Proactive Outreach Manager Agent API

Release 3.0.4

Issue 1.0

December 2016

AVAYA SOFTWARE DEVELOPMENT KIT LICENSE AGREEMENT

REVISED: March 24, 2016

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 SDK (COLLECTIVELY, AS REFERENCED HEREIN, "YOU", "YOUR", OR "LICENSEE") AND AVAYA INC. OR ANY AVAYA AFFILIATE (COLLECTIVELY, "AVAYA"). IF YOU ARE ACCEPTING THE TERMS AND CONDITIONS OF THIS AGREEMENT ON BEHALF OF A LEGAL ENTITY, YOU REPRESENT AND WARRANT THAT YOU HAVE FULL LEGAL AUTHORITY TO ACCEPT ON BEHALF OF AND BIND SUCH LEGAL ENTITY TO THIS AGREEMENT. BY OPENING THE MEDIA CONTAINER, BY INSTALLING, DOWNLOADING, COPYING OR OTHERWISE USING THE AVAYA SOFTWARE DEVELOPMENT KIT ("SDK") OR AUTHORIZING OTHERS TO DO SO, YOU SIGNIFY THAT YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT HAVE SUCH AUTHORITY OR DO NOT WISH TO BE BOUND BY THE TERMS OF THIS AGREEMENT, SELECT THE "DECLINE" BUTTON AT THE END OF THE TERMS OF THIS AGREEMENT OR THE EQUIVALENT OPTION.

1.0 DEFINITIONS.

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

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

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

1.4 "Change In Control" shall be deemed to have occurred if any person, entity or group comes to own or control, directly or indirectly, beneficially or of record, voting securities (or any other form of controlling interest) which represent more than fifty percent (50%) of the total voting power of or to 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 "Open Source Software" or "OSS" is as defined by the Open Source Initiative ("OSI") and is software licensed under an OSI approved license as set forth at http://www.opensource.org/docs/osd (or such successor site as designated by OSI).

1.9 "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.

1.10 "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.

1.11 “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.

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

1.13 “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 the Sample Application Code with Permitted Modifications, provided that such distribution is subject to an end user license agreement 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.

C. Except as expressly authorized by this Agreement, and unless otherwise permitted by the applicable law, Licensee acknowledges and agrees that the foregoing license does not include any right to distribute, license, translate, publish, or display the SDK, Specification Documents or Documentation or any copy or part thereof. Licensee represents and warrants that it will not use, modify, or distribute the redistributable Client Libraries in any manner that causes any portion of the redistributable Client Libraries that is not already subject to an OSS license to become subject to the terms of any OSS license.

D. 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).

E. 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.

F. 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.

2.2 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.

2.6 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.

2.7 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 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.

2.13 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 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, black boxes, malware, trapdoors, and other mechanisms to allow remote/hidden attacks or access through unauthorized computerized command and control, and will not contain any other computer software routines designed to spy, monitor traffic (network sniffers, keyloggers), 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, 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.

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, worldwide license under any and all of Licensee's Intellectual Property rights related to any Permitted Modifications, to use, employ, practice, make, have made, sell, and/or otherwise exploit any and all Permitted Modifications.

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 derivative 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 derivative 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 return to Avaya all Confidential Information (including copies thereof) in Licensee's possession, custody, or control upon termination of this Agreement at any time and for any reason. 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 AND WILLFUL MISCONDUCT, 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 AND WILLFUL MISCONDUCT, IN NO EVENT SHALL AVAYA'S TOTAL LIABILITY TO LICENSEE IN CONNECTION WITH, ARISING OUT OF OR RELATING TO THIS AGREEMENT EXCEED FIVE HUNDRED DOLLARS ($500). THE PARTIES AGREE THAT THE LIMITATIONS SPECIFIED IN THIS SECTION WILL APPLY EVEN IF ANY LIMITED REMEDY PROVIDED IN THIS AGREEMENT IS FOUND TO HAVE FAILED OF ITS ESSENTIAL PURPOSE.

9.0 INDEMNIFICATION.

Licensee shall indemnify and hold harmless Avaya, Affiliates and their respective officers, directors, agents, suppliers, customers and employees from and against all claims, damages, losses, liabilities, costs, expenses, and fees (including fees of attorneys and other professionals) 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, including, but not limited to, products liability claims and claims of infringement of third party Intellectual Property rights.

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 of this Agreement, Licensee will immediately cease using the SDK, and Licensee agrees to destroy all adaptations or copies of the SDK and Documentation, or return them to Avaya upon termination of this License.

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

11.0 ASSIGNMENT.

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

12.0 COMPLIANCE WITH LAWS.

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

13.0 WAIVER.

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

14.0 SEVERABILITY.

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

15.0 GOVERNING LAW AND DISPUTE RESOLUTION.

This Agreement and any dispute, claim or controversy arising out of or relating to this Agreement ("Dispute"), including without limitation those relating to 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.

Any Dispute shall be resolved in accordance with the following provisions. The disputing party shall give the other party written notice of the Dispute. The parties will attempt in good faith to resolve each Dispute within thirty (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. 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 these procedures and within these timeframes, 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 any or all other parties 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 this Agreement 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 each party will 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 shall 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.

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 above, 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 above with regard to arbitration of Disputes that arise anywhere other than in the United States or are based upon an alleged breach committed anywhere other than in the United States, each party to this Agreement consents to the exclusive jurisdiction of those courts, including their appellate courts, for the purpose of all actions and proceedings.

The parties agree that the arbitration provision in this section 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. Nothing in this section 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. In addition and notwithstanding the foregoing, Avaya shall be entitled to take any necessary legal action at any time, including without limitation seeking immediate injunctive relief from a court of competent jurisdiction, in order to protect Avaya's intellectual property and its confidential or proprietary information (including but not limited to trade secrets).

16.0 IMPORT/EXPORT CONTROL.

Licensee is advised that the SDK is of U.S. origin and subject to the U.S. Export Administration Regulations ("EAR"). The SDK also may be subject to applicable local country import/export laws and regulations. Diversion contrary to U.S. and/or applicable local country law and/or regulation is prohibited. Licensee agrees not to directly or indirectly export, re-export, import, download, or transmit the SDK to any country, end user or for any use that is contrary to applicable U.S. and/or local country regulation or statute (including but not limited to those countries embargoed by the U.S. government). Licensee represents that any governmental agency has not issued sanctions against Licensee or otherwise suspended, revoked or denied Licensee's import/export privileges. Licensee agrees not to use or transfer the SDK for any use relating to nuclear, chemical or biological weapons, or missile technology, unless authorized by the U.S. and/or any applicable local government by regulation or specific written license. Additionally, Licensee is advised that the SDK may contain encryption algorithm or source code that may not be exported to government or military end users without a license issued by the U.S. Bureau of Industry and Security and any other country's governmental agencies, where applicable.

17.0 AGREEMENT IN ENGLISH.

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

18.0 ENTIRE AGREEMENT.

This Agreement, its exhibits and other agreements or documents referenced herein, constitute the full and complete understanding and agreement between the parties and supersede all contemporaneous and prior understandings, agreements and representations relating to the subject matter hereof. No modifications, alterations or

amendments shall be effective unless in writing signed by both parties to this Agreement.

19. REDISTRIBUTABLE CLIENT FILES.

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

20. SCHEDULE 1 TO AVAYA SDK LICENSE AGREEMENT THIRD PARTY NOTICES

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit)

alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and limitations under the License.

Agent Desktop API

Introduction

Purpose

This document describes the methods and properties used for agent desktop APIs of Avaya Proactive Outreach Manager.

Intended Audience

This document is intended for users and development engineers who are involved in customization for Avaya Proactive Outreach Manager using agent desktop APIs.

Availability

The latest version of this document is available on the Avaya online support Web site:

http://support.avaya.com

What's new

- In callback call notification preview timer can be set to zero. So if the preview timer is zero and the call notification is of type callback, desktop need to dial the default number immediately. See section **Callback->Callback Preview Timer**
- For the requirement background call classification, the wrapup timer is received as zero with answering machine completion code (Answer_Machine). See API section **AGTAutoReleaseLine**.
- In external conference agent can transfer the call to external party and leave the conference. See section **Conference-> External Conference** and updated **Capability Matrix** for external conference.
- .Net framework of agent API DLL is upgraded to 4.5.
- Agent API DLL supports TLS1.2.

Overview

An Application Programming Interface (API) defines the way different software components can integrate with each other. Using POM, you can integrate the agent functionality by designing a set of procedures or routines and manage the agents using a desktop.

The API's are divided into 2 main categories: Commands (requests), Notifications and or Responses.

- Commands are requests sent by the agent desktop, which is, initiated by the

agent. Examples include login, hold, unhold, transfer, and conference.

- Notifications and or Responses are events sent by POM to the agent desktop. Examples include call notifications, state change notifications, call state change notifications, agent capabilities or responses to the commands.

The requests sent by the desktop can be asynchronous. Asynchronous events are when the system cannot or does not respond back immediately with the return values.
Depending on the other activities running on the system, or the number of parameters or arguments for the command, the system might give back delayed responses. To accommodate the asynchronous behavior of the POM system, the APIs also demonstrate asynchronous behavior.

For example, an agent issues a hold request or command using the API AGTHoldCall(). POM sends back a response to the call when the command reaches POM. Meanwhile POM completes the Hold request and sends the actual response, that is, success or failure, in an asynchronous manner by invoking the callback function AGTHoldCallRESP.

Note:

POM Agent Manager is a POM component which takes care of all agent activities. It maintains the agent state machine. It interacts/controls various other components like the call pacer, nailer, router and blender to achieve state transitions.

The API's are structured such that the Agent State Machine is closely associated with POM, so that POM has complete control of the agent state, and the agent desktop acts as an interface for the agent. This helps to minimize agent state handling within the desktop, and eliminate the need of two state machines – one for the desktop and the other in POM.

The desktop acts like a stateless interface and POM controls the desktop. POM defines desktop agent capabilities, such as CanHold, CanTransfer, CanConf, CanConsult.

At any point of time, POM provides these capabilities to the agent desktop via notifications so that various GUI controls can be enabled disabled accordingly. For example, when the agent performs a hold operation, POM sets the CanHold capability to false and enables CanUnhold. Similarly at various points in time the agent moves between Idle, Talking, Held, Consult, Conference, Wrapup call states.

However, it is advisible to handle the cases where agent can send multiple requests before getting the capabilities from POM server. POM notifies the desktop whenever the agent's call state changes. Similarly POM notifies the agent about the agent state such as Ready, NotReady, PendingNotReady.

POM also notifies the desktop about its nailing state such as NailedUp, PendingNailup, PendingNailupDrop, NailingLost, NotNailedUp.

Classes and interfaces

You can use the following classes and interfaces for the integration with the desktop:

- POMAgentFactory: Use to initialize the library.

- POMAgent: You must get the POMAgent API from the Software Development Kit (SDK) and then use it for invoking commands. POMAgent is the agent object.
- POMAgentHandlerInterface: You must implement the interface to set up the notifications and responses to commands. POM uses the interface as a callback mechanism for responding to commands and for sending notifications.

API Components

The desktop APIs are available as a combination of 3 Dynamic Link Libraries (DLL)'s

- POMDesktopAPI.dll: The API's are available through the POMDesktopAPI.dll.
- Avaya.POM.Agent.ObjectModel.dll: The data structures used for the API's are available through the Avaya.POM.Agent.ObjectModel.dll.
- log4net.dll: The log4net.dll acts as a helper DLL for the API's.

Commands, Notifications, and Responses

List of commands and responses

Command	Usage	Response
AGTLogon	Used to Login to POM	AGTLogonRESP
AGTLogoff	Used to Log off from POM	AGTLogoffRESP
AGTStateChange	Used to change agent state to Ready/NotReady/PendingNotReady	AGTStateChangeRESP
AGTHoldCall	Used to put customer call on hold.	AGTHoldCallRESP
AGTUnholdCall	Used to unhold customer call.	AGTUnholdCallRESP
AGTReleaseLine	Agent can use this to disconnect customer call.	AGTReleaseLineRESP
AGTGetCompCodes	Returns the list of custom codes linked with the current campaign.	AGTGetCompCodesRESP
AGTWrapupContact	Wraps up the customer call.	AGTWrapupContactRESP
AGTExtendWrapup	Permits agent to get more time to wrapup call.	AGTExtendWrapupRESP
AGTGetConsultTypes	Returns the types (Agent/Campaign) of consult supported for the current job.	AGTGetConsultTypesRESP
AGTGetConsultDestsForType	Returns the available destinations for the selected type of consult.	AGTGetConsultDestsForTypeRE
AGTConsultCall	Used to consult another party.	AGTConsultCallRESP
AGTCompleteTransfer	Used to complete the transfer for the ongoing consult call.	AGTCompleteTransferRESP
AGTCancelConsult	Used to cancel an ongoing/pending consult.	AGTCancelConsultRESP

AGTStartConf	To start a conference for an ongoing consult.	AGTStartConfRESP
AGTEndConf	Used by the agent to end the conference	AGTEndConfRESP
AGTConfChangeOwnership	Used by conference owner to transfer ownership to the passive agent.	AGTConfChangeOwnershipRESP
AGTRedial	Used by an agent to Redial the customer in Wrap-up mode.	AGTRedialRESP
AGTSendDTMF	Used by an agent to send DTMF over the line.	AGTSendDTMFRESP
AGTGetCallbackTypes	Used by an agent to determine the types of callbacks supported for the current job.	AGTGetCallbackTypesRESP
AGTGetCallbackDestsForType	Used by an agent to get a list of available destinations for the selected type of callback.	AGTGetCallbackDestsForTypeRESP
AGTCreateCallback	Used by an agent to create a callback.	AGTCreateCallbackRESP
AGTGetErrorString	Used by desktop to get error code details.	AGTGetErrorStringRESP
AGTPreviewDial	Used by agent to accept and dial the preview contact.	AGTPreviewDialRESP
AGTPreviewCancel	Used by agent to cancel the preview contact and move to wrapup state.	AGTPreviewCancelRESP
AGTGetCustomerDetails	Used by desktop to get details of the customer contact.	AGTGetCustomerDetailsRESP
AGTSetCustomerDetail	Used to set an attribute value of a contact.	AGTSetCustomerDetailRESP
AGTBlendToInbound	Sent by blender to POM informing it to move the agent to Inbound	AGTBlendToInboundRESP
AGTBlendToOutbound	Sent by blender to POM informing it	AGTBlendToOutboundRES
AGTNailupAgent	Sent by desktop after moving The agent from Inbound to outbound.	AGTNailupAgentRESP
AGTReadyForNailup	Sent by desktop after processing AGTNailupChange – PendingNailup notitication.	AGTReadyForNailupRESP
AGTLostNailing	Sent by desktop if desktop detects that the nailing is lost.	AGTLostNailingRESP
AGTPendingLogout	Sent by desktop in error situations,so that POM can logout the agent after the current call is wrapped up.	AGTPendingLogoutRESP
AGTAddAgentNote	Used by desktop to add an agent note.	AGTAddAgentNoteRESP
AGTRefreshAgentNotes	Used by desktop to refresh agent notes and get the new ones.	AGTRefreshAgentNotesRESP

AGTGetTimezones	Used by desktop to determine the supported time zones for callbacks.	AGTGetTimezonesRESP
AGTAvailableForNailup	Sent by desktop after the agent goes to Ready state for the first time after login.	AGTAvailableForNailupRESP
AGTAgentDisconnected	Sent by Proxy if it detects that the desktop is not reachable.	AGTAgentDisconnectedRESP
GetAgentStatusResponse	Desktop must send this in response to GetAgentStatus notification	GetAgentStatusResponseRESP
AGTAddToDNC	Sent by desktop to add an agent to DNC list.	AGTAddToDNCRESP
AGTGetZoneList	Command to get current zones on POM.	AGTGetZoneListRESP
AGTIsInDNC	Sent by desktop to check if a contact address is in DNC list.	AGTIsInDNCRESP
AGTSaveAgentForHA	Desktop must send this after logging in so that POM saves the agent state if the desktop comes back up after crashing..	AGTSaveAgentForHARESP
AGTSkillsChanged	Sent by AACC via AAAD Informing POM that the agents skill are modified.	AGTSkillsChangedRESP
AGTGetContactAttributes	Sent by desktop to determine all attributes linked with a particular contact.	AGTGetContactAttributesRESP

List of notifications

Notification	Usage
AGTStateChangedNotify	Sent by POM when the agent state changes (Ready/NotReady/PendingNotReady)
AGTCallNotify	Sent by POM when desktop receives a new contact.
AGTAutoReleaseLine	Sent by POM when the customer disconnects the call.
AGTConsultNotify	Sent by POM informing an agent that the agent is in consult now.
AGTConsultCancelled	Sent by POM to an agent when the consult gets over.
AGTTransferNotify	Sent by POM to the passive agent when the consult gets transferred to it.
AGTConferenceNotify	Sent by POM to the passive agent when the consult gets into a conference
AGTConferenceEnded	Sent by POM to an agent when the other agent ends the conference.

AGTConferenceOwnershipChanged	Sent by POM to the passive agent when the conference owner transfers conference ownership to the passive agent.
AGTCapabilitiesChanged	Sent by POM when it detects change in desktop capabilities based on ag nailing states.
AGTNailupChange	Sent by POM when it detects change in nailing state of an agent.
AGTCallStateChangedNotify	Sent by POM when it detects change in call state of an agent.
AGTDialFailed	Sent by POM when it detects failure while dialing a contact for either are preview dial.
AGTConsultDialFailed	Sent by POM when it detects failure while dialing for a consult.
AGTConsultPending	Sent by POM informing a busy agent that a consult is pending.
AGTPendingConsultComplete	Sent by POM to the initiator agent when the pending consult gets into a consult.
POMAvailable	Internal notification.
POMNotAvailable	Internal notification.
AGTPreviewCallbackPending	Sent by POM when a callback is pending for an agent.
AGTAgentLoggedOut	Sent by POM when it detects that the agent has logged out.
AGTEnableCancelConsult	Sent by POM informing the desktop that it can enable the cancel consult operation.
AGTPreviewCallbackCancelled	Sent by POM informing the agent that the current callback on the agents has been cancelled.
AGTInvalidCommandName	Sent by POM if it receives an unknown command.
AGTCustomerDetailsChanged	Sent by POM to a passive party in consult/conference if the owner changes customer attributes value.
AGTBlendedToInbound	Sent by POM informing the agent that it is blended to Inbound.
AGTBlendedToOutbound	Sent by POM informing the agent that it is blended to Outbound.
AGTZoneDown	Sent by POM when a zone is marked DOWN for failover.
AGTJobAttached	Sent by POM when an agent is attached to a new job.

Agent Activities

Login and Logout

To perform an outbound job, an agent must login to POM. POM provides APIs so that an agent can login to POM. The agent must provide AgentID/AgentExtension, password, zone name, timezone, locale with the login request in CCElite mode, and POM server fetches the skill information and password from Communication Manager using SMS webservice of AES and caches the skill information in database.

POM also has arrangement for an agent to force login to POM where the desktop crashes. After force login, the agent session on POM resets and a new session is created for the agent.

Note: Force login agent will drop the customer call and nailing call of the agent. So if the agent is in talk with customer, agent should wait for the current call to complete before force login.

API	Command/Response/Notification
AGTLogon	Command sent by desktop to POM
AGTLogonRESP	Asynchronous callback response to AGTLogon
AGTLogout	Command sent by desktop to POM
AGTLogoutRESP	Asynchronous callback response to AGTLogout

Agent state

An agent has to move to Ready state after logging-in so that it can receive outbound calls. After the agent moves to ready state, POM considers this agent for campaign assignment. An agent can also move to NotReady state while agent is in Ready state. But, the agent has to be in Idle (not in a call) state for it to move from Ready to NotReady state.

If the agent tries to move to NotReady state while handling a call, POM moves the agent to PendingNotReady state. The agent stays in PendingNotReady state until the agent wraps up the current call and also if it does not have any pending consults/callbacks. After the agent wrapps up the current call and handling the pending requests (consult/callbacks) POM moves the agent to NotReady state from PendingNotReady state.

POM notifies the agent about the current state through the AGTStateChangedNotify notification.

Note:

An agent has to handle a maximum of a current call, up-to 2 pending consult requests and one pending callback before POM moves it to NotReady state. If an agent issues a NotReady command when it does not have any pending requests (pending consults/pending callback), then it moves the agent to NotReady state when the agent wraps up the current call.

Walk-away agent

The API also has an arrangement to handle walk-away agents. For example, the agent has moved away from agent's desk while handling a call. On hearing nobody on the agent end, the customer hangs up the call. The call is autowrapped up by the agent if the

campaign has set a wrapup timer.

POM then gives the next outbound call to this agent. One more time the customer hangs up the call on hearing silence. One more time the call gets auto wrapped up. To prevent such situations, the desktop must detect such situations and automatically send AGTStateChange with the walkedAway flag set to true. POM then moves this agent to NotReady state and does not consider this agent for call pacing. One of the ways that the desktop can determine if the agent has walked away is, if no button clicks for two consecutive calls on the desktop by the agent.

