



Avaya Context Store Snap-in

Developer Guide

Release 3.1.0.1

Issue 3

January 2017

AVAYA SOFTWARE DEVELOPMENT KIT LICENSE AGREEMENT

REVISED: October 12, 2015

Using this document signifies your assent to the following SDK License Terms.

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 object code, Client Libraries, Specification Documents, Software libraries, application programming interfaces ("API"), Software tools, Sample Application Code, published specifications 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: (a) for copyrightable or copyrighted material, 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; (b) for patentable or patented material, any changes, additions, modifications or improvements thereon; and (c) for material which is protected by trade secret, any new material derived from such existing trade secret material, including new material which may be protected by copyright, patent and/or trade secret. Permitted Modifications will be considered Derivative Works.
- 1.6 "Documentation" includes, but is not limited to 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 tangible and intangible: (i) rights associated with works of authorship throughout the world, including but not limited to 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 Source 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 the high-level statement version of the Sample Application Code or Software written in the source language used by programmers and includes one or more programs. Source Code programs may include one or more files, but is not limited to, 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) and extensible markup language (.xml) files. Source Code files may also be provided in binary object format, may require explicit compilation into binary object format for execution, or may be interpreted natively using a separate application execution program or platform.

- 1.12 "Sample Application Code" means Source Code and/or executable Software provided for the purposes of demonstrating functionality of an Avaya product through the Avaya Software Development Kit.
- 1.13 "Software" means Avaya's intangible information constituting one or more computer or apparatus programs, including, but not limited to, Avaya software in 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)) to use the SDK (including Sample Application Code) 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 provided, however, that Licensee shall have no right to distribute, license (whether or not through multiple tiers) or otherwise transfer the SDK to any third party or incorporate the SDK in any software, product, or technology. Avaya further grants Licensee the right, if the Licensee so chooses, 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. Where SDK includes Specification Document(s), Licensee is granted a license to use such 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. Avaya's support obligations for the SDK, Sample Application Code and any Derivative Works are set forth in Section 4 of this Agreement.

B. The foregoing license to use Sample Application Code is contingent upon the following: (i) Licensee may use and modify 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, (ii) Licensee must ensure that the modifications made to the Sample Application Code as permitted in clause (i) of this Section 2.1B are compatible and/or interoperable with Avaya products and/or integrated therewith, (iii) Licensee may compile or otherwise prepare for distribution the Sample Application Code with Permitted Modifications, into an object code or other suitable program format for distribution, provided that such sublicense is subject to an end user license agreement

that is consistent with the terms of this Agreement and, if applicable, 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. Under no circumstances shall Licensee enable the use or activation of any of Avaya's Intellectual Property by an end user, without such end user having acquired the additional necessary licenses to Avaya Intellectual Property. Avaya's support obligations for the SDK, Sample Application Code and any Derivative Works are set forth in Section 4 of this Agreement.

C. Except as expressly authorized by this Agreement, and unless otherwise permitted by the applicable law, Licensee shall not: (i) translate, publish, or display the SDK, Specification Documents or Documentation or any copy or part thereof; or (ii) 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 agrees that it is licensed to use the SDK only in connection with Avaya products (and if applicable, in connection with services provided under the Avaya DevConnect Program Agreement). In the event of any conflict between the terms and conditions of this Agreement and the Avaya DevConnect Program Agreement (if applicable), the terms and conditions of the Avaya DevConnect Program Agreement shall prevail.

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: <https://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; provided, however that Licensee may also make one (1) copy for back-up purposes and any reproduction of the SDK (including derivatives thereof), either in whole or in part, shall include the Avaya copyright notice that was provided in the SDK.
- 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 and, except as expressly set forth herein, no license is granted by Avaya under this Agreement directly, by implication, estoppel or otherwise, under any patent, copyright, trade secret or trademark or other Intellectual Property right of Avaya. 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 Nonassertion by Licensee. Licensee agrees not to assert any patent rights related to the SDK or applications developed using the SDK against Avaya, Avaya's distributors, Avaya customers, or other licensees of the SDK for making, using, selling, offering for sale, or importing any products or technology developed using the SDK.
- 2.11 Avaya Independent Development. Licensee understands and agrees that Avaya or its Affiliates may acquire, license, develop for itself or have others develop for it, and market and/or distribute similar software to that which Licensee may develop. In absence of a separate written agreement to the contrary, Avaya or its Affiliates will be free to use any information Licensee provides, including problem reports or enhancement requests, to Avaya for any purpose, subject to any applicable patents or copyrights.
- 2.12 Feedback and Support. Licensee agrees to provide any comments and suggestions regarding the performance of the SDK (a) if applicable, on the developer forum of the DevConnect Program on www.avaya.com/devconnect; or (b) via the process otherwise indicated by Avaya with respect to the SDK. Avaya agrees to monitor the applicable forum but is under no obligation to implement any of the suggestions and/or proposals, or be required to respond to any questions asked in the forum. Self-support tools are available via the Avaya DevConnect program's portal and requires self registration. Licensee hereby assigns to Avaya all right, title, and interest in and to Feedback provided to Avaya.
- 2.13 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.

2.14 **No Endorsement.** Neither the name Avaya, its 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.

2.15 **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.

2.16 **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 Supplier Software 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, its 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.

2.17 **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 shall own and retain all proprietary rights, including all patent, copyright, trade secret, trademark and other 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 all of its right, title, and interest therein. Avaya 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, sub-licensable, 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.

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. Notwithstanding the above limitations, 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 a Avaya DevConnect Program Agreement or any support agreement with Avaya. Nothing herein shall be construed to require Avaya to provide support services or updates, upgrades, bug fixes or modifications to the SDK.

5.0 CONFIDENTIALITY.

5.1 **Protection of Confidential Information.** Licensee shall take all reasonable measures to maintain the confidentiality of the SDK, Specification Documents and other Avaya technical information obtained by it (collectively, the "Confidential Information"), and will not disclose the Confidential Information to any third party. Licensee agrees at all times to protect and preserve the SDK in strict confidence and perpetually, and shall not use such Confidential Information other than as expressly authorized by Avaya under this Agreement, nor shall Licensee disclose any such 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 review and 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 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, its 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 with other software, such as operating systems and codecs, and the, direct or indirect, distribution or sale of 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 and run concurrently with Licensee's membership in the Avaya DevConnect Program, if applicable, unless terminated as specified in Section 10.2 or 10.3 below, and, if applicable, provided Licensee is a member of the Avaya DevConnect Program in a good-standing as determined by Avaya at its sole discretion.

