Avaya Proactive Outreach Manager Event SDK

Release 4.0.2 SP3
Issue 2.5
July 2024

AVAYA SOFTWARE DEVELOPMENT KIT LICENSE AGREEMENT

REVISED: October 2023

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 LLC 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 LLC. For purposes of this definition, "control" means the power to direct the management and policies of such party, directly or indirectly, whether through ownership of voting securities, by contract or otherwise; and the terms "controlling" and "controlled" have meanings correlative to the foregoing.

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

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

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

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

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

1.7 “Intellectual Property” means any and all: (i) rights associated with works of authorship throughout the world, including copyrights, neighboring rights, moral rights, and mask works, (ii) trademark and trade name rights and similar rights, (iii) trade secret rights, (iv) patents, algorithms, designs and other industrial property rights, (v) all other intellectual and industrial property rights (of every kind and nature throughout the world and however designated) whether arising by operation of law, contract, license, or otherwise, and (vi) all registrations, initial applications, renewals, extensions, continuations, divisions or reissues thereof now or hereafter in force (including any rights in any of the foregoing).
1.8 “Permitted Modification(s)” means Licensee’s modifications of the Sample Application Code as needed to create applications, interfaces, workflows or processes for use with Avaya products.

1.9 “Specification Document” means any notes or similar instructions in hard copy or machine readable form, including any technical, interface and/or interoperability specifications that define the requirements and conditions for connection to and/or interoperability with Avaya products, systems and solutions.
1.10 “Source Code” means human readable or high-level statement version of software written in the source language used by programmers and includes one or more programs. Source Code programs may include one or more files, such as user interface markup language (.mxml), action script (.as), precompiled Flash code (.swc), java script (.js), hypertext markup language (.html), active server pages (.asp), C# or C# .Net source code (.cs), java source code (.java), java server pages (.jsp), java archives (.jar), graphic interchange format (.gif), cascading style sheet (.css), audio files (.wav) and extensible markup language (.xml) files.
1.11 “Sample Application Code” means Software provided for the purposes of demonstrating functionality of an Avaya product through the Avaya Software Development Kit.
1.12 “Software” means data or information constituting one or more computer or apparatus programs, including Source Code or in machine-readable, compiled object code form.
2.0 LICENSE GRANT.
2.1 SDK License.
A. Provided Licensee pays to Avaya the applicable license fee (if any), Avaya hereby grants Licensee a limited, non-exclusive, non-transferable license (without the right to sublicense, except as set forth in 2.1B(iii)) under the Intellectual Property of Avaya and, if applicable, its licensors and suppliers to (i) use the SDK solely for the purpose of Licensee's internal development efforts to develop applications, interfaces, value-added services and/or solutions, workflows or processes to work in conjunction with Avaya products; (ii) to package Client Libraries for redistribution with Licensee’s complementary applications that have been developed using this SDK, subject to the terms and conditions set forth herein; (iii) use Specification Documents solely to enable Licensee’s products, services and application solutions to exchange messages and signals with Avaya products, systems and solutions to which the Specification Document(s) apply; (iv) modify and create Derivative Works of the Sample Application Code, Specification Documents and Documentation solely for internal development of applications, interfaces, workflows or processes for use with Avaya products, integration of such applications, interfaces, workflows and processes with Avaya products and interoperability testing of the foregoing with Avaya products; and (v) compile or otherwise prepare for distribution the Sample Application Code with Permitted Modifications, into an object code or other machine-readable program format for distribution and distribute the same subject to the conditions set forth in Section 2.1B.
B. The foregoing license to use Sample Application Code is contingent upon the following: (i) Licensee must ensure that the modifications made to the Sample Application Code as permitted in clause (iv) of Section 2.1A are compatible and/or interoperable with Avaya products and/or integrated therewith, (ii) Licensee may distribute Licensee’s application that has been created using this SDK, provided that such distribution is subject to an end user pursuant to Licensee’s current end user license agreement (“Licensee EULA”) that is consistent with the terms of this Agreement and, if applicable, any other agreement with Avaya (e.g., the Avaya DevConnect Program Agreement), and is equally as protective as Licensee’s

standard software license terms, but in no event shall the standard of care be less than a reasonable degree of care, and (iii) Licensee ensures that each end user who receives Client Libraries or Sample Application Code with Permitted Modifications has all necessary licenses for all underlying Avaya products associated with such Client Libraries or Sample Application Code.
Your Licensee EULA must include terms concerning restrictions on use, protection of proprietary rights, disclaimer of warranties, and limitations of liability. You must ensure that Your End Users using applications, interfaces, value-added services and/or solutions, workflows or processes that incorporate the API, Client Libraries, Sample Code or Permitted Modifications adhere to these terms, and You agree to notify Avaya promptly if You become aware of any breach of the terms of Licensee EULA that may impact Avaya. You will take all reasonable precautions to prevent unauthorized access to or use of the SDK and notify Avaya promptly of any such unauthorized access or use.

C. Licensee acknowledges and agrees that it is licensed to use the SDK only in connection with Avaya products (and if applicable, in connection with services provided by or on behalf of Avaya).
D. With respect to Software that contains elements provided by third party suppliers, Licensee may install and use the Software in accordance with the terms and conditions of the applicable license agreements, such as "shrinkwrap" or "click-through" licenses, accompanying or applicable to the Software.
2.2 No Standalone Product. Nothing in this Agreement authorizes or grants Licensee any rights to distribute or otherwise make available to a third party the SDK, in whole or in part, or any Derivative Work in source or object code format on a standalone basis other than the modifications permitted in Section 2.1B of this Agreement.
2.3 Proprietary Notices. Licensee shall not remove any copyright, trade mark or other proprietary notices incorporated in the copies of the SDK, Sample Application Code and redistributable files in Licensee's possession or control or any modifications thereto. Redistributions in binary form or other suitable program format for distribution, to the extent expressly permitted, must also reproduce Avaya's copyright, trademarks or other proprietary notices as incorporated in the SDK in any associated Documentation or "splash screens" that display Licensee copyright notices.
2.4 Third-Party Components. You acknowledge certain software programs or portions thereof included in the SDK may contain software distributed under third party agreements ("Third Party Components"), which may contain terms that expand or limit rights to use certain portions of the SDK ("Third Party Terms"). Information identifying the copyright holders of the Third Party Components and the Third Party Terms that apply is available in the attached Schedule 1 (if any), SDK, Documentation, or on Avaya's web site at: http://support.avaya.com/Copyright (or such successor site as designated by Avaya). The open source software license terms provided as Third Party Terms are consistent with the license rights granted in this Agreement, and may contain additional rights benefiting You, such as modification and distribution of the open source software. The Third Party Terms shall take precedence over this Agreement, solely with respect to the applicable Third Party Components, to the extent that this Agreement imposes greater restrictions on You than the applicable Third Party Terms. Licensee is solely responsible for procuring any necessary licenses for Third Party Components, including payment of licensing royalties or other amounts to third parties, for the use thereof.
2.5 Copies of SDK. Licensee may copy the SDK only as necessary to exercise its rights hereunder.
2.6a No Reverse Engineering. Licensee shall have no rights to any Source Code for any of the software in the SDK, except for the explicit rights to use the Source Code as provided to Licensee hereunder. Licensee agrees that it shall not cause or permit the disassembly, decompilation or reverse engineering of the Software. Notwithstanding the foregoing, if the SDK is rightfully located in a member state of the European Union and Licensee needs information about the Software in the SDK in order to achieve interoperability of an independently created software program with the Software in the SDK, Licensee will first request such information from Avaya. Avaya may charge Licensee a reasonable fee for the provision of such information. If Avaya refuses to make such information available, then Licensee may take steps,

such as reverse assembly or reverse compilation, to the extent necessary solely in order to achieve interoperability of the Software in the SDK with an independently created software program. To the extent that the Licensee is expressly permitted by applicable mandatory law to undertake any of the activities listed in this section, Licensee will not exercise those rights until Licensee has given Avaya twenty (20) days written notice of its intent to exercise any such rights.

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

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

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

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

2.10 Independent Development.

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

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

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

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

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

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

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

2.15 No Virus. Licensee warrants that (i) the applications, interfaces, value-added services and/or solutions, workflows or processes Licensee develops using this SDK will not contain any computer program file that includes time code limitations, disabling devices, or any other mechanism which will prevent the Avaya product (including other software, firmware, hardware), services and networks from being functional at all times (collectively "Time Bombs"); and (ii) the applications, interfaces, value-added services and/or solutions, workflows or processes Licensee develops using this SDK will be free of computer viruses, malicious or other harmful code, black boxes, malware, trapdoors, and other mechanisms which could: a) damage, destroy or adversely affect Avaya product, or services and/or end users; b) allow remote/hidden attacks or access through unauthorized computerized command and control; c) spy (network sniffers, keyloggers), and d) damage or erase such applications, interfaces, value-added services and/or solutions, workflows or processes developed using this SDK or data, or any computer files or systems of Avaya,

Affiliates, and/or end users (collectively "Virus"). In addition to any other remedies permitted in the Agreement, if Licensee breaches its warranties under this Section, Licensee will, at its expense, take remedial action to eliminate any Time Bombs and/or Viruses and prevent re-occurrence (including implementing appropriate processes to prevent further occurrences) as well as provide prompt, reasonable assistance to Avaya to materially reduce the effects of the Time Bomb and/or Virus.

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

2.17 Third Party Licensed Software

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

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

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

3. OWNERSHIP.

3.1 As between Avaya and Licensee, Avaya or its licensors or suppliers shall own and retain all Intellectual Property rights, in and to the SDK and any corrections, bug fixes, enhancements, updates, improvements, or modifications thereto and Licensee hereby irrevocably transfers, conveys and assigns to Avaya, its licensors and its suppliers all of its right, title, and interest therein. Avaya or its licensors or suppliers shall have the

exclusive right to apply for or register any patents, mask work rights, copyrights, and such other proprietary protections with respect thereto. Licensee acknowledges that the license granted under this Agreement does not provide Licensee with title or ownership to the SDK, but only a right of limited use under the terms and conditions of this Agreement.

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

4.0 SUPPORT.

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

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

5.0 CONFIDENTIALITY.

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

pursuant to a court order, so long as Avaya is given adequate notice and the ability to challenge such required disclosure.

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

6.0 NO WARRANTY.

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

7.0 CONSEQUENTIAL DAMAGES WAIVER.

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

8.0 LIMITATION OF LIABILITY.

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

9.0 INDEMNIFICATION.

Licensee shall 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.

15.0GOVERNING 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. 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.

Contents

Preface

This section contains the following topics:

Purpose

The purpose of this guide is to provide detailed information about Avaya Proactive Outreach Manager Event SDK.

Audience

This guide is for personnel who develop client applications for Avaya Proactive Outreach Manager. The Avaya Proactive Outreach Manager Event SDK provides a JAVA based client for monitoring and reporting real-time operations.

Related documents

- Avaya Proactive Outreach Manager Overview and Specification
- Implementing Avaya Proactive Outreach Manager
- Administering Avaya Proactive Outreach Manager
- POM EventSDK-Kafka HA Guide

What's New

A new KAFKA topic CCTRENDS has been introduced.

What's Changed

Issue	Date	Summary of Changes
Release 4.0.2 SP2 Issue 2.4	October, 2023	Updated Appendix E: eventsdk.properties with a new value of key pom.enabled.cipher.suites

Overview

This section contains an overview of Avaya Proactive Outreach Manager and the Software Developer's Kit.

This section contains the following topics:

Avaya Proactive Outreach Manager Fundamentals

Avaya Proactive Outreach Manager (POM) is a multi-channel Outbound Contact Center solution. POM is a managed application of Avaya Experience Portal, linking the capabilities within the platform more closely with the management infrastructure and services. POM provides a solution for unified, outbound capability to communicate through different channels of interaction such as Short Message Service (SMS), or email, or voice. POM integrates with Avaya Aura® Contact Center and Avaya Aura® Call Center Elite to offer agent functionality such as agent blending, pacing, callbacks, conference, and transfer. POM also integrates with Avaya Oceana® Solution to offer limited set of outbound functionalities, i.e., predictive and progressive agent-based dialing.

Avaya Proactive Outreach Manager Modes

There are four Avaya Proactive Outreach Manager modes based on the integration:

- None
- CCElite
- AACC
- AACC-SBP
- Oceana

For more details about POM modes, see *Implementing Avaya Proactive Outreach Manager.*

Avaya Proactive Outreach Manager Major Components

- **Campaign Director**
  Campaign Director is a Linux service responsible for triggering campaigns and data imports at scheduled date and time. Campaign Director is also responsible for pausing and resuming campaigns based on user action and terminating campaigns if their finish criteria is specified. If you install POM as a multiple server configuration, then only one campaign director is in the active state and others are in dormant state. Campaign Director is responsible for assigning the

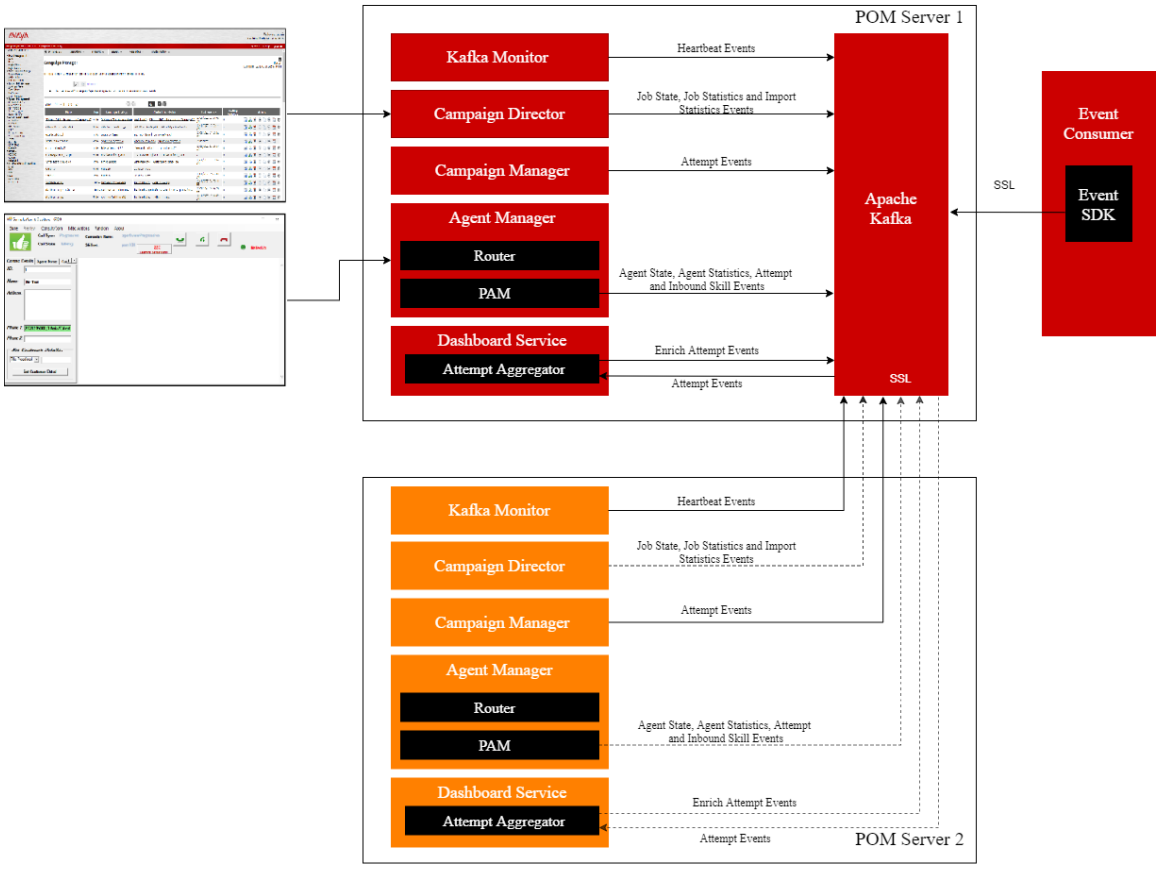
jobs to be processed across campaign managers. At a given time, only one job will be processed by one campaign manager. In the event of campaign manager failure, Campaign Director is responsible for redistribution of job to the next available campaign manager.

Stream Processor would be started on all POM boxes along with CD. And all will point to the leader while running in Kafka in HA mode. However, due to same groupID and single partition, only one will get the Attempt events.

- **Campaign Manager**
  Campaign Manager is a Linux service and is responsible for parsing a campaign strategy, making voice calls, and sending SMS or email messages. Campaign Manager interfaces with one or more EPM servers for making outbound calls.
  If you configure multiple EPM servers, the Campaign Manager uses all the servers in a synchronized manner, using all media resources available for load balancing and failover. If you install POM as a multiple server configuration, the Campaign Manager service runs on all POM servers. When a campaign is executed, a job is created for the campaign, and the campaign director assigns that job to one of the campaign managers for contact processing.
- **Agent Manager**
  POM Agent Manager (PAM) is a Linux service and is the core module to manage and run campaigns. You can either have agent-based campaigns or agent-less (notification) skill-based campaigns.PAM is responsible for distributing licenses to all voice-based campaigns.
  The high-level functions of PAM include:
  - Manages agent allocation and state for agents.
    Manage agents in a blended job. Only CC Elite configuration supports this module.
  - Update the POM database with current agent related information for reporting and HA related functionality.
  - Distribute the preview and predictive agent licenses among running agent-based campaigns and distributing outbound ports to voice notification campaigns.
  - Support real-time commands from POM Monitor such as minimum agents, priority, or agent-based commands such as Forced Logoff.

