



Transcript Access for Messaging and Email Contacts

Dec 2022
Version 3.10.0.0



AVAYA SOFTWARE DEVELOPMENT KIT LICENSE AGREEMENT

REVISED: January 14, 2022

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.

1. 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.
2. 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.

1. 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).
2. 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

1. "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".

1. 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.

1. 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.

1. 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 defend, 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 17 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 AND IMPORT/EXPORT CONTROL.

Licensee shall comply with all applicable laws and regulations, including without limitation those applicable to data privacy, intellectual property, trade secret, and fraud. Licensee is advised that the Technical Information is of U.S. origin and subject to the U.S. Export Administration Regulations ("EAR") and 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 Technical Information 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 Technical Information 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 Technical Information 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.

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 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.

17.0 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.

18.0 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](http://www.mpegla.com).

Overview

This document describes a mechanism to provide access for end-customers to transcripts from Oceana. For messaging contacts (Chat, SMS, and social media), this uses the transcript filtering service, which may also be used to remove sensitive data such as credit card details from messaging transcripts before they are persisted to the database. For email contacts, this occurs when the email transcript is sent to the configured end-point.

Terminology

This guide uses the following terms:

- "Legacy transcript" refers to any version prior to Oceana 3.6.0.0.
- "Enhanced transcript" refers to the extended version added in Oceana 3.6.0.0. This includes extra fields in the request and the sample application now provides the ability to store the filtered transcripts outside Oceana.
- "OCP" is an internal shorthand for the Omni-Channel components, e.g. the OCP cluster is the Avaya Breeze cluster that hosts the Omni-Channel components, such as the CustomerControllerService and EmailService.

API details

This section details the required API for Messaging and Email contacts, as expected by the CustomerControllerService and EmailService Oceana components.

Latency

For Oceana to operate effectively the max time for a response from the customer filtering service is 150ms.

ChatMessage.java

This is the syntax for a chat message in the legacy transcript service. The API expects to send a List of these, and to receive a List in return. It contains the following fields:

- MessageType: a static enum that holds the types of message that may be sent to the filtering service. The following types are available:
 - NORMAL - normal chat messages, visible to both agents and customers
 - WELCOMEMESSAGE - a welcome message that is played when the first agent joins the room.
 - WHISPER - a message that is visible to agents only.
 - COMFORTMESSAGE - a message that is sent automatically when the agent is inactive.
 - PAGEPUSH - a message that the agent sent as part of a PagePushMessage as configured in the Omnichannel Administration Utility. This only applies if the message was sent to the customer; if sent during a coaching session, it will be of type WHISPER.
 - FILETRANSFER - A file transfer message. The 'message' field will contain the URL from which the attachment can be retrieved
- sender: the display name of the sender, e.g. John Smith. If this is from an agent, it will be prefaced with "A:" (e.g. A: "John Smith"); if it is from the customer, it will be prefaced with "C:" (e.g. "C: John Smith"); if it is from a bot, it will be prefaced with "B:" (e.g. "B: Ava")
- message: the actual message text, e.g. "Hello"
- date: the number that holds the timestamp of the message in Unix format (Unix time is a date-time format used to express the number of milliseconds that have elapsed since January 1, 1970 00:00:00 (UTC))
- messageType: the value of TranscriptMessageType for this message.

The ChatMessage class contains the following methods:

- Getter and setter methods for the above fields. Getters return the current value of the field, and setters update the value of the field.
- An empty constructor. This is required for conversion to a JSON object.
- A constructor that takes a String for the sender, a String for the message, a long for the date, and a MessageType for the messageType.
- A getFilteredMessage() method that contains a sample logic for message filtering

The code for this is:

```
package com.avaya.customer.example.servlet.api;

/**
 * Representation of messages sent during a chat for transcript filtering
 */
public class ChatMessage implements Filter {

    /** The display name of the sender. */
    protected String sender;

    /** The message or page push URL. */
    protected String message;

    /** The timestamp of the message. */
    protected long date;
```

```
/** The message type. */
protected MessageType messageType;

/**
 * Empty constructor for Json
 */
public ChatMessage() {

}

/**
 * Instantiates a new chat message.
 * @param sender the sender
 * @param message the message
 * @param date the date
 * @param messageType the message type
 */
public ChatMessage(String sender, String message, long date, MessageType messageType) {
    this.sender = sender;
    this.message = message;
    this.date = date;
    this.messageType = messageType;
}

/**
 * Gets the sender.
 * @return the sender
 */
public String getSender() {
    return sender;
}

/**
 * Sets the sender.
 * @param sender the new sender
 */
public void setSender(String sender) {
    this.sender = sender;
}

/**
 * Gets the message.
 * @return the message
 */
public String getMessage() {
    return message;
}

/**
 * Sets the message.
 * @param message the new message
 */
public void setMessage(String message) {
    this.message = message;
}

/**
 * Gets the date.
 * @return the date
 */
public long getDate() {
    return date;
}

/**
 * Sets the date.
 * @param date
 */
public void setDate(long date) {
    this.date = date;
}
```

```

}

/**
 * @return the messageType
 */
public MessageType getMessageType() {
    return messageType;
}

/**
 * @param messageType the messageType to set
 */
public void setMessageType(MessageType messageType) {
    this.messageType = messageType;
}

/**
 * @return the filtered chat message
 */
@Override
public ChatMessage getFilteredMessage() {
    return new ChatMessage(sender,message + " been filtered",
        date, messageType);
}

@Override
public String toString() {
    return "ChatMessage [sender=" + sender + ", " +
        "message=***" + ", date=" + date + ", messageType=" + messageType + " ]";
}
}

```

MessageTranscriptV1.java

This is the syntax for the V1 enhanced transcript message in the transcript service. The API expects to send a List of these, and to receive a List in return. It contains the same fields as ChatMessage, along with the following additional fields:

- contactId: represents the contact in the Omnichannel database.
- customerId: represents the customer in the Omnichannel database.
- workRequestId: represents the contact in Oceana.
- channelType: represents the channel. Can be CHAT, SMS, SOCIAL or EMAIL.

The MessagingTranscript class contains the following methods:

- Getter and setter methods for the above fields. Getters return the current value of the field, and setters update the value of the field.
- An empty constructor. This is required for conversion to a JSON object.
- A getFilteredMessage() method that contains a sample logic for message filtering

The code for this is:

```

package com.avaya.customer.example.servlet.api;

public class MessageTranscript extends ChatMessage {

    /** The ID of Customer Contact. */
    private int contactId;

    /** The ID of Customer. */
    private int customerId;

    /** The ID of Chat Request. */
    private String workRequestId;

    public MessageTranscript() {
        // required for JSON conversion
    }

    public MessageTranscript(String sender, int contactId, int customerId, String workRequestId, String
message, long date, MessageType messageType, ChannelType channelType) {

        super(sender, message, date, messageType);
        this.contactId = contactId;
        this.customerId = customerId;
        this.workRequestId = workRequestId;
        this.channelType = channelType;
    }

    public int getContactId() {
        return contactId;
    }

    public void setContactId(int contactId) {
        this.contactId = contactId;
    }

    public int getCustomerId() { return customerId; }

    public void setCustomerId(int customerId) {
        this.customerId = customerId;
    }

    public String getWorkRequestId() {
        return workRequestId;
    }

    public void setWorkRequestId(String workRequestId) {
        this.workRequestId = workRequestId;
    }

    public ChannelType getChannelType () {
        return channelType;
    }

    public void setChannelType(ChannelType channelType) {
        this.channelType = channelType;
    }

    @Override
    public String toString() {
        return "ChatMessage [sender=" + sender + ", contactId=" + contactId + ", customerId=" + customerId + ",
workRequestId=" + workRequestId + "," +
            "message=***" + message + ", date=" + date + ", messageType=" + messageType + ", channelType="
+ channelType + "];";
    }
}

```

MessageTranscriptV2.java

This is the syntax for the V2 enhanced transcript message in the transcript service. The API expects to send a List of these along with a TranscriptContext object and to receive a List and context in return. It contains the same fields as ChatMessage, along with the following additional fields:

- messageId: represents the unique message id from external messaging aggregator application.
- customData: A structured JSON object which can be used to add any custom properties and will be shared with automation and agent desktop clients.
- richMediaType: identifier for the rich media message type (text, image, file, location, etc).