Note:

If acwMaxTime is received as zero with answering machine completion code (Answer_Machine), desktop needs to wrap up the call immediately without agent interaction with the completion code answering machine received in AGTAutoReleaseLine notification. In this case desktop should not mark agent as walk away agent.

Changing Aux Reason

An agent can also change the aux reason after going to NotReady state. The agent has to run AGTStateChange with new reason.

Note: POM does not provide any aux reason and the aux reason is not displayed in any reports.

API	Command/Response/Notification
AGTStateChange	Command sent by desktop to POM to change state. To move to either Ready or NotReady state, use the same
AGTStateChangeRESP	Asynchronous callback response to AGTStateChange
AGTStateChangedNotify	POM sends asynchronous notification when the agent state changes.

Nailing

POM nails the agent if a campaign is running which matches the skillset of agent. When the agent logs in for the first time, the agent sends AGTAvailableForNailup. POM can nail the agent only after POM receives the command.

POM sends a sequence of notifications with flags indicating the next nailing action. If POM wants to nail an agent it first sends AGTNailupChange with flag PendingNailup. On processing this flag the desktop must send AGTReadyForNailup. After POM receives AGTReadyForNailup, POM then attempts to send a nailing call to the agent. When the agent picks up the nailing call, POM again sends AGTNailupChange with flag NailedUp. POM sends AGTNailupChange with flag NotNailedUp, if the agent nailing call is not successful.

An agent gets unnailed if the job which attaches the agent gets over OR the agent tries to logout after moving to NotReady state from Ready state. POM sends AGTNailupChange with PendingNailupDrop flag. After the nailing call disconnects, POM sends AGTNailupChange with flag NotNailedup.

If the desktop drops the nailup unexpectedly then the desktop must send AGTLostNailing. Also, if POM detects that the nailing got dropped unexpectedly then POM sends AGTNailupChange with the flag set to NailingLost.

API	Command/Response/Notification
AGTAvailableForNailup	Command sent by desktop to POM. POM can decide this agent for nailing.
AGTAvailableForNailup RESP	Asynchronous callback response to *AGTAvailableForNailup*
AGTNailupChange	POM sends asynchronous notification when the nailing state changes.
AGTReadyForNailup	Command sent by desktop to POM on receiving AGTNailupChange – PendingNailup.
AGTReadyForNailupRESP	Asynchronous callback response to AGTReadyForNailup
AGTLostNailing	Command sent by desktop when it detects that the nailing got lost unexpectedly
AGTLostNailingRESP	Asynchronous callback response to AGTLostNailing

New call notification

Predictive/Progressive Campaigns: For Predictive and Progressive campaigns, it dials the customer first and then connect to the agent. POM sends a notification to the agent when it connects the customer call to the agent so that the agent has enough information about the customer. In the new call notification POM sends the agent script URL, campaign name, skillset and basic information about the customer.

API	Command/Response/Notification
AGTCallNotify	POM sends asynchronous notification when it connects the customer call to the agent.

Preview Campaigns: For Preview campaigns, it sends the new call(call data) notification to the agent first. The campaign dials the customer only if the agent decides to talk to the customer (AGTPreviewDial command). The agent can cancel the request (AGTPreviewCancel command) based on the configuration in strategy and in this case, the agent has to provide a completion code.

POM supports timed and nontimed preview campaigns. For timed preview, POM sends a time value with AGTCallNotify. This time value is the time POM gives to the agent to decide on dialing/ cancelling the preview. If the agent does not make a decision within the time value, then the desktop must send AGTPreviewDial automatically after the timer expires. For nontimed preview the agent has infinite time to make a decision on accepting the preview or rejecting it. The configuration for timed and nontimed preview is available in strategy

If the dial trail fails, the POM notifies the desktop by sending a notification AGTDialFailed with the reason code.

API	Command/Response/Notification
AGTCallNotify	POM sends asynchronous notification when the customer call connects to the agent.
AGTPreviewDial	Command sent by agent when agent wants to accept the preview request and the agent dials the customer OR supposed that the desktop sends on preview timer expiry.
AGTPreviewDialRESP	Asynchronous callback response to AGTPreviewDial
AGTPreviewCancel	Command sent by agent when agent wants to reject the preview request.
AGTPreviewCancelRESP	Asynchronous callback response to AGTPreviewCancel
AGTDialFailed	POM sends asynchronous notification if the customer call dial attempt fails.

Customer data

Desktop can send AGTGetCustomerDetails to get contact details. POM then sends back the system attribute values such as Title, First Name, and Last Name along with the custom attributes. With each attribute, POM also sends the attribute type. The attribute types POM supports are: READ_ONLY, WRITE, SCREEN_POP, MASKED_WRITE. The desktop must honor these types. AGTGetCustomerDetails can be sent after only after AGTCallNotify

Desktop can also change the value of the customers attribute by using AGTSetCustomerDetail. At a time desktop permits only one attribute. If the attribute is of READY_ONLY type POM sends back error.

API	Command/Response/Notification
AGTGetCustomerDetails	Command sent by agent when agent wants to get all customer details.
AGTGetCustomerDetailsRESP	POM sends asynchronous callback response in response to AGTGetCustomerDetails.
AGTSetCustomerDetail	Command sent by agent to change a customer attribute.
AGTSetCustomerDetailRESP	POM sends asynchronous callback response in response to AGTSetCustomerDetail.

Agent capabilities

POM controls the agent capabilities. Capabilities control the various actions that an agent can perform at any point of time. For more information about the capability matrix, see Capability Matrix . POM sends AGTCapabilitiesChanged notification when the agent's capabilities changes. Example: The capabilities control whether an agent can hold/consult another agent while talking to the customer. An agent can have

API	Command/Response/Notification

AGTCapabilitiesChanged	POM sends asynchronous notification when it detects change in agent desktop

approximately 17 capabilities. This permits the desktop developer to enable, disable, hide, or unhide buttons on the desktop based on these values. The desktop does not have to maintain desktop's own state machine. POM maintains that internally and the capabilities are a reflection of the current agent state on POM. The desktop developer adhere to these capabilities sent from POM.However, it is advisable to handle the cases where agent can send multiple requests. Desktop should not allow the agent to press the same command multiple times before the response/capabilities received from POM server.

Call state

POM notifies the desktop from time to time about the current call state. The desktop uses this information to notify the agent about the desktop's current call state according to POM. Some of the call states are: NoState, Preview, Idle, Dialing, Talking, Held, Wrapup, Consult, ConferenceOwner, ConferencePassive, Callback.

API	Command/Response/Notification
AGTCallStateChangedNotify	POM sends asynchronous notification when POM detects change in call state.

DNC

An agent can add a contact address to the DNC list. Desktop must run AGTAddToDNC to add a contact address to the DNC list. Agent can check whether the address is in DNC or not using AGTIsInDNC command.

API	Command/Response/Notification
AGTAddToDNC	Command sent by desktop to add a contact address to the contact list.
AGTAddToDNCRESP	Asynchronous callback response for AGTAddToDNC.
AGTIsInDNC	Command sent by desktop to check whether the address is in DNC or not
AGTIsInDNCRESP	Asynchronous callback response for AGTIsInDNC

Hold and unhold

An agent can put the customer on hold if required. The customer hears hold music (depending on the configuration) when the agent puts the customer on hold. After the customer is on hold, the agent can unhold the customer call.

API	Command/Response/Notification
AGTHoldCall	Command sent by desktop/agent to put the customer call on hold.
AGTHoldCallRESP	Asynchronous callback response for AGTHoldCall.
AGTUnholdCall	Command sent by desktop/agent to remove the customer from hold state back to talking state.
AGTUnholdCallRESP	Asynchronous callback response for AGTUnholdCall.

Send DTMF

An agent can send DTMF while talking to a customer. This feature is useful for situations wherein the agent is talking to an answering machine OR for consult/conference the consulted party is an IVR system. The agent can send one digit at a time.

For detecting Inband DTMF coming to MPP (IVR) you must turn on inband detection under **MPP Servers > VoIP Settings**, and set "Inband DTMF Detection Enabled" to "Yes". For hearing Inband DTMF ensures you have configured your gateway (CM) to send/receive Inband DTMF.

API	Command/Response/Notification
AGTSendDTMF	Command sent by desktop/agent to send DTMF.
AGTSendDTMFRESP	Asynchronous callback response for AGTSendDTMF

Consult

An agent can consult another party while talking to the customer. POM supports two types of consults: Agent, External. Consult is the first step required before attempting a transfer or a conference.The agent has to run AGTGetConsultTypes to get the supported consult types. After getting the supported types from POM, the agent has to select the type of consult and desktop needs to send AGTGetConsultDestsForType for a selected type of Consult. For Agent type of consult POM sends back a list of agents that belong to the same campaign as the consulting agent. If the agent selects External type of consult then, POM sends back the address list configured for the campaign. For external type destination list address list should be configured. Agent can also enter the free form number in external type of consult.

The agent has to use AGTConsultCall command request to start the consult. The other consulted agent gets a notification AGTConsultNotify when the two agents start to consult. POM supports only one consult at a time for an agent.

- Pending Consult: If Agent A tries to consult agent B, while agent B is already busy talking to a customer, then the Agent A sends the consult request to Agent B as a pending consult (AGTConsultPending notification) request to Agent B. Agent A can cancel this pending consult request if required, POM sends a maximum of two pending consult requests to any agent, if the agent is already in a call with the customer.

- Agents in Consult: Active agent is the consult starting agent, while POM considers Passive agent as the consulted agent. After both the agents are in consultation with each other, the customer hears music on hold (if configured on POM).
- Cancelling/Leaving a consult: After the agents are in consultation with each other, either of them can cancel the consult. The agents have to use AGTCancelConsult command request to cancel the consult. If the Active agent cancels the consult then the passive agent receives AGTConsultCancelled notification. If the Passive agent cancels the consult then the active agent receives AGTConsultCancelled notification from POM.
- External Consult: If an agent attempts to consult an external agent and if the dial attempt fails then POM sends back a notification to this agent AGTConsultDialFailed with appropriate reason.
- Customer: The agent cannot drop the customer directly while in consult. The agent has to end the consult before the customer can be dropped from the call. Ending a consult removes the passive agent from the consult. The customer call still stays active with the active agent. But the customer can drop the call at any point of time. If the customer drops the call, POM drops the consult and put the active agent in wrapup mode, moving the passive agent to Idle state.

API	**Command/Response/Notification**
AGTGetConsultTypes	Command sent by desktop to ensure the consult types by POM.
AGTGetConsultTypesRESP	Asynchronous callback response for AGTGetConsultTypes
AGTGetConsultDestsForType	Command sent by desktop to get a list of agents/external addresses which can be consulted for a particular type of transfer or conference.
AGTGetConsultDestsForTypeRESP	Asynchronous callback response for AGTGetConsultDestsForType
AGTConsultCall	Command sent by desktop/agent to start a consult.
AGTConsultCallRESP	Asynchronous callback response for AGTConsultCall
AGTCancelConsult	Command sent by desktop/agent to cancel a consult.
AGTCancelConsultRESP	Asynchronous callback response for AGTCancelConsult
AGTConsultNotify	POM sends asynchronous notification when a passive agent gets into a consult with an active agent.
AGTConsultPending	POM sends asynchronous notification when a passive agent is busy handling other calls.
AGTConsultCancelled	POM sends asynchronous notification to the other agent when an agent cancels an ongoing consult.
AGTConsultDialFailed	POM sends asynchronous notification when a consult attempt fails.

POM system plays a beep sound in following ways:

- When the agent clicks **Consult**, both the consult recipient and owner hears beep. The system does not play a beep to the contact.
- When the consult owner agent clicks **Cancel**, and the consult is established, both the consult recipient and owner hears beep. The system does not play a beep to the contact.
- When the consult recipient agent clicks **Cancel**, and the consult is established, the consult recipient, consult owner, and the contact hear the beep.
- When the agent clicks **Transfer** in the consult window, and the consult is established, the transfer recipient, transfer initiator, and the contact hear the beep.
- When the agent clicks **End Conference**, or **Leave Conference**, both the agents and the contact hears the beep.
- When the agent clicks **Changed Ownership** when the conference is going on, everyone in the conference, the agents and the contact hears the beep.
- When the contact disconnects the call while the conference is going on, both the agents hears the beep. The system does not play a beep to the contact.
- When the agent and the contact are talking, if the contact disconnects the call, the agent hears the beep.

Transfer

An agent (A) can transfer the current customer call to another agent (B). But first agent (A) has to enter in a consult with the other agent (B) before attempting a transfer. If an agent (A) transfers a call to the other consulted agent (B), POM sends AGTTransferNotify notification to the other agent (B) and handover the call control to the other agent (B). So, this means, POM transfers the call from the Active consulting agent (A) to the Passive consulted agent (B). POM removes the first agent (A) (earlier Active agent) from the call. So at the end of the transfer the consulted (B) (earlier Passive agent) becomes the owner of the customer call. The earlier Active agent (A) is free to accept another call from POM.

To perform a transfer an agent has to run AGTCompleteTransfer command. After the transfer is complete, it moves the customer out of hold state and is now able to talk to the agent (B).

API	Command/Response/Notification
AGTCompleteTransfer	Command sent by desktop/agent to start a transfer.
AGTCompleteTransferRESP	Asynchronous callback response for AGTCompleteTransfer.
AGTTransferNotify	POM sends asynchronous notification to the consulted agent when it completes the transfer

Callbacks

An agent can schedule a callback, while in a preview or while talking to a customer or during wrapup. An agent has to first determine the types of callbacks that the campaign

supports. As of now POM supports the following types of callbacks: Standard, Agent and Campaign. The agent has to run AGTGetCallbackTypes command to determine the supported callback types. After selecting the callback type the agent has to run AGTGetCallbackDestsForType command to get a list of destinations supported for a particular type of callback. To create a callback the agent has to run AGTCreateCallback. The agent can specify a free form number also while creating a callback.

Types of callback: A Standard type of callback means that any agent who has the same skillset as the callback creator can get the callback when the callback time arrives. An Agent type of callback means that the preferred agent for the callback (when the timer expires) is the callback creator, but if this agent is unavailable at that point of time, POM gives the callback to any agent which has the same skillset as the callback creator. A Campaign type of callback means to give the callback to any agent which matches the skillset of the selected campaign.

Callback Time: The time when the POM selects an agent depending on the callback type created. After the POM selects the agent, POM sends a pending callback notification AGTPreviewCallbackPending to the agent few seconds before the scheduled callback time (depending on the POM configuration). The callback notification notifies the agent about a pending callback requestA callback is a preview call for the agent. The agent can decide to cancel the callback, or reschedule it or accept it. If the agent accepts the preview, it has to run AGTPreviewDial command. If the dial fails POM sends back a notification AGTPreviewDial failed with an appropriate error code. If the agent decides to reject the pending callback, then the agent has to run AGTPreviewCancel command and then wrapup the call by providing an appropriate completion code. The agent can reschedule the callback also by running AGTCreateCallback again.

The callback preview timer can be set to zero on POM global configuration page. If the preview timer is received as zero in callback call notification (AGTCallNotify for callback), then desktop need to dial the default number immediately.

Desktop needs to check two conditions for that, timeout value as zero and the contact type is of type Callback to dial the default number immediately.

Callback expiry: If an agent receives a pending callback request and if by the time the agent accepts the callback then the callback expires. Then POM cancels the callback and sends a notification AGTPendingCallbackCancelled to this agent.

API	**Command/Response/Notification**
AGTGetCallbackTypes	Command sent by desktop/agent to determine types of callbacks supported by the current campaign.
AGTGetCallbackTypesRESP	Asynchronous callback response for AGTGetCallbackTypes.

AGTGetCallbackDestsForType	Command sent by desktop/agent to determine destinations permitted for a selected type of callback.
AGTGetCallbackDestsForTypeRESP	Asynchronous callback response for AGTGetCallbackDestsForType.
AGTCreateCallback	Command sent by desktop/agent to create a callback.
AGTCreateCallbackRESP	Asynchronous callback response for AGTCreateCallback.
AGTPreviewDial	Command sent by desktop/agent to dial the customer after receiving a pending callback.
AGTPreviewDialRESP	Asynchronous callback response for AGTPreviewDial.
AGTPreviewCancel	Command sent by desktop/agent to cancel the preview.
AGTPreviewCancelRESP	Asynchronous callback response for AGTPreviewCancel.
AGTPreviewCallbackPending	POM sends asynchronous notification to the selected agent who receives the callback.
AGTPreviewCallbackCancelled	POM sends asynchronous notification to the selected agent for the pending callback, when the callback expires.

Note: For POM server, AGTGetCallbackDestsForType may need more processing based on the Agent list or Campaign list (Agent or Campaign callback type) and data for this command may be huge in size. So it is recommended that use this command only when needed or only when agent selects that particular type of callback.. For Agent Owned Callback (self-agent callback) there is no need to send this command to get agent list, desktop has all the information to create callback for the current (self) agent.

Earlier for agent type of callback, the list of agents was fetched from Communication Manager. In the current release of POM, we fetch the list of agents from POM database cache

Conference

An agent (A) can conference the current customer call to another agent (B). But first agent (A) has to enter in a consult with the other agent (B) before attempting a conference. POM supports only 3 party conference (that is one customer and 2 agents). If an agent (A) conferences a call to the other consulted agent (B), POM sends AGTConferenceNotify notification to the other agent (B) and add the Passive Agent (B) in a conference with the customer.

To start a conference agent (A) has to run AGTStartConf command. When the agent completes the conference, the agent moves the customer from hold state to in-conference with both agents.

Ending the conference: Both agents can get themselves out of the conference albeit in a different way. Conference owner is the agent (A) who started the conference, while the consulted agent (B) is the passive party in the conference. If conference owner agent (A) wants to drop the conference, it has to run AGTEndConf command. In this case POM sends drop the passive agent (B) from the conference and send a notification AGTConferenceEnded to the passive agent (B). If the passive agent (B) wants to leave

the conference, then it has to also run AGTEndConf command. In this case POM sends a notification AGTConferenceEnded to the conference owner.

Customer: The agent cannot drop the customer directly while in conference. The agent has to end the conference before the customer can be dropped from the call. Ending a conference removes the passive agent from the conference. The customer call still stays active with the active agent. But the customer can drop the call at any point of time. If the customer drops the call, POM drops the conference and put the conference owner agent in wrapup mode, while it moves the passive agent to Idle state.

Conference Ownership: The conference owner agent (A) can transfer the conference ownership to the passive agent (B) after the conference is ON. To transfer the conference ownership from the conference owner agent (A) to passive agent (B), agent (A) has to run AGTConfChangeOwnership command. The agent transfers the ownership to the passive agent and then the passive agent receives a notification from POM AGTConferenceOwnershipChanged. After the agent transfers the ownership, the conference owner agent (A) now becomes the passive agent in the conference, while the earlier passive agent (B) now becomes the conference owner.

External Conference: In external conference, the agent can't change ownership to external party and thus can't leave the conference. If it leaves the conference, it will also drop the call between consulted party and customer. So in order to allow the agent to leave the external conference, the current release provides the capability to transfer the call to external party and can leave the conference.

In external conference, POM server will send the transfer capability set to true to enable the transfer button.

API	**Command/Response/Notification**
AGTStartConf	Command sent by desktop/agent to start the conference.
AGTStartConfRESP	Asynchronous callback response for AGTStartConf
AGTEndConf	Command sent by an agent in a conference to end the conference.
AGTEndConfRESP	Asynchronous callback response for AGTEndConf.
AGTConferenceNotify	POM sends asynchronous notification to the passive agent in the conference while has moved from consult to conference state.
AGTConferenceEnded	POM sends asynchronous notification to the conference owner if the passive agent leaves the conference. POM can also send this notification to the passive agent in the conference if the conference owner ends the conference.
AGTConfChangeOwnership	Command sent by conference owner desktop/agent to transfer the conference ownership to the passive agent in the conference.
AGTConfChangeOwnershipRESP	Asynchronous callback response for AGTConfChangeOwnership.

AGTConfOwnershipChanged	POM sends asynchronous notification to the passive agent when the conference owner transfers the conference

Wrapup

Redial: An agent can redial the customer while in wrapup mode. The agent can perform this action if the customer call drops or for other situations, by an error. The agent has to run AGTRedial command to perform a redial to the customer. The agent can select the number to dial from the available contact phone addresses. Agent can also enter free form number to redial.

Extending Wrapup: If a wrapup time is configured for the campaign, then when the configured wrapup time expires the desktop autowrapups the call with the completion code supplied with the release response/notification. But if required the agent can ask for an extension in wrapup time. Based on the configuration in the POM campaign, an agent can have multiple extensions.

Wrapping up: An agent has to send a completion code for wrapping up the call. Desktop can send AGTGetCompCodes command to determine the list of available completion codes for the current job.