For more details about POM components please refer to *Avaya Proactive Outreach Manager Overview and Specification.*

Event SDK Design



POM generates various events and sends it to the messaging system, i.e., Apache Kafka. All the POM Components acts as a producer and generates the real-time events for jobs and agents, as well as real time statistics events for jobs and agents.

POM provides the Event SDK which is the consumer of all the events published to Kafka. Event SDK abstracts out all the internal communication and provides cleaner and easier interface to clients. And enables the client to monitor the POM system by consuming the events. These events are provided as a POJO (Plain Old Java Object), for details refer JavaDoc of EventSDK shipped under Sample Client Package.

The communication between the Event SDK and POM happens over SSL. Client can connect only using the credential of a user authorized by Experience Portal and will be able to receive all the agent and job events of the user's organization only.

Working of Apache Kafka

Apache Kafka® is a distributed messaging platform which allows users to subscribe to it and publish data to any number of systems or real-time applications. It provides a

unified, high-throughput, low-latency platform for handling real-time data feeds. Refer https://kafka.apache.org/10/documentation.html for further details.

To enable HA for Kafka server refer to POM EventSDK-Kafka HA guide on support.avaya.com

POM creates one topic per event type per organization on Kafka server:

- *POM.<Org ID or Default>.JOB* for JOB events
- *POM.<Org ID or Default >. AGENT* for Agent events
- *POM.<Org ID or Default >.ATTEMPT* for *all* Attempt events
- *POM.<Org ID or Default >.ENRICHEDATTEMPTRESULT* for consolidated attempt event on contact 'Done' state.
- *POM.<Org ID or Default >.JOBSTATISTICS* for JOB statistics events
- *POM.<Org ID or Default >.AGENTSTATISTICS* for Agent statistics events
- *POM.<Org ID or Default >.IMPORTSTATISTICS* for Import statistics events
- *POM.<Org ID or Default >.CCTRENDS* for completion codes counts events

However, heartbeat and Inbound skill events are shared across all organizations, so it has common topics, *POM.HEARTBEAT and POM.INBOUNDSKILL* respectively.

The Kafka server transparently manages concurrent connections from producers and consumers as well as the transactional capacities.

Below are some of the Kafka related properties used to control the producer/consumer behavior.

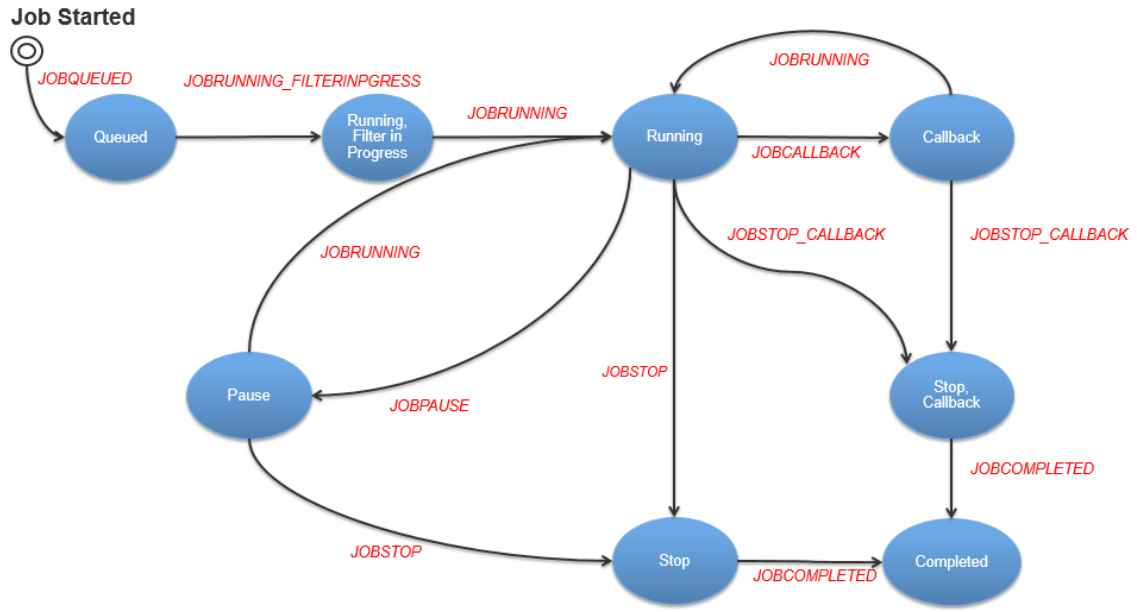
NAME	Description	Valid Values
key.serializer	Serializer class for key that implements the org.apache.kafka.common.serialization.Serializer interface.	org.apache.kafka.common.serialization.StringSerializer
value.serializer	Serializer class for value that implements the org.apache.kafka.common.serialization.Serializer interface.	org.apache.kafka.common.serialization.StringSerializer
Acks	The number of acknowledgments the producer requires the leader to have received before considering a request complete. This controls the durability of records that are sent.	All

Retries	Setting a value greater than zero will cause the client to resend any record whose send fails with a potentially transient error. Note that this retry is no different than if the client resent the record upon receiving the error. Allowing retries without setting max.inflight.requests.per.connection to 1 will potentially change the ordering of records because if two batches are sent to a single partition, and the first fails and is retried but the second succeeds, then the records in the second batch may appear first.	2
request.timeout.ms	The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted. This should be larger than replica.lag.time.max.ms (a broker configuration) to reduce the possibility of message duplication due to unnecessary producer retries.	120000 (millisecond)
max.block.ms	The configuration controls how long KafkaProducer.send() and KafkaProducer.partitionsFor() will block. These methods can be blocked either because the buffer is full, or metadata is unavailable. Blocking in the user-supplied serializers or partitioner will not be counted against this timeout.	50000 (millisecond)
retry.backoff.ms	The amount of time to wait before attempting to retry a failed request to a given topic partition. This avoids repeatedly sending requests in a tight loop under some failure scenarios.	10000 (millisecond)

For EventSDK Client/Consumer related properties please edit the file at <POM_EVENT_SDK_DIR>/conf/eventsdk.properties.

Job States

The following diagram depicts the Job State Model and the Job events generated on each state transitions.



Job State transition and corresponding events:

1. Once job gets kicked in by manually or via scheduler by default it will go to “Queued” state. This time “JOBQUEUED” event will be raised. Hence, the currentJobStatus will be JOBQUEUED and the *lastJobStatus* will be NULL.
2. Once job start then in triggers the contact record filtering. This time “JOBRUNNING_FILTERINPGRESS” event will be raised.
3. Once filtering completes or first batch of filtering completes, job moves to “Running” state. After successful state transition,” JOBRUNNING” event will be raised. Hence, the currentJobStatus will be JOB_ACTIVE and the *lastJobStatus* will be JOB_ACTIVE_FILTER_IN_PROGRESS.
4. From “JOBRUNNINGSTATE” state, job can be moved to “PAUSE” state. After successful state transition,”JOBPAUSE” event will be raised. Hence, the currentJobStatus will be JOB_PAUSING and the *lastJobStatus* will be JOB_ACTIVE.
5. From “JOBRUNNINGSTATE” state, job can be moved to “Stop” state. After successful state transition,” JOBSTOP” event will be raised.
6. From “JOBRUNNINGSTATE” state, job can be moved to “Callback” state if all the contact record for dialing is finished and there are callback pending on this job. After successful state transition,”JOBCALLBACK” event will be raised.
7. From “JOBRUNNINGSTATE” state, job can be moved to “Stop,Callback” state if admin has stopped the job while there are callbacks are pending on this job. After successful state transition,” JOBSTOP_CALLBACK” event will be raised.

8. When job gets completed from “Stop” or “Stop,Callback” state, job will move to “Completed” state. After successful state transition,” JOBCOMPLETED” event will be raised.
9. Job can be move to “Running” state from “Pause” or “Callback” state. This time “JOBRUNNING” event will be raised.

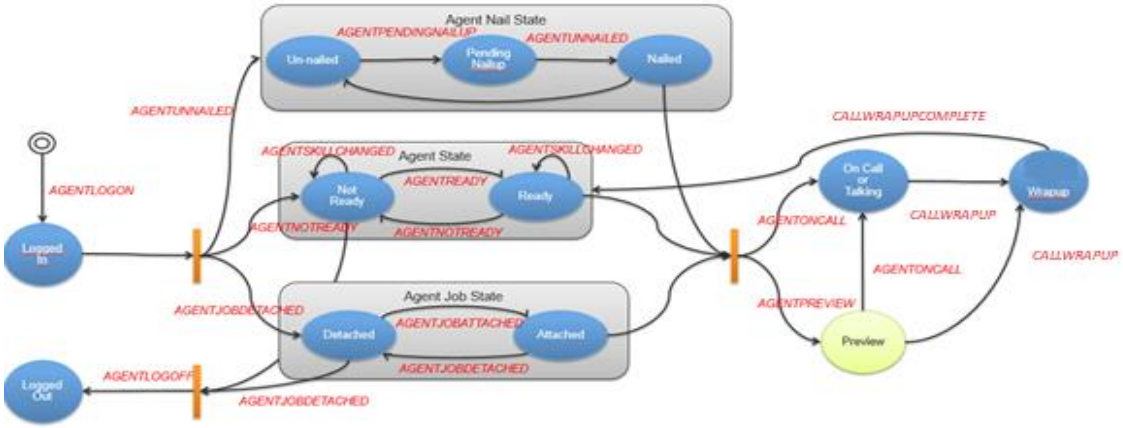
Agent States

POM maintains 4 different state models for each agent that login to POM. These state models are – Agent State, Agent Call State, Agent Job State and Agent Nail State.

- Agent State represents the current state of the agent like LOGGEDIN, READY, BUSY, WORK_NOT_READY, LOGGED_OUT.
- Agent Call state represents the state of call handled by the agent like IDLE, ONCALL, WRAPUP, WRAPUP_COMPLETE and so on.
- Agent Job State represents the state of job to which agent is associated like JOBATTACHED, JOBDETACHED, PENDING_JOB_INBOUND.
- Agent Nail State represents the state of the nailing call placed by POM like NAILED, UNNAILED, PENDING_NAILUP.

Depending upon the nature of operation being performed POM will raise a combination of these events. POM will also raise agent statistical event whenever any of the above state change occurs. For more information see *Administering Avaya Proactive Outreach Manager.*

The following diagram depicts the Agent State Model and the Agent events generated on each state transitions.



Agent states can include:

1. Agent logged in to POM System. After successful login, “AGENTLOGON” event will be raised. By default, agent will go to “Un-Nailed”, “Not Ready” and

“Detached” State. In this case “AGENTUNNAILED”, “AGENTNOTREADY”, “AGENTJOBDETACHED” events will be raised.

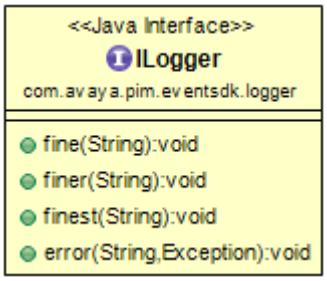
2. Once agent make available for themselves using desktop, agent will move to “Ready” state. “After successful state transition, “AGENTREADY” event will be raised.
3. Once agent will answer the nail up call, agent moved to “PENDINGNailup” state and “AGENTPENDINGNAILUP” event will be raised.
4. Once agent is nailed up, agent will move to “Nailed” state. After successful state transition, “AGENTNAILED” event will be raised.
5. Once Agent attach to job, agent will move to “Attached” state. After successful state transition,”AGENTJOBATTACHED” event will be raised.
6. Once agent will assign a contact record from non-preview campaign and take the call, agent will move to “OnCall” state. After successful state transition,”AGENTONCALL” event will be raised.
7. In case of preview campaign or manual campaign, agent will move to “Preview” state. After successful state transition,”AGENTPREVIEW” event will be raised.
8. After preview, once agent has dialed the contact, agent will move to “OnCall” state. After successful state transition,”AGENTONCALL” event will be raised.
9. Once agent disposes the call, agent will move to “WrapUp” state. After successful state transition,”CALLWRAPUP” event will be raised. On successful completion of contact, agent will move to “Ready” state. This time “CALLWRAPUPCOMPLETE” event will be raised.
10. Once agent is in “Attached” state, agent can move to “Detached” state. This time “AGENTJOBDETACHED” event will be raised.
11. Once agent is in “Ready” state, agent can move to “NotReady” state. This time “AGENTNOTREADY” event will be raised.
12. Once agent is in “NotReady” state, agent can move to “AGENTLOGOFF” state. This time “AGENTLOGOFF” event will be raised.
13. After log out, agent will be moved implicitly to “UnNailed” state. After successful state transition,”AGENTUNNAILED” event will be raised.

Developing Client Applications

To develop client application, one needs to use Event SDK Client library. Below are the Java Classes of Event SDK Client library that should be used by client application to subscribe and to consume the events from the Kafka Server.

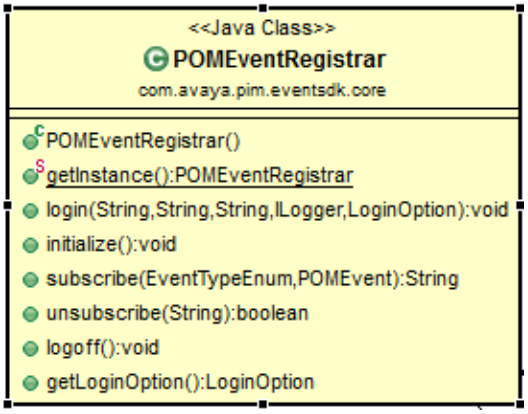
Interface ILogger

This interface must be implemented in client application. Its instance should be passed to the POMEventRegistrar and all the event notifier implementations for logging.



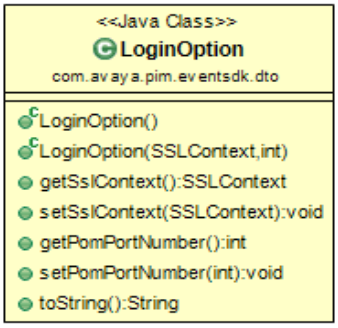
Class POMEventRegistrar

This is the main class of Event SDK client that connects and subscribes to the primary Kafka Server. It has various methods like login(), subscribe(), initialize() and unsubscribe().



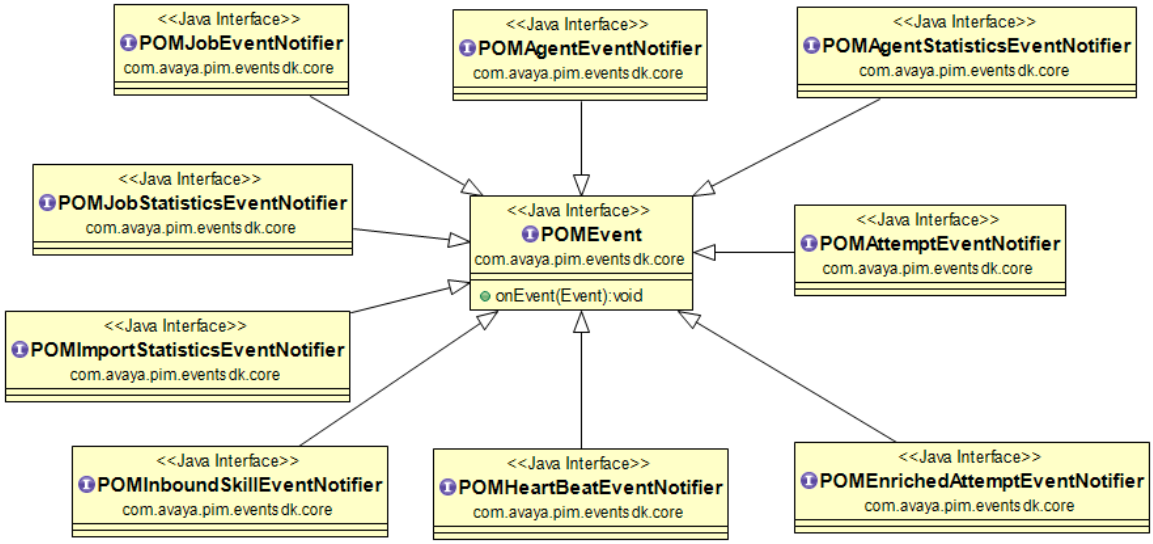
Class LoginOption

This class is used to hold the SSLContext and the port number used by POMEventRegistrar for authorization of client application while login(). If null value is passed, then POMEventRegistrar creates default LoginOption using default port (443) and KeyStore/TrustStore paths present in eventsdk.properties file.



