Avaya Oceana Email Open Interfaces

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.

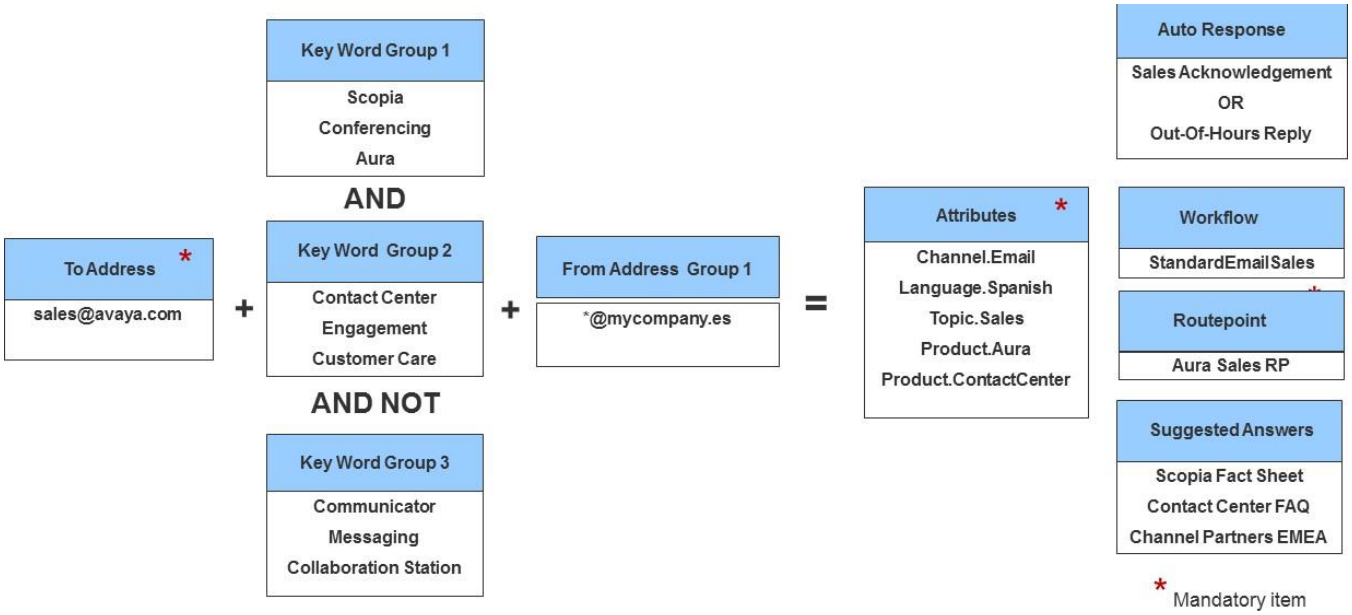
THE H.264 (AVC) CODEC IS LICENSED UNDER THE AVC PATENT PORTFOLIO LICENSE FOR THE PERSONAL USE OF A CONSUMER OR OTHER USES IN WHICH IT DOES NOT RECEIVE REMUNERATION TO: (I) ENCODE VIDEO IN COMPLIANCE WITH THE AVC STANDARD ("AVC VIDEO") AND/OR (II) DECODE AVC VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED 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.

Introduction

Avaya Oceana Email Open Interfaces is a capability to allow pre-processing of an incoming email to Oceana to enhance its routing or filter its content. This pre-processing step is bespoke to the Oceana deployment and will require custom development. The feature allows Oceana to use this custom code in the manipulation of email content or influence the routing decisions of email.

Email Rules

When an email is retrieved by Oceana from an Email server, it is analyzed by the Oceana email rules and based on the content of the email, a set of routing attributes are applied and a workflow is executed.



Input Criteria to a Rule

Based on the above example, emails To sales@avaya.com are analyzed. Consider the rule to determine True/False output to each part of the rule:

- To Address: If the email being analysed was sent TO sales@avaya.com, this part of the rule analysis will be true. This could be a physical mailbox or an alias.
- Keyword Matching: This part of the rule will return True if it contains any keyword from Keyword Group 1 AND any word from Keyword Group 2 BUT DOES NOT contain any keyword from Keyword Group 3.
    - The use of keyword searching is optional and you can configure if just the subject or the entire body of the email is searched. Up to 3 sets of keywords can be combined using AND & AND NOT operators. This mechanism can be used to filter out of office messages or other emails sent by automated attendants.
- From Address Group: This part of the rule will return True if the email being analyzed was sent from any address in the email address group defined.
    - This is an optional parameter to a rule. It is useful however to prioritize VIP customers/organizations or block emails based on senders or domains.

Assuming that **all** parts of the email rule (To Address, Keyword Matching (if used) and From Address Group (if used)) return True, then the output of the rule is applied to the email. If **any** part returns False, then the next rule is checked. If no rule returns True to **all** parts, the default rule is applied to provide basic default routing and treatment.

Rule Output

The output of a rule will provide the routing parameters to determine how that email should be treated. It is only applied if all parts of the input criteria to the rule holds true for the email being analyzed.

- **Attributes**: These are the routing attributes Oceana will use to attempt to route the email. You must apply attributes to the email.
- **Auto Response**: This is an automated message sent to the sender. It could be an acknowledgement of receipt of the email (and the email is subsequently routed to an agent) or an auto-reply attempting to answer the customer query (and the email is not routed to an agent). It is possible to send an alternate auto-response to the sender based on time of day.
  Auto-responses are optional.
- **Workflow**: This is the Engagement Designer workflow that will be executed to handle this email.
  Setting a workflow is mandatory.
- **Routepoint**: This is the route-point captured in Oceanalytics for reporting on this email. Setting a route-point is mandatory.
- **Suggested Answers**: These are supervisor prepared suggested email answers that are offered to the agent when answering the email. Setting suggested answers is optional.

Email Open Interfaces

An additional rule output to the above is a custom web service which can be called to influence the routing or treatment of the email. For emails matching the input criteria to a rule, a custom web service can be executed to perform custom tasks, examples of which are shown below.

Typical Uses

A custom web service may be called to perform the following:

- Filter the email content before being routed to an agent. In this case, we may want to pass the subject and body of an email to a custom web service that might remove/obfuscate credit card details, social security numbers or other sensitive information from the email. The web service will take the subject and body as input parameters, perform its filtering and return them as output parameters. Oceana will then use the filtered subject and body, discarding the originals.
- Enhance the routing decisions. In this case, a custom service might be passed the senders email address and/or the subject and body. Based on a lookup of the sender or language detection of the body, additional routing attributes could be applied to the email being processed.
- Selectively not route emails based on sender or content. In this case, a custom service might be passed the senders email address and/or body and subject. You may look up this sender in an external system and determine not to route an email from them. Alternatively, you may find abusive text in the subject or body and determine not to route the email. The service would return an output parameter which is mapped to the Auto-Close field. This decides if the workflow is executed or not. Although not routed, auto-closed contacts will be persisted in the database including transcript, and visible in customer history search.
- Find bespoke Ticket IDs in emails and handle accordingly. In this case, the custom service would take the subject and body and input parameters. The pattern of the Ticket ID could be retrieved and used to take a custom action e.g. update a CRM system. The email could then (optionally) be routed by Oceana.