The code for this is:

```
package com.avaya.customer.example.servlet.api;

public class MessageTranscriptV2 extends ChatMessage {

    private String messageId;

    private String customData;

    private String richMediaType;

    public MessageTranscriptV2() {
    }

    public MessageTranscriptV2(String sender, String message, long date, MessageType messageType,
        ChannelType channelType, String messageId, String customData, String richMediaType) {
        super(sender, message, date, messageType, channelType);
        this.messageId = messageId;
        this.customData = customData;
        this.richMediaType = richMediaType;
    }

    public String getMessageId() {
        return messageId;
    }

    public void setMessageId(String messageId) {
        this.messageId = messageId;
    }

    public String getCustomData() {
        return customData;
    }

    public void setCustomData(String customData) {
        this.customData = customData;
    }

    public String getRichMediaType() {
        return richMediaType;
    }

    public void setRichMediaType(String richMediaType) {
        this.richMediaType = richMediaType;
    }

    /**
     * @return the filtered transcript message
     */
    @Override
    public MessageTranscriptV2 getFilteredMessage() {
        MessageType type = this.getMessageType();
        return (type == MessageType.PAGEPUSH || type == MessageType.FILETRANSFER) ? this : new
MessageTranscriptV2(
            sender,
            message + " been filtered",
            date,
            messageType,
            channelType,
```

```

        messageId,
        customData,
        richMediaType
    );
}

@Override
public String toString() {
    return "MessageTranscriptV2{" + "messageId='" + messageId + '\'' + ", customData='" + customData + '\''
        + ", richMediaType='" + richMediaType + '\'' + ", sender='" + sender + '\'' + ", message='" +
message
        + '\'' + ", date=" + date + ", messageType=" + messageType + ", channelType=" + channelType +
}'';
}
}

```

TranscriptContext.java

This is the syntax for the V2 enhanced transcript context in the transcript service. The API expects to send this along with a List of TranscriptMessageV2. java objects and to receive the context and List in return. It has the following fields:

- contactId: represents the contact in the Omnichannel database.
- customerId: represents the customer in the Omnichannel database.
- workRequestId: represents the contact in Oceana.
- conversationId: represents the unique Id for a given conversation. (Specific to Async Messaging)
- channelSrc: identifier for the channel from which a message originated. May include one of "web", "ios", "android", "messenger", "whatsapp", "wechat", "line" or any number of other channels. (Specific to Async Messaging)
- endUserId: unique Id of external end-user for Oceana to correlate internal user for various features such as customer journey, history, etc. (Specific to Async Messaging)
- tenantId: tenant Id from messaging provider (aggregator appld); unique and immutable. (Specific to Async Messaging)

The code for this is:

```

package com.avaya.customer.example.servlet.api;

public class TranscriptContext {

    /**
     * The ID of Customer Contact
     */
    protected int contactId;

    /**
     * The ID of Customer
     */
    protected int customerId;

    /**
     * The ID of Chat Request
     */
    protected String workRequestId;

    /**
     * unique Id for a given conversation; unique and immutable
     */
    protected String conversationId;

    /**
     * identifier for the channel from which a message originated. May include one of "web", "ios", "android",
     * "messenger", "whatsapp", "wechat", "line" or any number of other channels.
     */
    protected String channelSrc;

    /**
     * unique Id of external end-user for Oceana to correlate internal user for various features such as
     customer
     * journey, history, etc.
     */
    protected String endUserId;
}

```

```
/**
 * Tenant Id from messaging provider (aggregator appId); unique and immutable
 */
protected String tenantId;

public TranscriptContext() {
}

public TranscriptContext(int contactId, int customerId, String workRequestId, String conversationId,
    String channelSrc, String endUserId, String tenantId) {
    this.contactId = contactId;
    this.customerId = customerId;
    this.workRequestId = workRequestId;
    this.conversationId = conversationId;
    this.channelSrc = channelSrc;
    this.endUserId = endUserId;
    this.tenantId = tenantId;
}

public int getContactId() {
    return contactId;
}

public void setContactId(int contactId) {
    this.contactId = contactId;
}

public int getCustomerId() {
    return customerId;
}

public void setCustomerId(int customerId) {
    this.customerId = customerId;
}

public String getWorkRequestId() {
    return workRequestId;
}

public void setWorkRequestId(String workRequestId) {
    this.workRequestId = workRequestId;
}

public String getConversationId() {
    return conversationId;
}

public void setConversationId(String conversationId) {
    this.conversationId = conversationId;
}

public String getChannelSrc() {
    return channelSrc;
}

public void setChannelSrc(String channelSrc) {
    this.channelSrc = channelSrc;
}

public String getEndUserId() {
    return endUserId;
}

public void setEndUserId(String endUserId) {
    this.endUserId = endUserId;
}

public String getTenantId() {
    return tenantId;
}
```

```

    }

    public void setTenantId(String tenantId) {
        this.tenantId = tenantId;
    }

    @Override
    public String toString() {
        return "TranscriptContext{" + "contactId=" + contactId + ", customerId=" + customerId + ",
workRequestId='"
            + workRequestId + '\'' + ", conversationId='" + conversationId + '\'' + ", channelSrc='" +
channelSrc
            + '\'' + ", endUserId='" + endUserId + '\'' + ", tenantId='" + tenantId + '\'' + '\''};
    }
}

```

TranscriptContextAndMessage.java

This is the syntax for the V2 enhanced transcript context and the list of MessageTranscriptV2 objects in the transcript service. The API expects to send this TranscriptContextAndMessage object and to receive the same object in return. It has the following fields:

- transcriptContext: represents the context object mentioned above.
- transcriptMessages: represents the list of MessageTranscriptV2 objects mentioned above.

The code for this is:

```

package com.avaya.customer.example.servlet.api;

import java.util.List;

public class TranscriptContextAndMessage {

    private TranscriptContext transcriptContext;

    private List<MessageTranscriptV2> transcriptMessages;

    public TranscriptContextAndMessage() {

    }

    public TranscriptContextAndMessage(TranscriptContext transcriptContext,
        List<MessageTranscriptV2> transcriptMessages) {
        this.transcriptContext = transcriptContext;
        this.transcriptMessages = transcriptMessages;
    }

    public TranscriptContext getTranscriptContext() {
        return transcriptContext;
    }

    public void setTranscriptContext(TranscriptContext transcriptContext) {
        this.transcriptContext = transcriptContext;
    }

    public List<MessageTranscriptV2> getTranscriptMessages() {
        return transcriptMessages;
    }

    public void setTranscriptMessages(List<MessageTranscriptV2> transcriptMessages) {
        this.transcriptMessages = transcriptMessages;
    }

    @Override
    public String toString() {
        return "TranscriptContextAndMessage{" + "transcriptContext=" + transcriptContext + ",
transcriptMessages="
            + transcriptMessages + '}';
    }
}