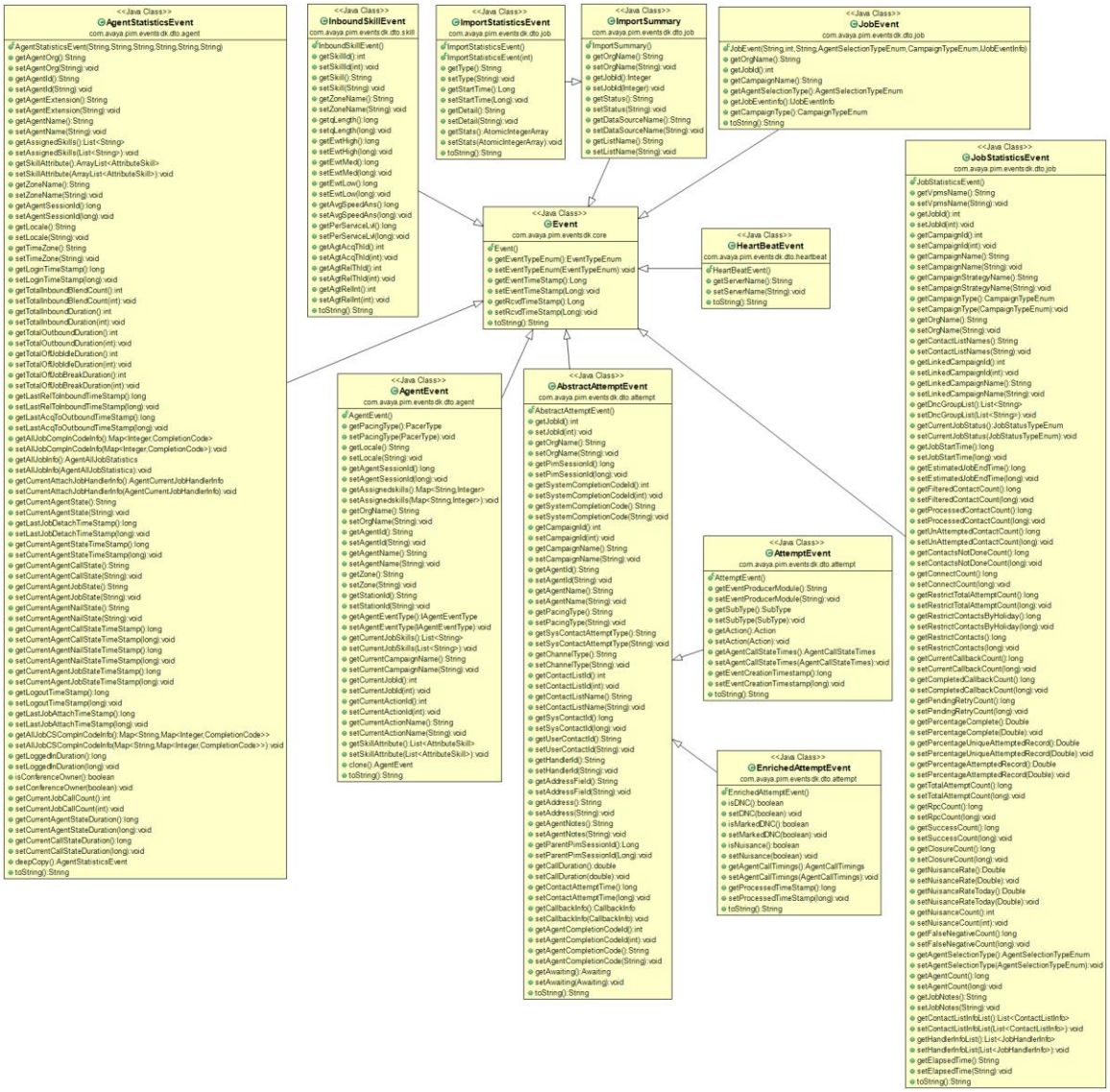
Event Notifier Interfaces

There are six interfaces for each type of Events whose onEvent() method should be implemented. And its instance should be passed to POMEventRegistrar subscribe methods along with the respective EventTypeEnum.



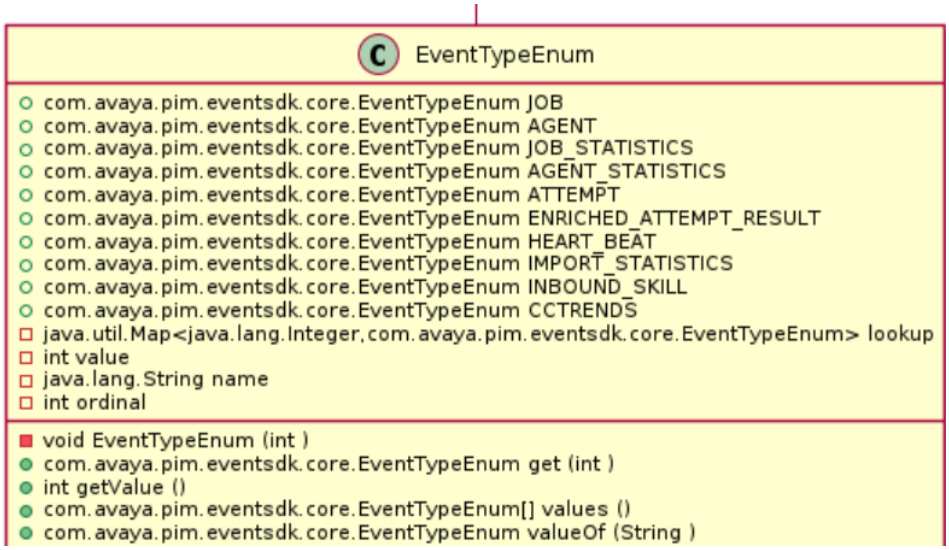
Event Classes

The Event SDK Client receives the events from the Kafka Server, passes the object of Event to onEvent() method of respective Event Notifier implementation.



Enum EventTypeEnum

There are seven Event type enum in this class for each type of Event.



Sample applications

The developer's kit includes sample applications that illustrate how you can implement Event SDK client applications.

The Java Sample Application demonstrates how an Event SDK can be used to subscribe and consume events from the Apache Kafka topics to which POM is publishing events. This sample code was written for customers who want to implement client applications using Java. This sample project is for reference. Refer Appendix B: Java sample application for complete Event SDK Sample Application code.

Using the Avaya Proactive Outreach Manager Event SDK

The Avaya Proactive Outreach Manager Event SDK was developed on top of Client API of Apache Kafka. Event SDK allow any client to consume the events published by POM on Apache Kafka topics. Currently, POM 3.1.x does not require a separate license for Event SDK. However, Avaya reserves the right to issue separate licensing terms for Event SDK.

This section contains the following topics:

Connect to the Avaya Proactive Outreach Manager Event SDK

To start getting the events from POM, you need to connect to the Event SDK Client library and subscribe for the events.

The important operations are:

Initialize the Logger

```
/* Initialize the logger */
ILogger logger = LoggerImpl.getTracer();
```

Login to the Event SDK Client library:

The Event SDK Client needs login credentials of an Experience Portal user.

```
/* This is starting point for Event SDK client. All the operations are
performed on pomEventRegistrar */

POMEventRegistrar pomEventRegistrar = POMEventRegistrar.getInstance();

pomEventRegistrar.login(loginName, password, pomIPAddress, logger, new
LoginOptions());
```

Create Callbacks and Subscribe for Events:

The above user will get subscribed to the events of its Organization only.

```
// For JOB events
POMJobEventNotifierImpl jobCallback = new
POMJobEventNotifierImpl(logger);
pomEventRegistrar.subscribe(EventTypeEnum.JOB, jobCallback);

// For Agent Events
POMAgentEventNotifierImpl agentCallback = new
POMAgentEventNotifierImpl(logger);
pomEventRegistrar.subscribe(EventTypeEnum.AGENT, agentCallback);

// For JOB statistics
POMJobStatisticsEventNotifierImpl jobStatisticsCallback = new
POMJobStatisticsEventNotifierImpl(logger);
pomEventRegistrar.subscribe(EventTypeEnum.JOB_STATISTICS,
jobStatisticsCallback);

// For Agent Statistics
POMAgentStatisticsEventNotifierImpl agentStatisticsCallback = new
POMAgentStatisticsEventNotifierImpl(logger);
pomEventRegistrar.subscribe(EventTypeEnum.AGENT_STATISTICS,
agentStatisticsCallback);

// For Attempt events
POMAttemptEventNotifierImpl attemptResultCallback = new
POMAttemptEventNotifierImpl(logger);
pomEventRegistrar.subscribe(EventTypeEnum.ATTEMPT,
attemptResultCallback);

// For EnrichedAttemptResult events
POMEnrichedAttemptEventNotifierImpl enrichedAttemptResultCallback = new
POMEnrichedAttemptEventNotifierImpl(logger);
pomEventRegistrar.subscribe(EventTypeEnum.ENRICHED_ATTEMPT_RESULT,
enrichedAttemptResultCallback);

// For Import statistics
POMImportStatisticsEventNotifierImpl importStatisticsCallback = new
POMImportStatisticsEventNotifierImpl(logger);
pomEventRegistrar.subscribe(EventTypeEnum.IMPORT_STATISTICS,
importStatisticsCallback);
```

```
// For Heartbeat events
POMHeartBeatEventNotifierImpl heartBeatResultCallback = new
POMHeartBeatEventNotifierImpl(logger);
pomEventRegistrar.subscribe(EventTypeEnum.HEART_BEAT,
heartBeatResultCallback);

// For Inbound Skill events
POMInboundSkillEventNotifier pomInboundSkillEventNotifier = new
POMInboundSkillEventNotifierImpl(logger);
 pomEventRegistrar.subscribe(EventTypeEnum.INBOUND_SKILL,
pomInboundSkillEventNotifier);

//For CC Trend event
POMCCTrendEventNotifierImpl pomCCTrendEventNotifier = new
POMCCTrendEventNotifierImpl(logger);
     PomCCTrendEventNotifier.subscribe(EventTypeEnu.CCTRENDS,
pomCCTrendEventNotifier)
```

Initialize POMEventRegistrar:

```
pomEventRegistrar.initialize();
```

Refer Sample Client Application code as per Appendix B: Java sample application.

Exception Handling

A set of POMEventException are thrown by the Event SDK Client library, refer Appendix A: Avaya Proactive Outreach Manager Event SDK exceptions.

All the exceptions are self-explanatory and should be handled by client application at each of the following steps of EventSDK connection:

- Login to EventSDK – Exceptions related to user authorization, invalid credential and SSL configuration (KeyStore/TrustStore).
- Subscribe for events - Exceptions related to subscribing to Invalid Event type or connection already initialized before subscribing for events.
- Initialize the connection - Exceptions related connection failure with kafka server or initialized connection without login.

Certificate Management

You must use the POM Trusted Certificate Management web user interface page for importing EventSDK client certificate.

You can import a trusted certificate either from the pem certificate file or from the https URL.

Importing EventSDK Client Certificate into POM Truststore

1. Log in to the **Avaya Experience Portal** web console with the Administrator user role.
2. In the navigation pane, click **POM Home**.
3. Click **Configurations** > **POM Trusted Certificates**.

4. The system displays all the trusted certificates.
5. Click **import**, and do the following:
    a. Click **Browse** and locate the file on the local system.
    b. Click **Continue**.
6. To fetch the certificate, do the following:
    a. Click **Fetch**.
    b. Click **alias** and type the certificate URL with the https prefix.
    c. Click **Continue**.

Importing POM Certificate to EventSDK Client Truststore

1. Using the browser window, log in to the EPM as an administrator.
   Note: In case of multiple POM servers, that is, primary or auxiliary, log in to the primary EPM.
2. In the navigation plane, click **POM** > **POM Home**.
3. Click **Configurations** > **POM Servers**.
4. Click **Export** on the listed certificate tab and save it on your local system.
5. On the EventSDK client box, copy the exported POM certificate and import it to EventSDK client Truststore under <POM_EVENT_SDK_DIR>/conf.
6. Run the following command to import the certificate
   keytool -import -trustcacerts -alias pom_cert -file <Path to Certificate file> -keystore <Keystore_Path>
7. Set the truststore and keystore properties in <POM_EVENT_SDK_DIR>/conf/eventsdk.properties.

Note: If EventSDK client doesn't have its own TrustStore and KeyStore, POM TrustStore and KeyStore can be reused by copying them from POM server at $POM_HOME/config

Event Settings

Before you start the SDK client, following configuration should be in place:

1. Adding the POM server's host entry on local-host: On the Client machine.
    a. **Windows**: add the POM server's entry in "C:\Windows\System32\drivers\etc\hosts" file.
    b. **Linux**: add the POM server's entry in "\etc\hosts" file.
2. Event Settings:
    a. Login to Avaya Experience Portal with the user having administrator privilege.
    b. Navigate to POM -> POM Home -> Configurations -> Global Configurations
    c. Go to Event Settings and check following events:
        i. Send Job State Events
        ii. Send Agent State
        iii. Send Attempt Events

iv. Send Enriched Attempt Events

Notes:

i. Job Statistics and Agent Statistics Events are enabled by default. To disabled it, set SendJobStatisticsEvents and SendAgentStatisticsEvents flag to false in pim_config table in database.
ii. To enable the Enriched Attempt Events, enabling the Attempt Events is a pre-requisite.

d. You can also adjust the frequency of receiving Job Statistics Events by modifying Job Statistics Event Frequency (sec) field.
e. Save the changes. The following message would be displayed on top of the page
   i. The information entered has been saved.
   ii. You must restart the VPMS service on all POM servers to take effect.

f. Restarts the VPMS service by running the following command:
   #/sbin/service vpms restart

Logging

Setting appropriate log levels

Updating Kafka consumer (EventSDK Client) log level:

The Kafka consumer logs can be managed by setting properties in the following file:

<POM_EVENT_SDK_DIR>/conf/EventSDKLog4j.properties

```
# Kafka related logs
log4j.logger.kafka=ERROR,kafkaAppender
log4j.logger.org.apache.kafka=ERROR,kafkaAppender
log4j.logger.org.I0Itec.zkclient.ZkClient=ERROR,kafkaAppender
log4j.logger.org.apache.zookeeper=ERROR,kafkaAppender
```

- Log file : kafka-consumer.log
- Printing event meta data in logs:
- To enable event meta data printing in logs, following property needs to be set to true in <POM_EVENT_SDK_DIR>/conf/eventsdk.properties file:

  pom.print.event.payload=true

Localization

The Event Data is not localized in the current release.

Heartbeat Event

Heartbeat Events are periodic notification, generated by POM server. These events will be useful to monitor the communication between POM Kafka and EventSDK client. Java process named kafkamonitor will be running as a background process continuously while POM is up and running, to generate heartbeat events.

The frequency of heartbeat event generation is 30 seconds by default. The event frequency can be configured by updating config_value of parameter **HEARTBEAT_EVENT_FREQUENCY_IN_SECONDS** in pim_config table of POM database.

For Example: To set frequency of 60 seconds, execute below command in database.

update pim_config set config_value='60' where config_name='HEARTBEAT_EVENT_FREQUENCY_IN_SECONDS';

Frequency changes to take effect user will need to restart kafkamonitor service.

systemctl restart kafkamonitor

Minimum value allowed for **HEARTBEAT_EVENT_FREQUENCY_IN_SECONDS** is 10 seconds. If frequency set to less than 10 seconds in database, then kafkamonitor will use minimum allowed value, i.e. 10 seconds. Similarly, if frequency set to greater than 120 seconds in database, then kafkamonitor will use maximum allowed value, i.e. 120 seconds.

Each POM server will generate one heartbeat event within configured frequency.
For Example: If there are 3 POM server and **HEARTBEAT_EVENT_FREQUENCY_IN_SECONDS** is set to 30 seconds, then consumer will receive maximum 3 heartbeat events per 30 seconds.

However, in some cases, if client receive only one event generated from any POM server, it should be fine, since the main motive behind heartbeat event, is to monitor communication between Kafka Server and EventSDK client. If communication between server and client breaks due to any reason like network failure, then client will not receive any heartbeat event. In such scenario client can have their own mechanism to generate alarm or notification.

The implementation of raising alarm or notification is depends on, after how many consecutive heartbeat event losses, client want to raise alarm or notification.
For example: When **HEARTBEAT_EVENT_FREQUENCY_IN_SECONDS** is set to 30 seconds, few clients may want to raise alarm or notification, after 3 consecutive event loss, that means, if client doesn't receive any heartbeat event within 90 seconds (3 time window x 30 seconds frequency). However, few may want to raise immediately after first event loss that means after 30 seconds.

Common Attributes across Heartbeat Events

Following data will be passed as part of this event.

Attribute Name	Attribute Value
serverName	Name of the server

eventTimeStamp	Event creation time stamp. This timestamp is in UTC EPOCH format.
rcvdTimeStamp	Timestamp when the event has been received by the EventSDK client
eventTypeEnum	HEART_BEAT

Job Events

The SDK generates real-time events when changes occur in the state of job. The SDK delivers the events to callback objects created by the client application.

This section identifies the data structures used to deliver the events to the method for the callback object. Each job event contains a common event data and has its own unique data segment information.

Common Attributes across Job Events

Following information is sent with all Job events.

Attribute Name	Attribute Value
orgName	Organization to which the campaign belongs.
jobId	The job ID for the campaign
campaignName	The name of the campaign
campaignType	Campaign type: FINITE: Campaigns stop after processing all contacts or when the campaigns meet the specified finish criteria. INFINITE: Campaigns do not stop after processing the contacts associated with their contact-lists gets over.
agentSelectionType	• DEFAULT • ATTRIBUTE_BASED • AGENT_ID • ATTRIBUTE_AGENT_ID
jobEventinfo	It is a generic interface for all Job related events. This is implemented by JobStateEventInfo, the base class for Job state related event meta data. Further, the sub-classes represent the meta-data of specific job state event related meta data. These include: • JobRunningEventInfo • JobCallbackEventInfo • JobCompletedEventInfo Each Implementation has different properties and are listed in corresponding tables below.
eventType	Types of events triggered at different states of Job: • JOBQUEUED • JOBRUNNING • JOBPAUSED • JOBCALLBACK • JOBSTOP

	• JOBSTOP_CALLBACK • JOBRUNNING_FILTERINPROGRESS • JOBCOMPLETED
eventTimeStamp	Timestamp when the event has been received by the Kafka server. This timestamp is in UTC EPOCH format.