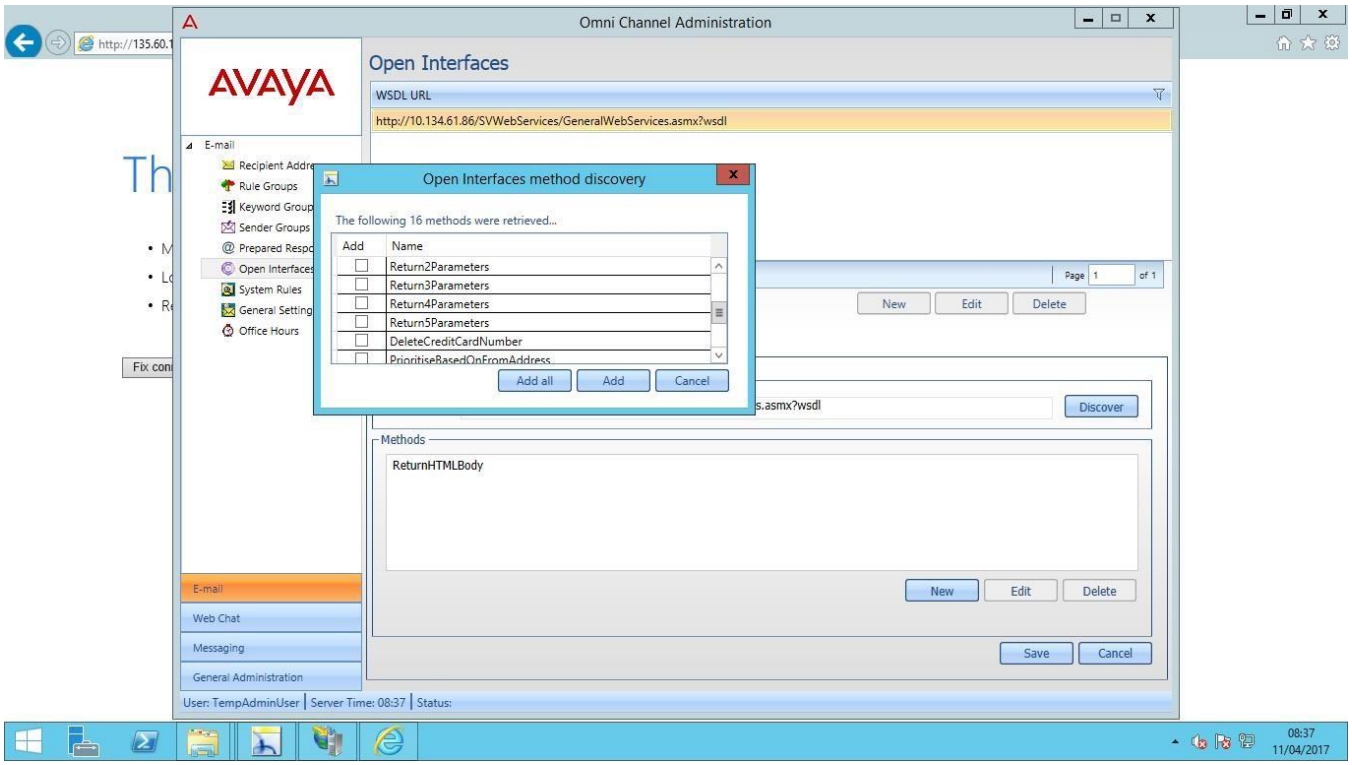
Limitations

- Only SOAP interfaces are supported. RESTful services cannot be called. The service must be developed to WSDL standards.
- The custom service must not be hosted on any Oceana cluster / node. If developing the service as a snap-in, you should provision a cluster outside of Oceana to host this.
- The feature in MM Admin to auto-detect the web service operations is not compatible with all WSDL formats. In some cases, the web service details may need to be entered manually.
- HTTPS redirects: Secure connections to the web service over HTTPS are supported. However if your web service makes the WSDL available over non-secure HTTP but redirects to HTTPS for performing operations, then you must enter a ‘HTTPS’ URL for the WSDL in the MM Admin, even if your WSDL is also available over HTTP. The default ‘http://’ prefix is auto-filled in the URL field by default - be sure to change this to ‘https://’ if your web service operations are secured via SSL. You will find more notes on SSL support including certs later in this document.
- If changes are made to your settings in the Admin, e.g. updated URL, input/output parameter(s) then a reboot is required for the changes to take effect.

Administration

The administration of this feature will be done via the Avaya Control Manager tool. Launch the MM Admin from where the user will be able to enter the WSDL URL and method name, provide other properties if necessary and specify what the input and output parameters should correspond to. When configuring a rule the user will be able to choose which web service should be associated with the rule.

Firstly, add the URL to the custom web service WSDL and click Discover. The administration client will then attempt to parse the SOAP envelope and list the methods retrieved. Select the method to add.

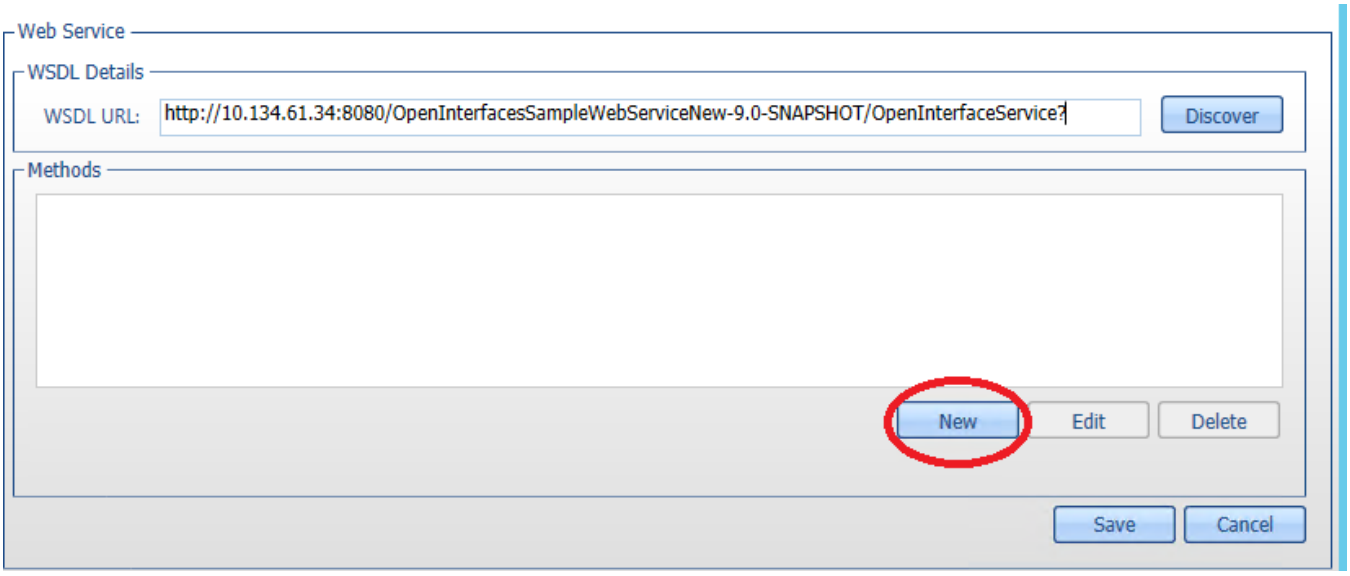


Step 1: Add WSDL details and discover the methods of the custom service.

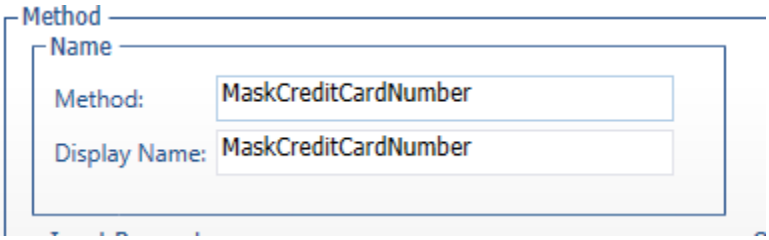
Note: Failure of the discovery feature in the Admin does not in itself indicate that the WSDL is not compatible with Oceana. This is a convenience feature *NOT* a compatibility test.

The Admin 'Discover' feature is limited to a specific WSDL format in order to auto-detect the web service operations successfully. This feature only works on single flat file WSDL format and will NOT work if your WSDL contains imports. In this case, you may simply enter the details manually as described below.

Alternatively, to enter the name(s) of your web service method(s) manually, click 'New' in the 'Methods' pane as shown below.



Y



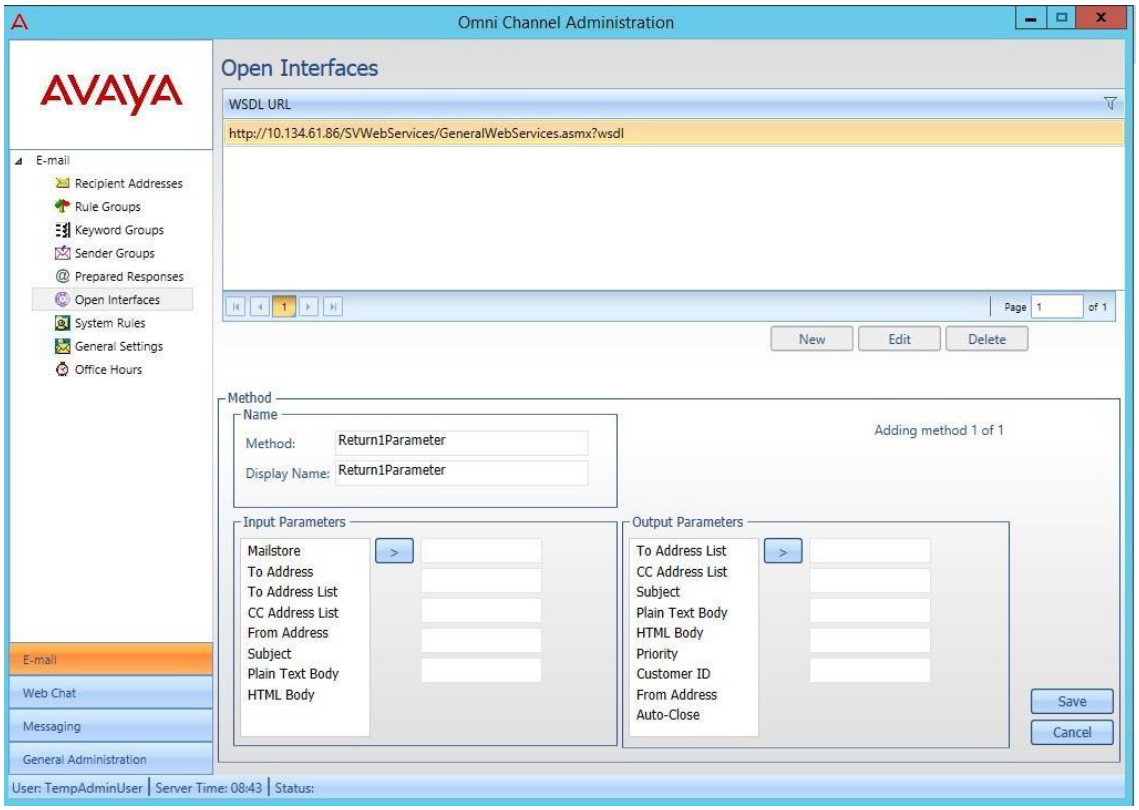
The method name should match the name of the publicly exposed web method name as implemented in the code e.g. MaskCreditCardNumber. This is usually the same as the display name but may not always be the case.

```
@WebMethod(action = "MaskCreditCardNumber", operationName = "MaskCreditCardNumber")
String[] MaskCreditCardNumber(String[] inputParameters);
```

The display name field should match the name of the operation as it appears in the WSDL. In a Java web service this is typically defined in the 'operationName' attribute of the @WebMethod annotation as shown in the code snippet above. In the WSDL, you will find the method display name as an attribute in the 'operation' element tags as shown in the sample WSDL snippet below.