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 license 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 Development Kit, 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 (excluding the Avaya DevConnect Program Agreement) 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.

Any file under the "examples" directory can be redistributed with or without modification.

Schedule 1 to Avaya SDK License Agreement Third Party Notices

Apache Log4j : 1.2.14. Copyright © 2001-2011 The Apache Software Foundation, Licensed under the Apache License, Version 2.0.

Joda - Time - joda-time : 2.1. Copyright © 2001-2011 Stephen Colebourne, Licensed under the Apache License, Version 2.0.

Jackson : 1.9.0. Copyright © 2001-2011 The Apache Software Foundation (originally written by Tatu Saloranta), Licensed under the Apache License, Version 2.0.

Jackson - org.codehaus.jackson:jackson-core-asl : 1.9.12. Copyright © 2001-2011 The Apache Software Foundation (originally written by Tatu Saloranta), Licensed under the Apache License, Version 2.0.

Data Mapper for Jackson : 1.9.0. Copyright © 2001-2011 The Apache Software Foundation (originally written by Tatu Saloranta), Licensed under the Apache License, Version 2.0.

Commons Logging - commons-logging:commons-logging : 1.0.4. Copyright © 2001-2004 The Apache Software Foundation, Licensed under the Apache License, Version 2.0.

Apache Commons Codec (for Apache Directory Studio) : 1.8. Copyright © 2001-2015 The Apache Software Foundation. All Rights Reserved. Licensed under the Apache License, Version 2.0.

Apache Commons IO (for Apache Directory Studio) : 2.4. Copyright © 2001-2011 The Apache Software Foundation.

Licensed under the Apache License, Version 2.0.

Apache ServiceMix :: Bundles :: commons-configuration : 1.9.2. Copyright © 2001-2011 The Apache Software Foundation. Licensed under the Apache License, Version 2.0.

Lang : 2.3, Apache Jakarta Commons Lang. Copyright © 2001-2007 The Apache Software Foundation. Licensed under the Apache License, Version 2.0.

Lang3 : 3.1, Copyright © 2001-20011. The Apache Software Foundation, All Rights Reserved. Licensed under the Apache License, Version 2.0.

httpcore : 4.*. Copyright © 1999- 2013 The Apache Software Foundation, Licensed under the Apache License, Version 2.0.

httpClient : 4.*. Copyright © 1999- 2013 The Apache Software Foundation, Licensed under the Apache License, Version 2.0.

Compiler assisted localization library (CAL10N) - API : 0.7.4. The MIT License (MIT), Copyright © 2009 QOS.ch All rights reserved.

SLF4J API Module : 1.7.2. Copyright © 2004-2011 QOS.ch. The MIT License (MIT) [OSI Approved License] The MIT License (MIT) Copyright © Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the 'Software'), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

SLF4J Extensions Module : 1.7.2. Copyright © 2004-2011 QOS.ch. The MIT License (MIT) [OSI Approved License] The MIT License (MIT) Copyright © Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the 'Software'), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Contents

1. Intended audience	15
1.1. Related Documentation.....	15
1.2. DevConnect Resources	15
2. Introduction	16
2.1. Platform	16
2.1.1. Platform Sizing	16
3. RESTful Interface	17
3.1. Overview	17
3.1.1. API Documentation	17
3.1.2. Case Sensitive	17
3.1.3. REST Clients.....	18
3.1.4. Sample API Requests	18
3.2. Audit Trail Feature (since CS 3.1).....	20
3.2.1. Overview	20
3.2.2. Audit Entry.....	20
3.2.3. Enabling and setting the size of Audit Trail	21
3.2.4. Considerations for Capacity Planning	22
3.3. Upsert Feature (since CS 3.1)	23
3.3.1. Overview	23
3.3.2. Upsert Parameters	23
3.4. CSRest API	25
3.4.1. Context Object Information	25
3.4.2. Query Parameters	33
3.4.3. POST – Adding a Context.....	36
3.4.4. UPSERT – Upserting a Context.....	37
3.4.5. UPSERT – Upserting a Context which has optional identifier fields.....	37
3.4.6. UPSERT – Upserting a Context by aliasId	38
3.4.7. UPSERT – Upserting a Context with aliasIds	38
3.4.8. UPSERT – Upserting a Context with aliasIds by aliasId.....	39
3.4.9. GET – Getting a Context.....	39
3.4.10. GET – Getting a Context value	40
3.4.11. GET – Getting a Context by aliasId.....	40
3.4.12. GET – Getting a Context value by aliasId.....	41
3.4.13. GET – Getting a Context's Audit Data	41
3.4.14. GET – Getting a Context's Audit Data by aliasId	42

3.4.15.	GET – Getting Contexts’ Metadata by groupId	42
3.4.16.	UPDATE – Updating a Context.....	43
3.4.17.	UPDATE – Updating a Context value	43
3.4.18.	UPDATE – Updating a Context by aliasId	44
3.4.19.	UPDATE – Updating a Context value by aliasId.....	44
3.4.20.	UPDATE – Adding an aliasId by contextId	45
3.4.21.	UPDATE – Adding an aliasId by aliasId.....	45
3.4.22.	DELETE – Deleting a Context.....	46
3.4.23.	DELETE – Deleting a Context value	46
3.4.24.	DELETE – Deleting a Context by aliasId	46
3.4.25.	DELETE – Deleting a Context value by aliasId.....	47
3.4.26.	DELETE – Deleting an aliasId.....	47
4.	Context Store SDK Tutorial	49
4.1.	Java Client.....	49
4.1.1.	Overview	49
4.1.2.	SDK operations	50
4.1.3.	Configuration	55
4.1.4.	Usage	56
4.2.	C# Client.....	57
4.2.1.	GUI layout	57
4.2.2.	Application Layout:.....	58
4.3.	Security	60
5.	Screen Pop	61
5.1.	Overview	61
5.2.	Screen Pop Operations on the Context Store.....	61
5.3.	Output Format	61
5.4.	Rules Engine	62
5.5.	Rest Service	62
5.6.	Data-grid.....	62
5.7.	URLs & Operations	62
5.7.1.	Create & View a Context.....	63
5.7.2.	Update & View a Context	64
5.7.3.	View a Context	65
5.7.4.	View a Context by Selecting contextId from Parameter (UCID)	66
5.7.5.	View contextIds by groupId	67
5.7.6.	Upsert & View a Context	67
5.7.7.	View a Context by Selecting contextId from Parameter (UCID)	69