Specific properties based on jobEventinfo implementation

JobStateEventInfo

Attribute Name	Attribute Value
jobEventType	JobStateEvent - Supported job type event
jobStateEventType	Possible values are: • JobQueuedEvent, • JobRunningEvent, • JobPauseEvent, • JobCallbackEvent, • JobCompletedEvent, • JobCompletedCreatingHistoryEvent, • JobStopEvent,
currentJobStatus	<current status of job> e.g. *JOB_QUEUED*
lastJobStatus	<last status of the job> e.g. if the *currentJobStatus* is *JOB_QUEUED* , the last status can be *null*. Refer the Job States section to understand currentJobStatus and lastJobStatus values.

JobRunningEventInfo

Attribute Name	Attribute Value
filteredContactCount	Number of contacts identified for attempting based on the Job's filter criteria.
callbackCount	Number of scheduled callbacks for the Job.
pendingRetryCount	Number of pending retry attempts.

JobCallbackEventInfo

Attribute Name	Attribute Value
callbackCount	Number of Callbacks assigned to current job.

JobCompletedEventInfo

Attribute Name	Attribute Value
totalJobTime	Total time taken to complete the job.
jobEndTime	Job End time
finishReason	Possible values are: • NATURAL_FINISH : Job finished after completion of contacts attempting. • CAMPAIGN_ABORTED_MANUALLY: User has stopped the Job manually. • ABORT_TIME_CRITERIA_SATISFIED: Job terminated after it runs for a specific time COMPLETION_CODE_CRITERIA_SATISFIED: Job terminated based on existing completion codes. • GOAL_BASED_CRITERIA_SATISFIED: Job terminated based on the goal settings in campaigns e.g. when sum of (specific contact attribute) exceeds certain value.

Agent Events

The SDK generates real-time events when changes occur in the state of agents. The SDK delivers the events to callback objects created by the client application.
This section identifies the data structures used to deliver the events to the method for the callback object. Each agent event contains a common event data and unique data segment information.

Common Attributes across Agent Events

Following information is sent with all Agent events.

Attribute Name	Attribute Value
orgName	Organization name to which the Agent is associated
agentId	<Agent logged in ID>
agentName	<Agent's Name>
zone	The zone to which the agent is logged in
stationId	<Agent logged in station ID>
locale	The locale value as provided by agent during login. For ex. "en_US"
agentSessionId	The unique id assigned by POM on agent login
assignedskills	Map of all the Skills associated with the agent as the key and respective skill levels as the value.
skillAttribute	List of configured agent skill attributes.
agentEventType	A generic interface named *IAgentEventType*, its different implementations are present in each Agent Event, depending on the following Agent Event type. • AgentStateEvent • CallStateEvent • JobStateEvent • NailStateEvent • AgentOperationEvent Each Implementation has different properties and are listed in corresponding tables below. **Note**: **AgentOperationEvent** is generated in AACC Mode only, when agent skills are changed.
currentJobSkills	Outbound skill of the Campaign to which agent is associated
currentCampaignName	Name of the Campaign to which agent is associated
currentJobId	<Campaign Job ID> of the currentCampaignName.
currentActionId	<Action ID> of the currentActionName
currentActionName	Action Name within the Campaign under which agent is associated
eventTypeEnum	AGENT
eventTimeStamp	The event timestamp (in milliseconds) when the event is received by the Kafka server. This timestamp is in UTC EPOCH format.
pacingType	Pacing type when agent attach to a Job otherwise null.

	Possible values are: • **ECR** • **CruiseControl** • **Progressive** • **Preview** • **Manual**

Specific Properties Based On agentEventType Implementation

AgentStateEvent

Attribute Name	Attribute Value
state	• LOGGED_IN • READY • BUSY • WORK_NOT_READY • NOT_READY • LOGGED_OUT
logoutReasonCode	If the Agent is in LOGGED_OUT state, the value would be one of the following else it would be empty string. • DESKTOP_CRASHED • FORCE_LOGOUT • MANUAL_LOGOUT
notReadyReasonCode	The reason code as provided by agent while going to not ready state. Available for NOT_READY state only.
previousAgentState	The last state of the agent, i.e., <AgentStateEvent.state>.
timestamp	The time stamp of agent state when agent moved to that specific state.

AgentCallStateEvent

Attribute Name	Attribute Value
callState	IDLE PENDING_CALL ONCALL WRAPUP WRAPUP_COMPLETE PREVIEW DIALING CALL_HELD CONSULT_INITIATED CONSULT_ANSWERED CONFERENCED CONFERENCE_ENDED

	CONFERENCE_HELD CALL_TRANSFERRED
calltype	REGULAR REDIAL CONSULT CONFERENCE CALLBACK
pimsessionid	The ID assigned by POM for each attempt presented to the agent
contactId	The contact ID
userContactId	The contact's userCon tactId
contactListId	ID of the contact list
contactListName	Name of the contact list
userdata	List of contact attributes
ani	The number from which call was placed
dnis	The number to which call was placed
defaultaddress	The number to which call should be placed. Applicable in case of preview dial
completionCode	The completion code as provided by the agent during wrapup.
previousCallState	The previous call state, i.e., < AgentCallStateEvent.callState>
callDropBy	AGENT CUSTOMER Note: The callDropBy attribute will be empty in the following call scenarios: Agent cancels the preview call. Nailing is lost during the call. Consulted agents leaves the conference/consult call.
callUCID	The UCID of the call as provided by the platform.
consultInfo	The information related to consult call. Include agentId / external number to which call was placed.
conferenceInfo	The information related to conference call. Include agentId / external number to which call was placed.
timestamp	The time stamp of agent state when agent moved to that specific state.

AgentJobStateEvent

Attribute Name	Attribute Value
jobState	Possible values are: • **JOBATTACHED:** • **JOBDETACHED** • **JOBEND** • **JOBINBOUND**

	• **PENDING_INBOUND** • **PENDING_OUTBOUND** • **PENDING_MANUAL_INBOUND** • **PENDING_MANUAL_JOB_MOVEMENT** • **PENDING_JOB_ATTACH**
pacingType	Possible values are: • **ECR** • **CruiseControl** • **Progressive** • **Preview** • **Manual**
timestamp	The time stamp of agent state when agent moved to that specific state.

AgentNailStateEvent

Attribute Name	Attribute Value
nailState	**NAILED**: Agent has picked up the Nail-up call on its endpoint. **PENDING_NAILUP:** Agent has been associated with the job but Nail-up call is yet to be placed or picked up. **PENDING_NAILUP_DROP:** Agent is disassociated with the job but Nail-up call is yet to be dropped. **UNNAILED:** No Nail-up on agent endpoint as Agent is no more associated with the job. **RENAILING:** Nail-up call got dropped while Agent was associated with the job so Renailing the agent.
timestamp	The time stamp of agent state when agent moved to that specific state.

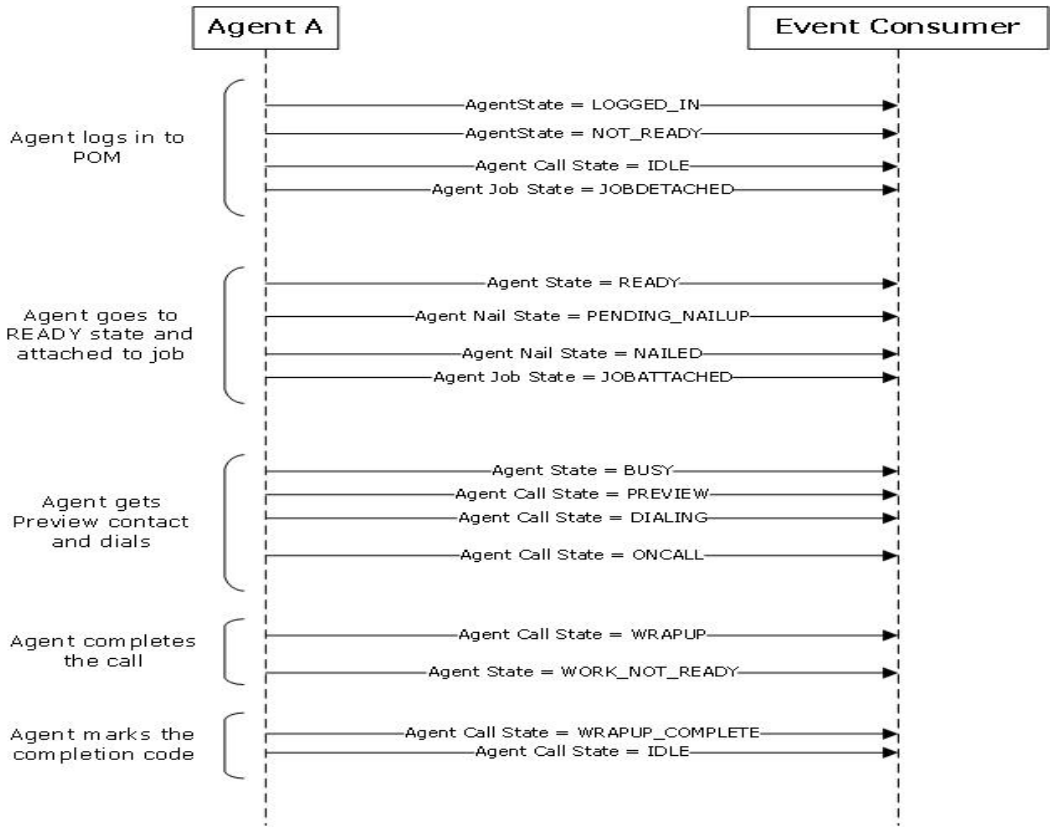
AgentOperationEvent

Attribute Name	Attribute Value
operation	**AgentSkillChanged**
timestamp	The time stamp of agent state when agent moved to that specific state.

Agent Event Sequence Diagrams

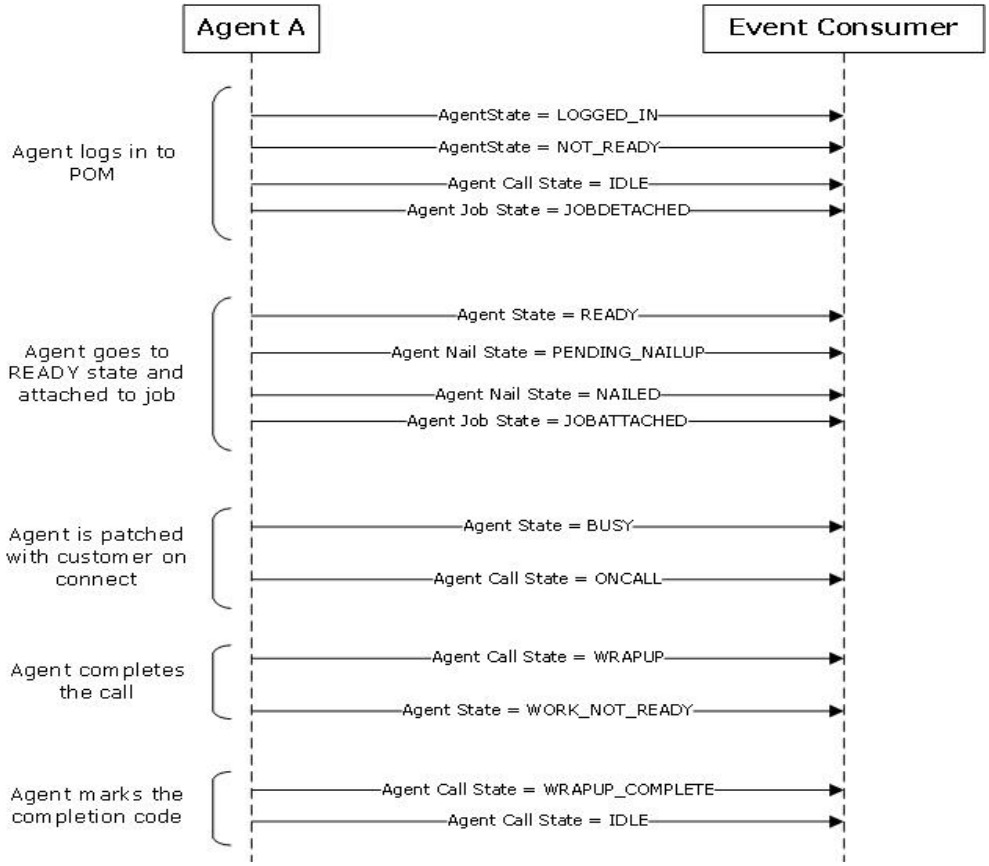
Below are the sequence diagram depicting the Agent Events generated during different POM call flows. During each operation one can see different Agent Events getting generated, i.e., Agent State Event/ Call State Event/Job State Event/Nail State Event.

Agent Event sequence – Preview /Manual

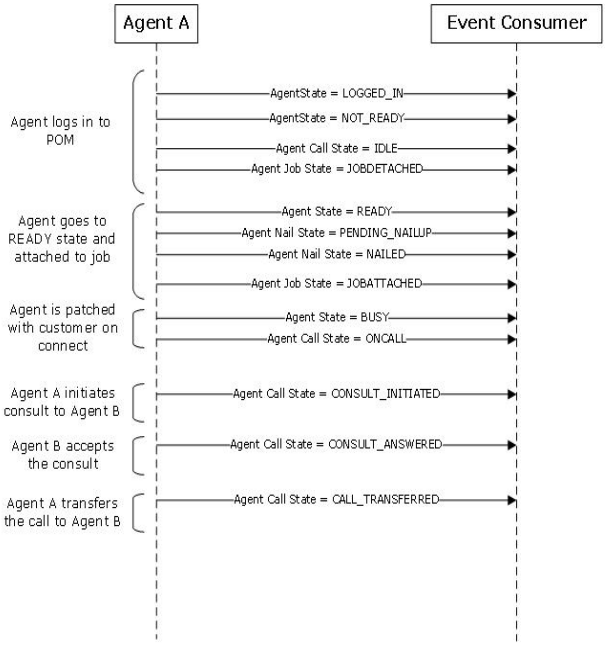


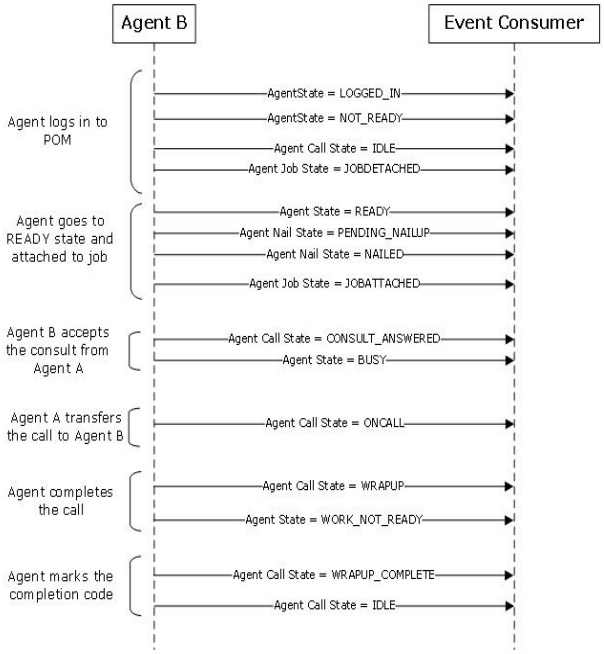
- The above sequence diagram is same for Preview and Manual Campaign.