```

MessageType

Defines the possible types of messages.

```

package com.avaya.customer.example.servlet.api;

public enum MessageType {
    /** A message visible to both agents and customers */
    NORMAL,
    /** A message configured to send when the first agent joins the room */
    WELCOMEMESSAGE,
    /** A message only visible to agents in the room */
    WHISPER,
    /** A comfort message automatically sent due to agent inactivity */
    COMFORTMESSAGE,
    /** A page push URL. The 'message' field will contain the URL. Only applies to URLs sent to the
customer. */
    PAGEPUSH,
    /** A file transfer message. The 'message' field will contain the file name */
    FILETRANSFER,
    /** An async message visible to both agents and customers. The 'message' field will contain the
text fallback */
    ASYNCMESSAGE
}

```

EmailTranscript.java

This is the syntax for an Email message in the transcript service. The API expects to send a single email transcript per request and to receive 200 OK response in return on success. There are 2 types of Email transcripts: plain transcript and transcript with mailbox. The transcript with mailbox contains all fields of the plain transcript plus a mailbox which is used by Oceana for receiving/sending the particular email. The plain transcript contains the following fields:

- direction: the direction of the email (Incoming, Outgoing etc.)
- customerId: represents the customer in the Omnichannel database.
- contactId: represents the contact in the Omnichannel database.
- workRequestId: represents the contact in Oceana.
- mailFrom: the email address of the originator
- mailTo: the email address(es) of the message's recipient(s)
- mailCC: the email address(es) of the carbon copy recipient(s)
- mailBCC: the email address(es) of the blind carbon copy recipient(s)
- subject: represents the Subject field of the email.
- timestamp: the number that holds the timestamp of the email in Unix format (Unix time is a date-time format used to express the number of milliseconds that have elapsed since January 1, 1970 00:00:00 (UTC))
- attachments: a List of the attachments links
- bodyEmail: the body of the email
- messageId: a representation of the email's [Message-ID](#) field. This will identify the email on the email server.

The EmailTranscript class contains the following methods:

- Getter and setter methods for the above fields. Getters return the current value of the field, and setters update the value of the field.
- An empty constructor. This is required for conversion to a JSON object.
- getFieldLength. This is used when building a toString method to obfuscate sensitive data when logging.

The code for this is:

```

package com.avaya.customer.example.servlet.api;

import com.fasterxml.jackson.annotation.JsonIgnore;
import java.util.List;

/**
 * Representation of Email transcript message
 */
public class EmailTranscript {

    private String direction;
    private Long customerId;
    private Long contactId;
    private String workRequestId;
    private String mailFrom;

    private List<String> mailTo = null;

```

```
private List<String> mailCC = null;
private List<String> mailBCC = null;
private String subject;
private Long timestamp;
private List<String> attachments = null;
private String bodyEmail;
    private String messageId;

    /**
     * Empty constructor for Json
     */
public EmailTranscript(){
    // required for JSON conversion
}

public String getDirection() {
    return direction;
}

public void setDirection(String direction) {
    this.direction = direction;
}

public Long getCustomerId() {
    return customerId;
}

public void setCustomerId(Long customerId) {
    this.customerId = customerId;
}

public Long getContactId() {
    return contactId;
}

public void setContactId(Long contactId) {
    this.contactId = contactId;
}

public String getWorkRequestId() {
    return workRequestId;
}

public void setWorkRequestId(String workRequestId) {
    this.workRequestId = workRequestId;
}

public String getMailFrom() {
    return mailFrom;
}

public void setMailFrom(String mailFrom) {
    this.mailFrom = mailFrom;
}

public List<String> getMailTo() {
    return mailTo;
}

public void setMailTo(List<String> mailTo) {
    this.mailTo = mailTo;
}

public List<String> getMailCC() {
    return mailCC;
}

public void setMailCC(List<String> mailCC) {
    this.mailCC = mailCC;
}
```

```

public List<String> getMailBCC() {
    return mailBCC;
}

public void setMailBCC(List<String> mailBCC) {
    this.mailBCC = mailBCC;
}

public String getSubject() {
    return subject;
}

public void setSubject(String subject) {
    this.subject = subject;
}

public Long getTimestamp() {
    return timestamp;
}

public void setTimestamp(Long timestamp) {
    this.timestamp = timestamp;
}

public List<String> getAttachments() {
    return attachments;
}

public void setAttachments(List<String> attachments) {
    this.attachments = attachments;
}

public String getBodyEmail() {
    return bodyEmail;
}

public void setBodyEmail(String bodyEmail) {
    this.bodyEmail = bodyEmail;
}

    public String getMessageId () {
        return messageId;
    }

    public void setMessageId(String messageId) {
        this.messageId = messageId;
    }

@JsonIgnore
private int getFieldLength (String field){
    return (field != null ? field.length() : 0);
}

@Override
public String toString() {
    return "EmailTranscript{" +
        "direction=" + direction + '\'' +
        ", customerId=" + customerId +
        ", contactId=" + contactId +
        ", workRequestId=" + workRequestId + '\'' +
        ", mailFrom=" + mailFrom + '\'' +
        ", mailTo=" + mailTo +
        ", mailCC=" + mailCC +
        ", mailBCC=" + mailBCC +
        ", subject=" + getFieldLength(subject) + '\'' +
        ", timestamp=" + timestamp +
        ", attachments=" + attachments +
        ", bodyEmail=" + getFieldLength(bodyEmail) + '\'' +
        '}';
}
}

```

RestTranscriptAPI.java

This interface describes the expected syntax of a request to the filtering service, and the expected response. Implement this to create a customised transcript processing mechanism (filtering, storing etc). It contains the following methods:

- processEmailTranscript
- processMessageTranscript
- filterContactChatConversation
- echo

They use a similar base URL, which for the below example is <http://127.0.0.1/CustomerFilterMessages/rest/filter>.

processEmailTranscript

There are 2 methods for processing email transcripts. One of them expects a single EmailTranscript passed in as a JSON object and another one - EmailTranscriptWithMailbox instance. They both return 200 OK HTTP Response, if the message is correct. The paths end with **/transcript/email** and **/transcript/emailwithmailbox** respectively. For ex., <http://127.0.0.1/CustomerFilterMessages/rest/filter/transcript/email> and <http://127.0.0.1/CustomerFilterMessages/rest/filter/transcript/emailwithmailbox>. The type of request is POST:

```
@POST@Path("/transcript/email")
@Consumes(MediaType.APPLICATION_JSON)
public Response processEmailTranscript(EmailTranscript transcript);

@POST
@Path("/transcript/emailwithmailbox")
@Consumes(MediaType.APPLICATION_JSON)
public Response processEmailTranscript(EmailTranscriptWithMailbox transcript);
```

processV1MessageTranscript

This is a method for processing V1 enhanced message transcripts. It expects a List of MessageTranscript objects passed in as a JSON array, and it returns this list after filtering out sensitive data. The path it expects relative to the base URL above is **/enhancedtranscript**, e.g. <http://127.0.0.1/CustomerFilterMessages/rest/filter/enhancedtranscript>. This must be accessed using a POST call:

```
@POST@Path("/enhancedtranscript")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public List<MessageTranscript>
processMessageTranscript(List<MessageTranscript> transcript);
```

processV2MessageTranscript

This is a method for processing V2 enhanced message transcripts. It expects a TranscriptContextAndMessage, and it returns this object after filtering out sensitive data. The path it expects relative to the base URL above is **/v2/enhancedtranscript**, e.g. <http://127.0.0.1/CustomerFilterMessages/rest/filter/v2/enhancedtranscript>. This must be accessed using a POST call:

```
@POST
@Path("/v2/enhancedtranscript")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public TranscriptContextAndMessage processV2MessageTranscript(TranscriptContextAndMessage transcript);
```

filterContactChatConversation

This is the main method for filtering standard chat contacts. It expects a List of ChatMessage objects passed in as a JSON array, and it returns this list after filtering out sensitive data. The path it expects relative to the base URL above is **/transcript**, e.g. <http://127.0.0.1/CustomFilterMessages/filter/transcript>. This must be accessed using a POST call:

```
@POST@Path("/transcript")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public List<ChatMessage>
filterContactChatConversation(List<ChatMessage> transcript);
```

echo

This method simply echoes a message that is passed in as a parameter at the end of the URL, e.g. <http://127.0.0.1/CustomFilterMessages/rest/filter/echo/Hello> will produce a plain text response to the message "Hello". A sample method might simply echo the word "Hello". The purpose of this method is to provide a testing mechanism to confirm that the filtering service is accessible:

```
@GET
@Path("/echo/{message}")
@Produces("text/plain")
public String echo(@PathParam("message") String message);
```