5.7.8.	View contextIds by groupId	70
5.7.9.	Older Browser Compatibility.....	70
5.7.10.	JSON Content Type	71
5.8.	Response Formats	71
5.8.1.	Format Selection	71
5.8.2.	Format Types	72
5.9.	Pre-Configured Rules.....	78
5.9.1.	Rules Engine Overview	78
5.9.2.	Default Rules Provided	78
5.10.	Editable Rules	81
5.10.1.	Configurable Properties	85
5.10.2.	Base for URL.....	86
5.10.3.	Context Store Rest Version.....	87
5.10.4.	CSS for HTML.....	87
5.10.5.	Identifier Delimit Character.....	87
5.10.6.	Identifier Delimit Position.....	87
5.10.7.	Identifier Parsing Position	87
5.10.8.	User Rules 01-20	88
5.10.9.	Example URLs & Their Functions	88
5.11.	Sample Usages.....	98
5.11.1.	Create and View.....	98
5.11.2.	Create, Update and View	98
5.11.3.	Create, Update Multiple Times, View then Redirect	99
5.11.4.	Create Context in CSRest, Update and Redirect.....	101
5.12.	Example Configuration for Communication Manager	102
5.12.1.	Configure CM to support Screen Pop	102
5.13.	Example Configuration for One-X Screen Pop	105
5.13.1.	Configuring Security Certificates for Agent Desktops	105
5.13.2.	Configuration of the Screen Pop feature in the One-X Agent Desktop client	105
6.	CRM Integration	107
6.1.	Overview	107
6.1.1.	Features	107
6.1.2.	Caveats	107
6.2.	High Level Design	107
6.2.1.	Basic Flow	108
6.2.2.	Example CRM & CS Integration.....	108
6.3.	Configuration	108

6.3.1.	Create EDP Event	109
6.3.2.	Create Workflow	110
6.3.3.	Configure Rule	111
6.3.4.	Firing a Rule	112
7.	Context Store PDC	115
7.1.	Overview	115
7.2.	Installation Prerequisites of the Context Store PDC in Eclipse/Orchestration Designer	115
7.2.1.	Software requirements	115
7.2.2.	Prerequisites	115
7.3.	Installing/Upgrading the Context Store PDC plugin	116
7.4.	Tomcat configuration in Orchestration Designer.....	118
7.5.	Configuring certificates in Orchestration Designer.....	119
7.6.	Using the Context Store Connector	120
7.6.1.	Configure the Context Store plugin	120
7.6.2.	Add the Context Store Connector in the workflow	121
7.6.3.	Context Store PDC input/output variables	123
7.6.4.	Retrieving output variable values	130
7.7.	Context Store PDC – Running Sample Projects.....	132
7.7.1.	Configuring the project to use the Context Store PDC	133
7.7.2.	Testing Context Store PDC Sample Applications	135
7.7.3.	Sample Audit Trail Application	136
7.8.	PDC - Experience Portal Test Setup.....	137
7.8.1.	Overview	137
7.8.2.	What's needed	137
7.8.3.	Sample Call Flow One.	137
7.9.	Tomcat configuration in Orchestration Designer.....	Error! Bookmark not defined.
7.10.	Configuring certificates in Orchestration Designer.....	Error! Bookmark not defined.
7.11.	Using the Context Store Connector	Error! Bookmark not defined.
7.11.1.	Configure the Context Store plugin	Error! Bookmark not defined.
7.11.2.	Add the Context Store Connector in the workflow	Error! Bookmark not defined.
7.11.3.	Context Store PDC input/output variables	Error! Bookmark not defined.
7.11.4.	Retrieving output variable values	Error! Bookmark not defined.
7.11.5.	Sample Call Flow Two.....	154
7.11.6.	Installing the sample application plus runtimeconfig on an Experience Portal system...	154
8.	Context Store Task Type for Engagement Designer	156
8.1.	Overview	156
8.2.	Usage	156

8.2.1.	Input and Output.....	156
8.2.2.	CS Task Type Operations.....	159
8.2.3.	Creating a workflow in ED.....	163
8.2.4.	Importing the Sample Workflow into the Designer.....	167
8.2.5.	Validation.....	168
8.2.6.	Re-use of Collaboration Designer 3.0 Workflows	168
9.	Notifications	170
9.1.	Overview	170
9.2.	Security Configuration.....	170
9.3.	Usage	171
9.4.	Performance/Capacity.....	173
9.5.	High Availability and Fault Tolerance.....	173
	Event Tracker (Agent Notifications)	174
9.6.	Overview	174
9.7.	Configuration	174
9.7.1.	Configure EDP and Context Store	174
9.7.2.	Install Security Certificates.....	175
9.8.	Usage	176
9.8.1.	Setup the JavaScript Test Client.....	176
9.8.2.	Register and Verify an Event Stream.....	176
9.8.3.	Advanced Event Stream Configuration	177
10.	Performance and Scalability Considerations	178
10.1.	Overview	178
10.2.	Capacity Planning	178
10.2.1.	Enabling Optional Features.....	179
10.3.	Hardware and Network	180
10.3.1.	CPU.....	180
10.3.2.	Network	180
10.3.3.	VMware	180
10.4.	High Availability.....	181
10.5.	Geo Redundancy	181
10.6.	External DataMart	182
10.6.1.	Provisioning from External DataMart	182
10.7.	Throttling	182
10.8.	Sequenced Apps.....	183
11.	Appendix	184
11.1.	Context Store API Documentation	184

11.1.1.	Overview	184
11.1.2.	Data Types	184
11.1.3.	Context Operations	186
11.1.4.	Interface Error Codes	201
11.2.	Certificate Based Authentication	204
11.2.1.	Configuring Client Certificate Challenge	204
11.2.2.	Create Client Keystore	204
11.2.3.	Download Avaya Aura® Engagement Development Platform Trusted Certificate from System Manager	206
11.2.4.	Import Trusted Certificate into Keystore.....	208
11.2.5.	Verifying Successful Authentication	208
11.2.6.	General Information Regarding Java SSL	208
11.2.7.	Thin Client Access.....	209
11.2.8.	Troubleshooting SSL Connections.....	209
11.3.	A10 Load Balancer Configuration	211
11.3.1.	A10 Installation.....	211
11.3.2.	HTTP Traffic Configuration	211
11.3.3.	HTTPS Traffic Configuration	216

1.Intended audience

This document is intended as a guide for developers who want to utilize the Context Store services and SDK's. It assumes that developers have a working knowledge of the technologies and products referenced in the document.

1.1. Related Documentation

The following table lists the related documentation that might be a useful source of information.