Agent Event sequence – predictive/progressive

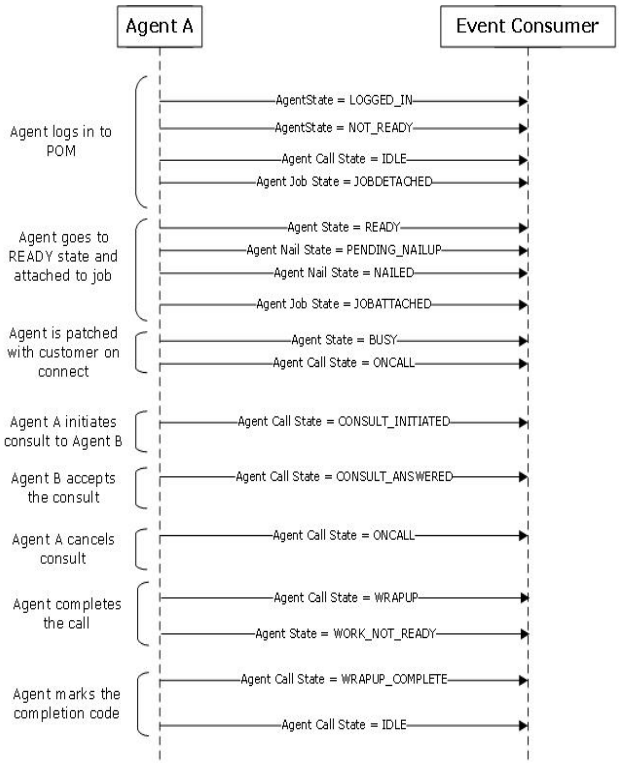


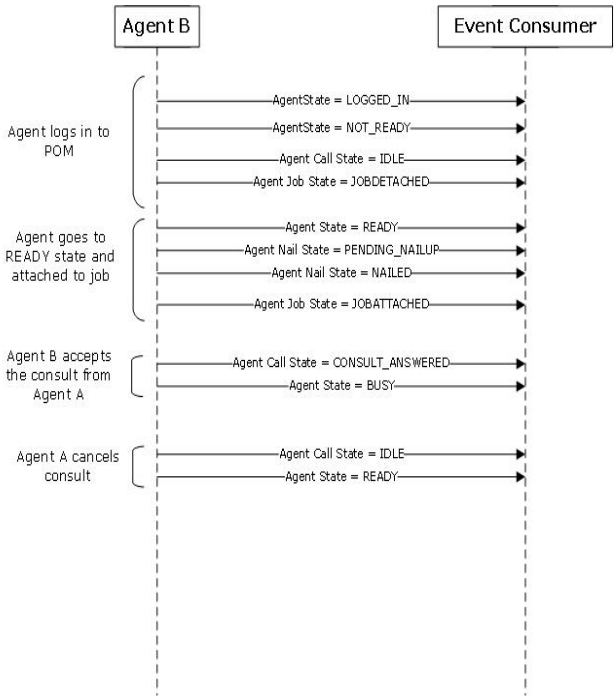
Agent Event sequence – consult transfer



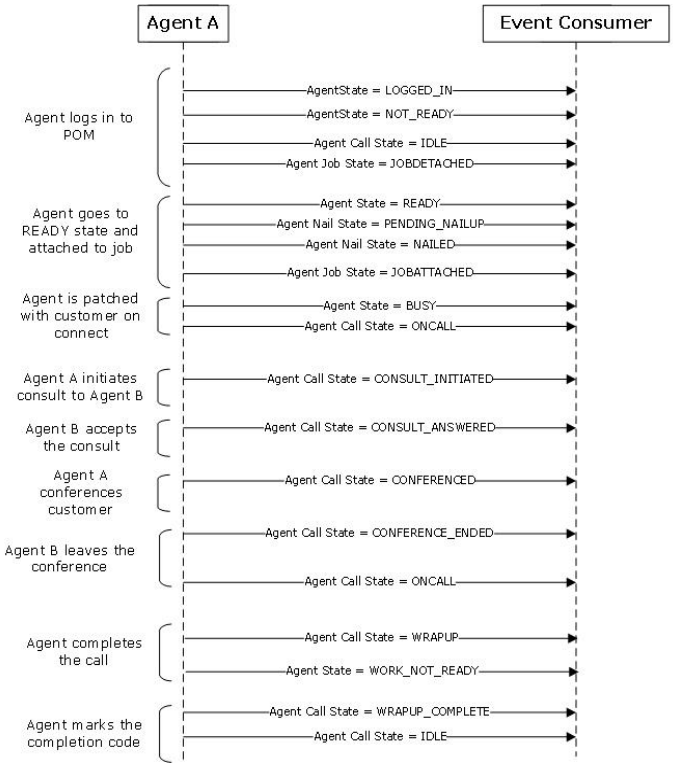


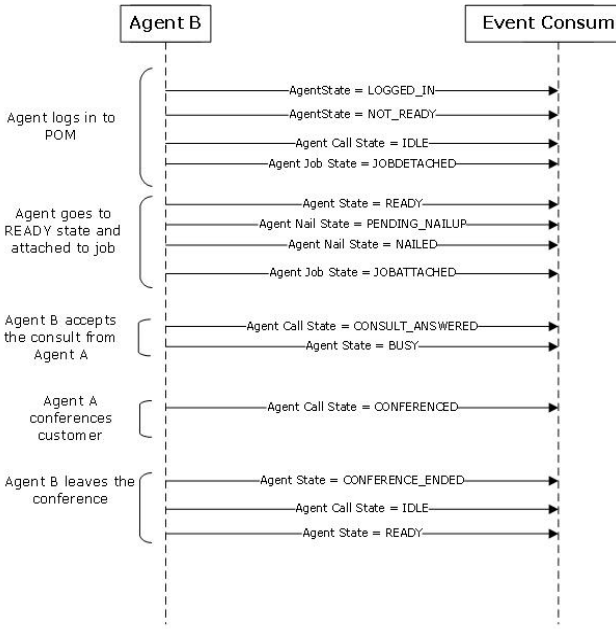
Agent Event sequence – consult



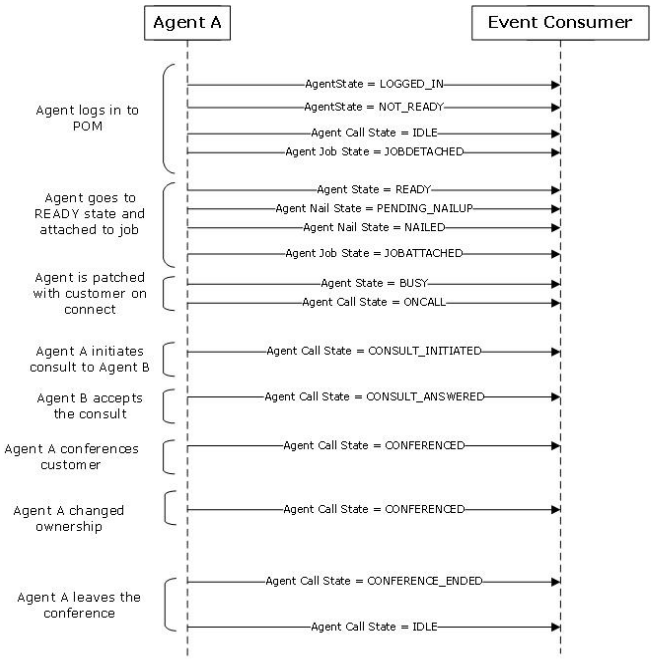


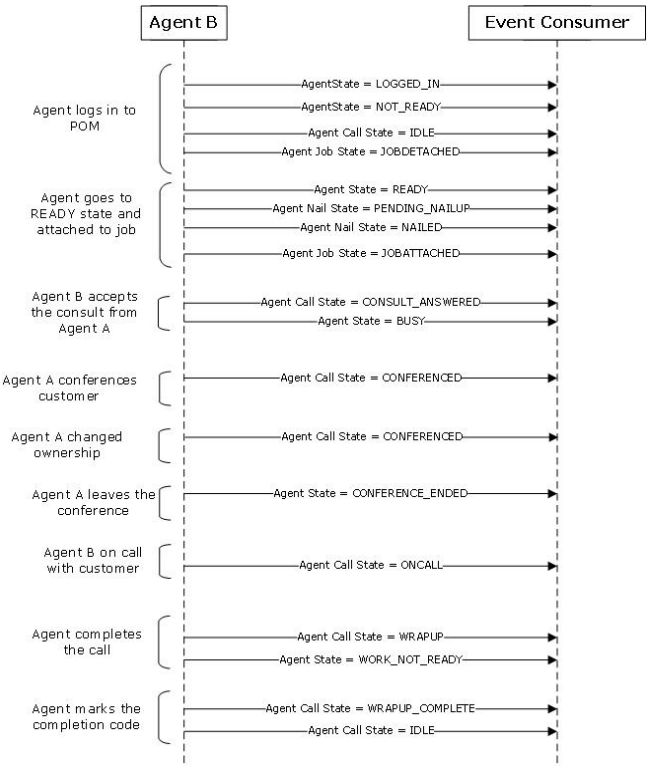
Agent Event sequence – conference



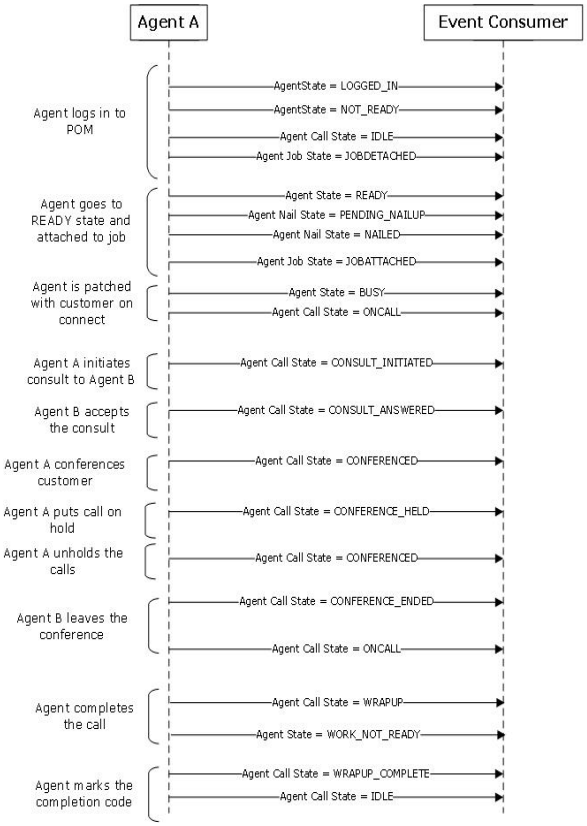


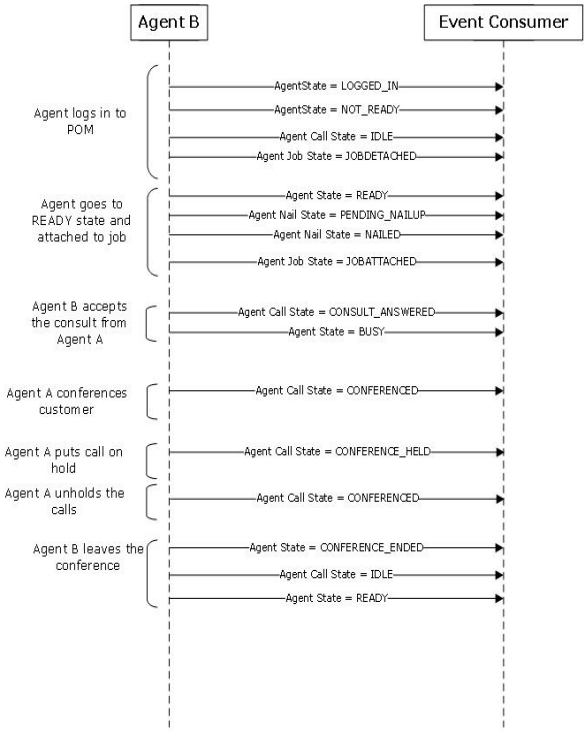
Agent Event sequence – conference ownership



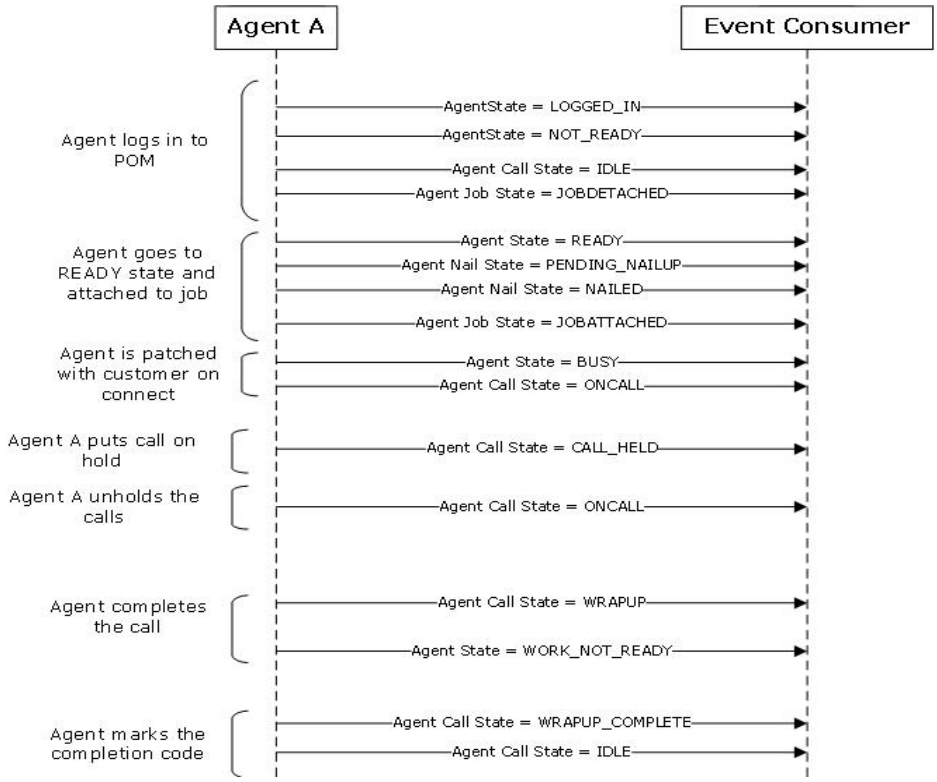


Agent Event sequence – conference held





Agent Event sequence – call held



Attempt Events

The Attempt Event gets generated when a new attempt gets created or updated. There can be multiple entities working in same attempt, hence, there will be multiple attempt events generated for a single attempt.

The event changing the contact state to ‘Done’ can be considered as the last event for this attempt event.

The presence of multiple events per attempt leads to the requirement of an Enriched Attempt, which is a consolidated event with the values updated by different entities/producers.

Attempt Event

Following data will be passed as part of this event.

Attribute Name	Attribute Value
pimSessionId	Auto generated ID of this contact attempt
systemCompletionCodeId	ID of the system completion code received for this contact attempt
systemCompletionCode	Name of the system completion code received for this contact attempt
campaignId	Id of the campaign
campaignName	Name of the Campaign
agentId	The agent Id as provided by agent during login
agentName	Name of the Agent
pacingType	It can be one of the following: **ECR** **CruiseControl** **CruiseControlW** **Progressive** **Custom** **Preview** **TimeBased** **SkillBased** **None** **Manual**
sysContactAttemptType	It can be one of the following: **REGULAR** **REDIAL** **CONSULT** **CONFERENCE** **CALLBACK**

channelType	The following channel types are available: **Voice** **Email** **SMS** **Custom**
contactListId	ID of the contact list
contactListName	Name of the contact list
sysContactId	system generated contact Id
userContactId	For Operational Use.
handlerId	This gives information about the Handler created in strategy
addressField	Address field name on which the call should be performed
Address	Value of the address
agentNotes	Any notes updated by the agent for this attempt
eventCreationTimestamp	Timestamp of event creation
parentPimSessionId	The session id of the parent attempt. Used in case of Redial
callDuration	Call duration of this attempt
contactAttemptTime	Timestamp of attempt creation
callbackInfo	It contains below information: type - type of callback(CAMPAIGN, STANDARD, AGENT) callbackId- id for the callback
eventTimeStamp	Timestamp when the event has been received by the Kafka server. This timestamp is in UTC EPOCH format.
eventTypeEnum	Attempt
eventProducerModule	The producer module for this event.
Subtype	It can be one of the following: **AttemptCreated** **AttemptUpdated** **AttemptCompleted**
Action	It can be one of the following: **CALLBACK_CREATED,** **AGENT_NOTES_ADDED,** **DNC_MARKED,** **AGENT_CALL_STATE_CHANGED,** **NUISSANCE_MARKED**
agentCallStateTimes	It contains below information type – It can be one of (PREVIEW_TIME, TALK_TIME, ACW_TIME)

Enriched Attempt Event

Following data will be passed as part of this event. There can be multiple entities or producers working in same attempt. The multiple events per attempt are consolidated into Enriched Attempt result event, which is final event with the values updated by different entities/producers. The final event is generated when contact is completed.

For real time addition of DNC by POM agent, it will contain separate flag. The callback creation detail are provided as well. The consumer could subscribe to Enriched Attempt Topic to read outcome or attempt result (completion code) from POM for every completed attempt.

Attribute Name	Attribute Value
eventTypeEnum	Attempt
pimSessionId	Auto generated ID of this contact attempt
sysContactId	system generated contact Id
userContactId	The User Id mapped to the provided Id field
campaignId	Id of the campaign
campaignName	Name of the Campaign
contactListId	ID of the contact list
contactListName	Name of the contact list
systemCompletionCodeId	ID of the system completion code received for this contact attempt
systemCompletionCode	Name of the system completion code received for this contact attempt
agentCompletionCodeId	ID of the custom completion code received for this contact attempt
agentCompletionCode	Name of the custom completion code received for this contact attempt
agentId	The agent Id as provided by agent during login
agentName	Name of the Agent
pacingType	It can be one of the following: ECR CruiseControl Progressive Custom Preview TimeBased SkillBased None Manual
sysContactAttemptType	It can be one of the following: REGULAR REDIAL CONSULT CONFERENCE CALLBACK
channelType	The following channel types are available:

	Voice Email SMS Custom
addressField	Address field name on which the call should be performed e.g. phoneNumber1
Address	Value of the address e.g. the contact number on which the attempt would be dialed
handlerId	This gives information about the Handler created in strategy
agentNotes	Any notes updated by the agent for this attempt
isDNC	True if the current is marked as DNC
markedDNC	True, if the address is marked DNC during this attempt
isNuisance	The Nuisance status for this attempt
parentPimSessionId	The session id of the parent attempt. Used in case of Redial
callDuration	Call duration of this attempt. It represent the time between Call-connect and Call-complete.
agentCallTimings	It has following information: talkTime acwTime previewTime
contactAttemptTime	Timestamp of attempt creation
processedTimeStamp	Timestamp when the event has been processed by Stream Processor
callbackInfo	It contains below information: type - type of callback(CAMPAIGN, STANDARD, AGENT) callbackId - id for the callback
eventTimeStamp	Timestamp when the event has been received by the Kafka server. This timestamp is in UTC EPOCH format.
Awaiting	It can be one of the following: AWAITING_CALLBACK AWAITING_REDIAL AWAITING_RETRY

Note : To receive Enriched Attempt Event, POM Dashboard service should be up and running.