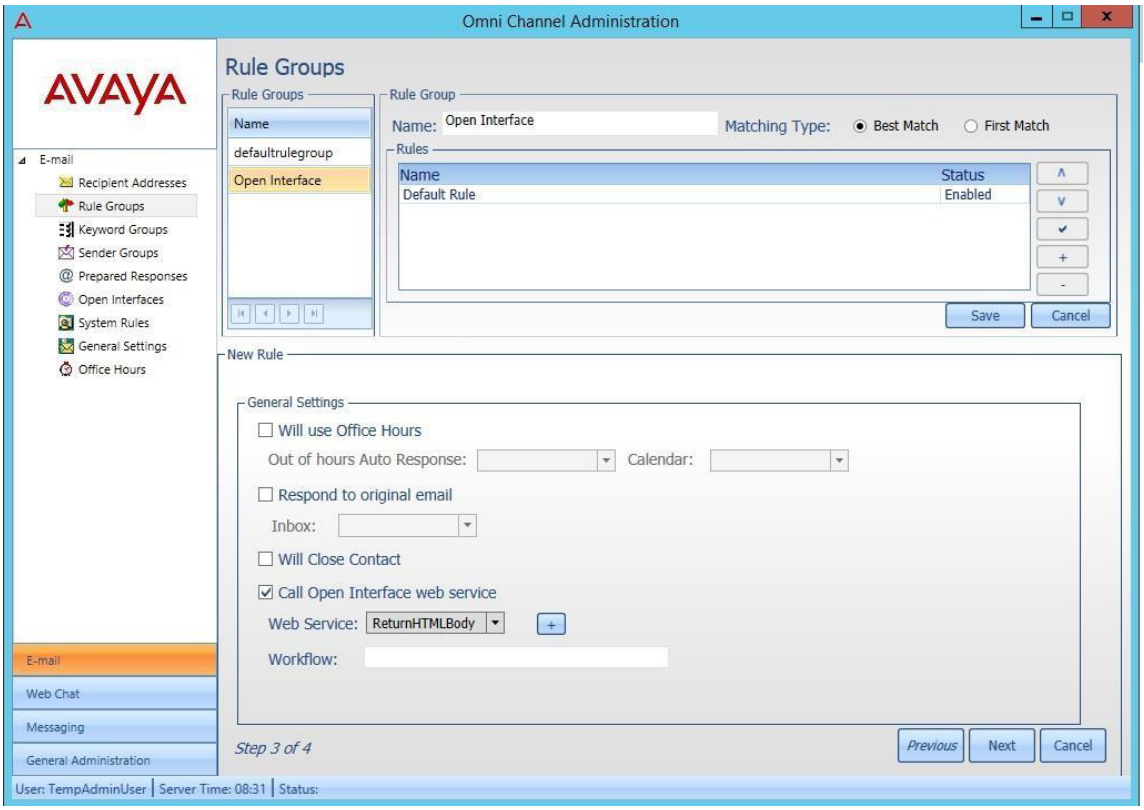
```
<operation name="MaskCreditCardNumber">
  <soap:operation soapAction="MaskCreditCardNumber"/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
```

Next, specify what Oceana fields the input parameters of the web service correspond to. For example, the first and second input parameters of the custom service might correspond to the subject and body of the email. Also, specify what Oceana fields the output parameters of the custom web service correspond to. For example, 3 fields might be returned: a filtered subject, a filtered body and an additional set of attributes.



Step 2: Set what the expected input and output parameters map to in Oceana

Lastly, when configuring the email rules, define what custom web service (if any) should be called when an email matches all the input criteria of that rule.



Step 3: Set the web service to be called if the email matches all the input criteria of that rule

Input and output parameters

When the configuring the web service the user will need to specify what the input and output parameters should map to. For example, a web service may take an input parameter of the body of the email and return back a modified body as well as a priority. These output parameters will then be used for the email instead of the original values. Note with Oceana 3.8.1 Release it will be possible to send 7 input parameters (instead of the previous limit of 5).

Input parameters

- Mailstore
- To address
- To address list
- CC address list
- From address
- Subject
- Body
- Message-ID (added in 3.8.1.0 Release)

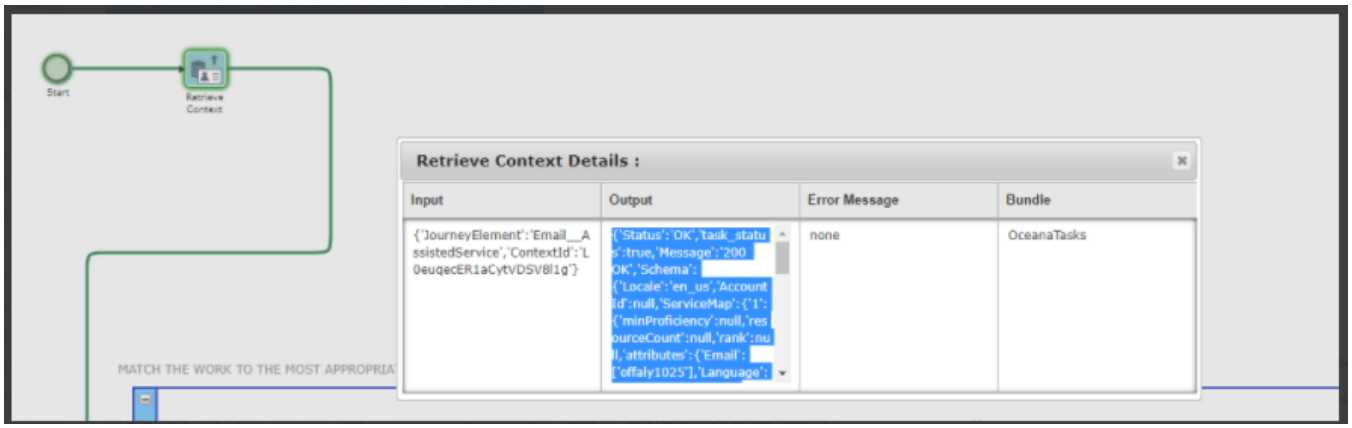
Please note Message-ID will also be available in the CS-Record for each contact.

Output parameters

- To address
- To address list
- CC address list
- Subject
- Body
- Priority
- Auto-close
- Attributes

Message-ID usage (Introduced with 3.8.1.0 release)

To allow matching of EmailOpenInterfaces record with CS records utilized by ED Flows, Message-ID is added to the list of available input parameters and it will also be available as External ID in the ContextStore record for this contact. This will allow customers with bespoke ED Flows to treat individual emails as desired.




```
{
        'Status':'OK',
        'task_status':true,
        'Message':'200 OK',
        'Schema':{
                'Locale':'en_us',
                'AccountId':null,
                'ServiceMap':{
                        '1':{
                                'minProficiency':null,
                                'resourceCount':null,
                                'rank':null,
                                'attributes':{
                                        'Email':['offaly1025'],
                                        'Language':['English'],
                                        'Channel':['Email'],
                                        'Service':['CorporateAccounts'],
                                        'Location':['Inhouse']
                                },
                                'priority':1,
                                'maxProficiency':null
                        }
                },
                'TransferServiceMap':{},
                'MatchingService':null,
                'Rating':null,
                'TransferResourceMap':{},
                'TranscriptRef':'jPjxEPhFTlqp0tzcAYg7hg',
                'CustomerId':'149',
                'ResourceMap':{},
                'DataCenter':'Primary',
                'Deferral':null,
                'CollectedDigits':null,
                'ApproverId':null,
                'Disposition':null,
                'Strategy':'Most Idle',
                'AgentId':null,
                'ApprovalMap':{}
        },
        'ExternalId':'<9b4f5a87-b6af-39fe-1d28-1f715ebbcce1@wf.aura.com>',
        'Data':{},
        'Topic':'EnglishCorporateAccounts'
}
```


Input and output parameters - dealing with Nulls

No nulls should be sent by the client. If a null value is passed from the Email Service the web service client will send a blank string.

If the webservice returns a null or an empty string as one of it's return parameters then the field that it has been matched to will not be changed.

Failure scenarios

There are a number of potential scenarios when the web service specified may not return the expected response. These include:

- Web service not available. For example, the web service may not be running, an incorrect WSDL may have been specified or there may be some network issue.
- Specified method not found. In this scenario a valid WSDL URL has been specified but the WSDL does not contain the method that have been defined.
- The method specified may not have the correct signature, i.e. it does not take an array of strings as input and return an array of strings.

Should any of these scenarios occur then the output of the web service call will be ignored and the email interaction processed without the web service customization. Please refer to the Troubleshooting section at the end of this document to help diagnose and prevent failure scenarios.

Adding additional routing attributes

Any additional attributes returned by a custom web service must:

- Be returned in comma separated string format e.g. “Language.Spanish,Location.Basque”
- The additional attributes must be pre-defined in ACM and returned from the custom service exactly as defined. Otherwise, it will be ignored.
- Be mapped to the Attribute List output parameter in Oceana.