Below is the full syntax for the interface.

```

package com.avaya.customer.example.servlet.api;

import java.util.List;
import javax.ws.rs.Consumes;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

/**
 * The interface that is expected for processing transcript messages.
 */
public interface RestTranscriptAPI {

    /**
     * Endpoint for processing Email transcripts
     */
    @POST
    @Path("/transcript/email")
    @Consumes(MediaType.APPLICATION_JSON)
    public Response processEmailTranscript(EmailTranscript transcript);

    @POST
    @Path("/transcript/emailwithmailbox")
    @Consumes(MediaType.APPLICATION_JSON)
    public Response processEmailTranscript(EmailTranscriptWithMailbox transcript);

    /**
     * Endpoint for processing V1 Messaging transcripts.
     */
    @POST
    @Path("/enhancedtranscript")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public List<MessageTranscriptV1> processV1MessageTranscript(List<MessageTranscriptV1> transcript);

    /**
     * Endpoint for processing V2 Messaging transcripts.
     */
    @POST
    @Path("/v2/enhancedtranscript")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public TranscriptContextAndMessage processV2MessageTranscript(TranscriptContextAndMessage transcript);

    /**
     * Endpoint for processing old chat transcripts.
     */
    @POST
    @Path("/transcript")
    @Produces(MediaType.APPLICATION_JSON)
    @Consumes(MediaType.APPLICATION_JSON)
    public List<ChatMessage> filterContactChatConversation(List<ChatMessage> transcript);

    /**
     * Endpoint for sanity testing
     */
    @GET
    @Path("/echo/{message}")
    @Produces("text/plain")
    public String echo(@PathParam("message")String message);
}

```

SampleTranscriptService.java

This is a sample implementation of the RestTranscriptAPI interface. It listens on the "filter" path. In this case, it takes the content of the message, filters and saves it as *.json file to the following paths accordingly:

- email transcripts: %TOMCAT_HOME%/temp/CustomFilterMessages/email/\${workRequestId}.json
 - attachments, if they exist: %TOMCAT_HOME%/temp/CustomFilterMessages/email/\${workRequestId}_attachments/*
- enhanced messaging transcripts: %TOMCAT_HOME%/temp/CustomFilterMessages/messaging/\${workRequestId}.json
- chat transcripts: %TOMCAT_HOME%/temp/CustomFilterMessages/messaging/\${date}.json

Do **not** edit this class. Instead, create your own class that implements the RestTranscriptAPI interface, and edit the FilterApplication class to reference that instead of this class.



```

package com.avaya.customer.example.servlet;

import com.avaya.customer.example.servlet.api.ChatMessage;
import com.avaya.customer.example.servlet.api.EmailTranscript;
import com.avaya.customer.example.servlet.api.MessageTranscript;
import com.avaya.customer.example.servlet.api.RestTranscriptAPI;
import com.avaya.customer.example.servlet.utils.FileUtils;
import javax.ws.rs.Path;
import javax.ws.rs.core.Response;
import java.util.ArrayList;
import java.util.List;

/**
 * A sample transcript service.
 */
@Path("filter")
public class SampleTranscriptService implements RestTranscriptAPI {

    @Override
    public Response processEmailTranscript(EmailTranscript transcript) {
        FileUtils.saveEmailTranscript(transcript);
        return Response.ok().build();
    }

    @Override
    public Response processEmailTranscript(EmailTranscriptWithMailbox transcript) {
        FileUtils.saveEmailTranscript(transcript);
        return Response.ok().build();
    }

    @Override
    public List<MessageTranscriptV1> processV1MessageTranscript(List<MessageTranscriptV1> transcript) {
        List<MessageTranscriptV1> returnedList = new ArrayList<>();
        for (MessageTranscriptV1 message : transcript) {
            returnedList.add(message.getFilteredMessage());
        }
        FileUtils.saveMessageTranscript(returnedList);
        return returnedList;
    }

    @Override
    public TranscriptContextAndMessage processV2MessageTranscript(TranscriptContextAndMessage transcript) {
        List<MessageTranscriptV2> returnedList = new ArrayList<>();
        for (MessageTranscriptV2 message : transcript.getTranscriptMessages()) {
            returnedList.add(message.getFilteredMessage());
        }

        TranscriptContextAndMessage transcriptContextAndMessage =
            new TranscriptContextAndMessage(transcript.getTranscriptContext(), returnedList);
        FileUtils.saveTranscriptAndContext(transcriptContextAndMessage);
        return transcriptContextAndMessage;
    }

    @Override
    public List<ChatMessage> filterContactChatConversation(List<ChatMessage> transcript) {
        List<ChatMessage> returnedList = new ArrayList<>();
        for (ChatMessage chatMessage : transcript) {
            returnedList.add(chatMessage.getFilteredMessage());
        }
        FileUtils.saveChatTranscript(returnedList);
        return returnedList;
    }

    @Override
    public String echo(String message) {
        return message;
    }
}

```

Access to attachments

The transcript service provides a sample token based mechanism to access email attachments.

The mechanism to download attachments is represented in **DownloadManager.java**. It will try to download all attachments links that are contained in the email transcript's `attachments` field. The default download path on a Tomcat server will be `%TOMCAT_HOME%/temp/CustomFilterMessages/email/{workrequestId}_attachments/`, where `%TOMCAT_HOME%` is the Tomcat root directory on Windows. On Linux servers, this would be `$TOMCAT_HOME/temp/CustomFilterMessages/email/{workrequestId}_attachments/`.

Authorization.java

This class contains a logic to receive access token from Oceana AuthorizationService

```
package com.avaya.customer.example.servlet.utils;

import com.avaya.collaboration.authorization.AccessToken;
import com.avaya.collaboration.authorization.AuthorizationClientHelper;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Authorization {

    private static final Logger LOG = LoggerFactory.getLogger(Authorization.class);

    static AccessToken getAccessToken(Configuration configuration) throws Exception {

        final String authorizationUrl = Utils.buildHttpSecureUrl(configuration.getAuthorizationHost() + ":"
            + configuration.getAuthPort(), "/services/AuthorizationService/token");

        AuthorizationClientHelper authorizationClientHelper = null;
        try {
            LOG.info("Getting access token from {} ...", authorizationUrl);
            authorizationClientHelper = new AuthorizationClientHelper.Builder()
                .tokenEndpoint(authorizationUrl)
                .clientIdentifier(configuration.getClientId())
                .keyStore(configuration.getClientKeyStore(), configuration.getKeyStorePassword(),
configuration.getKeyAlias())
                .trustStore(configuration.getClientTrustStore())
                .build();

            AccessToken accessToken = authorizationClientHelper.getAccessToken();
            LOG.info("Token received");

            return accessToken;
        } catch (Exception e) {
            LOG.error("Exception thrown during getting access token: ", e);
        } finally {
            if (authorizationClientHelper != null) {
                authorizationClientHelper.shutdown();
            }
        }
        return null;
    }

    private Authorization() {}
}
```

To configure token-based access, certificates must be configured.