Real-time statistics

Real-time statistics are organized into following categories:

- Job Statistics
- Agent Statistics
- Import Statistics

This section describes the data structures that the SDK uses to deliver statistical information to client applications.

Job Statistics Event

Job information exists from the time the job was started. For Infinite campaign such information exists for last 3 days. Campaign Director sends the job statistics event after every 6 seconds. Event name will be “JobStatNotify “.

JobStatisticsEvent

Following data will be passed as part of this event.

Attribute Name	Attribute Value
jobId	The Id of the campaign job
campaignName	The name of the campaign
campaignStrategyName	The name of the campaign strategy assigned to this campaign.
campaignType	Campaign type: • FINITE: Campaigns stop after processing all contacts or when the campaigns meet the specified finish criteria • INFINITE: Campaigns does not stop after processing the contacts associated with it
orgName	Organization to which the campaign belongs.
contactListNames	Contact list names assigned to this campaign
dncGroupList	DNC group list names assigned to this campaign
currentJobStatus	<current status of job> e.g. *JOB_ACTIVE* Refer the Job States section for all the possible Job State values.
jobStartTime	The time when the job was started (in milliseconds)
estimatedJobEndTime	The estimated time at which the job will end (in milliseconds)
filteredContactCount	The total number of records filtered for attempting.
processedContactCount	The number of records which were completed (marked as “done”)
unAttemptedContactCount	The number of records yet to be dialed.
contactsNotDoneCount	The number of records not yet done.
connectCount	Total number of contacts connected in the job.

	Connected contacts are the done contact with one of the System Completion Code from Call Answered, Call Answered (Human),Email Delivered or SMS Delivered.
restrictTotalAttemptCount	Total number of temporary restricted attempts.
restrictContactsByHoliday	Number of temporary restricted contacts due to holiday.
restrictContacts	Total number of temporary restricted contacts.
currentCallbackCount	The number of callbacks currently attached to job and yet to be handled
completedCallbackCount	The number of callbacks completed attached to this job.
pendingRetryCount	Number of pending retry attempts.
totalAttemptCount	Total number of attempts done in this job.
rpcCount	Total number of Right Party Connect count in this job. Right Party Connect count is the total number of custom completion code, with rpc flag true, marked.
successCount	Total number of success count in this job. Success count is the total number of custom completion code, with success flag true, marked.
closureCount	Total number of closure count in this job. Closure count is the total number of custom completion code, with closure flag true, marked.
percentageComplete	Percentage of completed records.
percentageUniqueAttemptedRecord	Percentage of records/contacts at least attempted once in a campaign out of total records/contacts in the campaign.
percentageAttemptedRecord	Percentage of all attempts made (incl. callbacks, retries, etc.) out of total records/contacts in the campaign.
nuisanceRate	The Nuisance call rate
nuisanceRateToday	The percentage of Nuisance calls for the calendar day.
nuisanceCount	Total nuisance call count
agentSelectionType	• DEFAULT • ATTRIBUTE_BASED • AGENT_ID • ATTRIBUTE_AGENT_ID
agentCount	Number of agents linked to this Job.

jobNotes	Possible values: • NATURAL_FINISH • CAMPAIGN_ABORTED_MANUALLY • ABORT_TIME_CRITERIA_SATISFIED • COMPLETION_CODE_CRITERIA_SATISFIED • GOAL_BASED_CRITERIA_SATISFIED • DIALING_PAUSED_AWAITING_IMPORT_TO_COMPLETE • DIALING_PAUSED_AWAITING_RECORD_FILTERING_TO_COMPLETE • DIALING_PAUSED_CONTACT_LIST_ADDITION_IN_PROGRESS • DIALING_PAUSED_CONTACT_LIST_REMOVAL_IN_PROGRESS • DIALING_PAUSED_CONTACT_LIST_IMPORT_IN_PROGRESS • RECORD_FILTERING_AND_DIALING_IN_PROGRESS • CONTACT_LIST_ADDITION_AND_DIALING_IN_PROGRESS • CONTACT_LIST_REMOVAL_AND_DIALING_IN_PROGRESS • CONTACT_LIST_IMPORT_AND_DIALING_IN_PROGRESS • DIALING_IN_PROGRESS • DIALING_STOPPED_ONLY_CALLBACK_TO_PROCESS • DIALING_COMPLETED_PENDING_CALLBACKS
contactListInfoList	List of ContactListInfo object containing contact-list specific properties. Refer ContactListInfo attributes table for complete listing of properties.
handlerInfoList	List of **JobHandlerInfo** object containing Job Handler specific properties for each handle associated with the job. Refer JobHandlerInfo attributes table for complete listing of properties
vpmsName	The EPM that will be used for SMS/Email/Voice
linkedCampaignName	Campaign name linked to running campaign
eventType	JOB_STATISTICS
eventTimeStamp	The event timestamp (in milliseconds) when the event is received by the Kafka server. This timestamp is in UTC EPOCH format.
campaignId	Unique Id for the campaign
linkedCampaignId	Id of the campaign which is linked to the running campaign
elapsedTime	Time for which the job is running. There is no end time for the infinite campaign.
falseNegativeCount	Answer Machine By Agent count for the job.

ContactListInfo

Attribute Name	Attribute Value
contactListName	Name of the contact list
contactListId	ID of the contact list
ftName	Filter Template Name associated with contact list
dialAlloc	Dialing association for contact list
fltrCnt	The total number of records filtered for attempting from this contact list.
procCnt	The number of records which were completed (marked as “done”) from this contact list
connCnt	Number of contacts connected from this contact list.
retryCnt	Number of pending retry attempts from this contact list
tmpRestCnt	Total number of temporary restricted contacts from this contact list.
tmpRestByHldCnt	Number of temporary restricted contacts due to holiday from this contact list.
tmpRestAtmpCnt	Total number of temporary restricted attempts from this contact list.
unAtmpCnt	Number of un-attempted contacts from this contact list
atmpCnt	Number of attempts from this contact list
uniqAtmpCnt	Number of unique attempts from this contact list
curtCbkCnt	The number of callbacks currently attached to job and yet to be handled from this contact list.
cmplCbkCnt	The number of completed callbacks currently attached to job from this contact list.
Rpc	rpc count from this contact list
Success	successs count from this contact list
Closure	closure count from this contact list
nseCnt	Nuisance count from this contact list
flsNgtCnt	False negative count from this contact list
percentageAttemptedRecord	Percentage of the total attempted records from this contact list.
percentageUniqueAttemptedRecord	Percentage of uniquely attempted records from this contact list.

notDoneCnt	Displays the count of contacts for all contact lists associated with the campaign, which are not dialed. This total can be: • Total Contacts • Processed Contacts • Un-attempted Contacts

JobHandlerInfo

Attribute Name	Attribute Value
handlerName	Name of the handler as set in the display name in the campaign strategy.
handlerState	State can have any name except, done and wait. When the campaign job starts, all records are in initial state. When the processing is over, the records move to done state. When the record reaches done state, it moves out of the campaign. You have to name one of your handlers as initial. You cannot add a handler for wait and done state. If a contact record is in wait state, it remains in that state until some call disposition moves the record to next state.
channelType	The following channel types are available: • Voice • Email • SMS • Custom
pacingParameter	The properties differ depending on the assigned pacing type of the job. Depending on the pacing, PacingParameter Object of the given pacing type is created and assigned to this property. The types are: • ECR • CruiseControl • Progressive • Custom • Preview • TimeBased • SkillBased • None • Manual The parameters based on pacing type can be found in PacingParameter table.
agentJobHandlerInfo	This provides assigned agents statistics for this job. Refer AgentJobHandlerInfo attributes for details.
actionId	actionId
attemptChannelCount	The Map containing each channelType within the handler as a key and the count of its attempt as value.

completionCodeMap	Map of the completion codes associated with this Job. Refer the CompletionCodeInfo attributes table for details.
zoneId	zoneId
Skill	Skill associated with the campaign strategy of this job.
currentCallbackCount	The number of callbacks currently attached to this job handler and yet to be handled
completedCallbackCount	The number of callbacks completed attached to this job handler.
connectCount	Total number of contacts connected in this job handler.
droppedCallCount	Total number dropped call in this job handler. Dropped call are those calls, which are marked with one of the following System Completion Code: **Disconnected By System-CCA** **Disconnected By User-CCA** **Disconnected By System-NuisanceApp** **Disconnected By User-NuisanceApp** *Note: The above configured System Completion Code for drop call, can be configured or change by updating **CompletionIDsDropCall** flag in pim_config table of POM database. The config_value for flag **CompletionIDsDropCall**, are comma separated System Completion Code Id. For the change to take effect Campaign Manager restart is required.
nuisanceCount	Nuisance call count in this job handler.
actionName	Action Name

licenseInfoList	List of LicenseInfo object containing License specific properties for each handler associated with the job.
	The properties differ depending on the assigned pacing. Depending on the pacing, LicenseInfo Object of the given pacing type is created and assigned to this property.
	• Manual
	  ○ ManualLicenseInfo
	• Preview
	  ○ PreviewLicenseInfo
	• ECR
	• CruiseControl
	• CruiseControlW
	• Progressive
	  ○ AgentBasedLicense
	• Custom
	• None
	• Timebased
	• SkillBased

	○ AgentLessLicense
inUsePortCount	Number of ports used by this job handler
callQueue	This provides calls in queue statistics for this job. Refer CallQueue attributes for details.

CallQueue

Attribute Name	Attribute Value
activeCnt	Displays the total number of calls that are currently in the queue for the campaign job.
totalCnt	Displays the total calls in queue count since the campaign job started.
totalTime	Displays the total time for which the calls are in the queue.

LicenseInfo

Attribute Name	Attribute Value
zoneId	Use to specify the zone Id.
zoneName	Displays the name of the zone to which the campaign belongs
allocationType	Displays the license allocation type. The following are the license allocation types: • Reserved: POM does not release a reserved license from a campaign job or task even if the campaign job or task does not need the license. The license is released and reallocated to other campaign job or task when a license recalculation event is triggered. • Dynamic: POM dynamically releases and reallocates the licenses to other campaign jobs or tasks as per the priorities.

AgentBasedLicense

Attribute Name	Attribute Value
agtWebAPILicCount	Displays the number of agent web API licenses that are currently used to run the corresponding task.
extSelLicCnt	Displays the number of external selection feature licenses that are currently used to run the corresponding task.
predLicCnt	Displays the number of predictive licenses that are currently used to run the corresponding task.

PreviewLicenseInfo

Attribute Name	Attribute Value
agtWebAPILicCount	Displays the number of agent web API licenses that are currently used to run the corresponding task.
extSelLicCnt	Displays the number of external selection feature licenses that are currently used to run the corresponding task.
predLicCnt	Displays the number of predictive licenses that are currently used to run the corresponding task.
prevLicCnt	Displays the number of preview licenses that are currently used to run the corresponding task.

ManualLicenseInfo

Attribute Name	Attribute Value
agtWebAPILicCount	Displays the number of agent web API licenses that are currently used to run the corresponding task.
extSelLicCnt	Displays the number of external selection feature licenses that are currently used to run the corresponding task.
predLicCnt	Displays the number of predictive licenses that are currently used to run the corresponding task.
prevLicCnt	Displays the number of preview licenses that are currently used to run the corresponding task.

manualLicCnt	Displays the number of manual licenses that are currently used to run the corresponding task.

AgentLessLicense

Attribute Name	Attribute Value
ports	Displays the number of ports that are currently used to run the corresponding task.

PacingParameter

Attribute Name	Attribute Value
pacingType	Use to specify the pacing type for the given task. This can be any of the following types: • ECR • CruiseControl • CruiseControlW • Progressive • Custom • Preview • TimeBased • SkillBased • None • Manual Depending upon the type, other properties can vary.

CruiseControlPacingParameterInfo

Attribute Name	Attribute Value

desiredServiceLevel	Specifies a service level against which the POM system compares the realized service levels to determine the magnitude and direction of service level errors.
currentServiceLevel	Current service level.
minAgent	Specifies the minimum number of agents required for a task.
maxAgent	Specifies the maximum number of agents required for a task.
priority	The priority of the task by specifying a number between 1 to 10. The higher the number, the higher the priority of the task. For example, a task with priority assigned as 7 gets more ports as compared to the task with priority 3.
acwTime	Specify the amount of time, the agent gets after the call, to wrap up the call with notes or other with other related work.
numOfACWExtns	Specifies the number of After Call Work (ACW) extensions required for an agent.
defaultCompCode	Specifies the completion code and assign the completion code to the call, if the agent does not provide the completion code.

PreviewPacingParameterInfo

Attribute Name	Attribute Value
timedPreview	It will be set to Yes if time-based preview is enabled for preview campaign.
canCancel	It will be set to Yes if cancel preview is enabled for preview campaign
minAgent	Minimum number of agents configured for an action.
maxAgent	Maximum number of agents configured for an action.
Priority	The configured priority of the task, possible number between 1 to 10. The higher the number, the higher the priority of the task. For example, a task with priority assigned as 7 gets more ports as compared to the task with priority 3.
acwTime	Amount of time, the agent gets after the call, to wrap up the call with notes or other with other related work.
numOfACWExtns	The number of After Call Work (ACW) extensions required for an agent.
defaultCompCode	The configured default completion code, if the agent does not provide/select any completion code this configured completion code will be used to wrapup the call.

ManualPacingParameterInfo

Attribute Name	Attribute Value
timedPreview	It will be set to Yes if time-based preview is enabled for manual campaign.
canCancel	It will be set to Yes if cancel preview is enabled for manual campaign

minAgent	Minimum number of agents configured for an action.
maxAgent	Maximum number of agents configured for an action.
Priority	The configured priority of the task, possible number between 1 to 10. The higher the number, the higher the priority of the task. For example, a task with priority assigned as 7 gets more ports as compared to the task with priority 3.
acwTime	Amount of time, the agent gets after the call, to wrap up the call with notes or other with other related work.
numOfACWExtns	The number of After Call Work (ACW) extensions required for an agent.
defaultCompCode	The configured default completion code, if the agent does not provide/select any completion code this configured completion code will be used to wrapup the call.

CustomPacingParameterInfo

Attribute Name	Attribute Value
maxPorts	Used to specify the maximum number of POM ports for the given task.
minPorts	Used to specify the minimum number of POM ports for the given task
Priority	The priority of the task by specifying a number between 1 to 10. The higher the number, the higher the priority of the task. For example, a task with priority assigned as 7 gets more ports as compared to the task with priority 3.

ECRPacingParameterInfo

Attribute Name	Attribute Value
percentHandleTime	The actual call time and the wrapping up time required for the agent to end the call.
percentUpdateTime	The time agents take to update the records.
minHitRate	The minimum hit rate, a number, determines the maximum number of calls to place to make an agent connection. The minimum and the maximum values are 1 and 100 respectively in the increments of 10. The default value is 30
initialHitRate	The initial hit rate, a number, determines the average number of calls per agent that the POM system makes during the first 5 minutes of the campaign job. The initial hit rate is the number of call completions compared with call attempts. If you set the rate too low, at 20 to 30, the dialer can make more connects than your agents can handle during the initial dialing period. If you set the rate too high, over 70, the POM system can fail to make enough connections to keep your agents busy.
ecrProbType	There are two types of ecrProbType: • Agent Update Time

	• Agent Work Time
minAgent	Specifies the minimum number of agents required for a task.
maxAgent	Specifies the maximum number of agents required for a task.
Priority	The priority of the task by specifying a number between 1 to 10. The higher the number, the higher the priority of the task. For example, a task with priority assigned as 7 gets more ports as compared to the task with priority 3.
acwTime	Specify the amount of time, the agent gets after the call, to wrap up the call with notes or other with other related work.
numOfACWExtns	Specifies the number of After Call Work (ACW) extensions required for an agent.
defaultCompCode	Specifies the completion code and assign the completion code to the call, if the agent does not provide the completion code.

ProgressivePacingParameterInfo

Attribute Name	Attribute Value
overdialRatio	Used to specify the dialing ratio with respect to the number of available agents. For example if you set the over dial ratio as 1, POM dials 1 call for 1 agent. You can also specify a float value like 1.2. If you specify the over dial ratio as 1.2, then POM dials 12 calls for 10 agents. The higher the over dial ratio, the chances of nuisance calls are also increased depending on the hit rate. You can specify any number between 1.0 and 100.
minAgent	Specifies the minimum number of agents required for a task.
maxAgent	Specifies the maximum number of agents required for a task.
Priority	The priority of the task by specifying a number between 1 to 10. The higher the number, the higher the priority of the task. For example, a task with priority assigned as 7 gets more ports as compared to the task with priority 3.
acwTime	Specify the amount of time, the agent gets after the call, to wrap up the call with notes or other with other related work.
numOfACWExtns	Specifies the number of After Call Work (ACW) extensions required for an agent.
defaultCompCode	Specifies the completion code and assign the completion code to the call, if the agent does not provide the completion code.