API	**Command/Response/Notification**
AGTRedial	Command sent by desktop/agent to redial the customer.
AGTRedial RESP	Asynchronous callback response for AGTRedial.
AGTGetCompCodes	Command sent by desktop/agent to determine the list of completion code.
AGTGetCompCodesRESP	Asynchronous callback response for AGTGetCompCodes.
AGTExtendWrapup	Command sent by desktop/agent to get an extension time in wrapup mode.
AGTExtendWrapup RESP	Asynchronous callback response for AGTExtendWrapup.
AGTWrapupContact	Command sent by desktop/agent to wrapup the call.
AGTWrapupContact RESP	Asynchronous callback response for AGTWrapupContact.

Agent notes

An agent can create some notes and save them while talking to the customer. These notes are useful for consult/conference/transfer. The agent has to run AGTAddAgentNote to add a note. To get the latest set of notes the agent has to run AGTRefreshAgentNotes. An agent can also create agent notes while creating a callback. So, when the agent invokes AGTRefreshAgentNotes, POM sends back all agent notes with the callback notes.

API	**Command/Response/Notification**
AGTAddAgentNote	Command sent by desktop/agent to add a note.

AGTAddAgentNote RESP	Asynchronous callback response for AGTAddAgentNote.
AGTRefreshAgentNotes	Command sent by desktop/agent to get all saved agent notes for the current call.
AGTRefreshAgentNotesRESP	Asynchronous callback response for AGTRefreshAgentNotes.

Error

Asynchronous responses of the commands can return error codes. The desktop must run AGTGetErrorString(obsolete method) or AGTGetErrorInfo to get the error message.

API	Command/Response/Notification
AGTGetErrorString	Command sent by desktop/agent for error details.(This is obsolete method. AGTGetErrorInfo should be used instead.
AGTGetErrorStringRESP	Asynchronous callback response for AGTGetErrorString.
AGTGetErrorInfo	Command sent by desktop/agent for error details
AGTGetErrorInfoRESP	Asynchronous callback response for AGTGetErrorInfo

Zones

POM supports zoning. So an agent has to notify POM which zone the agent belongs to. Agent needs to provide zone name while login into POM.

The agent manager process on POM can be managing one or more zones. So, the client can possibly make multiple socket connections to the agent manager, one for each zone. If different POM servers locate the agent managers then it makes the socket connections accordingly.

POM Agent Factory Methods and Commands

init

Syntax

Boolean init (PAMSocketInfo[] pamAddresses)

Parameters

pamAddress List of IP address and corresponding port of POM agent managers.

Description

The desktop developer must run this to initialize the library. Before using the library, the desktop developer must make the first API call that is Boolean init (PAMSocketInfo[] pamAddresses).

Note:

The init method return false in the first attempt if the connection between POM API DLL and POM server is not established and after that it always returns true whether the connection to POM server is established or not. So if the DLL init fails it is required to restart the desktop application.

---

deinit

Syntax

deinit()

Parameters

None

Description

The desktop developer must run this when the desktop is shutting down to clean-up POM Desktop library's resources used

Note: While doing deinit operation, there are some exceptions thrown in the log as the threads are interrupted at that time. However, there is no functionality impact because of this.

If you want to logoff and login agent again in the same instance of agent desktop, then do not initiate deinit() function after logoff. deinit() function should be used when you are closing the agent desktop to release the resources.

---

getPOMAgent

Syntax

POMAgent getPOMAgent(String id, POMAgentHandlerInterface handler)

Parameters

id	Agent ID
handler	Callback handler object.

Description

The desktop developer must run this to get an agent object. The desktop developer uses this agent object to send commands for this agent. POM in turn can invoke the handler for asynchronous responses and notifications using callback thread. Desktop should free this callback thread as soon as possible, so that this thread can process the responses of other commands. Ensure that it uses a valid POMAgent object before making any command request.

Note: This method should be called before every login request (AGTLogon) to create new POMAgent object.

---

removePOMAgent

Syntax

Boolean removePOMAgent (String id)

Parameters

id Agent ID.

Description

The desktop developer must invoke this to when the desktop developer has completed using the POMAgent object which was obtained with getPOMAgent. Ensure that this is invoked after processing the response for AGTLogoff command.

Commands

Asynchronous response for each command is available from POM in the corresponding command"RESP" method. Each command also has a int return type. Each command request can return one of the following error code based on the error:

Error Code: 9999 (POM_NOT_AVAILABLE)

Library sends it back if any command cannot be sent to POM. This gets generated internally and sent back with the appropriate response RESP.

Error Code: 9998 (PAM_NOT_AVAILABLE_FOR_ZONE)

Library sends it back if an agent manager does not manages the zone provided in the AGTLogon command. This is sent only for AGTLogon command.

Error Code: 9997 (INVALID_ARGUMENT)

Library sends it back if any argument provided to the API is not right. This error can be sent by any command.

Error Code 9996 (SDK_Failure)

If the library is unable to send the command to POM, then this error code is generated.

This error can be sent by any command.

AGTLogon

Command Syntax

int AGTLogon (String agentExt, String pwd, boolean isForce, String locale, String timeZone, String zoneName)

Parameters

agentExt This is agent extension

pwd Password used for authenticating the agent on CC Elite. POM server will accept empty string in password field as empty password is supported on Communication Manger for agent. So POM server will allow login if the agent password on Communication Manager is empty and empty password is provided from desktop.

Note: null in password will not be accepted.

isForce isForce is a flag, desktop uses to permit a forceful login to POM. isForce is true when the desktop wants to permit this agent to be able to login in the following scenarios:

- Even if this agent has already logged in from a different desktop.
- The desktop/agent PC crashed while the agent was already logged in.

Note:
Force login agent will drop the customer call and nailing of the agent. So if the agent is in talking state, agent should wait for the current call to complete before force login.

locale Locale of the agent.

timeZone Time zone of the agent.

Note:

POM restricts only Null values of locale and time Zone during login.

zoneName Zone of the agent.

Description

The agent sends this command from the desktop when the agent wants to login to POM. This logon API is for CCElite desktop. POM can then use this agent for a campaign when the agent becomes ready.

Logon for AACC

Command Syntax

int AGTLogon(String agentExt, String pwd, boolean isForce, String locale, String timeZone, String zoneName, String agentName, POMAgentSkill[] agentSkills)

Parameters:

agentExt This is the agent extension. This is the SIP URL that is used by POM to send the nailing call in AACC mode.

pwd This parameter is not used in case of AACC mode as AAAD is already authenticating agent credentials.

isForce This is a flag which can be used by the desktop to allow a forceful login to POM. It is set to true when the desktop wants to allow this agent to be able to login in the following scenarios:

- - even if this agent has already logged in from a different desktop.

  the desktop/agent PC crashed while the agent was already logged in.

locale Locale of the agent.

timeZone Timezone of the agent.

Note:

POM restricts only Null values of locale and timeZone during login.

zoneName Zone of the agent.

agentName Displays the name of the agent. The Agent Manager references the agent name while integrating with AAAD.

agentSkills The agent skillset array to be sent by AAAD.

Description

The agent sends this command from the desktop when the agent wants to login to POM. POM can then use this agent for a campaign when the agent becomes ready. This login command is for AAAD agent desktop

Response Syntax

AGTLogonRESP(int result)

Return Value Parameters

result "0" indicates success.

Description

Asynchronous callback response for AGTLogon.

Error codes

Error code	Error message	Description
2	This agent is not registered with the system	POM system does not recognize the logged in agent. You can force login again.
6	Unable to verify password	You can see this message only in POM integration with CC Elite. POM cannot get the password from AES. Check the password of the agent and check the AES Web service.
7	Agent is already logged in	POM displays the message if the agent has already logged in. You can force login again.
9	Agent skills not found	Check the agent ID. The agent ID must match the agent ID specified in Contact Center. If the error message persists, check the AES web services.
10	Unable to change the state of the agent	POM cannot change the agent state. Login again to the POM system.
11	Internal error. Unable to login	The agent cannot login to the POM system. Login again to the POM system.
12	Login failure. Zone not found	The agent cannot login because of zone error. Check the zone for which the agent logs in. The zone must match one of the zones specified in POM.
13	Login failure. Invalid locale	The agent cannot login because of locale error. Check the locale of the agent. The locale must not be Null. If Null, specify a locale for the agent. **Note:** POM checks only Null value for locale values, and does not restrict any other string value
15	Login failure. Invalid Timezone	The agent cannot login because of time zone error. Check the time zone of the agent. The time zone must not be Null. If Null, specify time zone of the agent. **Note:** POM checks only Null value for time zone values, and does not restrict any other string value
16	Login failure. Invalid Agent Name	You can see this message only if you have POM integration with AACC. The agent name cannot be Null. You must specify a value for agent name.

17	Login failure. Authentication of agent failed.	You see this message only if you have POM integration with CC Elite. Check the agent password. The agent password must match the agent password specified in the Contact Center.

AGTLogoff

Command Syntax

AGTLogoff()

Parameters

None

Description

The agent sends this command from the desktop when it wants to logout from POM. An agent can send this command ONLY when the command is in “NotReady” state.

Error code	Error message	Proposed solution
541	Unable to logoff. Please move to not ready state	Agent needs to move to not ready state and then retry to logoff

Response Syntax

AGTLogoffRESP(int result)

Return Value Parameters

result “0” indicates success.

Description

Asynchronous callback response for AGTLogoff.

AGTStateChange

Command Parameters

int AGTStateChange(POMAgentState agentState, String reasonCode, String reasonName, Boolean hasWalkedAway)

Parameters

agentState An agent can specify the state to which the agent wants to transition to from the current state. The current agent states are:

- Ready: The agent is ready to pick calls.
- NotReady: The agent wants to take a break
- PendingNotReady: The agent is in the middle of the call and has issued a request to go to NotReady state, but POM cannot send the agent to NotReady.

reasonCode reasonCode is AUX or reason code the agent selects from the desktop while going to NotReady state.

reasonName reasonName is the AUX or reason code the agent selects from the desktop while going to NotReady state.

hasWalkedAway This flag is set to true if it considers the agent as walkaway agent. If an agent does not perform any actions for two consecutive calls, then the desktop considers that the agent has walked away and changes the call state to NotReady, so that it does not send the next call to this agent. Except for a nontimed preview campaign an agent does not necessarily have to click buttons on the desktop to handle a call, considering that the customer disconnects the calls and implements the autowrapup

Description

The agent sends this command when it wants to go Ready or NotReady. An agent goes to Ready state when it wants to accept calls from POM. Unless the agent is ready POM will not handover a call to the agent. If the agent is in NotReady state, POM will not pass on the outbound call to the agent. An agent can go to NotReady state from a Ready state only after the current call is done OR to put it in other words, the agent can go to NotReady state only if it is in Idle state. If the agent is in the middle of a call OR in wrap up state, the agent can issue a NotReady request. POM accepts this request and puts the agent in PendingNotReady state. Once the agent is done with disposing the existing call, POM will immediately put the agent in NotReady state and will not give it a new call until it goes back to Ready State. An important point to note is that if this agent has pending consults + pending callbacks, POM will not move the agent from PendingNotReady to NotReady until the pending consults + pending callbacks are completed.

Error codes

Error code	Error message	Description
63	State change request sent more than once with the same reason code	When the agent tries to change the agent state to NotReady with the same reason, POM returns this error. The agent must select other reason code to mention different reason for NotReady.

64	Unable to change the current state	POM does not allow the agent to change the state to NotReady or Ready from the current state.

Response Syntax

AGTStateChangeRESP(POMAgentState pomAgentState, int result)

Return Value Parameters

pomAgentState	Agent state of type POMAgentState
result	"0" indicates success.

Description

Asynchronous callback response for AGTStateChange.

AGTHoldCall

Command Syntax

int AGTHoldCall (String sessionID)

Parameters

sessionID	Unique ID of the session for the entire contact processing duration.

Description

The agent sends this command when it wants to put the customer on hold. The customer can associate a Hold application with a strategy. The agent plays the hold application to the customer if an agent puts it on Hold. If the customer disconnects the call when an agent has put the customer on Hold, POM drops the call and moves the agent to Wrapup state.

Error codes

Error code	Error message	Description
101	Unable to hold	Unable to hold call due to telephony errors.
102	Unable to hold as call is disconnected	This error comes when agent tries to hold the call which is disconnected.
103	Unable to hold as agent is not on call	This error comes when agent is not on call.

Response Syntax

AGTHoldCallRESP(String sessionID, int result)

Return Value Parameters

sessionID Unique ID of the call for the entire contact processing duration.

result “0” indicates success.

Description

Asynchronous callback response for AGTHoldCall.

AGTUnholdCall

Command Syntax

int AGTUnholdCall (String sessionID)

Parameters

sessionID Unique ID of the contact for the entire contact processing duration.

Description

The agent sends this command when it wants to take the customer out of hold state.

Error codes

Error code	Error message	Description
121	Unable to hold.	Unable to unhold call due to telephony errors.
122	Unable to hold as call is disconnected.	This error comes when agent tries to unhold the call which is disconnected.
123	Unable to hold as agent is not on call.	This error comes when agent is not on call.

Response Syntax

AGTUnholdCallRESP(String sessionID, int result)

Return Value Parameters

sessionID Unique ID of the call for the entire contact processing duration.
result “0” indicates success.

Description

Asynchronous callback response for AGTUnHoldCall.

AGTReleaseLine

Command Syntax

int AGTReleaseLine(String sessionID)

Parameters

sessionID Unique ID of the call for the entire contact processing duration.

Description

The agent sends this command when the agent wants to disconnect the customer call. POM moves the agent to Wrapup state after the agent disconnects the customer call.

An agent can give this command only when the agent is in Talking state.

Error codes

Error code	Error message	Description
81	Unable to release the call	POM cannot release the call for the agent. The agent has some pending commands which the agent must address.
82	Call is already disconnected	This error comes when agent tries to disconnect the call which is already disconnected.

Response Syntax

AGTReleaseLineRESP (WrapupData wrapupDetails, String sessionID, int result)

Return Value Parameters

wrapupDetail This structure contains the following 3 values:

- *acwMaxTime*: Maximum time permitted to an agent to wrapup the call.
- *acwExtendable*: Boolean value indicating if the agent can request more wrapup time.
- *defaultCompCode*: The completion code POM uses to wrapup the call.

sessionID Unique ID of the call for the entire contact processing duration.

result “0” indicates success.

Description

Asynchronous callback response for AGTReleaseLine.

AGTGetCompCodes

Command Syntax

int AGTGetCompCodes(String sessionID)

Parameters

sessionID Unique ID of the call for the entire contact processing duration.

Description

The agent sends this command when the agent wants a list of disposition codes linked with the campaign that the agent is handling. POM sends back the custom completion code list linked with the campaign that this agent is part of.

The agent can give this command in Talking/Wrapup state.

Note:

It is not recommended to use this command every time before wrapping up the call. Use this command for the first customer call on a campaign (or for the first call after AGTJobAttached notification) and cache the completion code list at desktop side and refresh it at every first call after job attach. Use this cached completion code list while wrapping up the call.

Reponse Syntax

AGTGetCompCodesRESP(POMCompletionCode[] completionCodesList, String sessionID, int result)

Return Value Parameters

compCodeList List of custom completion codes linked with the current job.

sessionID Unique ID of the call for the entire contact processing duration.

result “0” indicates success

Description

Asynchronous callback response for AGTGetCompCodes.

AGTWrapupContact

Command Syntax

int AGTWrapupContact(POMCompletionCode completionCode, String sessionID)

Parameters

completionCode An agent can provide a completion code (disposition code) to dispose the call.

sessionID Unique ID of the call for the entire contact processing duration.

Description

The agent sends this command when the agent wants to wrap up the current call. The agent has to select a completion code to wrap up the call. Desktop needs to show the completion code list that is received by invoking command AGTGetCompCodes OR completion code received from AGTAutoReleaseLine or AGTReleaseLine.

Error codes

Error code	Error message	Description
201	Unable to wrapup as contact record not found	This error comes when contact record is cleared from the agent object.
202	Unable to wrapup with this completion code	POM cannot update the provided completion code for the contact. Try to update the contact with a different completion code.

Reponse Syntax

AGTWrapupContactRESP(String sessionID, int result)

Return Value Parameters

sessionID Unique ID of the call for the entire contact processing duration.

result “0” indicates success.

Description

Asynchronous callback response for AGTWrapupContact.

AGTExtendWrapup

Command Syntax

int AGTExtendWrapup(String sessionID)

Parameters

sessionID Unique ID of the call for the entire contact processing duration.

Description

The agent sends this command when it wants to extend the wrapup time of the current call. The agent gives this command only in wrapup state. Agent can specify a wrapup time and number of wrapup extensions in the strategy editor.

- Wrapup time: Time value in seconds before which the agent must provide a disposition for the call.
  Wrapup extensions: If an agent needs more time to wrapup a call, then it can ask for an wrapup extension from POM. If the POM user has provided extensions in the campaign strategy, then POM sends back the extension time permitted for this agent. The agent can get more than the wrapup time (above) to dispose the current call.

Reponse Syntax

AGTExtendWrapupRESP (WrapupData wrapupDetails, String sessionID, int result)

Return Value Parameters

wrapupDetail This structure contains the following 3 values:

- *acwMaxTime*: Maximum time permitted to an agent to wrapup the call.
- *acwExtendable*: Boolean value indicating if the agent can request more wrapup time.
- *defaultCompCode*: The completion code which POM uses to wrapup the call.

sessionID Unique ID of the call for the entire contact processing duration.

result “0” indicates success.

Description

Asynchronous callback response for AGTExtendWrapup.

AGTGetConsultTypes

Command Syntax

int AGTGetConsultTypes(String sessionID)

Parameters

sessionID Unique ID of the call for the entire contact processing duration.

Description

The agent sends this command to determine the types of consult supported for the current job that the attaches the agent

The supported types are:

- Agent: You can consult any agent which is ready + nailed + attached to the same job
- External – You can consult any external party. In POM you can define external party by adding a contact in Address book. You can also enter a free form number for consultation.

The agent can send this command in Talking state only.

Response Syntax

AGTGetConsultTypesRESP(POMDestinationType[] destinationTypes, bool allowFreeForm, String sessionID, int result)

Return Value Parameters

destinationTypes	POMDestinationType enumerator
allowFreeForm	Boolean flag indicating if the agent can enter a free form number for consultation.
sessionID	Unique ID of the call for the entire contact processing duration.
result	“0” indicates success.

Description

Asynchronous callback response for AGTGetConsultTypes.

AGTGetConsultDestsForType

Command Syntax

int AGTGetConsultDestsForType(POMDestinationType destinationType, String sessionID)

Parameters

destinationType	The selected transfer type Agent/External.
sessionID	Unique ID of the call for the entire contact processing duration.

Description

The agent sends this command after selecting a consultation type. The can send this command only after AGTGetConsultTypes and in Talking state.

Note:

POM supports maximum packet size of 16000 bytes. If the list of agents or list of external addresses exceeds 16000 bytes, POM truncates the list of agents or list of external addresses.

For POM server, AGTGetConsultDestsForType may need more processing based on the Agent list or External addresses and data may be huge in size. So it is recommended that use this command only when needed or only when agent selects that particular type of consult.

Error codes

Error code	Error message	Description
382	No destinations available	POM cannot send external destinations for consult. The administrator must select some addresses for the campaign for which the contact is being
383	Agents are not available	POM cannot send agent destinations for consult. No available agents for consult in the running job

Response Syntax

AGTGetConsultDestsForTypeRESP(POMDestination[] destinations, String sessionID, int result)

Return Value Parameters

destinations List of available agents for the selected type of consult.

sessionID Unique ID of the call for the entire contact processing duration.

result "0" indicates success.

Description

Asynchronous callback response for AGTGetConsultDestTypes.

AGTConsultCall

Command Syntax

int AGTConsultCall (POMDestination destination, String sessionID)

Parameters

destination POM sends the selected destination from the list in response to AGTGetTransferDestsForType or AGTGetConfDestsForType.

sessionID Unique ID of the call for the entire contact processing duration.

Description

The agent sends this command after selecting a destination that returns by either AGTGetConsultDestsForType. An agent has to consult before attempting a transfer or a conference.

Error codes

Error code	Error message	Description
141	Unable to consult as the selected agent is currently not ready	Agent cannot consult with other agent if the agent is in either Not Ready, or the agent is unnailed, If the agent selects such agent, POM returns the error. The agent must choose some other agent for consult.
142	Unable to consult as the selected agent has logged out	Agent cannot consult with other agent if the agent is in process of logging out from the system. If the agent selects such agent, POM returns the error. The agent needs to choose some other agent for consult.
143	Unable to consult as the selected destination is invalid	If the agent desktop sends invalid or null destination, POM returns the error.
144	Unable to consult as the selected agent already has maximum pending consults	Agent cannot consult with other agent if the agent has more than 2 pending consults.

Response Syntax

AGTConsultCallRESP(String sessionID, int result)

Return Value Parameters

sessionID Unique ID of the call for the entire contact processing duration.

result “0” indicates success.

Description

Asynchronous callback response for AGTGetConsultDestTypes.