WSDL format

The Web Service Description Language (WSDL) is the XML based document that describes a Web Service. It specifies the location of the service and the operation or methods the service exposes.

The WSDL must be of a standard format. The Operation(s) to be called should take an array of strings as its input parameter. It should return an array of strings.

The recommended format is a single flat file WSDL with no imports. This format promotes compatibility with the Admin ‘Discover’ feature to detect the WSDL details automatically. Please refer to pages 10 /11 of the Administration section for more details.

WCF (.NET) WSDL format

Default WCF projects generate a WSDL with nested internal files. This format is supported provided the operations are manually specified by the user in the admin (refer to page 11). There are a number of methods to ‘flatten’ a WCF WSDL in a single file, including using a custom WCF service host (http://blogs.msdn.com/b/dotnetinterop/archive/2008/09/23/flatten-your-wsdl-with-this-custom- servicehost-for-wcf.aspx) or using WCFExtras (http://wcfextras.codeplex.com/). It can be done natively in .NET 4.5.

Java Web Service WSDL format

Most Java based web service projects generate a WSDL at runtime with import tags by default. This format is supported by Oceana, provided the web service operations are manually entered by the user in the Admin (refer to page 11). In the case of Java web services, support for ‘flattening’ the WSDL into a single file format varies depending on your implementation, libraries used etc. Although not as straight- forward as with .NET projects, there are some options to achieve this format if desired:

Wsgen: The JAX-WS wsgen tool allows for the generated schema to be embedded inside the WSDL file by using the -inlineSchemas command line switch. Using Maven you can configure the JAX-WS Maven plugin to do the same in your POM.xml. As part of your wsgen Maven goal configuration, set the optional ‘inlineSchemas’ element to ‘true’ in your configuration.

```
<plugin>
  <groupId>org.jvnet.jax-ws-commons</groupId>
  <artifactId>jaxws-maven-plugin</artifactId>
  <version>2.2</version>
  <executions>
    <execution>
      <id>SomeId</id>
      <goals>
        <goal>wsgen</goal>
      </goals>
      <phase>prepare-package</phase>
      <configuration>
        <sei>some.class.Name</sei>
        <genWsdl>true</genWsdl>
        <keep>true</keep>
        <resourceDestDir>some/target/dir</resourceDestDir>
        <inlineSchemas>true</inlineSchemas>
      </configuration>
    </execution>
  </executions>
</plugin>
```

More information on wsgen: https://docs.oracle.com/javase/7/docs/technotes/tools/share/wsgen.html

Manual consolidation:

An alternative workaround to achieve a single unified WSDL file can be achieved as follows:

1. Publish your endpoint, then save the generated WSDL and the XSD files for editing manually.
2. Manually copy & paste the content of the XSD file(s) into the 'types' section of your WSDL, replacing the schema 'import' reference with the actual xsd content.
3. Save the merged WSDL file in a resource folder where your application can access it.
4. To configure your application to load the merged WSDL instead, specify the path in the 'wsdllocation' attribute inside the JAX-WS @Webservice annotation.

In the interface you specify the wsdlLocation:

```
@WebService(name = "Echo", targetNamespace = "http://echo/", wsdlLocation="META-INF/wsdl/EchoService.wsdl")
```

Using the ‘wsdlLocation’ attribute of the javax.jws.WebService (@Webservice) annotation, developers can manually specify a location of a relative or absolute URL of their own pre-defined WSDL file. This can be used in place of an auto-generated WSDL, thereby allowing for manual control / editing of the WSDL such as consolidating imports into one single file.

Oceana Email Service supports a broad range of WSDL formats including those with external schema imports, typical of Java web services. However, in the unlikely event that you encounter compatibility issues, simplifying the WSDL format may help to overcome this. The WSDL unification steps outlined above may be helpful if you encounter any difficulties using your WSDL with Oceana.

Sample Web Service

The sample Java project can be used as a reference for building Open Interfaces web services. This section provides an overview of the sample project and a step by step guide to setting it up.

Assumptions

- You are familiar with a modern Object Oriented Programming language and have some knowledge of Java.
- You are familiar with the following Web Technologies:
  - Apache Tomcat (or equivalent)
  - Java EE
  - SOAP web services
- You are using a Windows environment.

Software Requirements

- Java JDK
- Eclipse IDE
- Maven
- Apache Tomcat

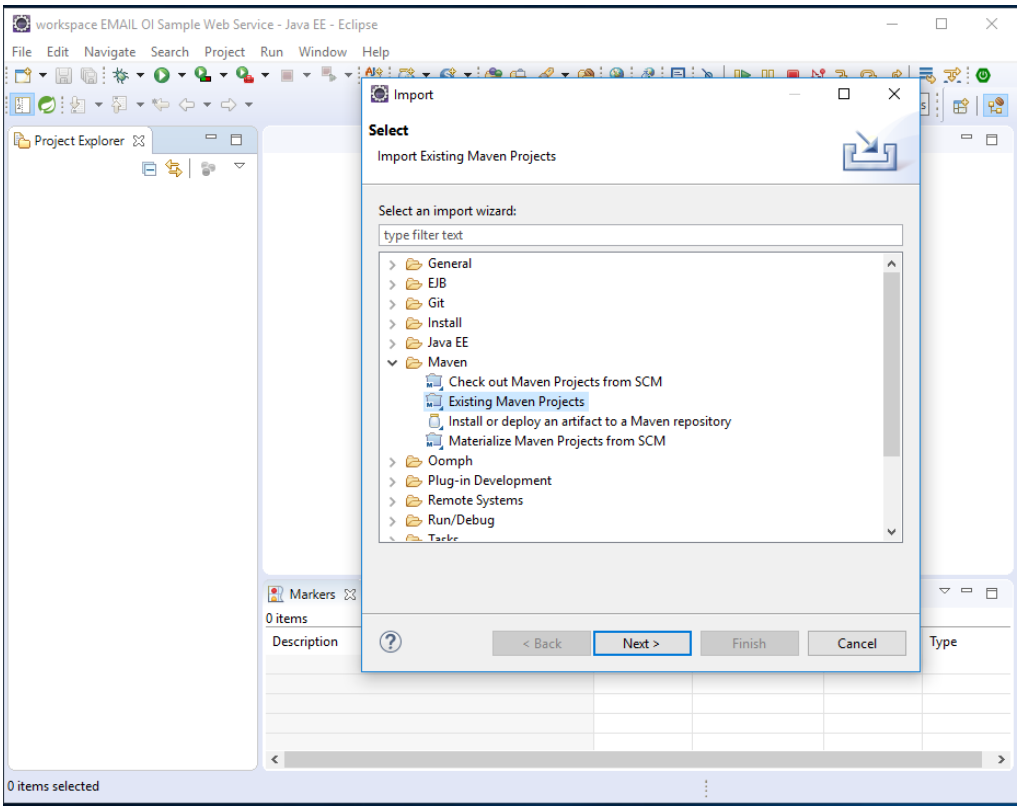
- SoapUI

Environment Set-up

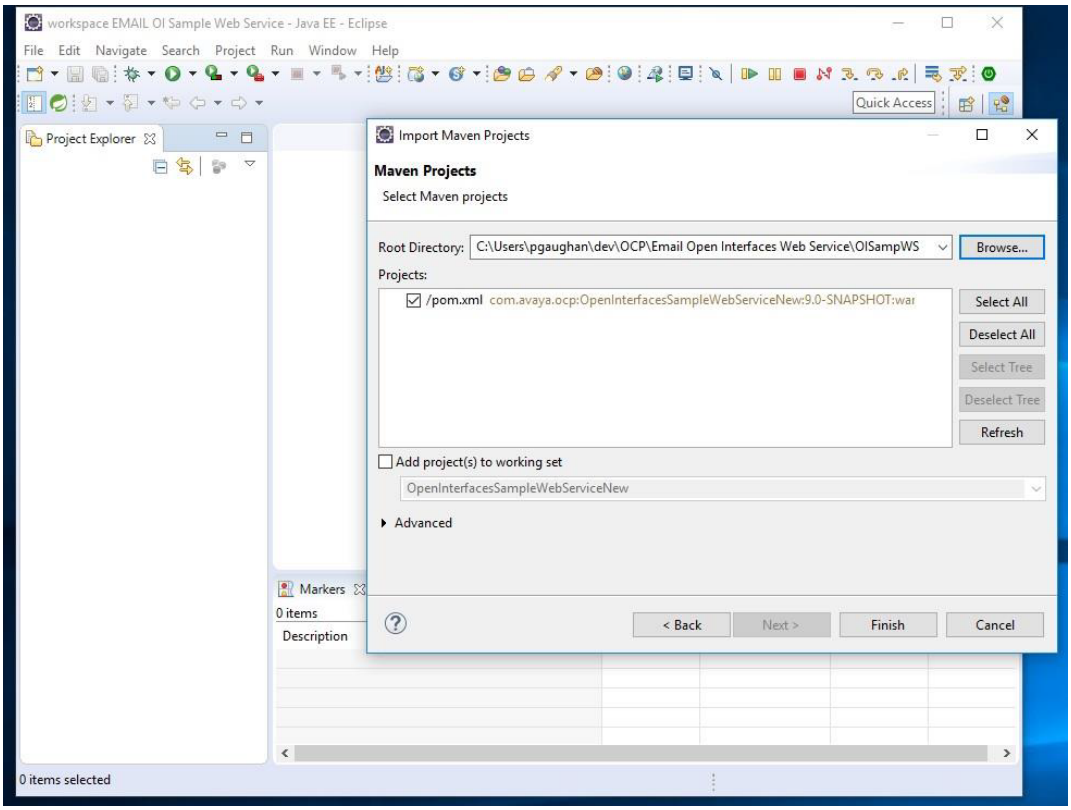
1. Download and install the latest JDK for your environment.
2. Download and install the latest version of Eclipse. You can find the latest version at https://www.eclipse.org/downloads/.
3. Download and install the latest version of Apache Tomcat. You can find the latest version at https://tomcat.apache.org/ . It is recommended that you download the zip artefact. When you have done so, extract it to a convenient location, such as the Desktop folder.