SkillBasedPacingParameterInfo

Attribute Name	Attribute Value
channelType	Type of the channel.
initialPace	Use to specify the initial value of the number of call attempts in the pacing interval
callPacingInterval	Use to specify the time unit for the call pacing. You can specify the value in either seconds, minutes, or hour.
inbSkillToMonitorName	Displays the inbound skill you select to monitor for the campaign.
InboundPacingEnum	• QueueLength • AvgSpeedAnswer • ExpectedWaitTime • ServiceLevel
desiredASA	Desired value of the average time the agent requires to finish the call. POM monitors the average speed for an inbound skill which you want to monitor.
desiredEWT	Desired value of the expected time for which the call waits in the queue before the call is answered.
EWTLevels	EWT level used for pacing: • high(default) • med • low
desiredQLength	Desired value of the number of calls in the wait queue for the inbound skill.
desiredSL	Desired value of the number of calls answered within the specified time in the inbound queue.
pacingInterval	The time unit for the call pacing. This can be specified in seconds, minutes or hours.
maxPace	Use to specify the maximum pacing value for the task. You can specify a value between 1 to 500000. POM never exceeds this pace.
maxPorts	Used to specify the maximum number of POM ports for the given task.
minPorts	Used to specify the minimum number of POM ports for the given task.
Priority	The priority of the task by specifying a number between 1 to 10. The higher the number, the higher the priority of the task. For example, a task with priority assigned as 7 gets more ports as compared to the task with priority 3.
inboundSkillParametersMap	Map of Inbound Skill Parameter as Key and its value. Refer InboundSkillParametersMap table for complete listing of parameter.

TimeBasedPacingParameterInfo

Attribute Name	Attribute Value
channelType	Type of channel
Pace	Specify a value between 1 to 1000000. This is rate of calls. For example if this value is 10 and unit is minute, POM tries to make 10 call attempts in 1 minute.
pacingInterval	Select the unit for the rate specified with **Call Pace**. The available values are 'Second', 'Minute' and 'Hour'.
maxPorts	Used to specify the maximum number of POM ports for the given task.
minPorts	Used to specify the minimum number of POM ports for the given task
Priority	The priority of the task by specifying a number between 1 to 10. The higher the number, the higher the priority of the task. For example, a task with priority assigned as 7 gets more ports as compared to the task with priority 3.

NonePacingParameterInfo

Attribute Name	Attribute Value
channelType	Type of channel
maxPorts	Used to specify the maximum number of POM ports for the given task.
minPorts	Used to specify the minimum number of POM ports for the given task
Priority	The priority of the task by specifying a number between 1 to 10. The higher the number, the higher the priority of the task. For example, a task with priority assigned as 7 gets more ports as compared to the task with priority 3.

InboundSkillParametersMap

Parameter Name	Parameter Value
Asa	Displays the average time in which the agent answers the call.
Sl	Displays the percentage of calls the agents answers to achieve the specified service level.

qLength	Displays the number of calls in the wait queue for the inbound skills.
ewtHigh	Displays the high expected time for which the call waits in the queue before the call is answered.
ewtMed	Displays the medium expected time for which the call waits in the queue before the call is answered.
ewtLow	Displays the low expected time for which the call waits in the queue before the call is answered.

AgentJobHandlerInfo

Attribute Name	Attribute Value
totalAgentCount	Number of agents associated with this Job.
totalTalkTime	The total talk time of the agents in this job.
avgTalkTime	The average talk time of the agents in this job.
totalAcwTime	The total after call work time of the agents in this job.
avgAcwTime	The average after call work of the agents in this job.
totalInJobIdleTime	The total in-job idle time of the agents in this job.
totalInJobIdleCount	The total in-job idle count of the agents in this job.
avgIdleTime	The average in-job idle time of the agents in this job.
totalInJobBreakTime	The total in-job break time of the agents in this job.
totalInJobBreakCount	The total in-job break count of the agents in this job.
avgBreakTime	The average in-job break time of the agents in this job.
totalHoldTime	The total hold time of the agents in this job.
totalHoldCount	The total hold count of the agents in this job.
avgHoldTime	The average hold time of the agents in this job.
totalTransferCount	The total call transfers done by the agents in this job.
totalTransferRecvCount	The total number of call transfers received or accepted by agent in this job.
totalPreviewTime	The total preview time of the agents in this job.
totalPreviewAcceptCount	The total dialed contact in preview by agents in this job.

totalPreviewRejectCount	The total canceled contact in preview by agents in this job.
avgPreviewTime	The average preview time of the agents in this job.
totalAbandonedHoldCount	The total number of calls handled by agent getting disconnected when customer is on hold in this job.
totalHoldInConfCount	The total number of calls agent has put customer on hold during conference in this job.
totalHoldInConfTime	The total time agent has put customer on hold during conference in this job.
callCount	The total call count of the agents in this job.
percIdleTime	The percentage of in-job idle time of the agents in this job.
percBreakTime	The percentage of in-job break time of the agents in this job.
agentUtilization	The agent utilization for this job.
achievedServiceLevel	The agent current service level for this job.
agentReadyCount	Number of Agents in Ready state in this job.
agentOnCallCount	Number of Agents in call in this job.
agentOnBreakCount	Number of Agents in break in this job.
zoneId	The id of the zone to which agent is associated.
totalJobAttachedDuration	Time for which Agent is attached to current job.

CompletionCodeInfo

Attribute Name	Attribute Value
Name	Completion code description for System completion code and completion code name for Custom completion code.
Count	Number of completion codes.
compCodeId	The database id of this completion code.
Rpc	Right Party Connect flag for completion code. One should use the flag only for custom completion codes.
Success	Success flag for completion code. One should use the flag only for custom completion codes.
Closure	Closure flag for completion code. One should use the flag only for custom completion codes.
codeName	Name of the completion code.

Agent Statistics Event

Agent statistical information exists from the time an agent logs on to POM until agent logs off. Agent Manager sends the agent statistics after every 6 seconds. Event name will be “AgentStatNotify “.

Following data will be passed as part of this event.

Attribute Name	Attribute Value
agentOrg	The organization to which agent is logged in
Agentid	The agent Id as provided by agent during login
Agentname	The agent name as provided by agent during login
assignedSkills	The skills which are configured in CC Elite and mapped with POM skills.
skillAttribute	List of configured agent skill attributes.
zoneName	The zone to which agent is logged in
agentExtension	The station Id as provided by agent during login
Locale	The locale value as provided by agent during login. For ex. “en_US”
agentSessionId	The unique id assigned by POM on agent login
timeZone	The time zone as provided by agent during login.
loginTimeStamp	The timestamp of agent login
logoutTimeStamp	The timestamp of agent logout
totalInboundBlendCount	Total number of times agent was blended to inbound within a login session
totalInboundDuration	Total time agent spent in inbound within a login session
totalOutboundDuration	Total time agent spent in outbound within a login session
totalOffJobIdleDuration	Total time agent was in READY state and not attached to job

totalOffJobBreakDuration	Total time agent was in NOT_READY state and not attached to job
lastRelToInboundTimeStamp	Timestamp when the agent was last released to inbound
lastAcqToOutboundTimeStamp	Timestamp when the agent was last acquired to outbound
currentAgentState	The current state of agent like Ready, Not_ready, Busy, Work_Not_Ready
currentAgentCallState	The current call state if handled by the agent like Idle, OnCall, WrapUp, WrapUpComplete
currentAgentJobState	The current job state for the agent like JobAttached, JobDetached
currentAgentNailState	The current nail state for the agent like Nailed, Unnailed
currentAgentStateTimeStamp	The time stamp when agent state changed to current state
currentAgentCallStateTimeStamp	The time stamp when agent call state changed to current call state
currentAgentNailStateTimeStamp	The time stamp when agent nail state changed to current nail state
currentAgentJobStateTimeStamp	The time stamp when agent job state changed to current job state
lastJobAttachTimeStamp	The time stamp when the agent was last attached to the job
lastJobDetachTimeStamp	The time stamp when the agent was last detached from the job
loggedInDuration	Total time duration for which agent is in logged In state.
allJobComplnCodeInfo	Map of completion codes IDs as Key and **CompletionCode** object as value. Refer CompletionCode attributes table for details. **Note:** This Map contains cumulative Completion Code information for all the jobs that Agent has handled within current login session.
allJobInfo	**AgentAllJobStatistics** class holding cumulative data for all the jobs that Agent has handled within current login session. Refer the AgentAllJobStatistics attributes table for details. **Note:** This attribute is populated only when agent is detached from the job and is not calculated periodically. Also, POM doesn't retain any data from previous login sessions of the agent.
currentAttachJobHandlerInfo	**AgentCurrentJobHandlerInfo** class holding current attached job data. Refer the AgentCurrentJobHandlerInfo attributes table for details. **Note:** This attribute is populated only when agent is detached from the job and is not calculated periodically.
isConferenceOwner	Specifies if Agent is owner of the Conference Call.

AgentCurrentJobHandlerInfo

Attribute Name	Attribute Value
jobId	The Id of the current campaign job to which agent is attached.
actionId	The id of action node.
actionName	Action Name within the Campaign under which agent is associated
zoneId	The id of the zone to which agent is associated.
campaignName	The name of the campaign in which agent is attached.
jobAttachTimeStamp	The time stamp when the agent was got attached to the current job.
callAndTimeInfo	**AgentCurrentJobStatistics** class containing different time related agent statistics. Refer AgentCurrentJobStatistics attributes table for details.
compInCodeInfo	Map of completion codes IDs as Key and **CompletionCode** object as value. Refer CompletionCode attributes table for details. **Note:** This Map contains Completion Codes information of the current job only.
lastJobAttachTimeStamp	The time stamp when the agent was last attached to the current job.
lastJobDetachTimeStamp	The time stamp when the agent was last detached from current the job.
handlerName	The name of the handler in which agent is attached.
pacingType	The pacing type of the handler in which agent is attached.
currentJobDuration	Time for which the agent is attached to the current job.

AgentAllJobStatistics and AgentCurrentJobStatistics

Both the class have same set of attributes except totalJobTime attribute available only in AgentAllJobStatistics class. But the main difference is AgentAllJobStatistics class holds the cumulative data for all the jobs handled by the agent within current login session, while AgentCurrentJobStatistics class holds the data for the current job being handled by the agent.

Attribute Name	Attribute Value
talkTime	The total talk time of the agents in all jobs or current job. Note: In case agent logs out and logs in again, this will reset to 0.
callCount	The total call count of the agents in all jobs or current job. Note: In case agent logs out and logs in again, this will reset to 0.
acwTime	The total after call work time of the agents in all jobs or current job. Note: In case agent logs out and logs in again, this will reset to 0.
acwCount	The total after call work count of the agents in all jobs or current job. Note: In case agent logs out and logs in again, this will reset to 0.
inJobBreakTime	The total in-job break time of the agents in all jobs or current job.

	Note: In case agent logs out and logs in again, this will reset to 0.
inJobBreakCount	The total in-job break count of the agents in all jobs or current job. Note: In case agent logs out and logs in again, this will reset to 0.
inJobIdleTime	The total in-job idle time of the agents in all jobs or current job. Note: In case agent logs out and logs in again, this will reset to 0.
inJobIdleCount	The total in-job idle count of the agents in all jobs or current job. Note: In case agent logs out and logs in again, this will reset to 0.
holdTime	The total hold time of the agents in all jobs or current job. Note: In case agent logs out and logs in again, this will reset to 0.
holdCount	The total hold count of the agents in all jobs or current job. Note: In case agent logs out and logs in again, this will reset to 0.
conferenceTime	The total conference time of the agents in all jobs or current job. Note: In case agent logs out and logs in again, this will reset to 0.
conferenceCount	The total conference count of the agents in all jobs or current job. Note: In case agent logs out and logs in again, this will reset to 0.
transferCount	The total transfer count of the agents in all jobs or current job. **Note**: In case agent logs out and logs in again, this will reset to 0.
transferRecvCount	The total number of calls transferred or received by agent in all jobs or current job. **Note**: In case agent logs out and logs in again, this will reset to 0.
consultTime	The total consult time of the agents in all jobs or current job. **Note**: In case agent logs out and logs in again, this will reset to 0.
consultCount	The total consult count of the agents in all jobs or current job. **Note**: In case agent logs out and logs in again, this will reset to 0.
previewTime	The total preview time of the agents in all jobs or current job. **Note**: In case agent logs out and logs in again, this will reset to 0.
previewAcceptCount	The total dialed contact in preview by agents in all jobs or current job. **Note**: In case agent logs out and logs in again, this will reset to 0.
previewRejectCount	The total canceled contact in preview by agents in all jobs or current job. **Note**: In case agent logs out and logs in again, this will reset to 0.
abandonedHoldCount	The total number of calls handled by agent getting disconnected when customer is on hold in all jobs or current job. **Note**: In case agent logs out and logs in again, this will reset to 0.
holdInConfCount	The total number of calls agent has put customer on hold during conference in all jobs or current job. **Note**: In case agent logs out and logs in again, this will reset to 0.
holdInConfTime	The total time agent has put customer on hold during conference in all jobs or current job. **Note**: In case agent logs out and logs in again, this will reset to 0.
avgACWTime	The average after call work of the agents in all jobs or current job. **Note**: In case agent logs out and logs in again, this will reset to 0.

avgTalkTime	The average talk time of the agents in all jobs or current job. **Note**: In case agent logs out and logs in again, this will reset to 0.
avgHoldTime	The average hold time of the agents in all jobs or current job. **Note**: In case agent logs out and logs in again, this will reset to 0.
avgPreviewTime	The average preview time of the agents in all jobs or current job. **Note**: In case agent logs out and logs in again, this will reset to 0.
avgBreakTime	The average in-job break time of the agents in all jobs or current job. **Note**: In case agent logs out and logs in again, this will reset to 0.
avgIdleTime	The average in-job idle time of the agents in all jobs or current job. **Note**: In case agent logs out and logs in again, this will reset to 0.
percIdleTime	The percentage of in-job idle time of the agents in all jobs or current job. **Note**: This is derived as (totalInJobIdelTime/(jobAttachedDuration-totalInJobBreakTime)) * 100. In case agent logs out and logs in again, this will reset to 0.
percBreakTime	The percentage of in-job break time of the agents in all jobs or current job. **Note**: This is derived as (totalInJobBreakTime/jobAttachedDuration) * 100. In case agent logs out and logs in again, this will reset to 0.
totalJobTime	The total time worked by the agents in all jobs. **Note**: This attribute is part of **AgentAllJobStatistics** class only. In case agent logs out and logs in again, this will reset to 0.
sessionCount	Displays the number of times the agent is attached to the current job action in the current session.

CompletionCode

Attribute Name	Attribute Value
Id	The database id of this completion code
Name	the name of the completion code
Rpc	Right Party Connect flag for completion code. One should use the flag only for custom completion codes.
Success	Success flag for completion code. One should use the flag only for custom completion codes.
Closure	Closure flag for completion code. One should use the flag only for custom completion codes.
Count	Total count of this completion code.

Import Statistics Event

Import statistical information exists from the time Contact List Import or DNC Import job starts until Import job completes. Campaign Director sends the import statistics after every 4 seconds.

The frequency of Import Statistics event generation is 4 seconds by default. The event frequency can be configured by updating config_value of parameter

IMPORT_STATISTICS_EVENT_FREQUENCY_IN_SECONDS in pim_config table of POM database.

For Example: To set frequency of 6 seconds, execute below command in database.

update pim_config set config_value='6' where config_name='IMPORT_STATISTICS_EVENT_FREQUENCY_IN_SECONDS';

Frequency changes to take effect user will need to restart vpms and cmpdir service.

systemctl restart vpms
systemctl restart cmpdir

Following data will be passed as part of this event.

Attribute Name	Attribute Value
jobId	Displays the job ID of the contact list or DNC import job.
dataSourceName	Displays the name of the data source from which data is imported in the contact list or DNC list.
listName	Displays the name of the contact list or DNC list in which the data is imported.
orgName	Displays the organization with which the contact list is associated. If the contact list is associated with multiple organizations, this field displays the organization name of the user who started the data import job. In case of DNC import job, this field displays the organization name of the user who started the DNC import job.
Type	Displays the import type of the job. Following are the possible Import Types: • CONTACT_FILE: Imports contacts into the contact list from a local, FTP, or SFTP source file. • DATABASE_SQL: Imports contacts into the contact list from a database. • CUSTOM: Imports contacts into the contact list by using a custom class. • UPLOAD_CONTACT_FILE: Imports contacts into the contact list from a .csv file that is uploaded in POM. • DNC_FILE_FOR_ADD: Adds contacts to the DNC list from a local, FTP, or SFTP source file. • DNC_FILE_FOR_REMOVE: if the contacts in the local, FTP, or SFTP source file match the contacts in the DNC list, this import job removes the matching contacts from the DNC list. • EXCLUDE_CONTACTS: Excludes or Marks contacts un-callable from a contact list.