Table 1 Additional Reading

Document
Avaya Aura® Avaya Aura® Engagement Development Platform Overview and Specification
Deploying Avaya Aura® Avaya Aura® Engagement Development Platform
Upgrading Avaya Aura® Avaya Aura® Engagement Development Platform
Administering Avaya Aura® Avaya Aura® Engagement Development Platform
Maintaining and Troubleshooting Avaya Aura® Avaya Aura® Engagement Development Platform
Quick Start to Deploying Avaya Aura® Avaya Aura® Engagement Development Platform Snap-ins
Avaya WebRTC Snap-in Reference
Engagement Designer Snap-in Reference
Getting Started with the Avaya Engagement Designer Snap-in
Avaya Engagement Designer Snap-in Developer's Guide
Avaya Context Store Snap-in Reference

1.2. DevConnect Resources

Useful materials for the various Context Store services and features, such as sample clients and workflows, are provided through the Avaya Context Store DevConnect site:

http://www.devconnectprogram.com/site/global/products_resources/engagement_development_platform/ava_ya_snap_ins/context_store/overview/index.gsp

2. Introduction

Context Store is designed to provide a centralized data cache to the various applications in a Contact Center or Unified Communications environment. Using a distributed cache, the solution provides a scalable, reliable and fault-tolerant system to store context information and provide a set of services for Context information changes.

Through standard open interfaces, Context Store provides a framework for customers to collect and share real time contextual information across the enterprise to be consumed by their various contact center components.

Context Store runs as a snap-in on the Avaya Aura® Engagement Development Platform (Engagement Development Platform) 3.1 Platform. The Engagement Development Platform is designed to host applications that require features like High Availability and the ability to scale.

Context Store integrates with the Engagement Designer 3.1 snap-in and this facilitates the creation of business or work flows that require interaction with CS3.1

Context Store is a centralized data cache that provides a scalable, reliable, and fault-tolerant system to store context information.

2.1. Platform

Context Store runs on Engagement Development Platform 3.1 and hence requires that the Avaya Aura® System Manager (System Manager) is upgraded to the supported lineup*. As Context Store is not directly involved in the media signaling, it has no explicit dependency on any other Avaya application software versions that might exist in the customer Environment.

If a customer is currently using SIP users on System Manager, they might be required to upgrade their Aura infrastructure as per Engagement Development Platform 3.1 requirements. Refer to Engagement Development Platform 3.1 documentation for further details on when to upgrade the Aura infrastructure.

2.1.1. Platform Sizing

Context Store 3.1 introduces a new scaling feature which allows the product to be deployed on a single Engagement Development Platform node or in a cluster of up to five nodes. The resources (i.e. CPU and RAM) allocated to the nodes on which Context Store is deployed can also be varied to suit business needs.

For detailed sizing configuration and capacity planning information see the Avaya Aura® Context Store 3.1 Reference Guide.

* For EDP 3.1 this is SMGR 7 but please refer to EDP documentation.

3. RESTful Interface

3.1. Overview

Context Server implements a RESTful Web Services interface to provide the required services to its clients. The use of REST allows many applications written in different technologies, running on different platforms, a mechanism to communicate and exchange data. This REST architecture implements a stateless communication between client and server. This means each request from any client contains all the information necessary to service the request, and any session state is held in the client.

Individual URL's are identified in requests and conceptually separate from the representations that are returned to the client. In the case of the Context Store web-based REST implementation, resources are identified using HTTP URIs and returned to clients using JavaScript Object Notation (JSON) representation.

Following the RESTful Web Service principles, Context Store exposes all the operations using HTTP Methods (e.g., GET, PUT, POST, or DELETE).

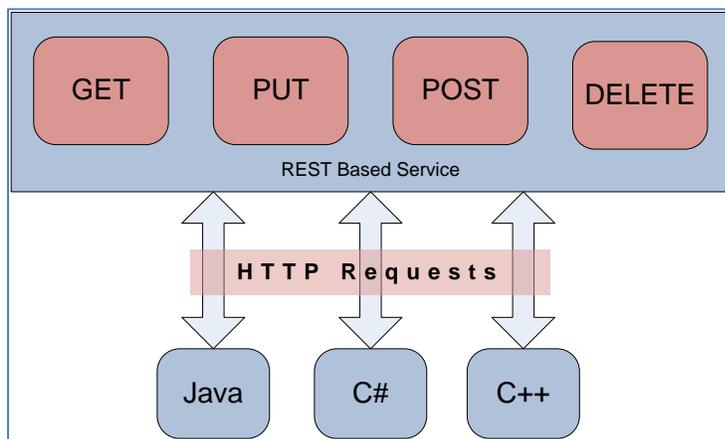


Figure 1: REST Commands

Context Store requests use HTTP so when waiting on a create or update to return any timeout value must be set to a minimum of 50ms to ensure enough time is given for the HTTP request to be processed

3.1.1. API Documentation

The ReST API documentation can be accessed by browsing to an instance of the Context Store Rest snap-in: http://<IP_ADDRESS>/services/CSRest/. This online documentation is only available through HTTP on either Firefox or Chrome browsers; IE is not supported. A copy of the full API documentation is provided in the appendix section of this document.

3.1.2. Case Sensitive

REST is case sensitive, so due care should be taken in creating the JSON queries. 'contextId' is not the same as 'ContextId' or 'contextID'. The following table contains the correct camel case for each of the parameters in the Context Store commands.

Parameters Case Sensitive
<ul style="list-style-type: none"> • lease • contextId

- groupId
- tenantId
- alias
- data
- persistToEDM
- persistTo
- rid
- touchpoint

3.1.3. REST Clients

There are many third party clients available for working with REST APIs. RESTClient for Firefox and Postman for Chrome, are two such options. The Context Store SDK distribution also bundles sample REST clients that can be used for testing purposes.

3.1.4. Sample API Requests

The following are example requests to get started (please refer to appendix for full API documentation). These requests are accessed through HTTP or, when security is enabled, HTTPS.

<IP_ADDRESS> - in a single-node Context Store deployment, the IP used in requests should be the security module IP address of the node itself, in a clustered deployment, the cluster's configured IP address must be used.