FilterApplication.java

This class defines the classes that are loaded. In this case, it loads the `SampleTranscriptService` class. To load your own customised version instead of `SampleTranscriptService`, replace `"SampleTranscriptService.class"` inside the `getClasses` method with a reference to your own class.

```
package com.avaya.customer.example.servlet;

import java.util.Collections;
import java.util.Set;

/**
 * Defines the classes that are included in the filtering application.
 * Change the class referenced in the "getClasses" method to load your own class.
 */
public class FilterApplication extends javax.ws.rs.core.Application {

    @Override
    public Set<Class<?>> getClasses() {
        return Collections.<Class<?>> singleton(SampleTranscriptService.class);
    }

    @Override
    public Set<Object> getSingletons() {
        return Collections.emptySet();
    }
}
```

Building the transcript service

1. Open the CustomerFilterMessages service in an IDE or text editor of your choice.
2. Run a Maven build for the service: `mvn clean install`.
3. Find the built WAR package in the target directory in the projects root, e.g: `./CustomerFilterMessages/target/CustomerFilterMessages-11.0-SNAPSHOT.war`

Deploying the transcript service

The sample transcript service is a WAR file. This should be deployed separately to Oceana, and should be accessed over a secure connection. Ideally, this would be deployed inside the corporate network, and would not be accessible from the public Internet. You may need to configure the firewall on the server that will host this to allow connections over ports other than 80 or 443.

There is a known issue where if the filtering service fails, the transcript will be persisted to the database unfiltered.

Prerequisites

You must have an application server that is capable of deploying WAR files, e.g. Tomcat. This should have a certificate or keystore generated.

Generating the certificate/keystore for TLS

1. Generate a private key and CSR file using OpenSSL as explained in the code block below. Do **NOT** use the password "tomcat" or any similarly obvious or weak password.
2. Log into System Manager and visit *Services Security Certificates Authority*.
3. Click "Add Entity" to add a new user that will represent the Tomcat server. Enter the following details and save:
 - a. Username (e.g. tomcat, tomcat-filtering, Tomcat). Take note of this.
 - b. Password or Enrollment Code. Take note of this.
 - c. Domain name attributes to identify the server.
 - d. An email address (optional).
 - e. Additional DNS names and an IP address. These are optional, but will allow greater flexibility.
4. Click the button for "Public Web". This will open the public EJBCA pages.
5. Click "Create Certificate from CSR" in the left-hand panel.
6. Enter the username and password from step 3a/b. Paste the contents of the CSR file into the "pasted request" box.
7. When the certificate is generate, download the PEM file and root cert. See the code block below for the OpenSSL instructions required to convert these to a PKCS12 keystore.
8. When the keystore is generated, configure Tomcat to support HTTPS as follows:
 - a. Put the tomcat.p12 file somewhere on the Tomcat host.
 - b. Open Tomcat's server.xml file and locate the HTTPS connector
 - c. Uncomment the connector and add the following attributes to it: `keystoreFile="path/to/tomcat.p12" keystorePass="tomcat"`

```
# Set OpenSSL configuration file. This could also be set as a permanent environment variable
export OPENSSL_CONF="path/to/openssl.cnf"

# Generate private key using default parameters (and password of tomcat)
# Note: DO NOT USE THIS PASSWORD IN PRODUCTION!
openssl genpkey -algorithm RSA -out tomcat.key -pass pass:tomcat

# Generate CSR file
openssl req -new -key tomcat.key -out tomcat.csr

# Convert to PKCS12, once you have the cert authority's root cert, the new Tomcat cert, and the private key.
openssl pkcs12 -export -in tomcat.pem -inkey tomcat.key -out tomcat.p12 -name tomcat -CAfile myCA.pem -caname
root -chain
```

Configuring the transcript service to access attachments

This section describes setting up a non snap-in environment to work with Authorization Service and get an access token. The write-up assumes that the system/machine has **OpenSSL** and the Java **keytool** available.

1. The Authorization Service expects valid signed JSON Web Tokens from clients to authenticate them. For this, a client would first need to have a private key. We use OpenSSL here to generate an example key pair:

```
# Generates a key pair for the application
openssl genrsa -aes256 -out client.key 2048
```

2. Next, we generate a CSR (Certificate Signing Request) to get it signed by a CA:

```
# Generates a CSR using the above key
openssl req -x509 -sha256 -new -key client.key -out client.csr
```

3. You will be asked to enter a few fields to create the CSR and DN (Domain Name). The key part is the Common Name entry.

- Country Name (2 letter code) [AU]:
- State or Province Name (full name) [Some-State]:
- Locality Name (eg, city) []:
- Organization Name (eg, company) [Internet Widgits Pty Ltd]:
- Organizational Unit Name (eg, section) []:
- Common Name (e.g. server FQDN or YOUR name) []:
- Email Address []:

4. Once the CSR is generated, it should be signed by a certificate authority. For testing, you can self-sign it as follows. The client.crt file will be used later when adding an external client in SMGR.

```
# Self-sign the CSR
openssl x509 -sha256 -days 3652 -in client.csr -signkey client.key -out client.crt
```

5. Next, we create a PKCS#12 store out of the generated key and certificate:

```
# Takes the cert and the private key to create a PKCS#12 keystore
openssl pkcs12 -export -name clientcert -in client.crt -inkey client.key -out keystore.p12
```

6. This step converts the PKCS#12 store to the format expected Java KeyStore:

```
# Converts a PKCS#12 store to JKS
keytool -importkeystore -destkeystore clientkeystore.jks -srckeystore keystore.p12 -srcstoretype pkcs12 -alias clientcert
```

7. Download the SMGR CA Cert which is Active and is the one which is trusted by the Breeze node onto the local machine where these steps are being carried out and copy to the current folder:

The screenshot shows the Avaya Aura System Manager 7.0 interface. The top navigation bar includes 'Home', 'Avaya Breeze™', 'Inventory', and 'Security'. The main content area is titled 'CA Structure & CRLs'. On the left, there is a sidebar menu with 'CA Functions' (CA Activation, CA Structure & CRLs, Certificate Profiles, Certification Authorities, Crypto Tokens, Publishers) and 'RA Functions' (Add End Entity, End Entity Profiles, Search End Entities). The main content displays 'Basic Functions for CA : tmdefaultca' with links for 'View Certificate' and 'View Information'. Below this, it shows 'Root CA : CN=System Manager CA,OU=MGMT,O=AVAYA' and three download links: 'Download binary/to IE', 'Download to Firefox', and 'Download PEM file' (highlighted in yellow), and 'Download JKS file'. At the bottom, it lists 'Latest CRL: Created 2016-02-08 14:28:05+05:30, Expired 2016-02-09 14:28:05+05:30, number 1 Get CRL' and 'Latest Delta CRL: Created 2016-02-08 14:28:05+05:30, Expired 2016-02-09 00:28:05+05:30, number 2 Get Delta CRL'.