	• RESET_EXCLUDED_CONTACTS: Resets the excluded contacts to callable in the contact list. • EXCLUDE_CONTACTS_FILE: Exclude contacts from a contact list using either local or FTP or SFTP file. • EXCLUDE_CONTACTS_DB: Exclude contacts from a contact list using a database as a source. • EXCLUDE_CONTACTS_CUSTOM: Exclude contacts from a contact list using a custom class. • EXCLUDE_CONTACTS_UPLOAD_FILE: Exclude contacts from contact list by uploading a .csv file.
startTime	Displays the start time of the data import or DNC import job.
Status	Displays the status of the import job. The following are the status types: • COMPLETED: The import job is completed. • QUEUED: The import job is in a queue. • RUNNING: The import job is running. • ERROR: The import job has encountered an error. • FILE_COPYING: The import job is copying a file. • PAUSING: The import job is pausing. • PAUSED: The import job is paused. • STOPPING: The import job is stopping. • WAITING_TO_RESUME: A paused import job is waiting to be resumed. • DELETING_CONTACTS: The import job is deleting contacts. • CREATING_HISTORY: Contacts are imported and job history is being created before the import job is marked as completed. • QUEUED_EXCLUDING_CONTACTS: The import job for excluding contacts is in a queued state. • QUEUED_RESETTING_EXCLUDED_CONTACTS: The import job for resetting the excluded contacts to callable is in a queued state. • EXCLUDING_CONTACTS: The import job is marking the contacts as excluded in the contact list. • RESETTING_EXCLUDED_CONTACTS: The import job is resetting the excluded contacts to callable in the contact list.
Detail	Displays the import job status change reason. Possible values are: • UserPausedSuccessful • FileRunning • Completed • Failed • Deleting • FileCopying • Begin • BeginForFailover

	• AbortedQueued • FileDirPause • IOException • UserStopSuccessful • FileDirStopAborted • Queued • FileCopied • UserPaused • UserResume • UserStop • FileDirStop • ErrorStoreHasActiveCampaigns • ImportCompleteCreateHistory • ErrorStoreHasExcludeJob • ErrorStoreHasActiveImportJob • ErrorStoreHasActiveCallbacks
Stats	Integer Array which maintains the counts for each contact's status in a contact list. Following are the details of the indexes and the status count that each index of the array holds: 0: no.of contacts imported successfully 1: no.of contacts updated 2: no.of contacts where run time error occurred 3: no.of contacts where validation failed 4: no.of contacts where duplicates are ignored 5: no.of contacts that match phone reject pattern 6: no.of contacts deleted 7: no.of contacts that match DNC 8: no.of contacts where phone format failed 9: no.of contacts excluded 10: no.of contacts reset from excluded. 11: no.of contacts where phone or email made empty 12: no.of contacts having invalid sip 13: no.of record not found contacts 14: no.of contacts excluded but callbacks on those contacts retained.

Inbound Skill Event

Inbound Skill event sends the information of Skills used for blending in POM skill based pacing.

The events are published whenever POM receives message from CMS server.

Following data will be passed as part of this event.

Attribute Name	Attribute Value

Skill	Displays the skill you have defined on the CC Elite page.
skillId	Displays the skill number you have defined on the CC Elite page.
zoneName	Name of the zone to which skill belongs.
qLength	Displays the number of calls in the wait queue for the inbound skills.
ewtHigh	Displays the high expected time for which the call waits in the queue before the call is answered.
ewtMed	Displays the medium expected time for which the call waits in the queue before the call is answered.
ewtLow	Displays the low expected time for which the call waits in the queue before the call is answered.
avgSpeedAns	Displays the average time in which the agent answers the call.
agtAcqThld	Displays the percentage of calls the agents answers to achieve the specified service level.
agtRelThld	The value indicates that when traffic is below the specified value, the inbound skill is in good shape and if any blend agents were released, the system can acquire the agents. This parameter is used for blending. For example, if the agent acquire threshold value is 20 and the current Queue Length is less than 20 and any blend agents are released, the system can acquire the agents.
eventTimeStamp	Event creation time stamp. This timestamp is in UTC EPOCH format.
rcvdTimeStamp	Timestamp when the event has been received by the EventSDK client
eventTypeEnum	INBOUND_SKILL

CCTRENDS Event

CCTrend event is generated by Campaign Director per minute. The purpose of this event is to accumulate count of each completion code generated against each handler of the campaign. For each handler of campaign, Campaign Director publishes count of completion code generated in last 1 minute interval.

Attribute Name	Attribute Value
Job id	Displays the job id of campaign.
Action id	Displays the id of actions performed over campaign.
Time	Displays the time of completion code.
Name	Displays the name of completion code
Count	Displays count of completion code

Jobstarttime	Displays start time of job.

Sample CCTrends Event Data:

{
"jobId":15,
"jobStartTime":"Jun 13, 2022 1:59:56 PM",
"trendsData":
{"100":[{"time":"Jun 13, 2022 3:37:35 PM","jobId":15,"actionId":100,"count":17,"name":"Call Answered"},
{"time":"Jun 13, 2022 3:37:35 PM","jobId":15,"actionId":100,"count":14,"name":"Call Busy"},
{"time":"Jun 13, 2022 3:37:35 PM","jobId":15,"actionId":100,"count":17,"name":"Ring No Answer"},
{"time":"Jun 13, 2022 3:37:35 PM","jobId":15,"actionId":100,"count":14,"name":"CC1"},
{"time":"Jun 13, 2022 3:37:35 PM","jobId":15,"actionId":100,"count":24,"name":"CC2"}]
}
}

Appendix A: Avaya Proactive Outreach Manager Event SDK exceptions

The following table defines the POMEventException Error IDs for the Avaya Proactive Outreach Manager Event SDK client/consumer.

Error IDs	POMEventException	Description
1000	USER_NOT_LOGGED_IN	User is not logged in from any of the EventSDK Client application.
1001	USER_ALREADY_LOGGED_IN	User is already logged in from other EventSDK Client application.
1002	INVALID_CREDENTIAL	Invalid User Credentials passed during login from EventSDK Client application.
1003	USER_NOT_AUTHORIZED	User is not authorized by POM to connect to Apache Kafka.
1004	INVALID_LOGGER	Invalid logger instance passed by EventSDK Client application during login.

1005	ALREADY_INITIALIZED	EventSDK Client application is trying to subscribe for Events after initialization of POMEventRegistrar.
1006	INVALID_IMPLEMENTATION	EventSDK Client application is trying to subscribe for invalid Events.
1007	UNABLE_TO_CONNECT_TO_MESSAGE_BUS	EventSDK Client application is unable to connect with Apache Kafka.
1008	UNABLE_TO_LOAD_CONFIG_FILE	EventSDK Client application is unable to read eventsdk.properties file.
1009	UNABLE_TO_CREATE_SSLCONTEXT	Invalid truststore/keystore path and password configuration in EventSDK Client application.
1010	EVENT_BUS_NOT_REACHABLE	Kafka event bus not reachable .
1050	FAILED_SERVICE_RESPONSE	EventSDK Client failed to get response from POM server
1051	UNABLE_TO_CONNECT_TO_SERVER	EventSDK Client encounters I/O Exception while connecting to POM Server.
1052	UNABLE_TO_GET_KAFKA_SERVER_LIST	EventSDK Client is unable to fetch Kafka server list from POM Server.

Appendix B: Java sample Application Using EventSDK Library

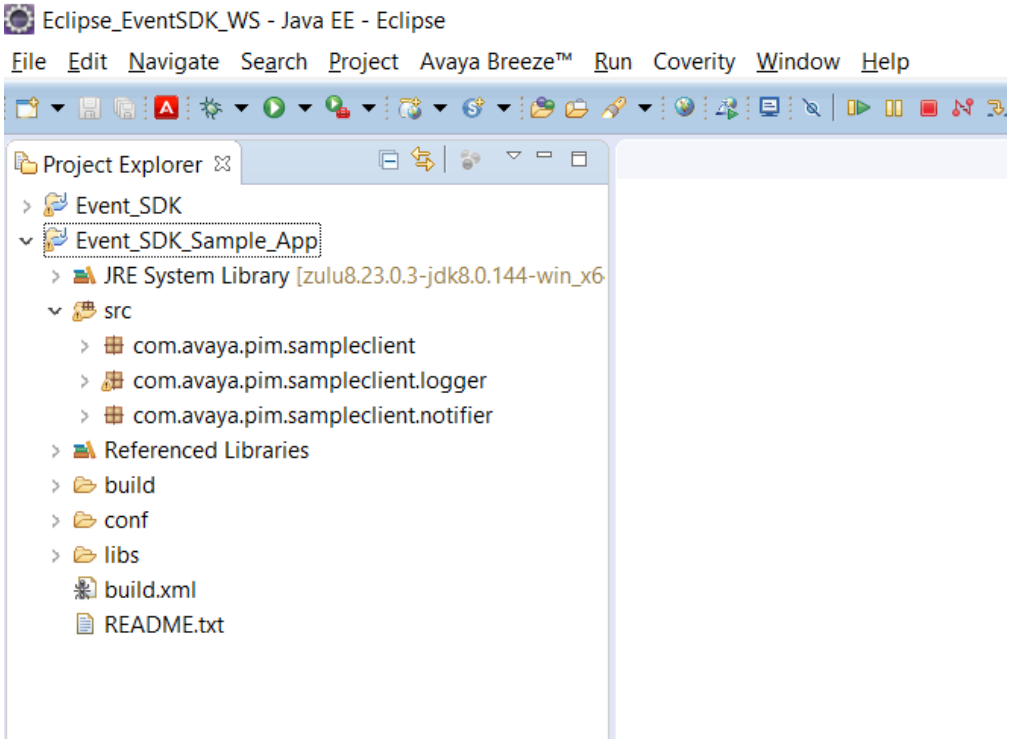
Sample Client Package

The sample client package consists of following files and directories

- *README.txt* : This file contains steps to import and execute the Sample Client App on Eclipse.
- *Sample/SampleJavaClient* : The eclipse project demonstrating the usage of various POM Event SDK APIs.
- *libs:* This directory contains dependent libraries for sample client app, listed under Required software.
- *javadocs*: Explaining POM Event SDK API's

Eclipse Project Import and Settings

1. Create a directory e.g. c:\EventSDK and set the environment variable POM_EVENT_SDK_DIR to c:\EventSDK.
   OR
   You can use -D option as below while executing the SDK sample project -DPOM_EVENT_SDK_DIR=C:\EventSDK\
2. Copy the pomKeyStore and pomTrustore from POM server (file location: <POM_HOME>/config) to <POM_EVENT_SDK_DIR>/conf i.e C:\EventSDK\conf
3. Add the POM server host entry in localhost's hosts file.
4. Import the sample project in eclipse IDE.



5. Set JRE System Library to select JRE 8 for the project build path.
6. Set compiler compliance level to 1.8.
7. Set the Run Configuration as follows:

a. Program arguments
   <POM System User Name> <POM System User Password> <POM Server IP Address/Hostname> <POM Server Port Number> <Subscription Option*>
   e.g. admin portal123 148.147.176.88 443 2

b. Subscription Option:
   - Job Events: 2
   - Agent Events: 4
   - Job Statistics Events: 8
   - Agent Statistics: 16
   - Attempt Events: 32
   - Enriched Attempt Event: 64
   - Import Statistics Events: 128
   - Heartbeat Events: 256
   - Inbound Skill Events: 512

You can sum up the number for multiple subscription i.e. 6 for Job and Agent events.

Required software

Platform

Supported JAVA: JAVA 1.8
Supported OS: Windows/Linux

Libraries

Below is the list of libraries required by the Avaya Proactive Outreach Manager Event SDK.

- avaya-pim-eventAPI.jar - Contains Core SDK API's for consuming POM Events.
- avaya-pim-security.jar - Contains Common Security Functionalities for POM
  - Dependencies:
    - bc-fips-1.0.1.jar - A Java implementation library of cryptographic algorithms certified to FIPS 140-2 level 1
    - camel-util-3.4.0.jar - A Utility Library part of Apache Camel
- gson-2.6.2.jar -> JSON to Java converter and vice versa.
- Kafka client libraries:
  - For POM 4.0
    - kafka-clients-2.7.0.jar
    - kafka-streams-2.7.0.jar
    - zookeeper-3.5.8.jar
- Logging and Sample logging implementation:
  - log4j-1.2.17.jar

- slf4j-api-1.7.30.jar
- slf4j-log4j12-1.730

Note: All the above 3rd party libraries must be of the specified version.
The **avaya-pim-eventAPI.jar** has no other version as of POM 3.1.x releases.

Appendix C: Create Client Consumer with Custom Code

First, we need to create the “consumer.properties” file that should hold the following information that is specific to the customer setup and Kafka.
Please note that the customer needs to change the IP_POM_SERVER to its own specific IP value and change the group.id which should could change for each consumer. (no spaces allowed).

```
bootstrap.servers=IP_POM_SERVER:9093
group.id=3
enable.auto.commit=true
key.deserializer=org.apache.kafka.common.serialization.StringDeserializer
value.deserializer=org.apache.kafka.common.serialization.StringDeserializer
```

After we have the above file created, we need to write a JAVA application (this is the case for our code snippet below), but Kafka consumer is not limited to only Java.
In the JAVA application, first we need to create the consumer object, similar to the code below:

```
KafkaConsumer<String, String> consumer;
Properties properties = new Properties();
try (InputStream props = Resources.getResource("consumer.props").openStream()) {
        properties.load(props);
        if (properties.getProperty("group.id") == null) {
                properties.setProperty("group.id", "group-" + new
                Random().nextInt(100000));
        }
} catch (IOException e) { e.printStackTrace(); }
consumer = new KafkaConsumer<>(properties);
topics = Arrays.asList("callback-requests ");
mapper = new ObjectMapper();
```

After we have created the consumer object, we need to start consuming the Kafka events that were generated. This behavior can be achieved with the below snippet:

```
try {
        consumer.subscribe(topics);
while (true) {
        ConsumerRecords<String, String> records = consumer.poll(Long.MAX_VALUE);
        for (ConsumerRecord<String, String> record : records) {
                        JsonNode msg = mapper.readTree(record.value());
                        System.out.println("crtMsg: " + msg);
                }
        }
        } catch (WakeupException e) {
        } catch (JsonProcessingException e) { e.printStackTrace();
        } catch (IOException e) { e.printStackTrace();
        } finally {
                consumer.close();
        }
```

Appendix D: Create Client Consumer with Custom Code in Kafka HA

We need to create a property file named “consumer.properties” that should hold the information below that is specific to the customer setup and Kafka.

Please note that the customer needs to complete the boostrap.servers value with the specific IP values for POM HA nodes. Refer to Appendix C for sample java code.

bootstrap.servers=
IP_POM_SERVER_1:9093,IP_POM_SERVER_2:9093,IP_POM_SERVER:9093
group.id=3
enable.auto.commit=true
key.deserializer=org.apache.kafka.common.serialization.StringDeserializer
value.deserializer=org.apache.kafka.common.serialization.StringDeserialize

Appendix E: eventsdk.properties

Key	**Value**	**Description**
ssl.truststore.location	conf/pomTrustStore	Provide the path for TrustStore. **Note:** If client doesn’t have its own TrustStore and KeyStore, POM TrustStore and KeyStore can be reused by copying them from POM server at $POM_HOME/config
ssl.truststore.password	Changeit	TrustStore password
ssl.keystore.location	conf/pomKeyStore	Provide the path for KeyStore. **Note:** If client doesn’t have its own TrustStore and KeyStore, POM TrustStore and KeyStore can be reused by copying them from POM server at $POM_HOME/config
ssl. keystore.password	Changeit	KeyStore password
ssl.enabled.protocols	TLSv1.2	Protocol to be used for secured connection
kafka.key.deserializer	org.apache.kafka.common.serialization.StringDeserializer	DeSerializer class to deserialize JSON object keys from event’s bytearray data.
kafka.value.deserializer	org.apache.kafka.common.ser	DeSerializer class to deserialize JSON

	ialization.StringDeserializer	object value from event's bytearray data.
kafka.group.id	3	The GroupID must be unique across all the different instances of the client
kafka.client.id	EVENT_SDK_CLIENT	Unique id for kafka client. Kafka will not authorize the client application if some other client has subscribed with the same client id.
pom.enabled.cipher.suites	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_EMPTY_RENEGOTIATION_INFO_SCSV	Comma separated cipher list to be used for secured connection with kafka topic.
pom.receive.message.after.start	True	If this parameter is set as true, consumer will receive messages after client starts.
pom.print.event.payload	False	If this parameter is set as true, all the event metadata will be printed to log files. Useful for debugging.
pom.eventsdk.use.encrypted.passwords	False	If this parameter is set as true, ssl.truststore.password and ssl.keystore.password will need to use encrypted passwords instead of plain passwords. To encrypt the password, you can use *$POM_HOME/bin/POMSecurityManagementTool.sh* script on the POM server. Please check "*Implementing Proactive Outreach Manager*" guide for more details.

Appendix E: Troubleshooting

Event SDK Client does not receive events

- On EP page, check the “Event Settings” under POM -> POM Home -> Configurations -> Global Configuration
- Check the host entry of Kafka Server in /etc/hosts file of the machine from where the client application is launched. This is required irrespective of you are using hostname or IP based configuration.
- Verify the Kafka server status on primary POM server. Kafka status can be checked from EP UI.
- Verify user belongs to correct organization. User will receive events only from the organization he belongs to.

Certificate Errors

- The POM Trust Store must have client certificate imported.
- The Client Trust Store must have POM certificate imported.
- Verify trust store paths and passwords are correct.

GroupId configuration

- The client GroupId is mentioned in of eventsdk.properties file.
- The GroupID must be unique across all the different instances of the client.
- Each EventSDK instance has separate copy of eventsdk.properties file.