AGTCompleteTranser

Command Syntax

int AGTCompleteTransfer(String sessionID)

Parameters

sessionID Unique ID of the call for the entire contact processing duration.

Description

The agent sends this command when it wants to transfer the consultation call to the passive agent in the consult. POM transfers the customer call to the passive agent, who now becomes the call owner. Call owner moves the active agent which issues this command directly to “Idle” state assuming it ready for the next call.

Error codes

Error code	Error message	Description
421	System error. Unable to consult	POM cannot transfer the call. The agent must retry the transfer or cancel the consult.

Response Syntax

AGTCompleteTransferRESP (bool canDispose, POMWrapupDetails wrapupDetails, String sessionID, int result)

Return Value Parameters

canDispose For an external transfer canDispose is true and the agent has to provide disposition for the call

wrapupDetails This structure contains the following 3 values:

- *acwMaxTime*: Maximum time permitted to an agent to wrapup the call.
- *acwExtendable*: Boolean value indicating if the agent can request more wrapup time.
- *defaultCompCode*: The completion code which POM uses to wrapup the call.

sessionID Unique ID of the call for the entire contact processing duration.

result “0” indicates success.

Description

Asynchronous callback response for AGTCompleteTransfer.

AGTCancelConsult

Command Syntax

int AGTCancelConsult (String destAgentID, String sessionID)

Parameters

destAgentID The selected destination for which the consult intends to cancel request.

sessionID Unique ID of the call for the entire contact processing duration.

Description

The active agent in a consult (customer call owner) sends this command to POM requesting it to cancel the ongoing consult with the passive agent. POM informs the passive agent about this action and makes the passive agent ready to accept a new call by moving the passive agent to Idle state. Using this command, you can cancel a pending consult request.

Error codes

Error code	Error message	Description
401	System error. Unable to cancel the consult	If the system cannot find the instance of consult, System displays the error on the agent desktop.

Response Syntax

AGTCancelConsultRESP(String sessionID, int result)

Return Value Parameters

sessionID Unique ID of the call for the entire contact processing duration.

result “0” indicates success.

Description

Asynchronous callback response for AGTCancelConsult.

AGTStartConf

Command Syntax

int AGTStartConf (String sessionID)

Parameters

sessionID Uique ID of the call for the entire contact processing duration.

Description

The agent sends this command when it wants to conference with the passive agent in the consult. POM joins the customer call with the active and the passive agents. The active agent’s state changes to ConferenceOwner, while the passive agent becomes ConferencePassive.

Response Syntax

AGTStartConfRESP(String sessionID, int result)

Return Value Parameters

sessionID Unique ID of the call for the entire contact processing duration.

Return Values

result "0" indicates success.

Description

Asynchronous callback response for AGTStartConf.

AGTEndConf

Command Syntax

int AGTEndConf (String sessionID)

Parameters

sessionID Unique ID of the contact for the entire contact processing duration.

Description

Agents involved in a conference can send this command when agent have to end the conference. The agent can send this command only in conference state. When the agent ends the conference, it informs the passive agent about it and the active agent moves back from conference state to Talking state. The passive agent can now receive new calls and it moves the conference owner from Conference to Talking state.

Error codes

Error code	Error message	Description
303	Unable to end the conference as passive agent not found	If the system cannot find any passive agent in the agent buffer, the system displays the error on the agent desktop.

Response Syntax

AGTEndConfRESP(String sessionID, int result)

Return Value Parameters

sessionID Unique ID of the call for the entire contact processing duration.

result "0" indicates success.

Description

Asynchronous callback response for AGTEndConf.

AGTConfChangeOwnership

Command Syntax

int AGTConfChangeOwnership (String sessionID)

Parameters

sessionID	Unique ID of the call for the entire contact processing duration.

Description

The Conference owner can give this command to transfer the conference ownership to the passive agent in the conference. POM then moves the passive agent from ConferencePassive state to ConferenceOwner state and vice-versa for the original conference owner. This command should be used for internal conference between two POM agents.

Response Syntax

AGTConfChangeOwnershipRESP(String sessionID, int result)

Return value parameters

sessionID	Unique ID of the call for the entire contact processing duration.
result	“0” indicates success.

Description

Asynchronous callback response for AGTConfChangeOwnership.

AGTRedial

Command Syntax

int AGTRedial (POMContactNumber contactNumber)

Parameters

contactNumber	The contact which needs to be redialed.

Description

An agent sends this command when it wants to redial the customer that it was talking to before disconnecting the call. AGTRedial is a useful command which is used if the customer call disconnects because of some reason (such as network not reachable/weak signal/disconnect by mistake). The agent can dial any number from the available numbers assigned to the contact or free from number. The agent desktop can send this command in wrapup state.

Error codes

Error code	Error message	Description
362	Unable to redial as the address is invalid	If the customer number to redial is Null, the system displays the error on the agent desktop.

Response Syntax

AGTRedialRESP(String sessionID, int result)

Return Value Parameters

sessionID Unique ID of the call for the entire contact processing duration.

result "0" indicates success.

Description

Asynchronous callback response for AGTRedial.

AGTSendDTMF

Command Syntax

int AGTSendDTMF (String dtmf, String sessionID)

Parameters

dtmf The dtmf digit that agent wants to send on the call.

sessionID Unique ID of the call for the entire contact processing duration.

Description

An agent can send this command if it wants to send dtmf during the call. This command is useful if the agent is interacting with an IVR or an answering machine. The agent can send one dtmf at a time.

Error codes

Error code	Error message	Description
341	Cannot send DTMF as invalid data received	If the DTMF input is invalid or Null, the system displays the error on the agent desktop.
342	Unable to send DTMF	Unable to send DTMF because of telephony error.

343	System error. Unable to send DTMF	If the telephony fails to send DTMF input, the system displays the error on the agent desktop.

Response Syntax

AGTSendDTMFRESP(String sessionID, int result)

Return Value Parameters

sessionID Unique ID of the call for the entire contact processing duration.

result “0” indicates success.

Description

Asynchronous callback response for AGTSendDTMF.

AGTGetCallbackTypes

Command Syntax

int AGTGetCallbackTypes (String sessionID)

Parameters

sessionID Unique ID of the call for the entire contact processing duration.

Description

An agent can send this command to check the types of callbacks which POM supports. POM supports the following callback types:

- Standard: Any agent having the right skill (skill assigned to the job in which schedules the callback) receives the callback.
- Agent: Preferrably POM tries to send the callback to the agent which scheduled the callback OR Any agent having the right skill (skill assigned to the job in which schedules the callback).
- Campaign: Any agent having the right skill (skill assigned to the campaign selected for the callback) receives the callback.

An agent has to be ready to receive a callback.

Response Syntax

AGTGetCallbackTypesRESP(POMCallbackType[] callbackTypes, String defaultExpiryTime, String sessionID, int result)

Return Value Parameters

callbackTypes	POMCallbackTypes enumerator (Agent, Standard, Campaign)
defaultExpiryTimes	Callback default expiry time value.
sessionID:	Unique ID of the call for the entire contact processing duration.
result:	"0" indicates success.

Description

Asynchronous callback response for AGTGetCallbackTypes.

AGTGetCallbackDestsForTypes

Command Syntax

int AGTGetCallbackDestsForTypes (String sessionID)

Parameters

sessionID	Unique ID of the contact for the entire contact processing duration.

Description

An agent can send this command to request POM to send back the list of available destinations for the selected callback type. An agent can set callbacks only in Talking/Wrapup/Preview state.

Note:

POM supports maximum packet size of 16000 bytes. If the list of campaigns or list of agents exceeds 16000 bytes, POM truncates the list of campaigns or list of agents.

Command Syntax

int AGTGetCallbackDestsForType(POMCallbackType callbackType, String zoneName, String sessionID)

Parameters

callbackType	The callback type which needs to be scheduled.
zoneName	Callback destination list of only this zone is expected.
sessionID	Unique ID of the contact for the entire contact processing duration.

Description

An agent can send this command to request POM to send back the list of available destinations for the selected callback type. An agent can set callbacks only in Preview, Talking and Wrapup state.

Note:

POM supports maximum packet size of 16000 bytes. If the list of campaigns or list of agents exceeds 16000 bytes, POM truncates the list of campaigns or list of agents.

Error codes

Error code	Error message	Description
221	Cannot get destinations as	If the callback type in API is Null, the system displays the error on the agent desktop,
222	Unable to get zone of the contact	If the system cannot find the zone of a contact on which the system creates the callback, the system displays the error on the agent desktop.
223	Unable to get campaigns for the organization	If the system cannot find any campaign for the organization, the system displays the error on the agent desktop.

Note: For POM server, AGTGetCallbackDestsForType may need more processing based on the Agent list or Campaign list (Agent or Campaign callback type) and data for this command may be huge in size. So it is recommended that use this command only when needed or only when agent selects that particular type of callback.. For Agent Owned Callback (self-agent callback) there is no need to send this command to get agent list, desktop has all the information to create callback for the current (self) agent.

Response Syntax

AGTGetCallbackDestsForTypeRESP(POMCallbackType callbackType, POMCallbackDest[] callbackDests, String sessionID, int result)

Return Value Parameters

callbackType	Type of the callback destinations sent back from POM.
callbackDestinations	List of available destinations for callback.
sessionID	Unique ID of the call for the entire contact processing duration.
result	“0” indicates success.

Description

Asynchronous callback response for AGTGetCallbackDestsForType.

AGTCreateCallback

Command Syntax

int AGTCreateCallback (POMCallbackType callbackType, POMCallbackDest callbackDest, String callbackTime, String callbackTimezone, String callbackExpiryTime, POMContactNumber contactNumber, String agentNotes, String sessionID)

Parameters

callbackType	The callback type which needs to be scheduled.
callbackDest	The callback destination which must be dialed
callbackTime	The callback time in UTC time zone.
callbackTimezone	The time zone for which the callback is tobe dialed
callbackExpiryTime	Time limit in minutes after which the callback expires, starting from callbackTime.
contactNumber	The contact number which must be dialed
agentNotes	The agent can set notes which can be referred during the callback.
sessionID	Unique ID of the call for the entire contact processing duration.

Description

An agent can send this command to request POM to create a callback. The agent can give this command in Talking/Wrapup/Preview state only. According to the callback type, POM can send a pending callback notification to the agent few minutes (configurable value) before the scheduled time of the callback.

Error codes

Error code	Error message	Proposed solution
481	Invalid callback type selected	This error comes when desktop provides invalid callback type.
484	Unable to create callback as the campaign is not found	For a standard callback, if the system cannot find the campaign job which creates the callback, the system displays the error on the agent desktop.
485	Unable to create the callback as invalid callback type received	If the system receives an invalid or Null callback type from the desktop, the system displays the error on the agent desktop.

486	Unable to create callback as callback time received in invalid format	If the date format provided in this API does not match with yyyy/MM/dd/ HH:mm, the system displays the error on the agent desktop.
487	Unable to create the callback as the expiry time is invalid	If the expiry time sent in this API is invalid or lesser than 0, the system displays the error on the agent desktop.
488	Unable to create callback as the callback time is invalid	If the callback scheduled time is earlier than current time, the system displays the error on the agent desktop.

Reponse Syntax

AGTCreateCallbackRESP(String sessionID, int result)

Return Value Parameters

sessionID Unique ID of the call for the entire contact processing duration.
result "0" indicates success.

Description

Asynchronous callback response for AGTCreateCallback.

AGTGetErrorString

Note: This is obsolete method, use AGTGetErrorInfo instead of AGTGetErrorString

Command Syntax

int AGTGetErrorString (String errorCode)

Parameters

errorCode POM sends the error code in response to a command failure

Description

Desktop can run this function to get details of the error sent in response to a command. The error code sent back is a number. AGTGetErrorString returns a string which provides some details about the error.

.

Error Codes:

Error code	Error message	Description

1001	Unable to find error string for this error codecallback type is invalid	This error comes when server is not able to find error string from the resource for provided error code.

Response Syntax

AGTGetErrorStringRESP(String errorMsg, String localizedErrorMsg, int result)

Return Value Parameters

errorMsg	Error message.
localizedErrorMsg	Localized error message based on the locale passed during agent login.
result	"0" indicates success.

Description

Asynchronous callback response for AGTGetErrorString.

AGTGetErrorInfo

Command Syntax

int AGTGetErrorInfo (POMErrorCode errorCode)

Parameters

errorCode	Error code information sent by desktop to agent manger. It has errorCode and apiName parameter that has to be filled by desktop. apiName field is optional , it is recommended that it should be filled by desktop.

Description

Desktop can run this function to get details of the error sent in response to a command.

Response Syntax

int AGTGetErrorInfoRESP(POMErrorInfo errorInfo, int result)

Return Value Parameters

errorInfo	Error information sent back to desktop. This obect contains errorCode, errorType, errorString and localizedErrorString and apiName.

result "0" indicates success.

Description:

Errors that are sent from POM to desktop may be displayed as pop up based on the desktop implementation and thus the agent becomes irritated as he has to press OK button for all the errors. So in POM 302, this API is added which provides error type along with error string. Based on this error type desktop can decide to display the error as pop-up or to display in status bar or to suppress it. This errorType is enum parameter of errorInfo object.

Error Type:

1. Info
    - Desktop can suppress this information or can display it as information in status bar
2. Minor
    - Desktop can display it as error in status bar.
3. Major
    - Desktop can display this error as pop-up or in status bar with red color. If there is form on top of main form and the error is received as major, then it is recommended to display it in status bar of child form in red. E.g. if a callback form is open on top of main form and if any error occurs related to callback and it is better to display it on status bar in red instead of pop-up.
4. Critical
    - Display this type error as a pop-up.

Refer the "POM API Error Codes" for categorization of errors.

Error Codes:

Error code	Error message	Description
1001	Unable to find error string for this error codecallback type is invalid	This error comes when server is not able to find error string from the resource for provided error code.

AGTPreviewDial

Command Syntax

int AGTPreviewDial (POMContactNumber contactNumber, String sessionID)

Parameters

contactNumber The contact number that needs to be dialed for preview.

sessionID Unique ID of the call for the entire contact processing duration.

Description

Agent can send this command if it wants to accept the preview notification and it wants to call the customer. Agent can select from the available numbers and the agent can runs AGTPreviewDial. The agent runs this command only for a PreviewCampaign and in Preview state. For dialing free form number, desktop needs to set “Name” field of contactNumber object to “ExternalNumber”

Response Syntax

AGTPreviewDialRESP(String sessionID, int result)

Return Value Parameters

sessionID Unique ID of the call for the entire contact processing duration.

result: “0” indicates success.

Description

Asynchronous callback response for AGTPreviewDial.

AGTPreviewCancel

Command Syntax

int AGTPreviewCancel (String sessionID)

Parameters

sessionID Unique ID of the call for the entire contact processing duration.

Description

Agent can send this command if the agent does not need to accept a preview contact and instead cancel the dial request. POM moves this agent from Preview state to Wrapup state. An agent can decide if the agent wants to reject a contact dial attempt for any reason and provide those details during wrapup.

Response Syntax

AGTPreviewCancelRESP(WrapupData wrapupDetails, String sessionID, int result)

Return Value Parameters

wrapupDetails This structure consists of the following 3 values:

- *acwMaxTime*: Maximum time permitted to an agent to wrapup the call.

- *acwExtendable*: Boolean value indicating if the agent can request more wrapup time.
- *defaultCompCode*: The completion code POM uses to wrapup the call.

sessionID Unique ID of the call for the entire contact processing duration.

Return Values

result "0" indicates success.

Description

Asynchronous callback response for AGTPreviewCancel.

AGTGetCustomerDetails

Command Syntax

int AGTGetCustomerDetails (String sessionID)

Parameters

sessionID Unique ID of the call for the entire contact processing duration.

Description

Agent can run this command if it wants complete details of a contact. Details include Title/ FirstName/LastName/Address/Phone/email fields and custom fields. Usually this command must run after receiving a new call notification, that is, AGTCallNotify.

Note:

You can use the AGTGetCustomerDetails to get customer data after call is connected to agent. If the database is updated with new data after the call is connected, AGTGetCustomerDetails does not give the updated data if you call again from the agent desktop.

Response Syntax

AGTGetCustomerDetailsRESP(POMCustomerDetails customerDetails, String sessionID, int result)

Return Value Parameters

customerDetails POMCustomerDetails object containing customer data.

sessionID Unique ID of the call for the entire contact processing duration.

result: "0" indicates success.

Description

Asynchronous callback response for AGTGetCustomerDetails.

AGTSetCustomerDetail

Command Syntax

int AGTSetCustomerDetail (POMKeyValuePair pomKeyValuePair, String sessionID)

Parameters

pomKeyValuePair New value that needs to be changed for a key.

sessionID Unique ID of the call for the entire contact processing duration.

Description

Agent can run this command if it wants to change a field value of a customer data

Error codes

Error code	Error message	Description
641	Attribute is read only, cannot update this attribute	If the agent tries to update a read-only attribute through the agent desktop, the system displays the error on the agent desktop.
642	Invalid value entered. Unable to save attribute	If the validation of the value of the contact attribute fails on the system, the system displays the error on the agent desktop. For example, if the attribute is float type provides a string value such as "ABCD" for the attribute, the system displays the error on the agent desktop.

Response Syntax

AGTSetCustomerDetailRESP(String sessionID, int result)

Return Value Parameters

sessionID Unique ID of the call for the entire contact processing duration.

result “0” indicates success.

Description

Asynchronous callback response for AGTSetCustomerDetail.

AGTBlendToInbound

Command Syntax

int AGTBlendToInbound()

Parameters

None

Description

This command is sent by AACC blender to move an agent from Outbound to Inbound.

Error codes

Error code	Error message	Description
581	Agent is already assigned to Inbound	If using this API, an inbound agent is tried moving to inbound again; the system displays the error on the agent desktop.

Response Syntax

AGTBlendToInboundRESP(int result)

Return Value Parameters

result "0" indicates success.

Description

Asynchronous callback response for AGTBlendToInbound.

AGTBlendToOutbound

Command Syntax

int AGTBlendToOutbound()

Parameters

None

Description

This command is sent by AACC blender to move an agent from Inbound to Outbound.

Error codes

Error code	Error message	Description
601	Agent is already assigned to Outbound	If using this API, an outbound agent is tried moving to outbound again; the system displays the error on the agent desktop.

Response Syntax

AGTBlendToOutboundRESP(int result)

Return Value Parameters

result "0" indicates success.

Description

Asynchronous callback response for AGTBlendToOutbound.

AGTNailupAgent

Command Syntax

int AGTNailupAgent()

Parameters

None.

Description

Blender sends this command when it moves the agent from Inbound to the Outbound queue.

Response Syntax

AGTNailupAgentRESP(int result)

Return Value Parameters

result "0" indicates success.

Description

Asynchronous callback response for AGTNailupAgent.

AGTReadyForNailup

Command Syntax

int AGTReadyForNailup()

Return value parametersNone.

Description

The desktop sends this command when it has processed the AGTNailupChange-PendingNailup notification. The notification gives an agent enough time to prepare for the nailup. The notification also acts as an indication to the desktop (with a softphone) making the next call as a nailing call which must be auto-answered and the call is not a generic inbound call.

Response Syntax

AGTReadyForNailupRESP(int result)

Return Value Parameters

result "0" indicates success.

Description

Asynchronous callback response for AGTReadyForNailup.

GetAgentStatusResponse

Command Syntax

int GetAgentStatusResponse (POMAgentStatus status)

Parameters

status Value having the agent state, call state, and call state value.

Description

Desktop sends this command on receiving GetAgentStatus notification. Agent manager sends GetAgentStatus when it wants to check the current agent state on the desktop. So, when Agent Manager is restarted, it sends GetAgentStatus when the desktop connects to Agent Manager after restart. The desktop must set desktop's current known Agent State, Nailing State, and Call State.

Response Syntax

GetAgentStatusResponseRESP(int result)

Return Value Parameters

result "0" indicates success.

Description

Asynchronous callback response for GetAgentStatusResponse.

Also see:

GetAgentStatus

AGTLostNailing

Command Syntax

int AGTLostNailing()

Parameters

None

Description

Desktop sends this command when it detects loss of nailing in the softphone or hardphone.

Response Syntax

AGTLostNailingRESP (WrapupData wrapupDetails, Sting sessionID, int result)

Return value parameters

wrapupDetails This structure consists of the following 3 values:

- *acwMaxTime*: Maximum time permitted to an agent to wrapup the call.
- *cwExtendable*: Boolean value indicating if the agent can request more wrapup time.
- *adefaultCompCode*: The completion code POM uses to wrapup the call.

sessionID Unique ID of the call for the entire contact processing duration.

result “0” indicates success.

Description

Asynchronous callback response for AGTLostNailing.

AGTPendingLogout

Command Syntax

int AGTPendingLogout()

Parameters

None.

Description

An agent sends this command at any state to logout from the desktop. If the agent is in the

middle of the call, then POM waits for wrap up and moves the agent to not ready state and then logout the agent. This operation is mandatory to perform, if the desktop detects an erroneous condition.