To create a context entry, use the following format:

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/`

HTTP: POST

Header: Content-Type application/json

JSON Body: `{"contextId":"mycontextId","data": {"mykey":"myvalue"}}`

Expected Response: 200 OK

To retrieve this context:

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/mycontextId`

HTTP: GET

Header: Content-Type application/json

Expected Response: 200 OK

JSON: `{"data": {"mykey":"myvalue"}}`

To update this contexts data:

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/mycontextId`

RESTful Interface

HTTP: PUT

Header: Content-Type application/json

JSON Body: {"data": {"mynewkey": "mynewvalue"}}

Expected Response: 200 OK

To delete this contexts data:

URL: http://<IP_ADDRESS>/services/CSRest/cs/contexts/mycontextId

HTTP: DELETE

Header: Content-Type application/json

Expected Response: 200 OK

3.2. Audit Trail Feature (since CS 3.1)

3.2.1. Overview

The Audit Trail will allow a customer to do analysis of any given Context and the Context Store operations which have been executed on this object during its life cycle. The Audit Trail shows a history of when, and by which touchpoint, the Context was created, modified or retrieved. The operationIds which are logged provide detailed information about the action; for example it is possible to identify whether the object was addressed by contextId or aliasId, or if was created using a standard create request or an upsert request. For a full list of applicable operationIds and the operation they represent see the table of [Operation Ids](#) in section 3.2.2.

Context Store will work as usual however the existing APIs will accept a new parameter touchpoint to describe the touchpoint. This parameter will be an alpha/numeric ID.

For example touchpoint=Desktop1 or touchpoint=Phone5 or touchpoint=Email or touchpoint=IM or touchpoint=Twitter. The new parameter should be a well formed valid string.

An example of the modified Get Context Request will be as follows:

```
https://<IP_ADDRESS>/services/CSRest/cs/contexts/{contextId}?touchpoint={touchpoint}
```

N.B: The existing functionality of the Context Store will stay intact.

If a well formed string is not provided as the touchpoint parameter then a 400 status will be returned. If no touchpoint parameter is provided, then a default value of "undefined" will be used for touchpoint identifier when the interaction is recorded in service logs and added to the audit trail of the Context.

For usage information, see the CSRest API in section [3.4](#).

3.2.2. Audit Entry

Each request will append the following information to the audit trail of the context.

Timestamp	A long value in milliseconds
Touchpoint	A string value defining the touchpoint
OperationId	A string value defining the operation performed
Version	An integer value representing the version of the Context

Timestamp

This value is the Unix Time Stamp at which the context was interacted with.

Touchpoint

This is a string value defining the application which interacted with the Context. If no value or a blank value was provided as a touchpoint parameter, then the default value of "undefined" will be used.

Version Id

In CS 3.1, the version of the Context will increase with each interaction with Context – create, read, update and delete portion of Context. Note, the version Id will not be incremented with the invocation of 'Get Audit Data' or 'Get Group' operations.

Operation Ids

The following table lists the codes for operation ids and what each code represents. These operation Id codes are predefined and cannot be modified by users.

Operation	Id
Create Context without contextId	CR_C_N
Create Context with contextId	CR_C_C
Create Context with aliasId	CR_C_A
Create Context without contextId and with aliasId	CR_C_N_A
Read Context by contextId	RD_C_C
Read Context by aliasId	RD_C_A
Read Context data key/value by contextId	RD_K_C
Read Context data key/value by aliasId	RD_K_A
Deleted aliasId using aliasId	DT_A_A
Deleted key/value pair from data using contextId	DT_K_C
Deleted key/value pair from data using aliasId	DT_K_A
Updated value of key using contextId	UD_K_C
Updated value of key using aliasId	UD_K_A
Updated all data using contextId	UD_C_C
Updated all data using aliasId	UD_C_A
Updated alias list using contextId	UD_A_C
Updated alias list using aliasId	UD_A_A
Upsert Create Context with contextId	UP_C_C_CR
Upsert Create Context with aliasId	UP_C_A_CR
Upsert Updated Context using contextId	UP_C_C_UD
Upsert Updated Context using aliasId	UP_C_A_UD

3.2.3. Enabling and setting the size of Audit Trail

This limit on stored entries is configurable through the **CS Audit: Event limit** attribute in EDP Element Manager. By default, this is set to 0 i.e. it is disabled - CS will not store any events/interactions for Context objects.

To enable the retention of an audit-trail for Contexts, the administrator must set this attribute to an integer in the range 1-50; Context Store will then record the most recent interactions as audit-trail entries in the Context object.

This feature is enabled/disabled for all Contexts in the cluster; it is not possible to apply different audit-trail settings to individual Contexts.

The CS administrator can reduce or increase the limit at any time or can turn off the Audit data recording all together by setting the limit back to 0. Changes made to this limit attribute will not take effect until the next interaction with the Context. If for example the limit was reduced from 10 to 5, the 5 oldest audit entries will be removed when the next audit entry is written to that Context.

3.2.4. Considerations for Capacity Planning

The Audit Trail records successful interactions with a Context and stores this information inside the Context object as an extra field. Storing a large number of audit entries will significantly increase the size of the Context object stored in the data-grid; therefore the number of entries configured for the system is an important factor for capacity planning and must be taken into consideration when sizing the data-grid.

Note: Enabling audit trail requires a substantial amount of space in the data-grid, particularly for longer audit trails. All benchmarking and performance testing for Context Store is executed using 2KB Context objects. To achieve the certified performance rate, the size of the base Context must be reduced when Audit Trails are being stored. If, for example, an audit trail of 10 entries is enabled and this is estimated to be 500 Bytes, then using 1.5 KB Contexts while have the same memory requirements at the standard, supported traffic rate stated for a cluster.

CS Audit: Event Limit (length of Audit Trail)	Reduction in Context size	Notes
21+	1 KB	NB: Additionally the throughput and/or lease must be reduced when storing very long audit trails
11 - 20	1 KB	Throughput and lease can remain at certified levels if Context object size is reduced by 1KB. If 2KB Context data (or larger) is required, throughput and/or lease must be reduced
1 - 10	0.5 KB	Throughput and lease remain at certified levels if Context object size is reduced by 0.5KB. If 2KB Context data (or larger) is required, throughput and/or lease must be reduced

3.3. Upsert Feature (since CS 3.1)

3.3.1. Overview

The new upsert (update or insert) method on the context store API call will invoke either a *create* or an *update* operation on a Context. Using the previous CS 3.0 ReST interface implementation, updating existing context objects requires two separate operations, the first a GET operation to check if the context already exists in the space followed by either a PUT operation to update it if it does exist or a POST operation to create the context if it does not exist. If the GET check is not executed, the PUT or POST operation may fail based on the existence or absence of that context object. This mechanism of checking before update or creation doubles the number of requests which must be processed by the ReST interface.

This new 'upsert' functionality would reduce the amount of ReST operations by handling the required functionality internally. The user executes a single operation (an Upsert request) without knowing the existence of the context object; Context Store will either update an existing context with new data or create a new context if it does not already exist. The combination of the two operations into a single request reduces the amount of total API calls and increases efficiency. Note the new method path `/upsert/` used for handling requests for the upsert functionality.