Importing the Project in Eclipse

Open your Eclipse IDE and click on File -> Import. Under Maven, select the Existing Maven Projects option and click Next:

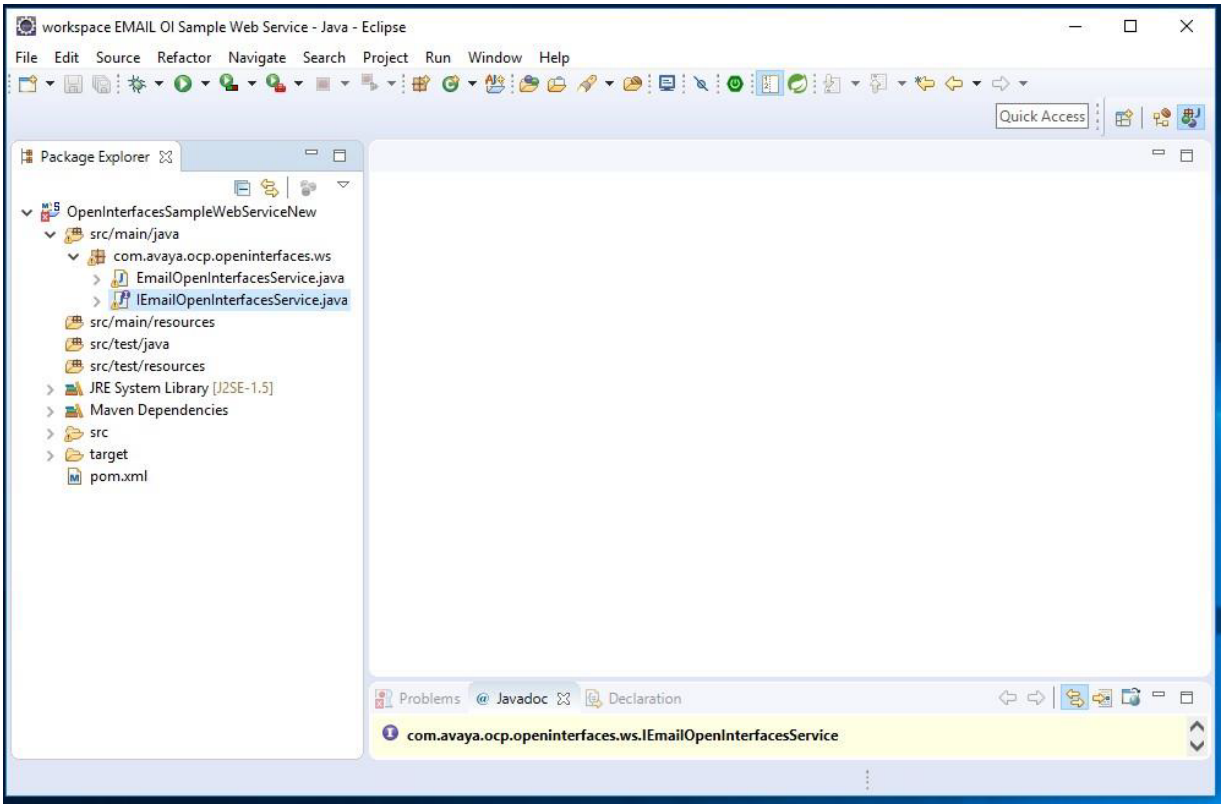


On the next page click Browse and find the Sample Open Interfaces project that you downloaded previously. Select it and click Finish.



Eclipse should then begin importing the project into your workspace. The import will take a few minutes to complete.

When the project has finished importing, you should be presented with something similar to the following:



Overview

The sample project contains 2 .java files in the 'com.avaya.ocp.openinterfaces.ws' package.

- IEmailOpenInterfacesService.java: simple interface to define the contract for the web service method
- EmailOpenInterfacesService.java: Service class implements above and performs the business logic

IEmailOpenInterfacesService.java *EmailOpenInterfacesService.java

```
1  package com.avaya.ocp.openinterfaces.ws;
2
3  import javax.jws.WebMethod;
9
10 @WebService
11 @SOAPBinding(style=Style.DOCUMENT, use=Use.LITERAL, parameterStyle=ParameterStyle.BARE)
12 public interface IEmailOpenInterfacesService {
13
14       /// <summary>
15       /// Input array:
16       ///     [0]: Subject
17       ///     [1]: Plain text Body of email
18       /// Output array:
19       ///     [0]: Subject
20       ///     [1]: Plain text Body of email
21       /// </summary>
22       /// <param name="inputParameters"></param>
23       /// <returns></returns>
24     @WebMethod(action = "MaskCreditCardNumber", operationName = "MaskCreditCardNumber")
25     String[] MaskCreditCardNumber(String[] inputParameters);
26 }
27
28
```

T

'MaskCreditCardNumber' which takes a String Array parameter. Please note that all publicly exposed Open Interfaces web methods must take a String Array parameter and return a String Array.

```
1  package com.avaya.ocp.openinterfaces.ws;
2
3  import java.util.regex.Matcher;
6
7  @WebService(portName="EmailOpenInterfacesServicePort",
8  serviceName = "EmailOpenInterfacesServiceService",
9  targetNamespace = "http://ws.openinterfaces.ocp.avaya.com/",
10 endpointInterface = "com.avaya.ocp.openinterfaces.ws.IEmailOpenInterfacesService")
11 public class EmailOpenInterfacesService implements IEmailOpenInterfacesService{
12
13     private String CreditCardPattern = "^(?:(?<visa>4[0-9]{12}(?:[0-9]{3})?)|" +
14             "(?<mastercard>5[1-5][0-9]{14})|" +
15             "(?<discover>6(?:011|5[0-9]{2})[0-9]{12})|" +
16             "(?<amex>3[47][0-9]{13})|" +
17             "(?<diners>3(?:0[0-5]|[68][0-9])?[0-9]{11})|" +
18             "(?<jcb>(?:2131|1800|35[0-9]{3})[0-9]{11}))$";
19
20
21   /// <summary>
22   /// Input array:
23   ///     [0]: Subject
24   ///     [1]: Plain text Body of email
25   /// Output array:
26   ///     [0]: Subject
27   ///     [1]: Plain text Body of email
28   /// </summary>
29   /// <param name="inputParameters"></param>
30   /// <returns></returns>
31   public String[] MaskCreditCardNumber(String[] inputParameters)
32   {
33       String[] outputParameters = new String[2];
34
35       if (inputParameters != null && inputParameters[0] != null && inputParameters[1] != null)
36       {
37           try{
38               outputParameters[0] = DetectCreditCardNumber(inputParameters[0]);
39               outputParameters[1] = DetectCreditCardNumber(inputParameters[1]);
40           }
41           catch (Exception e)
42           {
43               outputParameters[0] = inputParameters[0];
44               outputParameters[1] = inputParameters[1];
45           }
46       }
47
48       return outputParameters;
49   }
50
```