Response Syntax

AGTPendingLogoutRESP(int result)

Return Value Parameters

result "0" indicates success.

Description

Asynchronous callback response for AGTPendingLogoutRESP.

AGTAddAgentNote

Command Syntax

int AGTAddAgentNote (String agentNote)

Parameters

agentNote Notes to be stored by POM.

Description

An agent can use this command to store agent notes for a contact in POM. For consult/transfer/ conference/callbacks, the agent uses these notes as reference.

Error code	Error message	Description
561	Invalid data received. Unable to add agent	If the agent notes are Null or empty, the system displays the error on the agent desktop.
562	System error. Unable to add agent notes	If the system fails to add the agent notes, the system displays the error on the agent desktop.
563	System error. Unable to add agent notes	If the system fails to add agent notes because it cannot find the contact, the system displays the error on the desktop.

Response Syntax

AGTAddAgentNoteRESP(String sessionID, int result)

Return Value Parameters

sessionID	Unique ID of the call for the entire contact processing duration.
result:	"0" indicates success.

Description

Asynchronous callback response for AGTAddAgentNote.

AGTRefreshAgentNotes

Command Syntax

int AGTRefreshAgentNotes (String sessionID)

Parameters

sessionID	Unique ID of the call for the entire contact processing duration.

Description

An agent can use this command to have agent notes stored for this contact for the current job. The agent can have the notes saved during talking/consult/transfer/conference/callbacks.

Note:

POM supports maximum packet size of 16000 bytes. If the notes exceed 16000 bytes, POM truncates the notes.

Response Syntax

AGTRefreshAgentNotesRESP(String[] agentNotes, String sessionID, int result)

Return Value Parameters

agentNotes	Agent notes added earlier for the call.
sessionID	Unique ID of the call for the entire contact processing duration.
result	"0" indicates success.

Description

Asynchronous callback response for AGTRefreshAgentNotes.

AGTGetTimeZones

Command Syntax

int AGTGetTimezones()

Parameters

None

Description

POM uses this command to get a list of all supported time zones. This command must run before creating callbacks.

Response Syntax

AGTGetTimezonesRESP(POMKeyValuePair[] timezones, int result)

Return Value Parameters

timeZones	List of supported time zones.
result	"0" indicates success.

Description

Asynchronous callback response for AGTGetTimeZones

AGTAvailableForNailup

Command Syntax

AGTAvailableForNailup()

Parameters

None

Description

Desktop sends this command only when the agent gets Ready for the first time after logging in.

Response Syntax

AGTAvailableForNailupRESP(int result)

Return Value Parameters

result	"0" indicates success.

Description

Asynchronous callback response for AGTAvailableForNailup.

AGTAgentDisconnected

Command Syntax

int AGTAgentDisconnected()

Parameters

None

Description

A component must send this command which monitors the desktop. Sending this command permits POM to remove this agent from pacing and POM stops sending new notifications to this agent.

Response Syntax

AGTAgentDisconnectedRESP(int result)

Return Value Parameters

result	"0" indicates success.

Description

Asynchronous callback response for AGTAgentDisconnected.

AGTAddToDNC

Command Syntax

int AGTAddToDNC (POMAttribute[] addressList)

Parameters

addressList	List of addresses to be added to DNC list.

Description

The agent can add a contact address or more addresses into the DNC list, POM maintains.

Error Codes

Error code	Error message	Description

621	Unable to add to DNC. Contact address already present.	If the contact address is already in DNC, the system displays the error on the agent desktop.
622	System error. Unable to add to DNC	If the system cannot add the contact address to DNC, the system displays the error on the agent desktop.
623	Invalid data received. Unable to add to DNC	If the contact attribute for which the system tries to add the contact address to DNC is Null, or not present in the system, the system displays the error on the agent desktop.

Response Syntax

AGTAddToDNCRESP(String sessionID, int result)

Return Value Parameters

sessionID Unique ID of the call for the entire contact processing duration.

result "0" indicates success

Description

Asynchronous callback response for AGTAddToDNC.

AGTIsInDNC

Command Syntax

int AGTIsInDNC (String addressValue, String sessionID)

Parameters

addressValue Address to be verified in POM DNC list.

sessionID Unique ID of the call for the entire contact processing duration.

Description

The agent can run this command to verify if the address is already in the POM DNC list.

Response Syntax

AGTIsInDNCRESP(bool present, int result)

Return Value Parameters

present True if the value is in the POM DNC list, false otherwise.

result “0” indicates success.

Description

Asynchronous callback response for AGTIsInDNC.

AGTGetZoneList

Command Syntax

int AGTGetZoneList()

Parameters

None.

Description

The agent can run this command to get a list of zones configured on POM.

Response Syntax

AGTGetZoneListRESP(String[] zoneList, int result)

Return Value Parameters

zoneList List of zones configured on POM.

result “0” indicates success.

Description

Asynchronous callback response for AGTGetZoneList.

AGTSaveAgentForHA

Command Syntax

int AGTSaveAgentForHA()

Parameters

None.

Description

The agent must run this if it is interested in POM restoring the agent desktop state in a scenario where the desktop crashes and was restarted. So, if the agent invokes AGTSaveAgentForHA and after some time while performing some activity for a customer

the desktop crashes, then if the agent re-logs in then POM tries to restore the agent state back to what it was before the crash situation.

Note: The current release does not support Agent HA. Ensure that you do not use the AGTSaveAgentForHA command through the agent desktop.

Response Syntax

AGTSaveAgentForHARESP(int result)

Return Value Parameters

result "0" indicates success.

Description

Asynchronous callback response for AGTSaveAgentForHA.

AGTSkillsChanged

Command Syntax

int AGTSkillsChanged (POMAgentSkill[] agentSkills)

Parameters

agentSkills The current agent skill set.

Description

AAAD sends this when it detects any changes in the agent skillset on AACC.

Response Syntax

AGTSkillsChangedRESP(int result)

Return Value Parameters

result 0" indicates success.

Description

Asynchronous callback response for AGTSkillsChanged.

AGTGetContactAttributes

Command Syntax

int AGTGetContactAttributes()

Parameters

None.

Description

The agent can send this command to get the list of attributes and attributes types, linked with the current contact. This command does not send the attribute values in response to this command request. To get contact attribute values, the agent must run the command AGTGetCustomerDetails.

Response Syntax

AGTGetContactAttributesRESP(int result)

Return Value Parameters

result	"0" indicates success.

Description

Asynchronous callback response for AGTGetContactAttributes.

Notifications

AGTStateChangeNotify

Notification Syntax

AGTStateChangedNotify (POMAgentState agentState)

Notifcation Parameters

agentState	Agent state. Value can be one of Ready/NotReady/PendingNotReady

Description

POM sends this notification when it detects change in agent state. If the agent requests to go to Ready state by invoking AGTStateChange, with a response for AGTStateChange command POM sends back this notification with the agentState as Ready. If the agent requests to go to NotReady state during a call, POM sends back AGTStateChange notification with the agentState as PendingNotReady. But, when POM finds that the agent has finished agent's calls, the agent which was put in PendingNotReady state is

sent to NotReady state and POM informs this to the desktop by sending AGTStateChange notification.

AGTCallNotify

Notification Syntax

AGTCallNotify (POMContact contact, String sessionID)

Notifcation Parameters

contact POMContact object containing the customer and job related information about the contact which is (preview campaign)/got (Predictive + Progressive campaign) dialed.

sessionID sessionID: Unique ID of the call for the entire contact processing duration.

Description

POM sends notification for Predictive and Progressive campaigns when POM connects the outbound call to the customer with the agent. POM sends this notification to the selected agent, so that the agent gets to know the customer that got dialed. The desktop must then run AGTGetCustomerDetails command to get more details about the customer.

AGTAutoReleaseLine

Notification Syntax

AGTAutoReleaseLine (WrapupData wrapupDetails, String sessionID)

Notifcation Parameters

wrapupDetails This structure contains the following 3 values:

- *acwMaxTime*: Maximum time permitted to an agent to wrapup the call.
- *acwExtendable*: Boolean value indicating if the agent can request more wrapup time.
- *defaultCompCode*: The completion code POM uses to wrapup the call.

sessionID Unique ID of the call for the entire contact processing duration.

Description

POM sends this notification when it detects that the customer has disconnected the call. The desktop must permit the agent to wrapup the call by creating a timer with the time value set to acwMaxTime. If the agent needs more time to wrapup the call and if acwExtendable is set to true, then the agent can run AGTExtendWrapup and get some

more time from POM. POM uses defaultCompCode to auto wrapup the call, if the agent does not select a completion code during the wrapup time. Desktop can perform an autowrapup after the acwMaxTime expires and the agent does not run AGTWrapupContact. Desktop can use the defaultCompCode for AGTWrapupContact.

If acwMaxTime is received as zero with answering machine completion code (Answer_Machine), desktop needs to wrap up the call immediately without agent interaction with the completion code answering machine received in AGTAutoReleaseLine notification.

AGTConsultNotify

Notification Syntax

AGTConsultNotify (POMContact contact, String requestingAgentId, String sessionID)

Notifcation Parameters

contact	POMContact object containing the customer and job related information about the contact for which POM sends this consult request.
requestingAgentID	Agent ID of the consult starting agent.
sessionID	Unique ID of the call for the entire contact processing duration.

Description

POM sends this notification when an agent (active) requests a consult with another agent (passive). If the passive agent is busy on another call, POM sends AGTPendingConsult to the passive agent. POM sends AGTConsultNotify only when it successfully takes the passive agent into the consultation call.

AGTConsultCancelled

Notification Syntax

AGTConsultCancelled (String requestingAgentId, String sessionID)

Notifcation Parameters

requestingAgentID	Agent ID of the agent which initiated the cancel consult request.
sessionID	Unique ID of the call for the entire contact processing duration.

Description

POM sends this notification when a consult is cancelled/dropped. The other agent performs cancel operation and POM sends this notification to an active agent. POM

sends this notification either because of AGTCancelConsult or AGTAutoReelaseLine operations.

AGTTransferNotify

Notification Syntax

AGTTransferNotify (POMContact contact, String sessionID)

Notifcation Parameters

contact POMContact object containing the customer and job related information about the contact for which this transfer happened.

sessionID Unique ID of the call for the entire contact processing duration.

Description

POM sends this notification when an agent (active) completes the transfer to the consulted agent (passive). POM sends AGTTransferNotify to the passive agent informing it about the call ownership. POM moves the earlier active agent to Idle state and the new active agent to Talking/Busy state.

This release supports the transfer capability in external conference, so that agent can leave the external conference. It will transfer the call to external party and drop from the conference.

AGTConferenceNotify

Notification Syntax

AGTConferenceNotify (POMContact pomContact, String sessionID)

Notifcation Parameters

contact POMContact object containing the customer and job related information about the contact for which this conference happened.

sessionID Unique ID of the call for the entire contact processing duration.

Description

POM sends this notification when an agent (active) starts the conference with the consulted agent (passive). POM sends AGTConferenceNotify to the passive agent informing it about the conference. POM moves the earlier active agent to ConferenceOwner state and the passive agent to ConferencePassive state.

AGTConferenceEnded

Notification Syntax

AGTConferenceEnded (String sessionID)

Notifcation Parameters

sessionID Unique ID of the call for the entire contact processing duration.

Description

POM sends this notification to an agent informing it that the other agent has ended the conference. POM can also send this when the customer releases the call.

AGTConferenceOwnershipChanged

Notification Syntax

int AGTConferenceOwnershipChanged (String sessionID)

Notifcation Parameters

sessionID Unique ID of the call for the entire contact processing duration.

Description

The Conference owner can give this command to transfer the conference ownership to the passive agent in the conference. POM then moves the passive agent from ConferencePassive state to ConferenceOwner state and vice-versa for the original conference owner.

AGTCapabilitiesChanged

Notification Syntax

AGTCapabilitiesChanged (POMCapabilities capabilities)

Notifcation Parameters

capabilities Almost 17 capabilities are associated with an agent. This object contains the status of those capabilities.

Description

POM sends this notification from time to time, when it detects a Call/Agent/Nailed/Job status change. Some of the capabilities are CanHold/CanTransfer/CanConsult. This notification is a critical notification because this notification helps the desktop in becoming stateless. The desktop can hide/ unhide various controls based on values in capabilities.

AGTNailupChange

Notification Syntax

AGTNailupChange (POMNailupStatus nailupStatus)

Notifcation Parameters

nailupStatus Values are PendingNailup / NailedUp / PendingNailupDrop / NotNailed / NailingLost

Description

POM sends this notification to the desktop informing it about the agent's current nailing state. This notification is send when POM detects a nailing state change.

AGTCallStateChangedNotify

Notification Syntax

AGTCallStateChangedNotify (POMCallState callState)

Notifcation Parameters

callState Provides current call state value. Some values are Preview/Dialing/Talking/Held/Wrapup.

Description

POM sends this notification to the desktop when it detects a change in the current call state. The call state changes based on the agent state machine inside POM and also based on actions taken by the agent during a call.

AGTDialFailed

Notification Syntax

AGTDialFailed (WrapupData wrapupDetails, POMDialFailReason dialFailReason, String sessionID)

Notifcation Parameters

wrapupDetails This structure contains the following 3 values:

- *acwMaxTime*: Maximum time permitted to an agent to wrapup the call.
- *acwExtendable*: Boolean value indicating if the agent can request more wrapup time.

• *defaultCompCode*: The completion code POM uses to wrapup the call.

dialFailReason An enumerator value linked with a possible dial fail reason.

sessionID Unique ID of the call for the entire contact processing duration.

Description

POM sends this notification to the agent if the Preview Dial trail fails or a Redial attempt fails. POM moves the agent to Wrapup state after sending this notification. The agent can use the defaultCompCode to auto dispose the call. The action POM takes after receiving this notification is similar to AGTReleaseLine and AGTAutoReleaseLine.

AGTConsultDialFailed

Notification Syntax

AGTConsultDialFailed (POMDialFailReason dialFailReason, String sessionID)

Notifcation Parameters

dialFailReason An enumerator value linked with a possible dial fail reason.

sessionID Unique ID of the call for the entire contact processing duration.

Description

POM sends this notification to the agent that attempts a consult to another agent. The agent stays in Talking state. A consult can fail because of various reasons especially if the consulted party is an external number.

AGTConsultPending

Notification Syntax

AGTConsultPending (String requestingAgent, String requestingAgentID, String requestingCampaign, String sessionID)

Notifcation Parameters

requestingAgent Agent requesting the consult.

requestingAgentId Agent ID of the agent requesting the consult.

consultrequestingCampaign Name of the campaign to which the consult requesting agent is part of.

sessionID Unique ID of the call for the entire contact processing duration.

Description

POM sends this notification when the agent (active) requests a consult with another agent (passive) when the passive agent is already busy in a call that is the passive agent is either in Busy state. The receiving agent can decide on an action with the current call based on this notification.

AGTPendingConsultComplete

Notification Syntax

AGTPendingConsultComplete (String sessionID)

Notifcation Parameters

sessionID Unique ID of the call for the entire contact processing duration.

Description

POM sends this notification to the initiator agent when the recipient agent successfully completes the consult.

AGTPreviewCallbackPending

Notification Syntax

AGTPreviewCallbackPending (String dueTime, String sessionID)

Notifcation Parameters

dueTime Time at which the callback is due.

In POM 303, POM server can send time in UTC (yyyy/MM/dd HH:mm), if the UTC is set in *TimeZonePendingCallbackDueTime* parameter in database (POM configuration table pim_config). So desktop can convert the time as per agent desktop machine time zone.

The existing functionality in which POM sends due time as per the POM server time zone will work as it to maintain backward compatibility and this is the default behavior.

sessionID Unique ID of the call for the entire contact processing duration.

Description

POM sends this notification to an agent a few seconds (configurable value on POM) before the scheduled callback time. This notification is useful for the agent in deciding the agent's action for the current call.

AGTPreviewCallbackCancelled

Notification Syntax

AGTPreviewCallbackCancelled (String dueTime, String sessionID)

Notifcation Parameters

sessionID Unique ID of the call for the entire contact processing duration.

Description

POM sends this notification to an agent if POM cancels the callback because it has expired.

AGTAgentLoggedOut

Notification Syntax

AGTAgentLoggedOut()

Notifcation Parameters

None

Description

POM sends this notification when it successfully logs out an agent. This notification is useful if the agent has issued a AGTPendingLogout request. This notification provides the agent confirmation about agent's logout operation.

AGTCustomerDetailsChanged

Notification Syntax

AGTCustomerDetailsChanged (POMAttribute attribute, String sessionID)

Notifcation Parameters

attribute POMAttribute that has changed.

sessionID Unique ID of the call for the entire contact processing duration.

Description

POM sends this notification to determine the current agent state. POM sends this notification either when agent manager restarts while the desktop is in use OR if agent manager does not detect any activity from the agent for a considerable time. The desktop developer must fill up the current agent state in the POMAgentStatus structure and send back in the notification method response.

AGTEnableCancelConsult

Notification Syntax

AGTEnableCancelConsult (String sessionID)

Notifcation Parameters

sessionID Unique ID of the call for the entire contact processing duration.

Description

POM sends this notification to the active agent so that the desktop can enable the cancel consult feature. Certain scenarios/situations are there in which POM cannot cancel the consult. POM uses this notification to take care of such scenarios.

AGTInvalidCommandName

Notification Syntax

AGTInvalidCommandName (String commandName)

Notifcation Parameters

commandName Details of the data that POM received.

Description

POM sends this notification to the desktop if POM receives an invalid command.

GetAgentStatus

Notification Syntax

POMAgentStatus GetAgentStatus (String agentID)

Notifcation Parameters

agentID ID of the requested agent's status.

Description

POM sends this notification to determine the current agent state. POM sends this either when agent manager restarts while the desktop is in use OR if agent manager does not detect any activity from the agent for a considerable time.

The desktop developer must fill up the current agent state in the POMAgentStatus structure

and send back in the notification method response.

POM sends this notification to determine the current agent state. POM sends this notification either when agent manager restarts while the desktop is in use OR if agent manager does not detect any activity from the agent for a considerable time.

The desktop developer must fill up the current agent state in the POMAgentStatus structure and send back in the notification method response.

Also see:

GetAgentStatusResponse

POMAvailable

Notification Syntax

POMAvailable()

Notifcation Parameters

None

Description

POM sends this notification when the agent manager connects to the library for the first time OR if the agent manager restarts while the desktop is still in use.

POMNotAvailable

Notification Syntax

POMNotAvailable()

Notifcation Parameters

None.

Description

The library sends this notification when it detects that the connection to the agent manager is lost.

AGTBlendedtoOutbound

Notification Syntax

AGTBlendedtoOutbound()

Notifcation Parameters

None

Description

POM sends this notification when the agent is moved to Outbound.

AGTBlendedToInbound

Notification Syntax

AGTBlendedToInbound()

Notifcation Parameters

None

Description

POM sends this notification when the agent is moved to Inbound.

AGTZoneDown

Notification Syntax

AGTZoneDown (String zoneName, int gracePeriodInMillis)

Notifcation Parameters

zoneName	Name of the zone.
gracePeriodinMillis	Grace period in milliseconds after which POM forcefully logs out the agent

DescriptionPOM sends this notification when it detects that a zone is going down. The agent must finish off all his current activities before gracePeriodInMiilis, otherwise POM forcefully logs out the agent.

AGTJobAttached

Notification Syntax

AGTJobAttached (String campaignName)

Notifcation Parameters

campaignName	Name of the campaign.