- `http://<IP_ADDRESS>/services/CSRest/cs/contexts/upsert/DemoContextId`
- `http://<IP_ADDRESS>/services/CSRest/cs/contexts/upsert?alias=DemoAliasId`

For usage information, see the CSRest API in section [3.4](#).

3.3.2. Upsert Parameters

Context metadata can be provided to upsert via parameters. Some of the properties are immutable. This means that they cannot be altered after the contexts creation.

- In Upsert create scenarios the provided properties are set with the newly created context.
- In Upsert update scenarios the properties in the request are compared against those in the existing context. If immutable properties do not match the existing context the upsert request is rejected. Mutable properties provided in the request are updated to the provided values.

Property	Parameter Key	Type	Modifiability	Description
Context ID	n/a	Path parameter	Immutable	Can be set for creates but cannot be modified by updates.
Routing ID	rid	Query parameter	Immutable	Can be set for creates but cannot be modified by updates.
Lease	lease	Query parameter	Mutable	Can be set to new contexts or updated on existing contexts.
Alias	alias	Query parameter	Immutable	aliasIds are mutable but this parameter is used only for querying against the space, aliasIds cannot be altered from the URL.

Alias	alias	Provided in Request Body	Mutable	aliasIds can be provided through the "alias" field in the body. These (if not already existing) will be added to the context up to the maximum allowed.
Group ID	groupId	Provided in Request Body	Immutable	Can be set for creates but cannot be modified by updates.
Tenant ID	tenantId	Provided in Request Body	Immutable	Can be set for creates but cannot be modified by updates.
Data	data	Provided in Request Body	Mutable	Cannot be blank. Data provided here will be applied to new contexts and updated onto existing contexts.
Persist to EDM	persistToEDM	Provided in Request Body	Immutable	EDM persistence can be set for creates but cannot be modified by updates.
Provisioned Context	persistTo	Provided in Request Body	Immutable	EDM provisioning can be set for creates but cannot be modified by updates.

Touchpoint Parameter

As with any requests to CSRest, a touchpoint may be specified as query parameter if the Audit feature is in use. The touchpoint provided will be logged in the Audit Trail and the Upsert operation is recorded.

Touchpoint	touchpoint	Query parameter	Not Applicable	Can be provided with query parameter to be used for audit purposes if Audit feature is enabled.
------------	------------	-----------------	----------------	---

3.4. CSRest API

3.4.1. Context Object Information

Field	Description
contextId Mandatory	<pre>"contextId": "xxxxx"</pre> <p>contextId is a unique case sensitive immutable identifier that can be supplied by the Customer or can be generated by the context Store. It can have a max length of 255 characters and can only support a sub set of the ASCII character set.</p> <p>Please refer to section contextId field for more details.</p>
data Mandatory	<pre>"data": {"key1", "value1" }</pre> <p>Key & Value pair(s) that contain the context information to store</p> <p>Please refer to the data field section for more details.</p>
groupId Optional	<pre>"groupId": "xxxxx"</pre> <p>The group identification to which this context belongs.</p> <p>Please refer to section groupId field section for more details.</p>
tenantId Optional	<pre>"tenantId": "xxxxx"</pre> <p>The tenant identification to which the context belongs.</p> <p>Please refer to the tenantId field section for more details.</p>
aliasId Optional	<pre>"aliasIds": ["aliasId1", "aliasId2"]</pre> <p>An aliasId is an alternative identifier which can be used to retrieve a Context. Multiple aliasIds may be associated with a single context</p> <p>Please refer to the aliasId field section for more details.</p>
routingId Optional	<pre>"routingId": "xxxxx"</pre> <p>The routing identifier is used in a Geo-redundant deployment to identify in which cluster the Context was created and thereby ensure uniqueness of contextIds and/or aliasIds in Context data replicated between the two clusters.</p>

	Please refer to the routingId field section for more details.
persistToEDM Optional	<pre>"persistToEDM": "true"</pre> <p>Boolean flag that overrides the Cluster Wide setting an operation level. Default value = False.</p> <p>Please refer to section persistToEDM field for more details.</p>
persistTo Optional	<pre>"persistTo": "CS_PROVISION"</pre> <p>Boolean flag that overrides the Cluster Wide setting an operation level. Default value = False.</p> <p>Please refer to section persistTo field for more details.</p>
versionId Automatic	Every Context object contains an automated versionId field which represents the number of times the Context has been read or updated since its creation. This versionId will increase with each interaction with Context – create, read, update and delete portion of Context. Note: The version Id will not be incremented with the invocation of 'Get Audit' or 'Get Group' requests.

contextId field

Field Name	contextId
Case Sensitive	Yes

The contextId is a unique identifier that can be supplied by the customer or can be generated by Context Store. The choice of contextId by the customer is important as it must be supported and identifiable and accessible by all clients and applications that interact with Context Store and are part of the customer solution. Examples of identifiers that can be used are UCID or Web Session ID.

Properties	Description
Unique	The contextId must be unique with respect to the Context Store Cluster. A customer cannot use the same contextId to represent two separate Contexts at the same time. If the lease on a Context expires, the contextId can be reused. Context Store will generate a unique identifier if no contextId is specified as part of the post.
Case Sensitive	contextId is case sensitive and case is always preserved. For example, contextId1 and CONTEXTID1 are not equivalent and are considered to be two distinct ids.
Immutable	The contextId cannot be changed once the context has been created.
Max Length	The contextId supports a max length of 255 characters.
Supported Characters	<ul style="list-style-type: none"> Context Store does not accept blank or null keys as Context data. The contextId cannot contain any spaces (includes leading and trailing

	<p>spaces).</p> <ul style="list-style-type: none"> Any attempt to POST or PUT a contextId containing spaces will receive a '400 Bad Request' error and the status message will be "contextId has invalid characters". <p>☐ Only the ASCII characters listed here are permitted and the associated rules must be followed.</p> <ul style="list-style-type: none"> Uppercase and lowercase English letters (A–Z, a–z) (ASCII: 65–90, 97–122) Digits 0 to 9 (ASCII: 48–57) Characters - _ ~ (ASCII: 45, 95, 126). <ul style="list-style-type: none"> RULE: Provided that they are not the first or last character. Character. (dot, period, full stop) (ASCII: 46) <ul style="list-style-type: none"> RULE: Provided that it is not the first or last character, and provided also that it does not appear two or more times consecutively. null object may not be passed into the context store, i.e., "name":null <p>☐ Note: An asterisk (*) is a special character that is reserved for use in key name where it indicates a sensitive data field which should not be logged.</p> <ul style="list-style-type: none"> E.g. {"*key1*":"sensitiveField", "key2":"regularField"}
--	---

Special Considerations

UUI

UUI can be used to associate a contextId with a call to allow other applications retrieve relevant information from Context Store. Customers must ensure that their choice of contextId length does not exceed the maximum available length that UUI supports or the ID will be truncated. This length is typically 32 characters, but may change depending on customer’s usage of UUI.