T

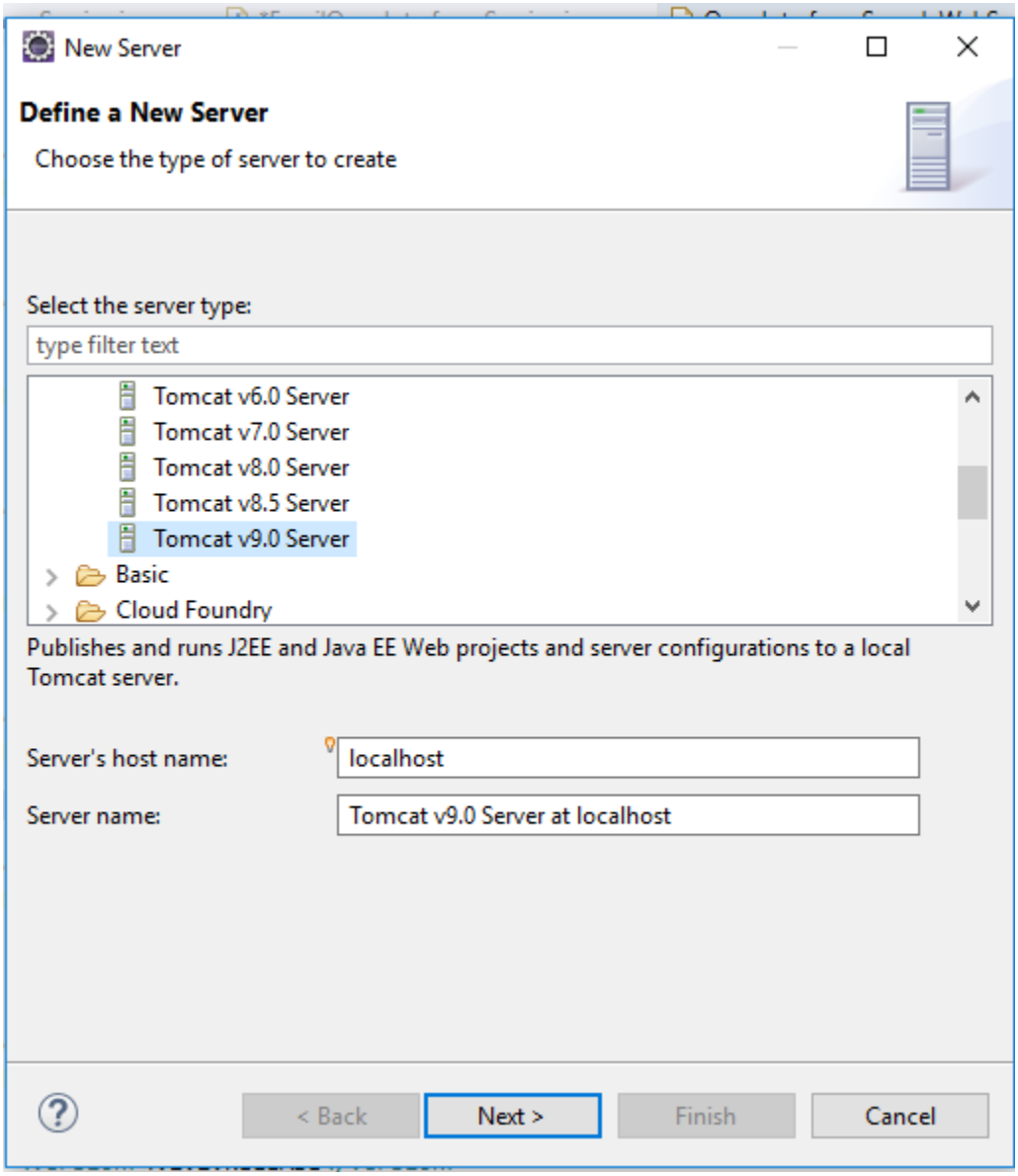
'MaskCreditCardNumber' demo method. This simple example method takes a String array parameter, expecting two entries that map to the subject and body of an email. The service checks these strings for the presence of a credit card number. If a credit card number is detected, the numeric characters will be replaced, to sanitize this sensitive data. The service will return the updated String array with the sensitive credit card data masked. If no card number is found, the strings are simply returned back unmodified.

Configuring Tomcat

Tomcat must be configured in order to run the example web service. Examine the Servers tab at the bottom of your Eclipse IDE. You will be informed that no servers are currently available. Click on the link to configure Tomcat for use with Eclipse:

```
  private String DetectCreditCardNumber(String input){
      String output = null;

      try{
          Pattern pattern = Pattern.compile(CreditCardPattern);
            //Strip all hyphens
            input = input.replaceAll("-", "");
            //Match the card
            Matcher matcher = pattern.matcher(input);

          if (matcher.matches())
          {
              output = ScrubCreditCard(matcher, input);
          }
          else
          {
              output = input;
          }
      }
      catch (Exception e)
      {
          output = input;
      }


      return output;
  }

  private String ScrubCreditCard(Matcher match, String input)
  {
      String output = input;
      Pattern pattern = Pattern.compile(CreditCardPattern);
      output = pattern.matcher(input).replaceAll("++++++++++++++++++");

      return output;
  }
}
```

Under the Tomcat server type, select the version of Tomcat you have downloaded (e.g. 9.0). Leave everything at its default setting and click Next:



On the next page, click Browse. Navigate to the folder where Tomcat is located, then click Next: On the next page you will be asked to upload a WAR file to run on the server. You should see a WAR file with the project name. Select it, then click Add, then click Finish. Go back to the Servers tab, right click the Tomcat server you just configured and click 'Start'.

By default the Tomcat application server will be running on port 8080 at http://localhost:8080 and the example web service WSDL is available at http://localhost:8080/OpenInterfacesSampleWebServiceNew- 9.0-SNAPSHOT/OpenInterfaceService?wsdl

You should see something like the below WSDL in your browser:

```
<!--
    Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.3-b02-.
 -->
<!--
    Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.3-b02-.
 -->
<definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://ws.openinterfaces.ocp.avaya.com/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://ws.openinterfaces.ocp.avaya.com/" name="EmailOpenInterfacesServiceService">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://ws.openinterfaces.ocp.avaya.com/" schemaLocation="http://10.134.61.34:8080/OpenInterfacesSampleWebServiceNew-9.0-SNAPSHOT/OpenInterfaceService?
      xsd=1"/>
    </xsd:schema>
    <xsd:schema>
      <xsd:import namespace="http://jaxb.dev.java.net/array" schemaLocation="http://10.134.61.34:8080/OpenInterfacesSampleWebServiceNew-9.0-SNAPSHOT/OpenInterfaceService?xsd=2"/>
    </xsd:schema>
  </types>
  <message name="MaskCreditCardNumber">
    <part name="MaskCreditCardNumber" element="tns:MaskCreditCardNumber"/>
  </message>
  <message name="MaskCreditCardNumberResponse">
    <part name="MaskCreditCardNumberResponse" element="tns:MaskCreditCardNumberResponse"/>
  </message>
  <portType name="IEmailOpenInterfacesService">
    <operation name="MaskCreditCardNumber">
      <input message="tns:MaskCreditCardNumber"/>
      <output message="tns:MaskCreditCardNumberResponse"/>
    </operation>
  </portType>
  <binding name="EmailOpenInterfacesServicePortBinding" type="tns:IEmailOpenInterfacesService">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="MaskCreditCardNumber">
      <soap:operation soapAction="MaskCreditCardNumber"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>
  <service name="EmailOpenInterfacesServiceService">
    <port name="EmailOpenInterfacesServicePort" binding="tns:EmailOpenInterfacesServicePortBinding">
      <soap:address location="http://10.134.61.34:8080/OpenInterfacesSampleWebServiceNew-9.0-SNAPSHOT/OpenInterfaceService"/>
    </port>
  </service>
</definitions>
```

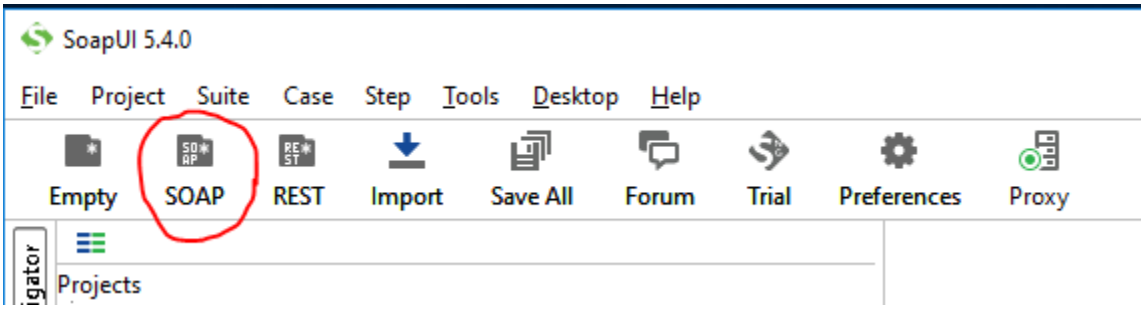
Testing the Web Service Example

To call the service, you can use an open source SOAP web service testing tool such as SoapUI to generate a request and view the response.