8. Next, import this certificate into a Java trust store:

```
# Import SMGR CA cert onto a trust store
keytool -import -noprompt -alias smgrca -keystore clienttruststore.jks -file SystemManagerCA.cacert.pem -
storepass <password>
```

9. The following files are of interest in the following steps:

- **client.crt** - Certificate to be used while provisioning the external client in SMGR
- **clientkeystore.jks** - The client keystore
- **clienttruststore.jks** - The client truststore

10. Go to Breeze > Configuration > Authorization > Client tab > Select New:

The screenshot shows the 'New External Authorization Client' form in the Avaya Breeze interface. The breadcrumb path is 'Home / Elements / Avaya Breeze™ / Configuration / Authorization'. The form has a left sidebar with navigation options like 'Server Administration', 'Cluster Administration', and 'Configuration'. The main content area has a title 'New External Authorization Client' and a description: 'This page allows you to create a new External Authorization Client'. There are two 'Commit' and 'Cancel' buttons. The form contains two fields: '* Name' with the value 'TestExternalClient' and '* Certificate' with a 'Browse...' button and the text 'No file selected.'.

Enter a name for the client, click on Browse and select the **client.crt** file that was generated previously in Step 4. Click on Commit to save.

11. This generates a client identifier for the application as shown below:

The screenshot shows the 'Authorization Configuration' page in the Avaya Breeze interface. The breadcrumb path is 'Home / Elements / Avaya Breeze™ / Configuration / Authorization'. The page has a left sidebar with navigation options. The main content area has a title 'Authorization Configuration' and a description: 'This page allows you to administer authorization clients, resources and Authorization service instances.' There are three tabs: 'Clients', 'Resource Servers', and 'Service Instances'. The 'Clients' tab is active, showing a table of 'Authorization Clients'. The table has columns for 'Name', 'Id', 'Cluster Name', and 'Type'. There are three rows of data, with the second row highlighted in yellow.

| Name | Id | Cluster Name | Type |
|----------------------|--------------------------------------|--------------|----------|
| SampleExternalClient | 8b6ebb43-ae9d-4788-b0eb-695cfc487f9 | | EXTERNAL |
| TestExternalClient | e104ed7e-918e-48b3-b3b2-ad347f6d2070 | | EXTERNAL |
| authorizationclient | df03af82-c11a-4554-9596-55b372cb769b | GajaCluster | BREEZE |

12. If your lab does not use the default certificate authority on System Manager:

- Navigate to Services / Inventory / Manage elements in System Manager.
- Highlight the node hosting the Authorization service.
- Select More actions Manage Identity Certs.
- Highlight Authorization service name and click the Export button.

The screenshot shows the Avaya Aura System Manager 8.0 interface. The top navigation bar includes 'Users', 'Elements', 'Services', 'Widgets', and 'Shortcuts'. The left sidebar shows the 'Inventory' menu with 'Manage Elements' selected. The main content area is titled 'Manage Identity Certificates' and contains a table with the following data:

| Выбрать | Expand List | Service Name | Common Name | Valid To | Expired | Service Description |
|----------------------------------|-------------|---------------|---------------|------------------------------|---------|----------------------------|
| <input checked="" type="radio"/> | | Authorization | Authorization | Thu Dec 10 16:58:46 GMT 2020 | No | Authorization Service |
| <input type="radio"/> | | spiritalias | spiritalias | Thu Dec 10 16:58:40 GMT 2020 | No | SPIRIT Service |
| <input type="radio"/> | | .. | .. | Thu Dec 10 16:58:43 GMT | .. | Internal TLS communication |

Import the received certificate into the trust store as follows:

```
# Import UAC CA cert into a trust store
keytool -import -noprompt -alias uacca -keystore clienttruststore.jks -file UACCluster.pem -storepass <password>
```

13. The configuration for the CustomerFilterMessages service is stored inside the Configuration folder inside the Tomcat temp directory. This folder is automatically created by the sample service upon startup.

- On Windows, this will be "%TOMCAT_HOME%/temp/CustomerFilterMessages/config/", where %TOMCAT_HOME% is the Tomcat root directory.
- On Linux servers, this will be "\$TOMCAT_HOME/temp/CustomerFilterMessages/config/" .

14. Copy "clientkeystore.jks" and "clienttruststore.jks" into the certs subfolder inside the configuration folder.

15. Fill in the "oceana.properties" file located inside the configuration folder.

oceana.properties

This properties file contains the settings required for getting the token access and additional settings described below.

```

# CustomerFilterMessages properties

# ===== Authorization properties =====

# AuthorizationService cluster host
authorizationHost=

# OCP Cluster host and security
ocpHost=
ocpSecure=true
ocpAttachmentPath=/services/AgentControllerService/attachment/

# The ID assigned to the client by SMGR.
clientId=

# SSL properties
keyStore=certs/clientkeystore.jks
keyStorePassword=your_keystore_password
keyAlias=clientcert
trustStore=certs/clienttruststore.jks
trustStorePassword=your_truststore_password
username=
password=
# Port which uses for authorization. Default value is 9443
authPort=9443

# The access token. Set automatically.
ssoToken=

# ===== Configuration properties =====

saveChatTranscripts=true
saveMessageTranscripts=true
saveEmailTranscripts=true
downloadAttachments=true

```

Set the following properties in this file:

- **authorizationHost** - FQDN or IP address of the AuthorizationService cluster host. E.g.: 10.10.10.40
- **ocpHost** - FQDN or IP address of the OCP cluster, e.g.: 10.10.10.41
- **ocpSecure** - if a secure connection should be used to contact OCP
- **ocpAttachmentPath** - the path for the OCP attachments REST API - should not be changed
- **clientId** - client identifier generated in step 11.
- **keyStorePassword** - password from step 6.
- **trustStorePassword** - password from step 8.

It has additional settings to enable / disable transcript saving and attachments downloading:

- **saveChatTranscripts** - true/false
- **saveMessageTranscripts** - true/false
- **saveEmailTranscripts** - true/false
- **downloadEmailAttachments** - true/false

Adding transcript service to chat

To add your filtering mechanism to Oceana Web Chat:

1. Open the Omni-Channel Admin.
2. Click on the "Transcript" entry in the bottom-left menu.
3. In the Transcript Filtering Web Service field, type the full URL of the filtering service, e.g. [http://127.0.0.1/CustomerFilterMessages-{\\$version}/rest/filter](http://127.0.0.1/CustomerFilterMessages-{$version}/rest/filter). If your service listens on a port other than 80 or 443, make sure to include this.
4. In the Sending message transcripts field, select the appropriate option:
 - "Do not send"
 - "Current transcript" - to send chat transcripts. (Transcripts will be sent to [http://127.0.0.1/CustomerFilterMessages-{\\$version}/rest/filter/transcript](http://127.0.0.1/CustomerFilterMessages-{$version}/rest/filter/transcript))
 - "V1 Enhanced transcript" - to send v1 enhanced transcripts. (Transcripts will be sent to [http://127.0.0.1/CustomerFilterMessages-{\\$version}/rest/filter/enhancedtranscript](http://127.0.0.1/CustomerFilterMessages-{$version}/rest/filter/enhancedtranscript))
 - "V2 Enhanced transcript" - to send v2 enhanced transcripts. (Transcripts will be sent to [http://127.0.0.1/CustomerFilterMessages-{\\$version}/rest/filter/v2/enhancedtranscript](http://127.0.0.1/CustomerFilterMessages-{$version}/rest/filter/v2/enhancedtranscript))
5. Optionally, set the start and end time for the retry mechanism to a quiet period.
6. Click "Save".

The CustomerControllerService will read this from the OmniChannel Database upon termination of a chat. If you have active chats in progress while this is being set up, any change to the URL will be read as soon as the chat finishes.

