enumeration provides values for DTMF keys 0-9, *, #, and A-D. However, A-D are not supported in the current release of Avaya Oceana™.

- On Android, if the buildToolsVersion is less than 26.0, you will receive the following error when adding Java 1.8 target compatibility - “Jack is required to support Java 8 language features. Either enable Jack or remove sourceCompatibility JavaVersion.VERSION_1_8”. In order to remove this error, either specify buildToolsVersion 26 or newer or add the following to your build.gradle file:

```
defaultConfig {
jackOptions {
   enabled true
      }
   }
```