For full details on UUI please refer to Avaya Aura Elite documentation

OneX

The supported character set enforced by contextId is chosen to support the largest number of customer interactions, but some potential clients of Context Store might have additional restrictions outside that of Context Store. OneX does not support the use of ~ and is not a suitable character for the contextId or for the groupId, or Key if OneX is part of your Solution that integrates with Context Store.

data field

Field Name	data
Case Sensitive	Yes

The data variable is a comma separated list of Key & Value pair(s) that contains the client specified context. The client cannot add a context without specifying at least one Key & Value pair.

The Context should not contain contextual data greater than 500K. The size of data stored in Context Store can have performance implications please refer to [Capacity Planning](#) section for details on how to configure your system.

Key

The Key is a unique identifier that is supplied by the Client and is used to retrieve a value within a context. The choice of key by the client is important as in conjunction with the contextId; it can be used to retrieve the value using the GET command. Therefore, if it is required to be retrieved this way, it must be supported and accessible by all clients and applications that interact with Context Store and are part of the customer solution.

Properties	Description
Unique	The Key is unique to a context. If a client adds a context with multiple duplicate keys the value associated with the last key will be stored.
Case Sensitive	Key is case sensitive and case is always preserved i.e. KEY1 and key1 are not equivalent and are considered to be two distinct keys
Immutable	The Key is not changeable but the value associated with a key can be copied to a new Key name
Max Length	The Key cannot be blank or null and supports a max length 255 characters.
Supported Characters	The Key supports the same character sets as the contextId.

Special Considerations

As the key is an index that can be used by clients, in conjunction with the contextId, to retrieve Context, it must follow the same considerations used when deciding on the contextId

Value

A value cannot exist without the equivalent key.

Properties	Description
Unique	Values can be duplicated as long as the associated Key is unique.
Max Length	There is no MAX length to a value, but it is recommended that the overall Context size does not exceed 500K for performance reasons.
Supported Characters	Supports UTF-8

persistToEDM field

Field Name	persistToEDM
Case Sensitive	Yes

Context Store supports the ability to persist add and update operations to an external data base. If the EDM feature is configured and enabled at a cluster level, Customers have the ability to specify that a context will be persisted to the External Database. A context can only be marked for persistence when it is created so the value can only be specified on a create, i.e. POST, operation. All updates on this marked context will be persisted.

Upsert: Persistence can be specified when creating Contexts using Upsert operations but note that if the Context already exists, the persistence status of existing Context will not be modified. For more information about Upsert operations see section 3.3 Upsert Feature (since CS 3.1).

Persisting data to an external database has performance implications, please refer to section 10.6 External DataMart for more information.

Properties	Description
Value	<ul style="list-style-type: none"> "persistToEDM": "true" "persistToEDM": "false"
Immutable	This value can only be set when the Context is created and cannot be changed through the lifecycle of the Context.

persistTo field

Field Name	persistTo
Case Sensitive	Yes

Context Store supports the ability to persist Contexts to an external data base to be provisioned automatically in the data-grid if the data-grid is restarted. If the EDM feature is configured and enabled at a cluster level, Customers have the ability to specify that a context will be persisted to the provisioning table in the External Database. A context can only be marked for persistence when it is created so the value can the value can only be specified on a create, i.e. POST, operation. All updates on this marked context will be persisted.

Upsert: Persistence can be specified when creating Contexts using Upsert operations but note that if the Context already exists, the persistence status of existing Context will not be modified. For more information about Upsert operations see section 3.3 Upsert Feature (since CS 3.1).

Persisting data to an external database has performance implications, please refer to section 10.6 External DataMart for more information.

Properties	Description
Value	<ul style="list-style-type: none"> "persistTo": "CS_PROVISION"
Immutable	This value can only be set when the Context is created and cannot be changed through the lifecycle of the Context.

groupId field

Field Name	groupId
Case Sensitive	Yes

A client has the option to associate several contexts together using the groupId. This groupId can be used with a GET command to return the metadata for all contexts that have a matching groupId. There is no limit

to the number of contexts that can be associated with a GroupId, but for performance reasons the GET will only return a maximum of 1000 associated contextIds.

Properties

A context can only be assigned a groupId when it is created so the value can only be specified on a POST.

The choice of groupId by the client is important as in conjunction with the contextId or an aliasId, it can be used to retrieve the value using the GET command (GET group request followed by GET context request) and hence must be supported and accessible to all clients and applications that interact with Context Store and are part of the customer solution.

Properties	Description
Case Sensitive	groupId is case sensitive and case is always preserved i.e. groupId1 and GROUPID1 are not equivalent and are considered to be two distinct ids
Immutable	The groupId cannot be changed using a PUT command
Max Length	The groupId cannot be blank or null and supports a max length of 255 characters.
Supported Characters	The groupId supports the same character sets as the contextId.

Special Considerations

As the groupId is an index that can be used by clients, in conjunction with the contextId, to retrieve Context it should abide by the same considerations used when deciding on the contextId

tenantId field

Field Name	tenantId
Case Sensitive	Yes

This parameter provides additional filtering for Context Store Notifications. A client can subscribe for notifications only for contexts which are created with a specific tenantId. This allows for application developers to create a one to one relationship between a producer application feeding into the Context Store and a consumer application feeding off the Context Store.

The tenantId has the same considerations as the groupId.

Properties	Description
Case Sensitive	tenantId is case sensitive and case is always preserved i.e. tenantId1 and TENANTID1 are not equivalent and are considered to be two distinct ids
Immutable	The tenantId cannot be changed using a PUT command
Max Length	The tenantId cannot be blank or null and supports a max length of 255 characters.
Supported Characters	The tenantId supports the same character sets as the contextId.

aliasId field

Field Name	aliasId
Case Sensitive	Yes

The Context object may contain a list of aliasIds, each of these alternative identifiers can be used to retrieve the same Context object from the data cache.

A new query parameter, **alias**, is added to the CSRest API to support this new feature functionality. See section [Query parameter: alias](#) for usage information.