Adding transcript service to Email

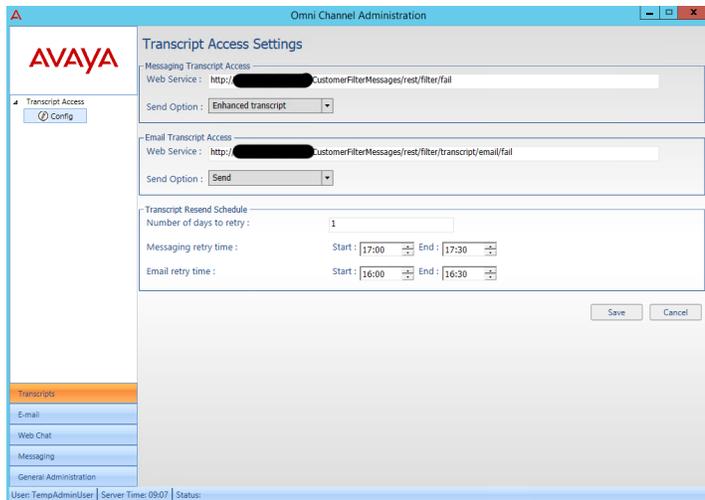
To add your filtering mechanism to Oceana Email, follow this procedure:

1. Open the Omni-Channel Admin.
2. Click on the "Transcripts" entry in the bottom-left menu.
3. Enter the full URL of your transcript service into the box labelled "Transcript Filtering Web Service", e.g. [http://127.0.0.1/CustomerFilterMessages-{\\$version}/rest/filter/transcript/email](http://127.0.0.1/CustomerFilterMessages-{$version}/rest/filter/transcript/email). If your service listens on a port other than 80 or 443, make sure to include this.
4. Set the "Send Option" field accordingly. The available options are "Send", "Send with mailbox", and "Do not send".
5. Optionally, set the start and end time for the retry mechanism to a quiet period.
6. Click "Save".

The EmailService will read this from the OmniChannel Database upon reading an email from the email server.

Layout of the configuration panel

The following image shows the layout of the transcript panel. It allows the administrator to configure the URLs, sending mechanism and times at which to reschedule a retry.



Scheduled retry mechanism

Upon failure to send the transcript, the CustomerControllerService and EmailService will make 4 more attempts to resend the transcript to account for temporary network conditions. If these attempts fail (due to e.g. the service not being deployed or configured correctly), the transcript will be marked as failed. In these cases, the service will attempt to resend at regular intervals. The flow is as follows:

1. Read the scheduled retry time from the database.
2. When the time on the Omnichannel node(s) matches the scheduled retry time, they will attempt to resend any transcripts that are failed.
3. If this attempt fails 5 times (to account for network conditions), an internal counter is incremented for the number of failed attempts.
4. If that counter reaches thirty (i.e. the transcript repeatedly failed to send over a 30-day period), it will be marked as permanently failed.

Testing the transcript service

The quickest way to check if this works is to access the following URL in a browser, replacing "127.0.0.1" with the IP address or FQDN of the server: [http://127.0.0.1/CustomFilterMessages-{\\$version}/rest/filter/echo/Hello](http://127.0.0.1/CustomFilterMessages-{$version}/rest/filter/echo/Hello). This will result in the browser displaying the word "Hello".

V1 Enhanced transcripts

To test the filtering mechanism with enhanced transcript messages, send the following as a JSON object to [http://127.0.0.1/CustomFilterMessages-{\\$version}/rest/filter/enhancedtranscript](http://127.0.0.1/CustomFilterMessages-{$version}/rest/filter/enhancedtranscript):

```
[
  {
    "sender" : "A: Mr Smith",
    "contactId" : "1",
    "customerId" : "1",
    "workRequestId" : "labdaiuhui2",
    "message" : "Hello, world!",
    "date" : 0,
    "messageType" : "NORMAL",
    "channelType" : "CHAT"
  },
  {
    "sender" : "C: John Doe",
    "contactId" : "1",
    "customerId" : "1",
    "workRequestId" : "labdaiuhui2",
    "message" : "This is a test",
    "date" : 0,
    "messageType" : "NORMAL",
    "channelType" : "CHAT"
  }
]
```

If you are using the out-of-the-box version, the following should be returned upon a successful request:

```
[{
  "sender": "A: Mr Smith",
  "contactId" : "1",
  "customerId" : "1",
  "workRequestId" : "labdaiuhui2",
  "message": "Hello, world! been filtered",
  "date": 0,
  "messageType": "NORMAL",
  "channelType" : "CHAT"
}, {
  "sender": "C: John Doe",
  "contactId" : "1",
  "customerId" : "1",
  "workRequestId" : "labdaiuhui2",
  "message": "This is a test been filtered",
  "date": 0,
  "messageType": "NORMAL",
  "channelType" : "CHAT"
}]
```

If message transcripts saving is enabled (see `oceana.properties` above), the filtered transcript will be saved to the following path: `%TOMCAT_HOME%/temp/CustomFilterMessages/messaging/labdaiuhui2.json`

V2 Enhanced transcripts

To test the filtering mechanism with enhanced transcript messages, send the following as a JSON object to [http://127.0.0.1/CustomFilterMessages-{\\$version}/rest/filter/v2/enhancedtranscript](http://127.0.0.1/CustomFilterMessages-{$version}/rest/filter/v2/enhancedtranscript):

```

{
  "transcriptContext": {
    "contactId": "1",
    "customerId": "1",
    "workRequestId": "luuid",
    "conversationId": "123",
    "channelSrc": "ios",
    "endUserId": "111",
    "tenantId": "avaya"
  },
  "transcriptMessages": [
    {
      "sender": "agent",
      "message": "fallback",
      "date": 0,
      "messageType": "ASYNCMESSAGE",
      "channelType": "MESSAGING",
      "messageId": "mid",
      "customData": "{ \"exampleKey1\": \"exampleValue1\", \"exampleKey2\": \"exampleValue2\", \"
asyncMessageInfo\": {\"message\": {\"text\": \"Hi! Do you have time to chat?\", \"received\": 1575303766664, \"
channelSrc\": \"ios\"}}}",
      "richMediaType": "text"
    }
  ]
}

```

If you are using the out-of-the-box version, the following should be returned upon a successful request:

```

{
  "transcriptContext": {
    "contactId": 1,
    "customerId": 1,
    "workRequestId": "luuid",
    "conversationId": "123",
    "channelSrc": "ios",
    "endUserId": "111",
    "tenantId": "avaya"
  },
  "transcriptMessages": [
    {
      "sender": "agent",
      "message": "fallback been filtered",
      "date": 0,
      "messageType": "ASYNCMESSAGE",
      "channelType": "MESSAGING",
      "messageId": "mid",
      "customData": "{ \"exampleKey1\": \"exampleValue1\", \"exampleKey2\": \"exampleValue2\", \"
asyncMessageInfo\": {\"message\": {\"text\": \"Hi! Do you have time to chat?\", \"received\": 1575303766664, \"
channelSrc\": \"ios\"}}}",
      "richMediaType": "text"
    }
  ]
}

```

If message transcripts saving is enabled (see `oceana.properties` above), the filtered transcript will be saved to the following path: `%TOMCAT_HOME%/temp/CustomerFilterMessages/messaging/luuid.json`

Email Transcripts

To test the transcript service with email transcript messages, send the following JSON object to [http://127.0.0.1/CustomerFilterMessages-{\\$version}/rest/filter/transcript/email](http://127.0.0.1/CustomerFilterMessages-{$version}/rest/filter/transcript/email). The service should respond with "200 OK".

```
{
  "direction" : "Incoming",
  "customerId" : 93530,
  "contactId" : 117221,
  "workRequestId" : "dsfQz12v2pdQeed66zQkh4viw",
  "mailFrom" : "testfrom@site.com",
  "mailTo" : [ "test1@sample.com", "test2@sample.com" ],
  "mailCC" : [ "abc@sample.com", "xyz@sample.com" ],
  "mailBCC" : [ "maill@test.com", "mail2@test.com" ],
  "subject" : "TMA Test Sales",
  "timestamp" : 1543998137967,
  "attachments" : [ "http://10.10.10.159/services/AgentControllerService/attachment/CFD3EE17-E4E3-44B2-A54A-6D1954DF18C4", "http://10.10.10.159/services/AgentControllerService/attachment/10EA7E0E-6FEB-4207-AB0B-14B286BEBDDD" ],
  "bodyEmail" : "<div><br></div><div>bbbbbb &lt;b>&gt;Re: Sales 02&lt;/b></div><div>Regards,<br></div><div>John<br>"
}
```

If you are using the out-of-the-box version, this should simply return HTTP 200 OK.