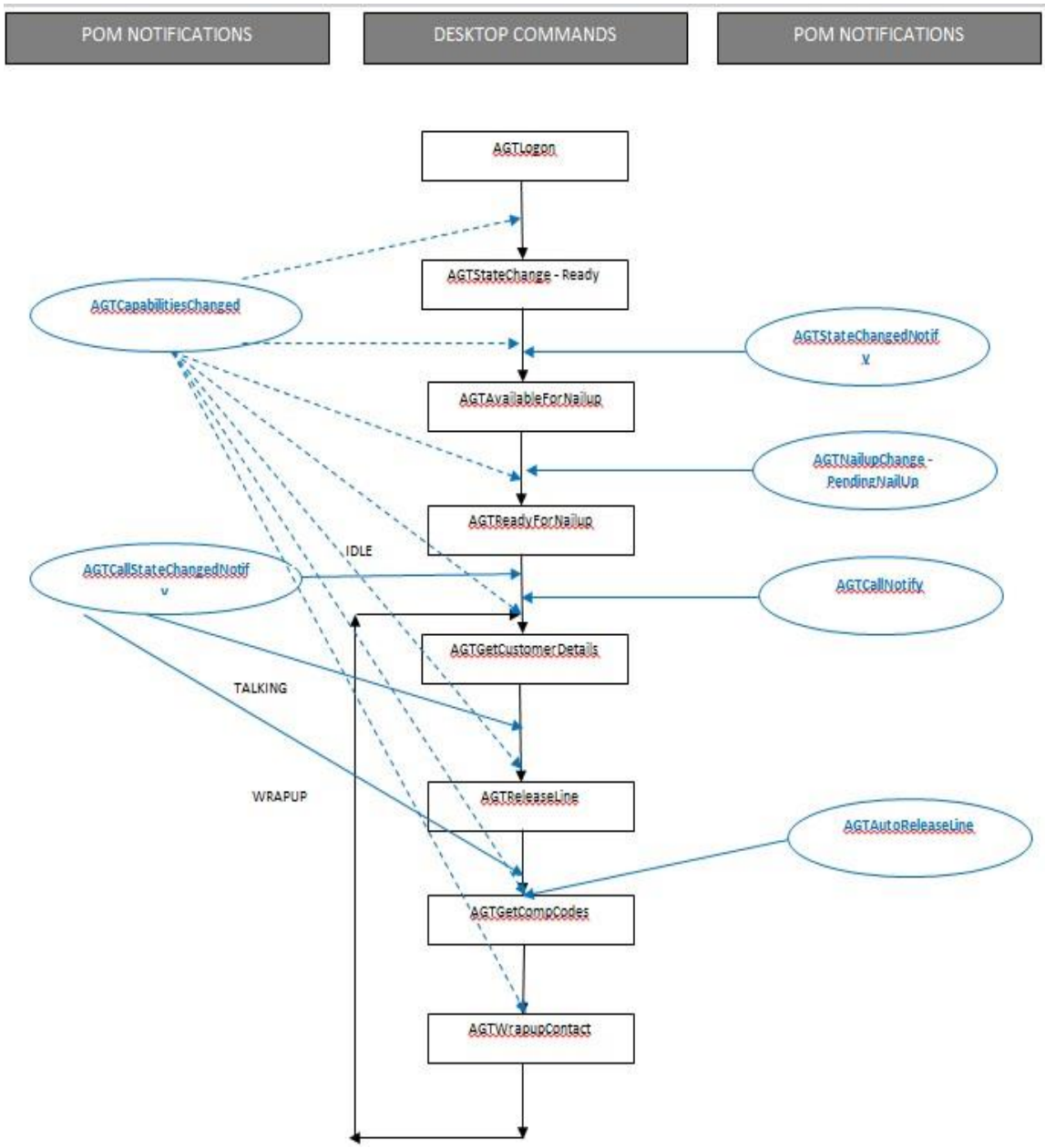
Description

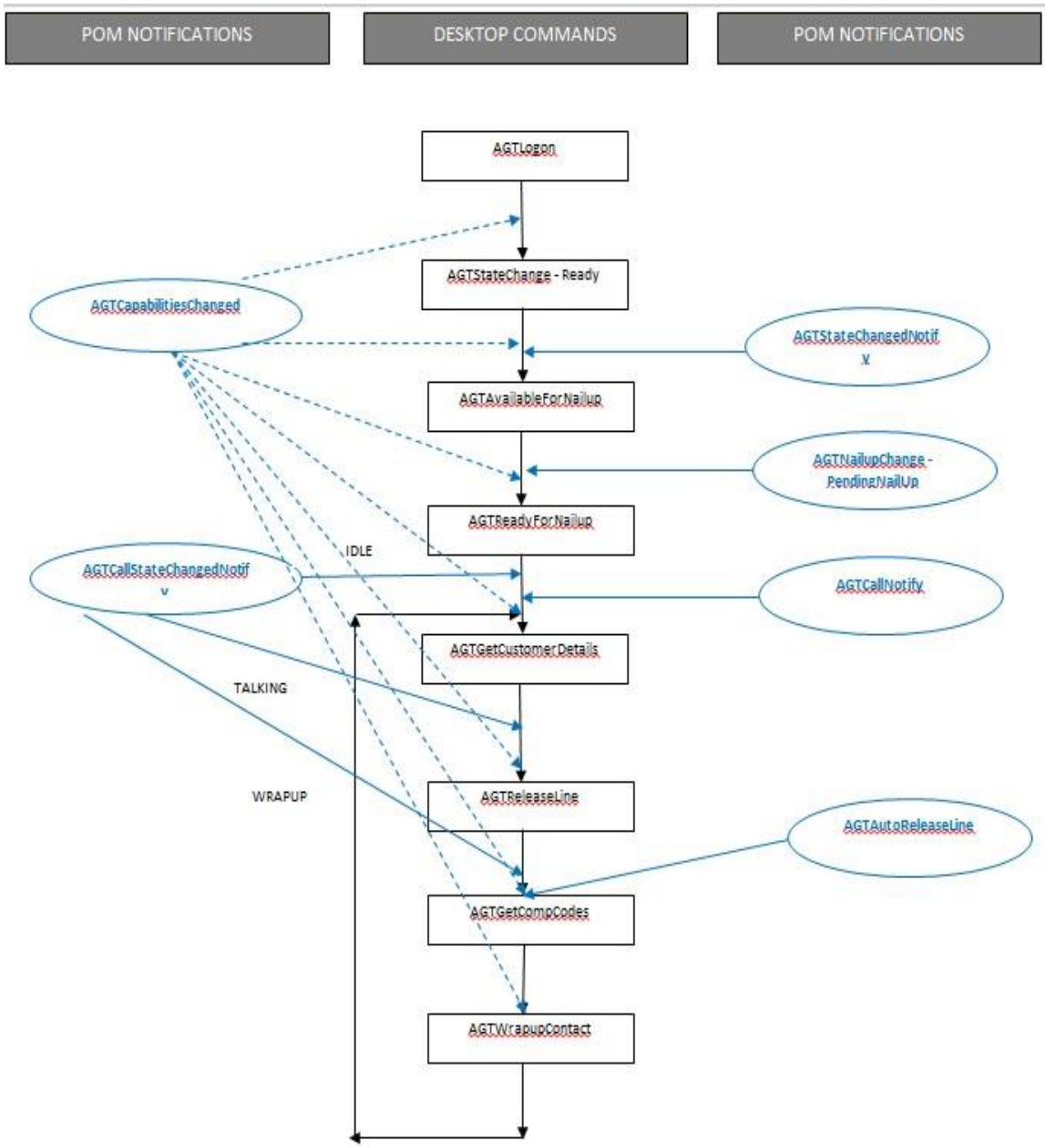
POM sends this notification when the agent is attached to a new job. The associated campaign name is sent as an argument.

Earlier, In CCElite mode the skill information was refreshed from Communication Manager while attaching to a job. Any request to refresh the skill would take time to reflect depending on Communication Manager work load. In the current release, POM does not refresh the skill information while attaching an agent to a job. If the agent skill information changes, the agent needs to logout and login again for the changes to take effect.

Call Flow and Capability Matrix

Simple Call Flow


POM NOTIFICATIONS
DESKTOP COMMANDS
POM NOTIFICATIONS
AGTLogon
AGTStateChange - Ready
AGTCapabilitiesChanged
AGTStateChangedNotif
v
AGTAvailableForNailup
AGTNailupChange -
PendingNailUp
AGTReadyForNailup
IDLE
AGTCallStateChangedNotif
v
AGTCallNotify
AGTGetCustomerDetails
TALKING
WRAPUP
AGTReleaseLine
AGTAutoReleaseLine
AGTGetCompCodes
AGTWrapupContact


POM NOTIFICATIONS
DESKTOP COMMANDS
POM NOTIFICATIONS
AGTLogon
AGTStateChange - Ready
AGTCapabilitiesChanged
AGTStateChangedNotif
v
AGTAvailableForNailup
AGTNailupChange -
PendingNailUp
AGTReadyForNailup
IDLE
AGTCallStateChangedNotif
v
AGTCallNotify
AGTGetCustomerDetails
TALKING
WRAPUP
AGTReleaseLine
AGTAutoReleaseLine
AGTGetCompCodes
AGTWrapupContact

Capability matrix

	Hold	Unhold	Transfer	Conference	Release	Disposition	Originate	CreateCallback	SendDTMF	UpdateRecord	LeaveConference	LeaveConsult	EndConference	ChangeOwnership	Ready	NotReady	Record
Idle	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N
Held	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N
Talking	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	Y	N
Consult (Owner)	N	N	Y	Y	N	N	N	N	N	N	N	Y	N	N	N	Y	N
Consult (Recipient)	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	Y	N
Agent Conference (Owner)	N	N	N	N	N	N	N	N	N	Y	N	N	Y	Y	N	Y	N
External Conference (Owner)	N	N	Y	N	N	N	N	N	N	Y	N	N	Y	Y	N	Y	N
Conference (Recipient)	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	Y	N
Wrapup	N	N	N	N	N	Y	N	Y	N	N	N	N	N	N	N	Y	N
NotReady	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N
Pending NotReady	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N
Ready	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N
Recording Present	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N

Preview/ Callback	N	N	N	N	N	N	N	Y	N	Y	N	N	N	N	N	Y	N
Dialing	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N

Logs and traces, Error messages, Troubleshooting, and Miscellaneous Notes

Logs and traces

The library generates own logs for it using Apache log4net library. A configuration file is shipped with the library, which stores the path locations of the dependent libraries and also the log4net library. The log4net configuration file log4net.xml is also shipped with the POM Desktop API libraries.

Error messages

If an API sends back an error code, then the desktop developer invokes AGTGetErrorString to get the appropriate error message. AGTGetErrorString localizes the error messages and the appropriate localized message are available based on the locale provided with AGTGetErrorString. POM receives the API request and the returns the value back. POM sends the error codes with the response for a command. All response messages ending with RESP have an error code resolved with a localized error message.

Note:

All error codes do not have an appropriate error message linked with them. There are certain error codes for which the AGTGetErrorString must not be invoked because the error codes are detected at the API level, POM does not receive the request . Only errors that POM detects, have an appropriate error message response with AGTGetErrorString. POM sends back all such error codes while invoking the commands.

Following error codes are not associated with error messages:

Error Code: 9999 (POM_NOT_AVAILABLE)

POM sends back the error code if the POM does not receive the command. POM generates the error code internally and sends the error code back with the appropriate response RESP.

Error Code: 9998 (PAM_NOT_AVAILABLE_FOR_ZONE)

POM sends back the error code if POM the agent manager does not manage the zones provided in the AGTLogon command.

Error Code: 9997 (INVALID_ARGUMENT)

POM sends back if API receives any incorrect argument

Error Code 9996 (SDK_Failure)

POM generates the error code if the library cannot send the command to POM.

Note:

Ensure that you localize or translate the error codes if required, and manage the error codes.

Troubleshooting

The library logs are important and useful in debugging issues in the first phase. The logging levels must be set to FINEST level to track issues. The logs lists all communication messages between the desktop library and POM. Logs also logs exceptions/errors which are useful in understanding issues. The logs also indicate the flow of data between the library user and POM.

In AACC mode agent login failed after restarting CCMM POMProxy and POM services. To resolve this issue sequence needs to be maintained. First start POM agent manager and when it is up and running/initialized start AACC CCMM proxy.

POM API error codes

Error Code	Error Type	Message
0		Success
1	Major	Command Failure
2	Major	This agent is not registered with the system
3	Minor	Unable to login. Check media server.
5	Minor	Agent is already nailed.
6	Major	Unable to verify password.
7	Major	Agent is already logged in.
8	Minor	Agent is forcefully logging in to the system.
9	Major	Agent skills not found.
10	Major	Unable to change the state of the agent.
11	Major	Internal error. Unable to login agent.
12	Major	Login failure. Zone not found.
13	Major	Login failure. Invalid Locale.
14	Major	Login failure. Invalid Agent Extension.
15	Major	Login failure. Invalid Timezone.
16	Major	Login failure. Invalid Agent Name.
17	Major	Login failure. Authentication of agent failed.
61	Info	Agent is already in ready state.
62	Minor	Agent walkaway received when agent is handling a call.
63	Info	State change request sent more than once with the same reason code.
64	Major	Unable to change the current state.
81	Minor	Unable to release the call.
82	Info	Call is already disconnected.
101	Major	Unable to hold.
102	Minor	Unable to hold as call is disconnected.
103	Minor	Unable to hold as agent is not on call.

121	Major	Unable to unhold.
122	Minor	Unable to unhold as call is disconnected.
123	Minor	Unable to unhold as agent is not on call.
141	Minor	Unable to consult as the selected agent is currently not ready.
142	Minor	Unable to consult as the selected agent has logged out.
143	Major	Unable to consult as the selected destination is invalid.
144	Minor	Unable to consult as the selected agent already has maximum pending consults.
181	Minor	Unable to conference.
182	Minor	Unable to conference as agent selected is not in a consult with this agent.
183	Minor	Unable to conference as agent selected for conference has logged out.
184	Minor	Unable to conference as agent selected for conference is currently not ready.
201	Major	Unable to wrapup as contact record not found.
202	Major	Unable to wrapup with this completion code.
221	Major	Cannot get destinations as callback type is invalid.
222	Minor	Unable to get zone of the contact.
223	Info	Unable to get campaigns for the organization.
241	Minor	Unable to dial as agent is not in a preview.
242	Minor	Unable to cancel as agent is not in a preview.
261	Minor	Moving agent to pending not ready state.
281	Minor	Conference owner cannot leave the conference.
301	Minor	Cannot end the conference as the agent is not in a conference.
302	Minor	Cannot end the conference as the agent is not the owner of the conference.
303	Major	Unable to end the conference as passive agent not found.
321	Minor	Cannot change ownership as this is not an agent type of conference.
341	Minor	Cannot send DTMF as invalid data received.
342	Minor	Unable to send DTMF as agent is not in a call.
343	Major	Unable to send DTMF.
361	Minor	Unable to redial as agent is not in wrapup.
362	Minor	Unable to redial as the address is invalid.
382	Info	No destinations available.
383	Info	Agents are not available.
401	Major	Unable to cancel the consult.
402	Minor	Unable to cancel the pending consult as it has already started.
421	Major	Unable to consult.
441	Minor	System error. Unable to start recording.
461	Minor	System error. Unable to stop recording.
481	Minor	System error. Unable to stop recording.
482	Minor	not found. Unable to create the callback.
483	Minor	Unable to create the callback as the agent is not in a call.
484	Major	Unable to create callback as the campaign is not found.
485	Major	Unable to create the callback as invalid callback type received.
486	Major	Unable to create the callback as callback time received in invalid format.
487	Major	Unable to create callback as the expiry time is invalid.
488	Major	Unable to create callback as the callback time is invalid.

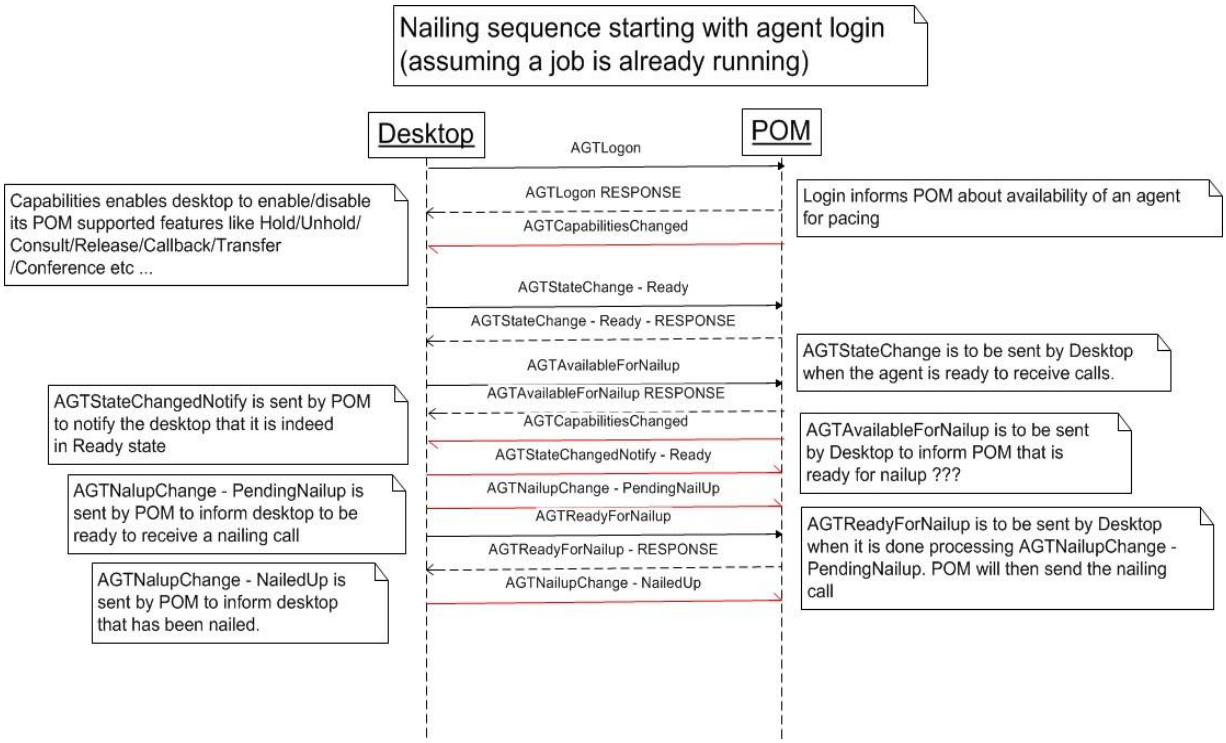
501	Info	Unable to extend the wrapup time as agent is not in wrapup state.
521	Minor	Unable to find the customer details.
541	Minor	Unable to logoff. Move to not ready state.
561	Info	Invalid data received. Unable to add agent notes.
562	Major	Unable to add agent notes.
563	Major	Unable to add agent notes.
581	Info	Agent is already assigned to Inbound.
601	Info	Agent is already assigned to Outbound.
621	Info	Unable to add to DNC. Contact address already present.
622	Major	Unable to add to DNC.
623	Major	Invalid data received. Unable to add to DNC.
624	Minor	Invalid address. Unable to add to DNC.
641	Minor	Attribute is read only, cannot update this attribute.
642	Major	Invalid value entered. Unable to save attribute.
1001	Info	Unable to find error string for this error code.
9001	Minor	Initialization failed as server is not reachable.
9002	Major	Agent is not registered.
9003	Major	System error. Unexpected command received
9004	Minor	This request cannot be processed in the current agent state.
9005	Minor	System is not available.
9006	Major	Unable to process request as agent is not nailed.
9007	Major	System error. Check media server.
9008	Major	System error. Invalid data received in the request.
9009	Critical	System error. Media server not reachable.
9010	Major	Parameters received in request are less than expected.
9011	Major	System error. Invalid command name received.
9012	Minor	Please wait while system is initializing.
9050	Major	System error. Unable to process the request.

Miscellaneous notes

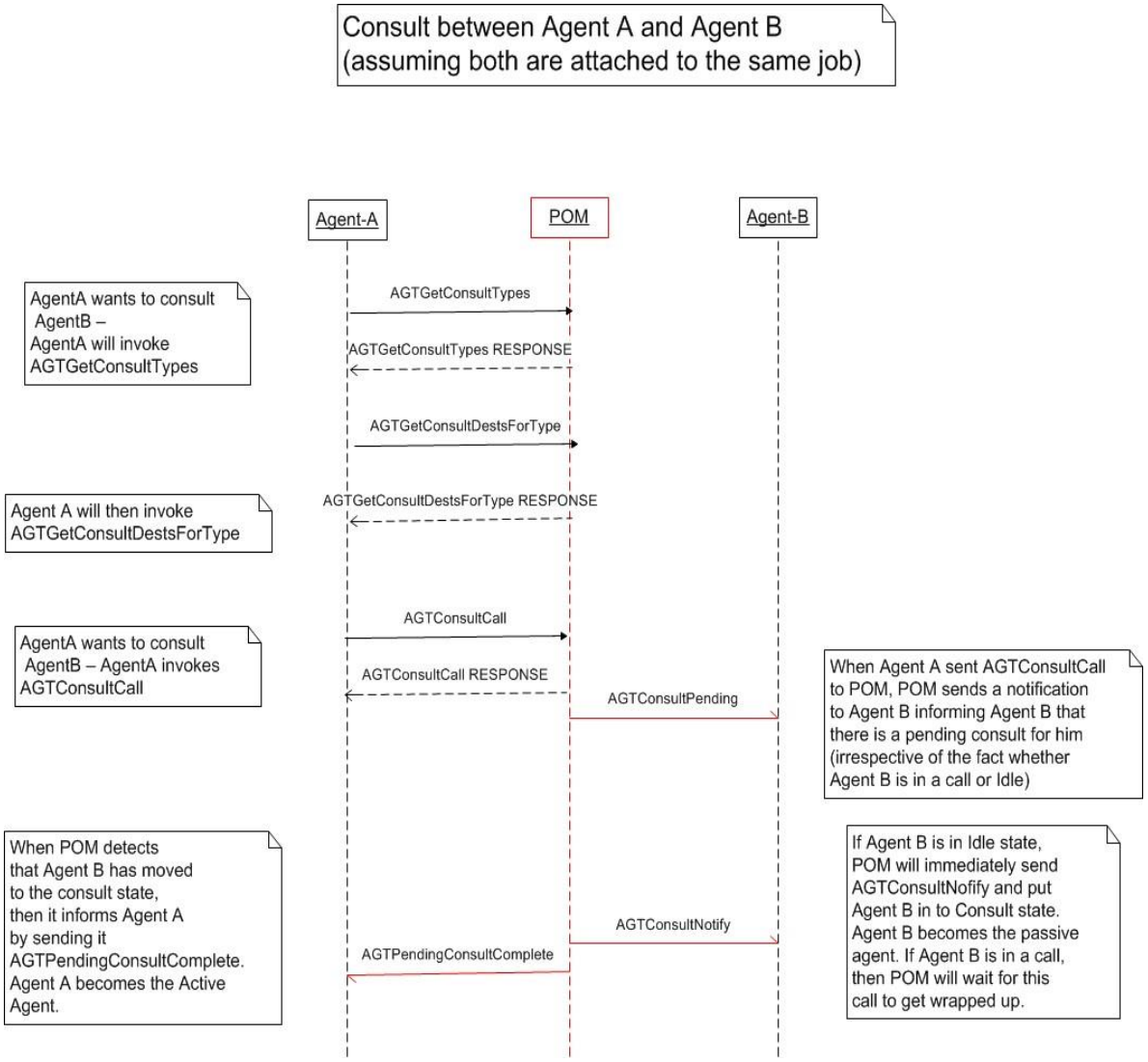
1. Consult is not a capability. So you can use the following logic to control the accessibility of the consult button/menu:
    a. Enable Consult button IF (LEAVECONSULT : FALSE, CANTRANSFER: TRUE, CANCONFERENCE : TRUE)
    b. Disable Consult button IF (LEAVECONSULT : TRUE, CANTRANSFER: TRUE, CANCONFERENCE : TRUE)
2. Callback – Callbacks can be set for Free form number (i.e. a contact number not currently associated with the contact on POM) and in this scenario it is expected the user specifies the string “ExternalNumber” in the contact number object.
3. Agent walkaway – The desktop developer track agent actions, such that if for two consecutive calls the agent has not clicked anything on the desktop, then the desktop must send AGTStateChange (NotReady) with the walkaway flag set to true. This operation ensures that POM stops sending calls to the agent if the agent has walked away from his station.
4. It is the customer’s preference to allow/disallow the passive agent involved in a consult/ conference to modify/update/work on the desktop. Only relevant buttons should be enabled. The passive party should not be allowed to modify customer details. Only agent notes should be enabled.