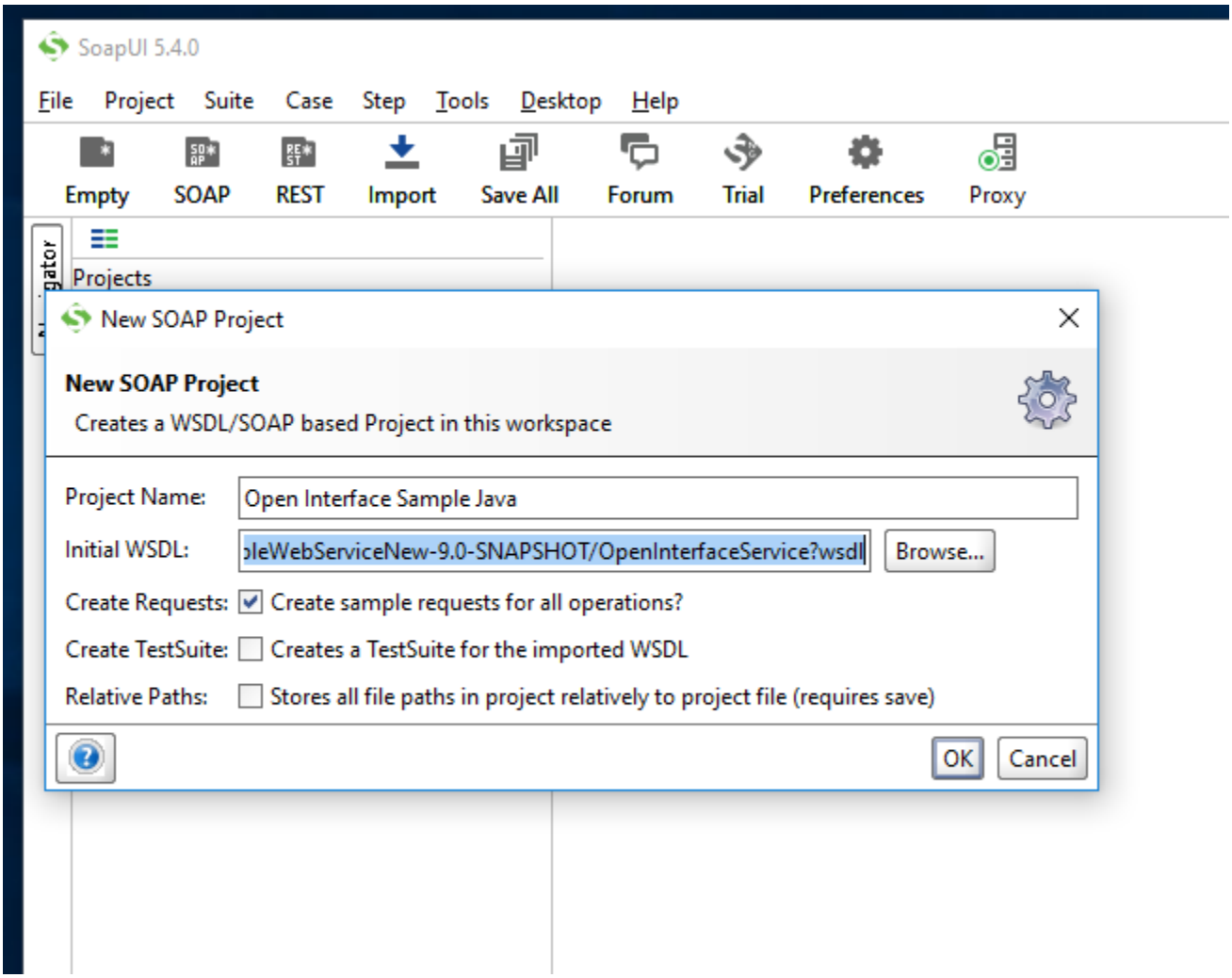
An open source version of SoapUI is available to download from here: https://www.soapui.org/downloads/soapui.html

Testing with SoapUI

To test the web service with SoapUI, click the SOAP icon in the navigation bar to create a new SOAP project:



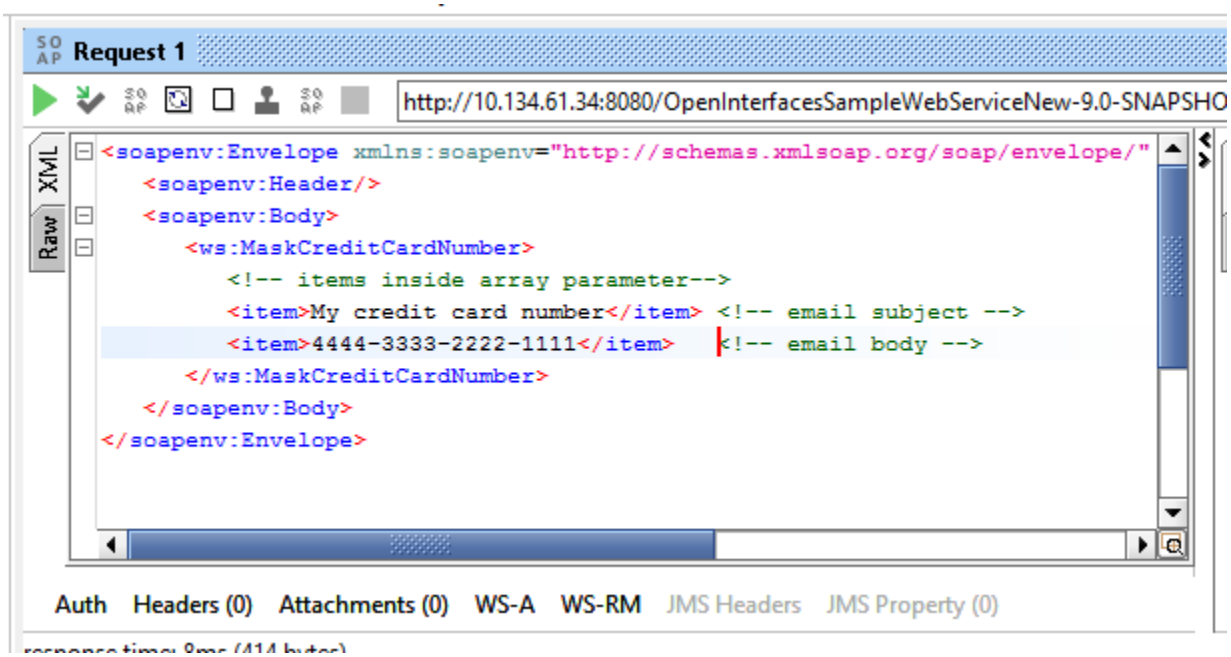
Enter a name for your project and copy and paste your WSDL URL in the ‘initial WSDL’ field as shown:



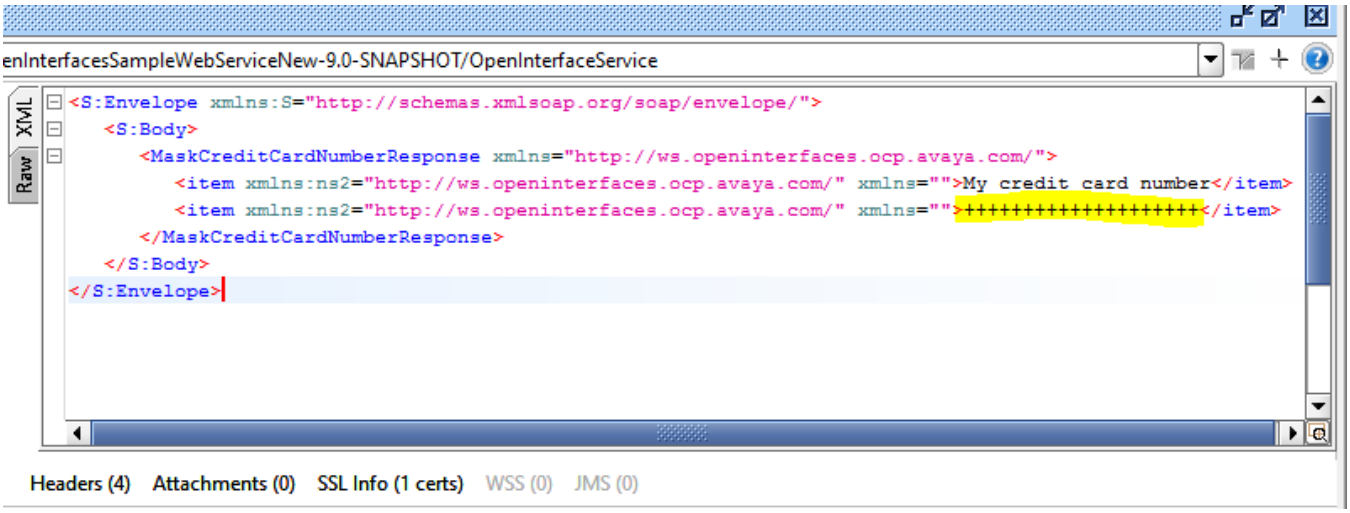
Make sure to leave the 'Create Requests' check box ticked and click OK. SoapUi will then connect and parse the WSDL from your web service and use the details to automatically generate a sample request in the correct format.

The generated request can be edited to enter various parameters of your choice for testing purposes. Remember that for compatibility purposes, the service method signature can only accept a single parameter in the form of a String Array. You may pass multiple strings as needed as long as they reside inside one single String Array collection.

In the example request below, the 'item' tags represent individual Strings inside the String Array parameter. The first 'item' tag represents the subject field and the second represents the body. In the example below, we have entered the text 'My credit card number' for the subject field and '4444-3333- 2222-1111' as an example 16 digit credit card number in the email body.



Upon executing the request, we can view the sample response returned by the service in the right-hand pane. If your service is up and running, you should see something similar to the example response shown below. The subject remains unchanged in this case but the credit card number in the body has been successfully masked in the response.