If the service is configured to save email transcripts to disk, (see `oceana.properties` above), the email transcript will be saved to the following path: `%TOMCAT_HOME%/temp/CustomFilterMessages/email/dsfQz12v2pdQeed66zQkh4viw.json`. If the transcript contains links to attachments, and access to attachments is properly configured (see the "Access to attachments" section), the attachments will be saved to the following path: `%TOMCAT_HOME%/temp/CustomFilterMessages/email/dsfQz12v2pdQeed66zQkh4viw_attachments/*`

Troubleshooting

The first step is to check the logs for the service (e.g. the CustomerControllerService)

Log files

Log files can be found under the following directories:

- CustomerController: `/var/log/Avaya/dcm/pu/CustomerControllerService`
- EmailService: `/var/log/Avaya/dcm/pu/EmailService`
- CustomerFilterMessages: search the Tomcat log directory under `%TOMCAT_HOME%/logs`

ELK Logging

There are 3 types of log errors that will be reported to the JSON logger for cases when transcript sends to fail to the external filtering service. Each of which will highlight different issues to the observer:

1. An attempt was made to send it to an empty filtering service URL
 - a. This will highlight a possible misconfiguration of the service URL in the Omnistore admin.
2. An attempt was made to send a transcript to the filtering service which exceeded the maximum number of retry attempts.
 - a. This will highlight cases where the CustomerControllerService could not communicate with the filtering service.
3. An unexpected exception was encountered that prevented the transcript from being sent to the filtering service.
 - a. This will highlight any cases where a transcript could not be sent due to an unexpected exception, e.g. can't communicate with Omnistore.

Each message type contains the following pieces of common information:

- The effected Work Request ID.
- Log level ERROR.
- The keywords "Cannot filter transcript".
- A message explaining the issue.

Log for Empty Service URL

| Time | _source |
|------------------------------|--|
| March 7th 2019, 13:24:23.946 | <pre>level: ERROR message: EMPTY_URL: Cannot filter transcript, empty filtering service url, check Omnistore admin configuration id: nG0ts8L1TAS1Hg1L4Tj5LQ offset: 3,592 @timestamp: March 7th 2019, 13:24:23.946 @version: 1 beat.version: 6.2.2 beat.hostname: dmz-breeze-24 beat.name: dmz-breeze-24 logger: com.avaya.ocp.customer.transcriptfilter.TranscriptFiltering prospector.type: log host: dmz-breeze-24 source: /var/log/Avaya/dcm/pu/CustomerControllerService/CustomerControllerWeb-3.6.0.0.1 1000.ison.log thread: ChatLib-1-Listener-? tags: beats input codec plain applied pu: CustomerControllerWeb-3.6.0.0.11000</pre> |

```
{
  "timestamp": "2019-03-06 12:21:57,531",
  "level": "ERROR",
  "logger": "com.avaya.ocp.customer.transcriptfilter.TranscriptFiltering",
  "thread": "ChatLib-1-Listener-1",
  "pu": "CustomerControllerWeb-3.6.0.0.11000",
  "id": ["1NrIUFIIMRC2HdfMsA3ji7A"],
  "message": "EMPTY_URL: Cannot filter transcript, empty filtering service url, check Omnistore admin configuration"
}
```

Log for when Maximum number of Retry Attempts Reached

```

Time ^      _source
┆ March 6th 2019, 11:56:40.580 level: ERROR message: MAX_RETRIES: Cannot filter transcript, exceeded max retries, filtering service unavailable url=[http://10.134.61.30:8080/CustomFilterMessages/rest/filter/] offset: 679 @timestamp: March 6th 2019, 11:56:40.580 @version: 1
beat.version: 6.2.2 beat.hostname: dmz-breeze-42 beat.name: dmz-breeze-42 logger: com.avaya.ocp.customer.transcriptfilter.TranscriptFiltering prospector.type: log id: od83y7d5Tw0KovbK8eJ3Rg host: dmz-breeze-42 source: /var/log/Avaya/dcm/pu/CustomControllerService/CustomControllerWeb-3.6.0.0.11000.ison.log thread: ChatLib-1-Listener-1 tags: beats innot coder plain applied nu:

```

```

{
  "timestamp": "2019-03-06 13:23:34,599",
  "level": "ERROR",
  "logger": "com.avaya.ocp.customer.transcriptfilter.TranscriptFiltering",
  "thread": "ChatLib-1-Listener-2",
  "pu": "CustomerControllerWeb-3.6.0.0.11000",
  "id": ["gDSIs5F3RW6ciyLjTBcGIg"],
  "message": "MAX_RETRIES: Cannot filter transcript, exceeded max retries, filtering service unavailable url=[http://10.134.61.30:8080/CustomFilterMessages/rest/filter/]"
}

```

Log for Unexpected Exceptions

```

┆ March 6th 2019, 14:14:44.907 level: ERROR message: EXCEPTION: Cannot filter transcript, encountered unexpected exception, type=[java.lang.NullPointerException], message=[null] offset: 2,421 @timestamp: March 6th 2019, 14:14:44.907 @version: 1 beat.version: 6.2.2 beat.hostname: dmz-breeze-24 beat.name: dmz-breeze-24 logger: com.avaya.ocp.customer.transcriptfilter.TranscriptFiltering prospector.type: log id: 50iARuv8TbmsR3sMiX7Vhg host: dmz-breeze-24 source: /var/log/Avaya/dcm/pu/CustomControllerService/CustomControllerWeb-3.6.0.0.11000.ison.log thread: ChatLib-1-Listener-1 tags: beats innot coder plain applied nu: CustomerControllerWeb-3.6.0.0.11000

```

```

{
  "timestamp": "2019-03-06 14:14:44,907",
  "level": "ERROR",
  "logger": "com.avaya.ocp.customer.transcriptfilter.TranscriptFiltering",
  "thread": "ChatLib-1-Listener-1",
  "pu": "CustomerControllerWeb-3.6.0.0.11000",
  "id": ["SOiARuv8TbmsR3sMiX7Vhg"],
  "message": "EXCEPTION: Cannot filter transcript, encountered unexpected exception, type=[java.lang.NullPointerException], message=[null]"
}

```

Common problems

| Symptoms | Cause | Resolution |
|---|---|---|
| Certificate issues | The certificate for Tomcat is not trusted | Import it into System Manager |
| HTTP 404 when trying to filter the transcript | The URL is probably incorrect | Check the URL in the Admin |
| No errors occur, but the transcript is not sent | The SendMessageTranscriptParameter parameter is not set correctly in the database | Check the setting in the admin, or run the following SQL query inside the OmniChannel Database's MULTIMEDIA namespace. The text value can be 0(disabled), 1(legacy filtering), or 2(enhanced transcript): UPDATE cls.siteparameters SET Text = '1' WHERE id = 275; |