Sequence Diagrams

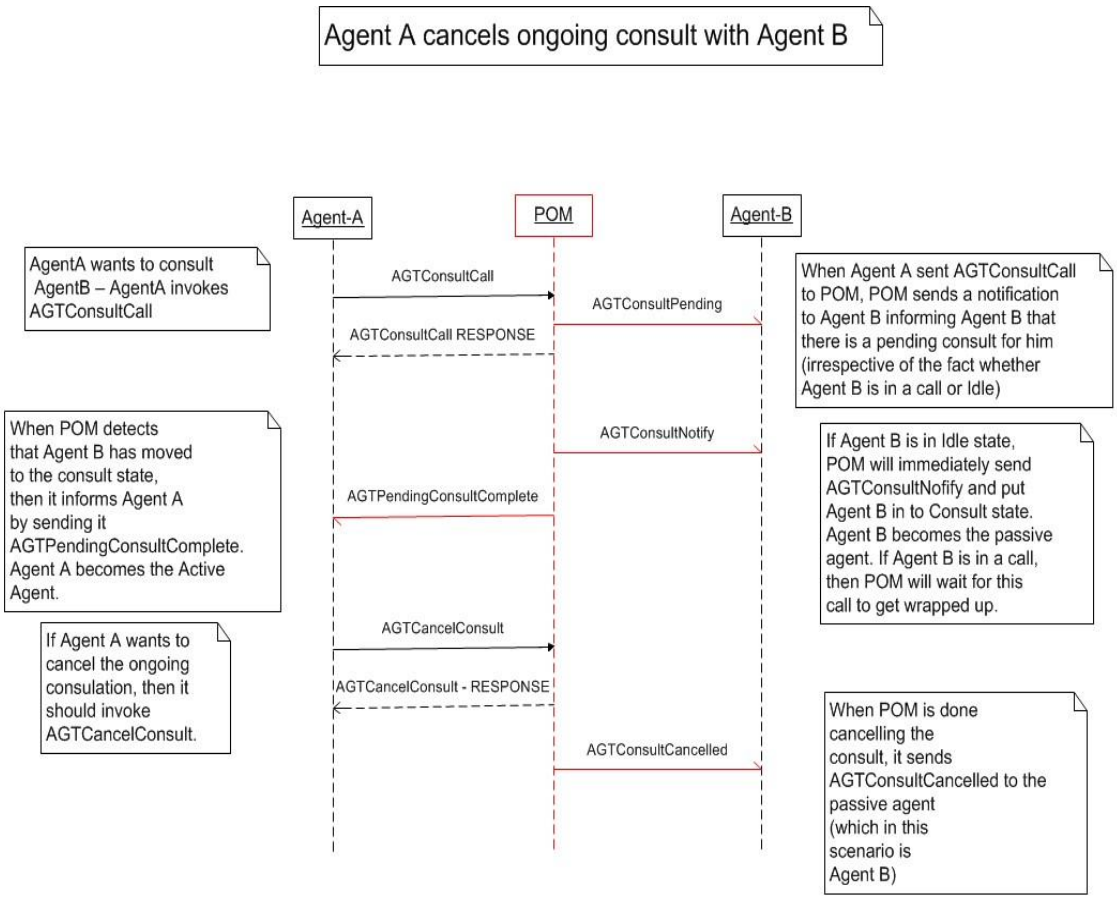
Sequence 1 - Nailing



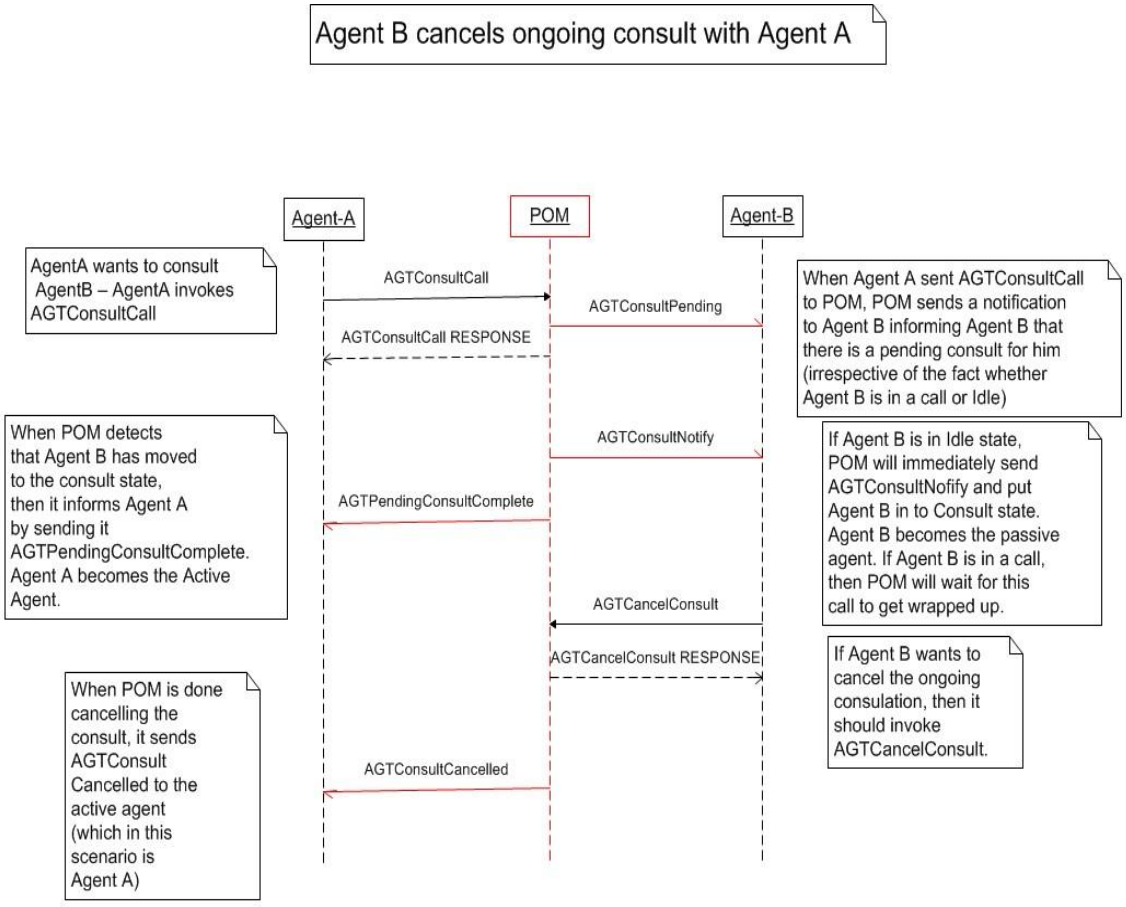
Sequence 2 - Consult



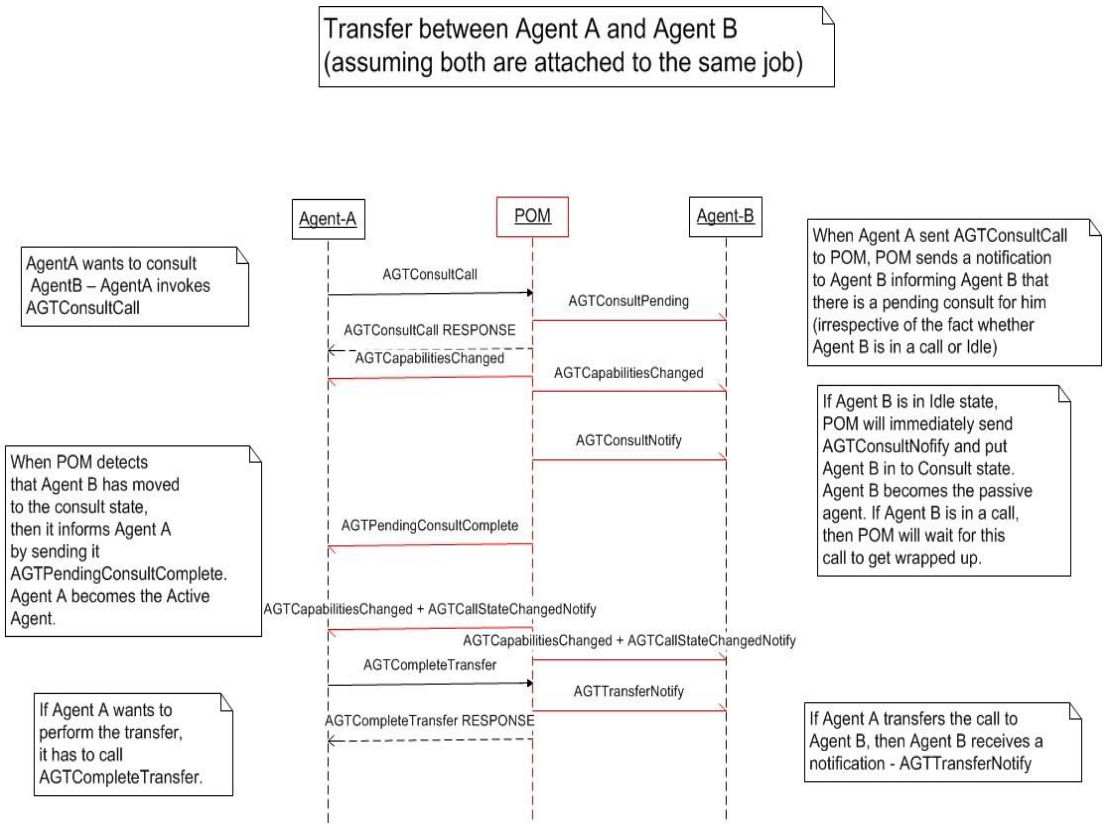
Sequence 3 - Cancel consult by active agent



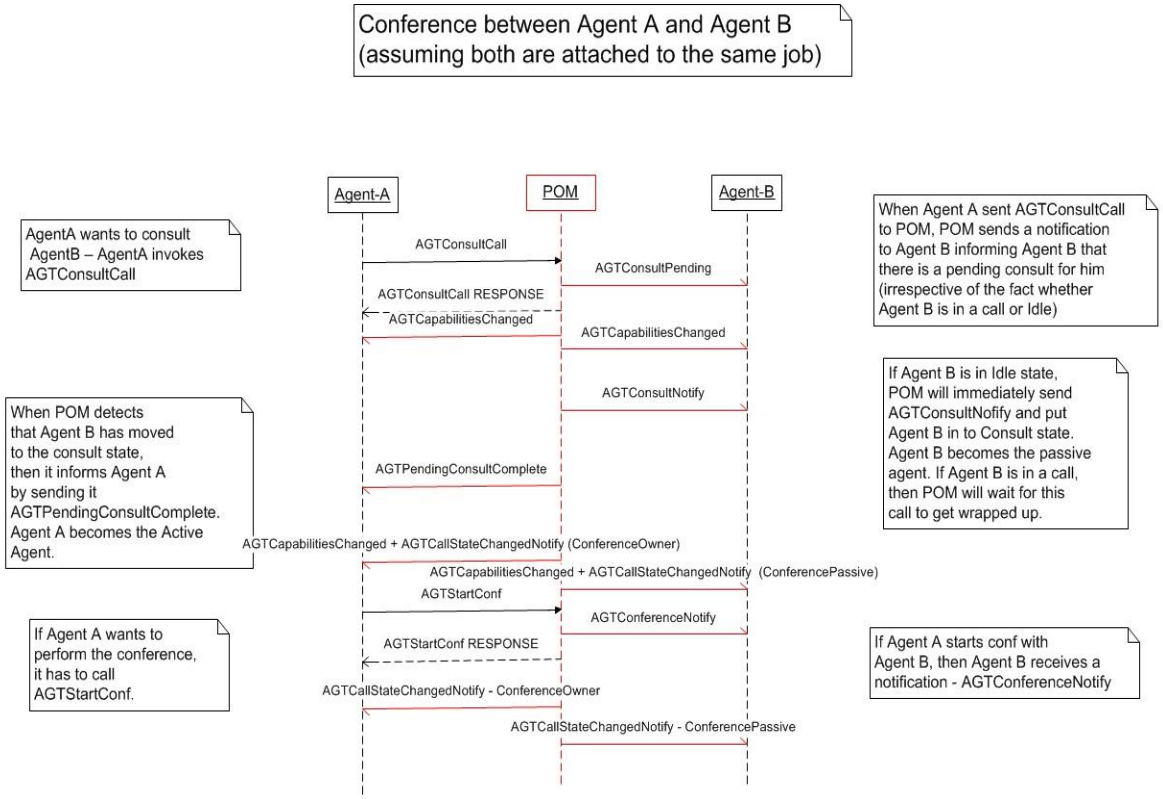
Sequence 4 - Cancel consult by passive agent