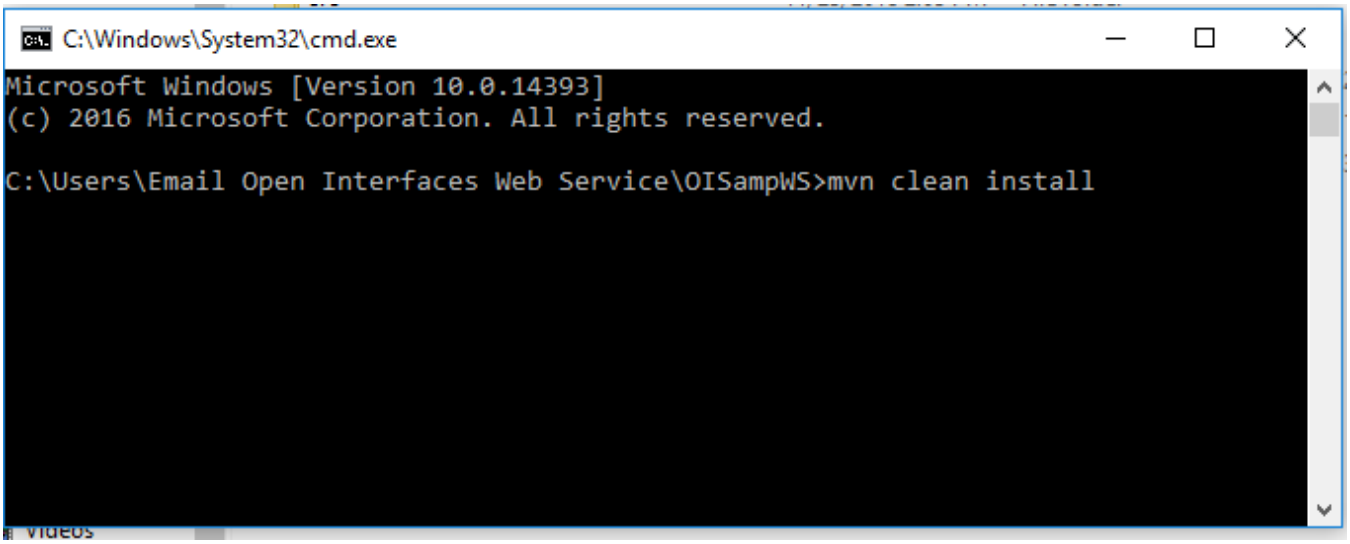


How To Deploy Remotely

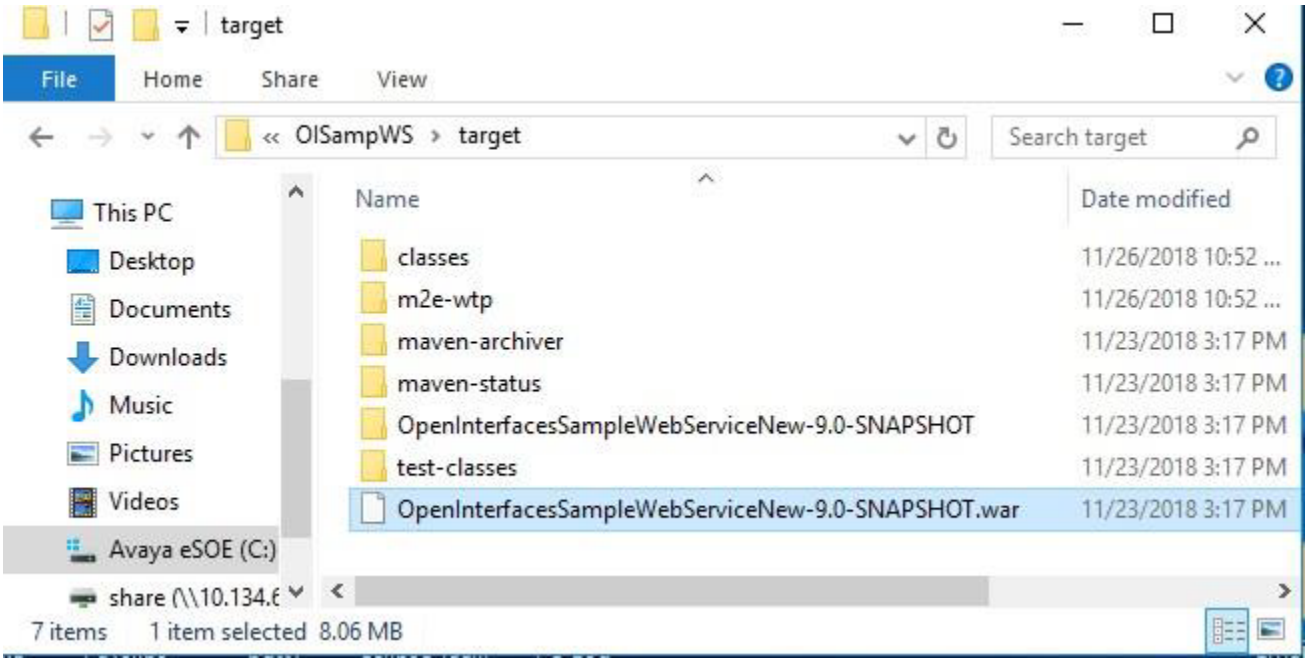
In this section we are going to build a WAR file that will enable us to deploy the web service on a Tomcat server other than our local one. This server will be referred to as a “Remote” Tomcat.

You can download Apache Tomcat from here: http://tomcat.apache.org/ and install it on your local system and / or a remote server of your choice.

To build the WAR file for remote deployment outside of Eclipse, navigate to the directory where you saved the Open Interfaces Web Service project and open a command line interface. Enter the command ‘mvn clean install’ as shown below.



When the build completes, you find the WAR file in the ‘target’ folder of the project.



Go to your Remote Tomcat install directory and find the “webapps” subdirectory, paste the .war file into here. If the Tomcat server is not already running, you can start it via “[tomcat_base]\bin\startup.bat”.
After the Tomcat server has started you should be able to view the WSDL in your browser. The Tomcat server is now running independently of our Eclipse. If you wish to update the web service, simply remove the WAR file and replace it with the updated version. For a more detailed guide on Tomcat web application deployment refer to the Tomcat docs.

Integration with Oceana Email Service

To connect to the sample service via Oceana Email Service you will need to enter the web service details in the OCP Admin, following the steps as described earlier in the Administration section.

Unlike running locally, you will need to explicitly specify the IP address of your web service host in the Admin URL field, replacing ‘localhost’ with the actual IP address. If you are running the service on your local machine via Eclipse or a local Tomcat instance, then you will simply enter your IP address of your local machine. If you have deployed your service on a remote host server, then enter the IP address of your remote host in the URL e.g. http://<ip address>:8080/OpenInterfacesSampleWebServiceNew-9.0- SNAPSHOT/OpenInterfaceService?wsdl

(Hint: on a Windows computer, you can check your IP address in the command terminal using the command ‘ipconfig’).

HTTPS support

Oceana supports secure connections to the Open Interfaces web service via HTTPS.

It is common for a WSDL ‘GET’ request to be accessible over HTTP while only the ‘POST’ requests to call the service require HTTPS.

However, If the web service operations must be called via a secure HTTPS connection, then a HTTPS URL for the WSDL must be entered in the Admin, even if the ‘GET’ request for the WSDL file itself is also available over HTTP.

The non-secure HTTP URL for the WSDL must not be entered in the admin, if the POST requests to call the service are using HTTPS as the redirect to HTTPS from HTTP will cause a failure.

By default, the MM Admin will auto-fill the URL field with the non-secure prefix ‘http://‘ but this must be changed to ‘https:// ’ if the web service operations require a secure connection.

Enabling HTTPS on sample Java web service

To enable HTTP on the example project, open the web.xml file located in the WEB-INF folder.

```
57
58    <security-constraint>
59    <web-resource-collection>
60        <web-resource-name>securedapp</web-resource-name>
61        <url-pattern>/*</url-pattern>
62        <!--
63                JSR-109 specifies that http-method POST must be used.
64                If we include GET here, access to the WSDL will also be
65                protected, otherwise not.
66            -->
67            <http-method>POST</http-method>
68    </web-resource-collection>
69    <user-data-constraint>
70        <transport-guarantee>NONE</transport-guarantee>
71        <!-- To enable HTTPS replace NONE above with CONFIDENTIAL as per below -->
72        <!--  <transport-guarantee>CONFIDENTIAL</transport-guarantee> -->
73    </user-data-constraint>
74 </security-constraint>
75
76 </web-app>
```

HTTPS is not enabled by default. To enable it, simply edit the transport-guarantee element as per the comment shown, replacing NONE with CONFIDENTIAL.

Also note that only POST requests will be secured as per the default settings shown above. The WSDL URL itself will be accessible over via GET requests over non secure HTTP. (This default behaviour can be changed if desired by including GET here also as per the comment shown in blue).

If enabling SSL then you must update the WSDL URL in the Admin with the HTTPS prefix and port number e.g. https://<ip address>:8443 /OpenInterfacesSampleWebServiceNew-9.0- SNAPSHOT/OpenInterfaceService?wsdl

This is required even if GET requests remain unsecured, since redirects from HTTP to HTTPS are currently not supported.

Troubleshooting

The following provides some guidelines for troubleshooting issues with the Open Interfaces service.

Failure to detect WSDL operations in OCP Admin

Don’t be alarmed if auto discovery is not working since this feature is NOT a compatibility test. The Admin auto-discovery feature has limitations and cannot support all WSDL formats. The details can be added manually if auto discovery fails. The WSDL parsing within Oceana is more complex and supports a broader range of formats.

Configuration changes in OCP Admin not picked up

A reboot of both OCP nodes is required after any updates to the Open Interfaces settings in the Admin.

Parameter mismatch

Parameter Sequence: Be sure to take care when setting the sequence of input and output parameters of your web service in the OCP Admin. The order in which you add the input parameters should match the order in which your web service method expects to receive them. Likewise, the sequence of parameters returned back from the web service should match the configuration in the Admin.

Parameter Quantity: Take care to match the quantity of input and output parameters exactly with those received / returned by the web service.

Failure to call HTTPS web service

If you encounter problems calling your HTTPS service, you could try temporarily disabling HTTPS on your service to test and confirm whether the issue is related. If you suspect a possible HTTPS related issue, below are some possible causes:

HTTPS redirect error: Ensure that you have entered a HTTPS URL for the WSDL in MM Admin as a redirect from HTTP to HTTPS is not supported and will cause a connection failure.

Certificate error: Ensure that you have imported the trusted certificate for your web service in System Manager for both OCP nodes.

Ensure the Common Name (CN) / Subject Alternative Name (SAN) field on your certificate matches the host name or IP address of your web service host.