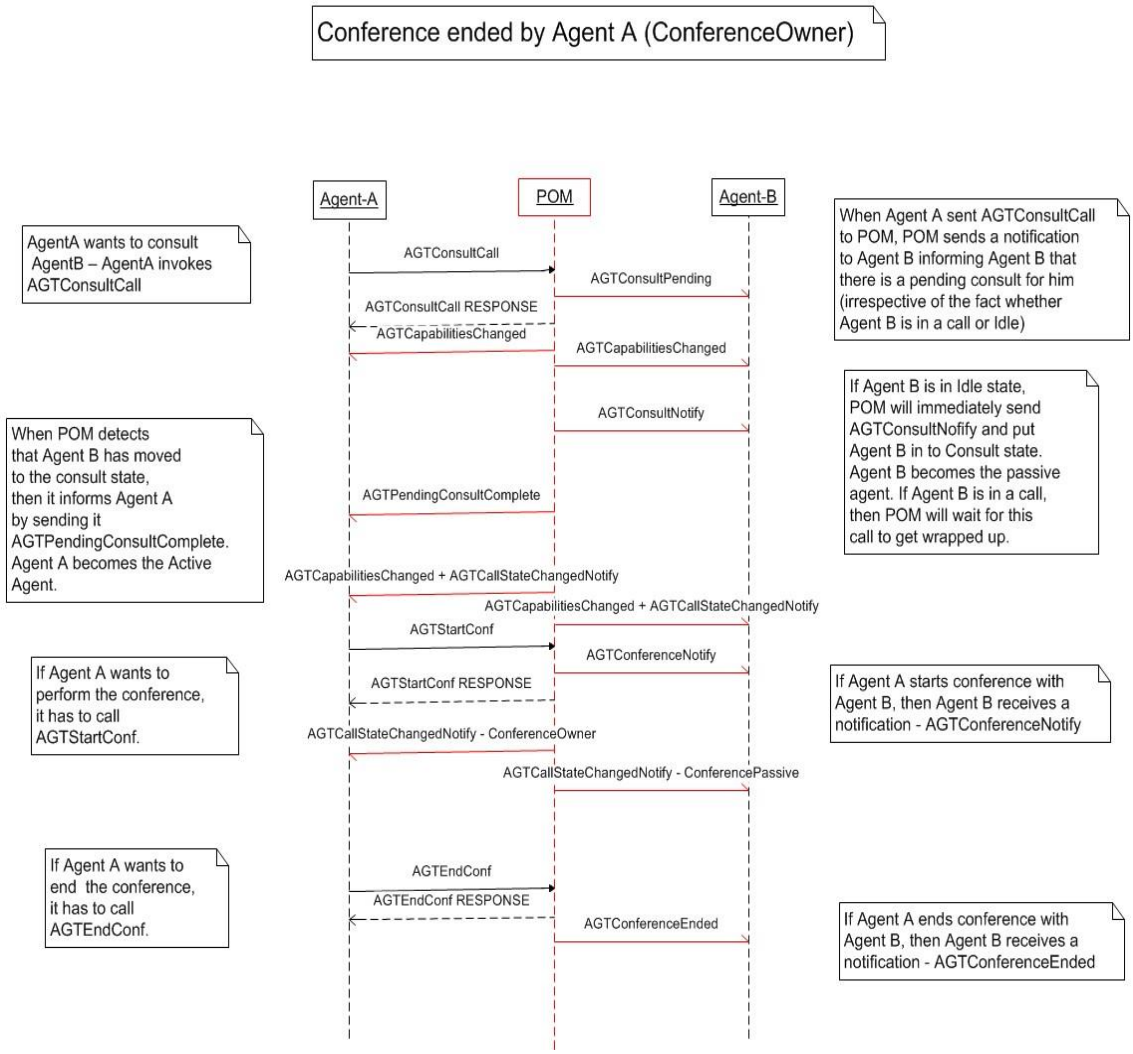
Sequence 5 - Transfer by active agent



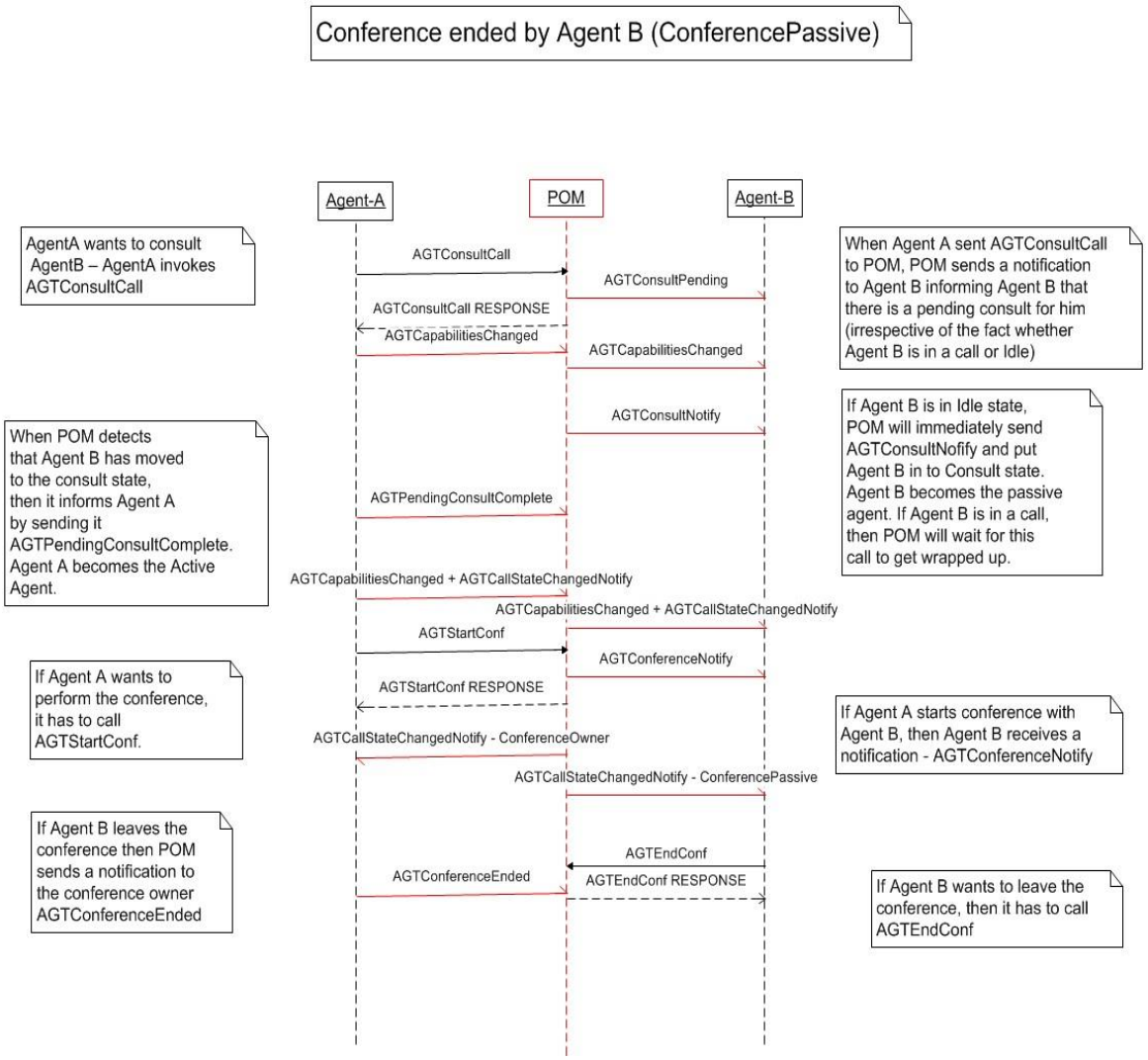
Sequence 6 - Conference by active agent



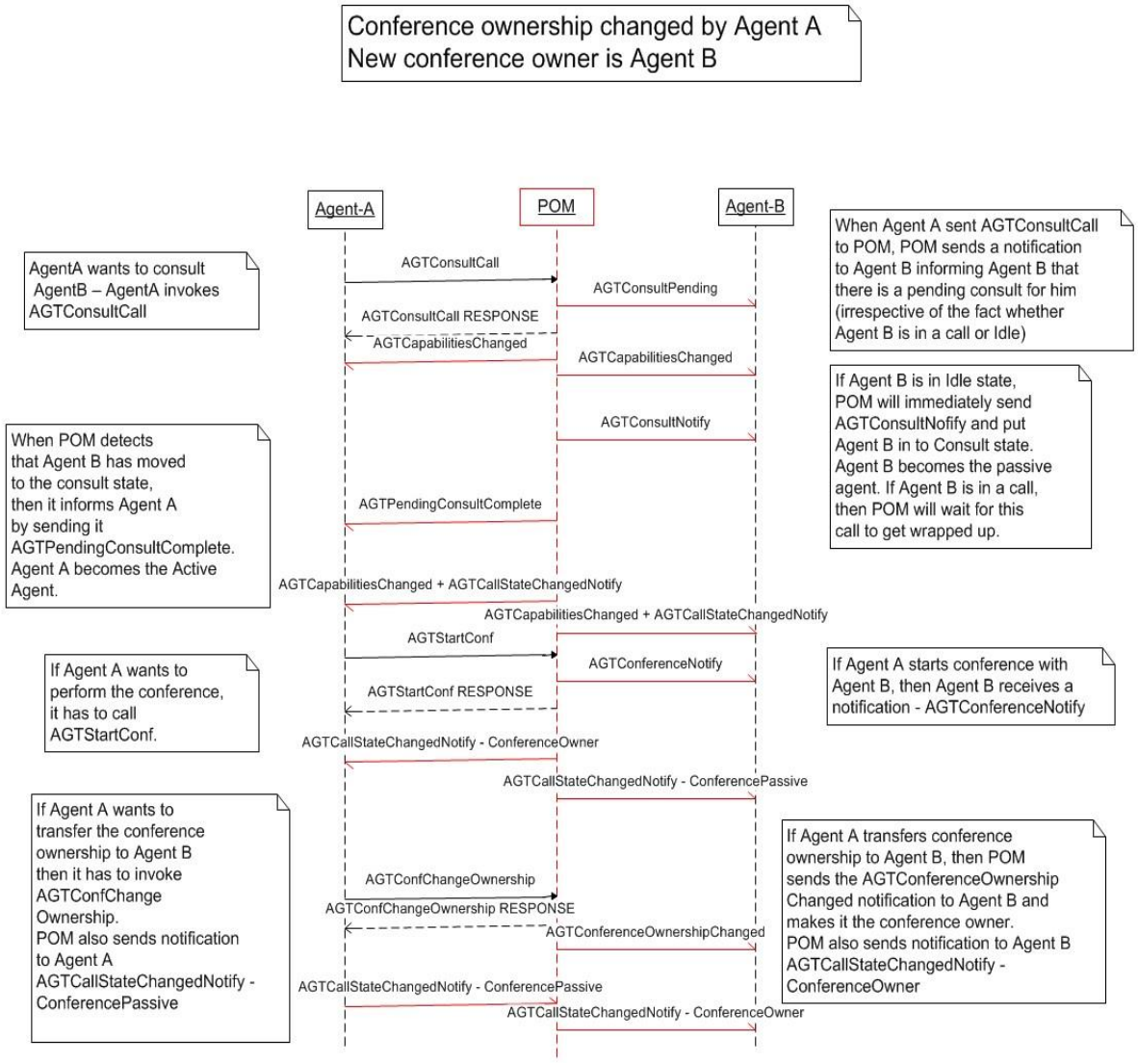
Sequence 7 - Conference end by conference owner



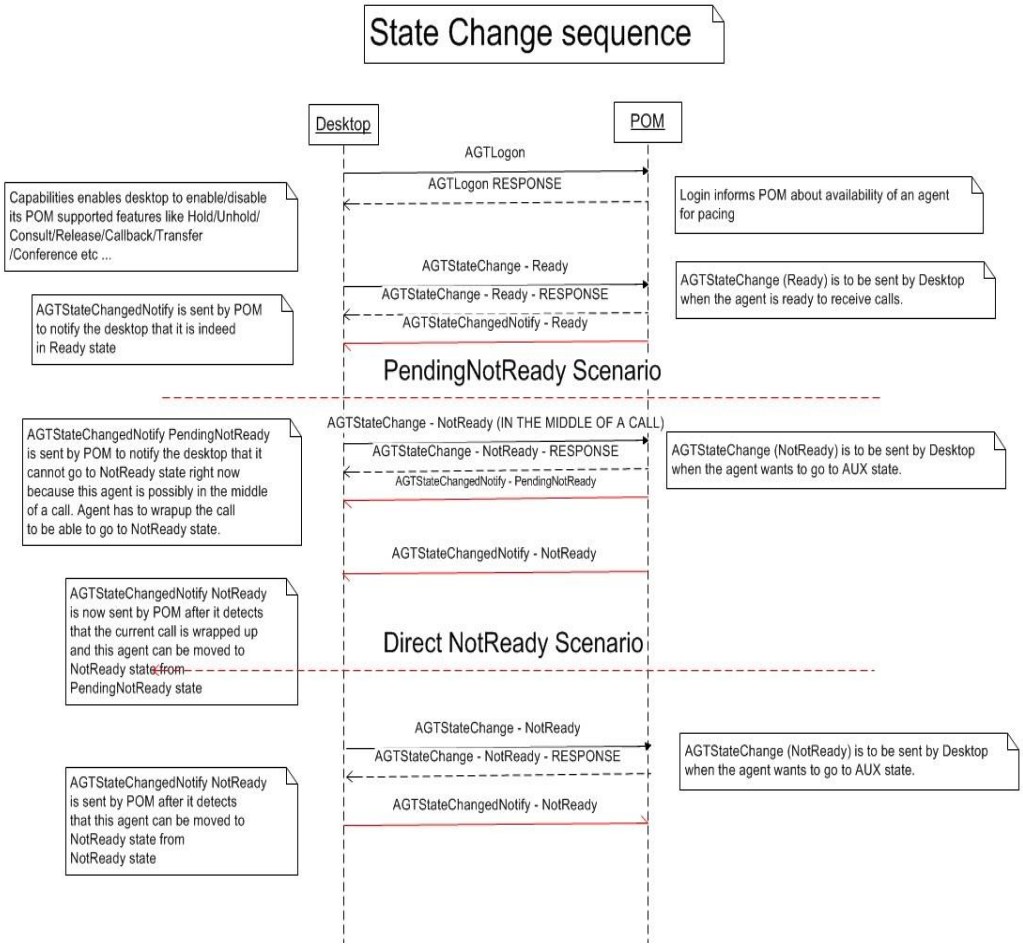
Sequence 8 - Conference left by passive agent in a conference



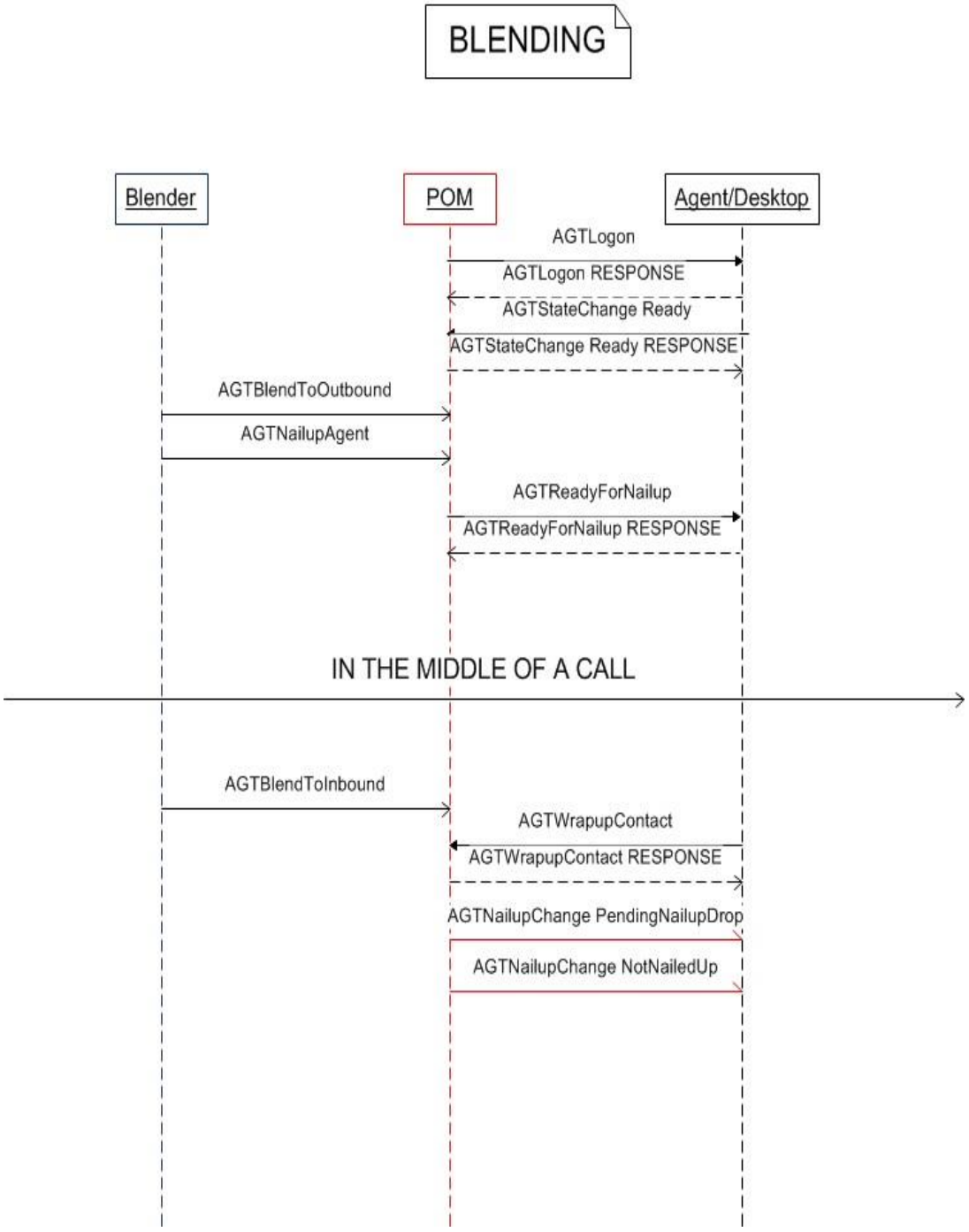
Sequence 9 - Conference ownership changed by conference owner



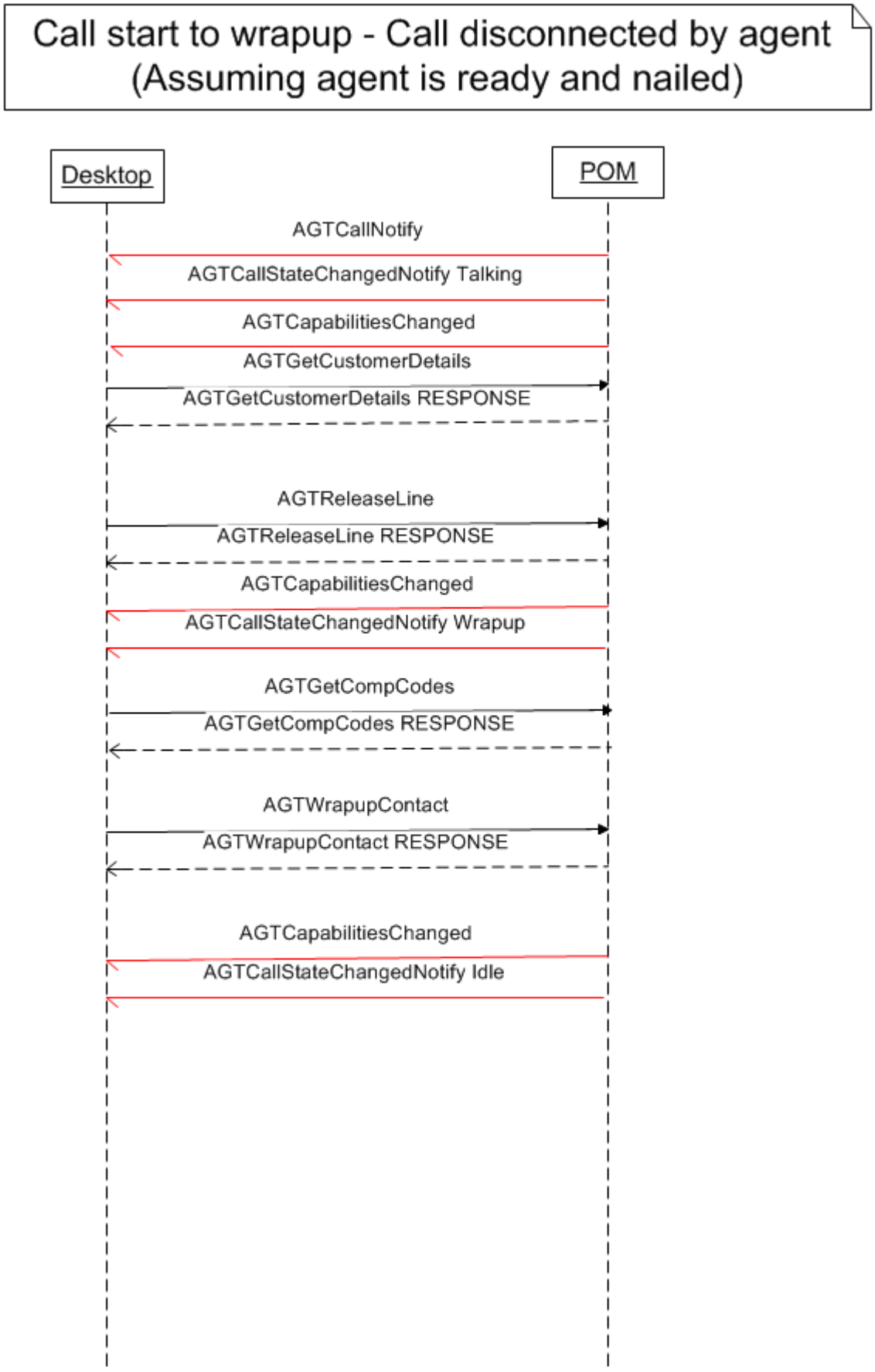
Sequence 10 - State change sequence



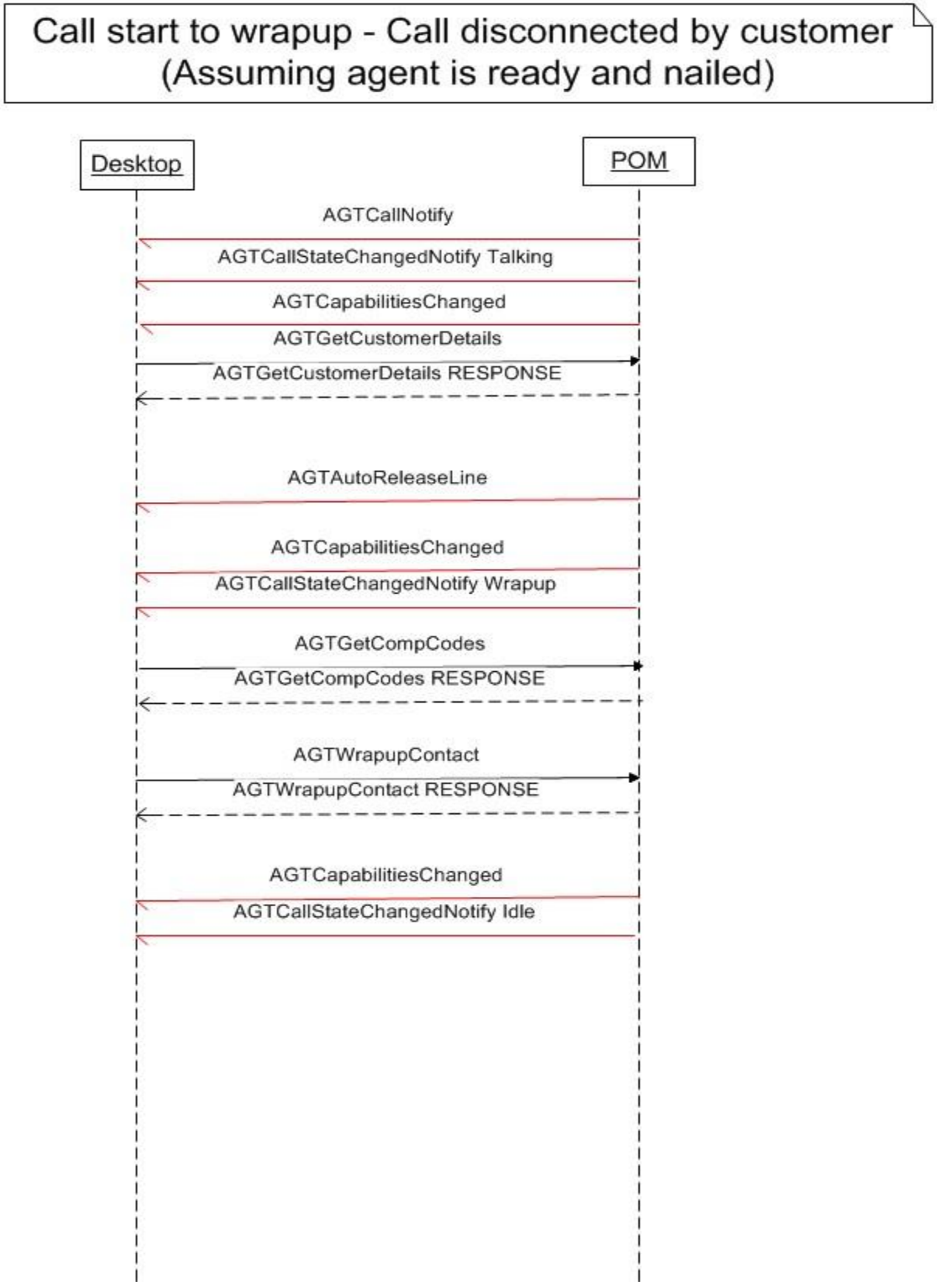
Sequence 11 - Blending



Sequence 12 - Call drop by agent



Sequence 13 — Call drop by customer



Sample Code

Sample code to use the API libraryThe code is written in VB.net

Initialize the library

```
Dim libInitDone As Boolean = False
libInitDone = POMAgentFactory.init(pamSocketInfoarray)

If libInitDone = False Then
MsgBox("Library could not be initialized. None of the PAM are reachable.")
      loginForm.Visible = True
End If
```

Create agent and log in

```
Private agentHandler As POMAgentHandler
agentHandler = New POMAgentHandler
agent = POMAgentFactory.getPOMAgent(AgentID, agentHandler)

Dim returnCode As Int32 = agent.AGTLogon(AgentExt, Password, forceLogin, agentLocale,
timeZone, zoneName)
```

Log off agent

```
agent.AGTLogoff()
```