The aliasId has the same considerations as the contextId.

Properties	Description
Case Sensitive	aliasId is case sensitive and case is always preserved i.e. aliasId1 and ALIASID1 are not equivalent and are considered to be two distinct ids The aliasId query parameter, alias , is also case sensitive.
Mutable	The list of aliasIds associated with a Context object can be modified replaced using DELETE and PUT commands
Max Length	The aliasId cannot be blank or null and supports a max length of 255 characters.
Supported Characters	The aliasId supports the same character sets as the contextId.

routingId field

Field Name	routingId
Case Sensitive	Yes

The rid parameter is used in a Geo-redundant deployment to route requests by the Load Balancer. A corresponding field routingId has been added to the Context Store POJO. The routingId is immutable and is only set when a context is created. The routingId is assigned the value of the rid parameter if supplied or the default value "0" otherwise. See the section [Query parameter: rid](#) for rid usage information.

The routingId has the same considerations as the contextId.

Properties	Description
Case Sensitive	routingId is case sensitive and case is always preserved i.e. routingId1 and ROUTINGID1 are not equivalent and are considered to be two distinct ids The routingId query parameter, rid , is also case sensitive
Immutable	The routingId cannot be changed using a PUT command
Max Length	The routingId cannot be blank or null and supports a max length of 255 characters.

Supported Characters	The routingId supports the same character sets as the contextId.
----------------------	--

3.4.2. Query Parameters

Query parameter: lease

This parameter specifies the maximum time (in milliseconds) that data is to be stored in the data cache. The lease parameter is common to POST and PUT and specifies the duration the client wishes the context to exist in Context Store. The duration a Context is stored in Context Store has performance implications. For more information, see the [Capacity Planning](#) section.

This optional variable allows Context Store clients to override the Cluster wide setting at an operation level, that is, this Context can exist for longer or shorter than the system default setting. If the client does not specify the setting in POST operation, the Cluster wide setting will be used. If no lease is specified on PUT operation, the context maintains its existing lease time.

Properties	Description
Optional	If no lease is provided on POST the default setting for the cluster will be used. If no lease is provided on PUT the existing lease time will continue to count down. The lease query parameter for a request can be provided but left blank. This sets the request to use the default configured lease
Mutable	The lease of a Context can be changed using a PUT operation
Max Length	The valid lease must be in the range 1 to 2147483647 . If client wants the context to exist indefinitely in Context Store then the lease can be set to either -1 , 0 , or infinite †(Case Insensitive).
Supported Characters	Numeric or max (Case Insensitive)

Query parameter: touchpoint

With the introduction of the new Audit Trail feature, All requests on the CSRest interface now accepts an additional new parameter “touchpoint” to describe the touchpoint used to invoke the ReST url. This parameter will be an alpha/numeric ID. For example touchpoint=Desktop1 or touchpoint=Phone5 or touchpoint=Email or touchpoint=IM or touchpoint=Twitter.

Properties	Description
Optional	If no touchpoint parameter is provided, then a default value of "undefined" will be used for touch-point identifier when the interaction is recorded in service logs and added to the audit trail of the Context.

† Infinite here indicates that maximum allowable time a context can exist in context store without the Context Store being restarted.

Immutable	The touchpoint provided is written in the Audit trail, once written it cannot be changed. Any valid touchpoint string can be submitted as a query parameter with each request submitted via the CSRest interface.
Max Length	The touchpoint can be blank. If provided and not blank then it must be in the range of 1 to 255 characters.
Supported Characters	The touchpoint supports the same character sets as the contextId.

Query parameter: rid

Due to the new Geo service preservation feature for CS 3.1 which utilizes two active/active Context Store clusters the sole unique identifier contextId previously employed for contexts can no longer guarantee that two simultaneous create context requests to each CS cluster in a geo system with the same contextId would not conflict when replication between each cluster occurs. Therefore an optional rid parameter has been added to the request and a routingId field has been added to the context POJO. In a Geo-redundant environment, the Load Balancer must route the traffic based on this rid parameter, after an object is created on one cluster it will then be replicated to the other one but the routingId field remains the same.

In all other operations, the rid parameter is used to identify the routingId to be used, in conjunction with contextId or aliasId, to identify the correct context object to retrieve, modify or delete in a geo redundant deployment.

Properties	Description
Optional	If no rid is provided, the request will be routed to the default cluster
Immutable	The routingId field of a context object cannot be changed using PUT
Max Length	The rid cannot be blank or null and supports a max length 255 characters.
Supported Characters	The rid supports the same character sets as the contextId.

Query parameter: alias

The Context object may contain a list of aliasIds; each of these alternative identifiers can be used to retrieve the same Context object from the data cache. The **alias** query parameter has been added to the CSRest API to support the aliasId feature functionality.

NB: In earlier release of Context Store, all requests to the CSRest interface required that the contextId be specified in the URL path for all RUD operations

E.g. `https://<IP_ADDRESS>/services/CSRest/cs/contexts/{contextId}`

The base URL for aliasId based requests must not include the contextId in the path and should instead use this alias query parameter

E.g. `https://<IP_ADDRESS>/services/CSRest/cs/contexts/?alias={aliasId}`

The POST request (i.e. create Context) accepts multiple **alias** parameters; up to a maximum of three. Each of these aliasIds will be associated with the Context object that is created.

All other requests by aliasId (i.e. read, update, delete, upsert) accept a single **alias** parameter only. Only the first **alias** parameter is acted upon, all subsequent **alias** parameters will be ignored.

E.g.1 POST request:

```
https://<IP_ADDRESS>/services/CSRest/cs/contexts/?alias=aliasId1&alias=aliasId2&alias=aliasId3
```

E.g.2 GET request: `https://<IP_ADDRESS>/services/CSRest/cs/contexts/?alias=aliasId`

Some additional CSRest requests have been created for managing the mutable set of aliasIds associated with a Context. These requests will use new `/aliases/` path in the CSRest API for aliasId management.

E.g. `https://<IP_ADDRESS>/services/CSRest/cs/contexts/aliases/`

This aliasing capability will impact the performance of Context Store, in particular latency, due the increased number of lookups and data comparison required to find the correct object in the data cache.

- To lessen the performance impact of this feature, a limit of three aliasIds per Context is enforced.
- If an attempt is made to add a 4th aliasId, the request will fail.

Properties	Description
Optional	Used in CRUD operations as an alternative to contextId
Mutable	The list of aliasIds associated with a Context object can be modified replaced using DELETE and PUT commands
Max Length	The aliasId field cannot be blank or null and supports a max length 255 characters.
Supported Characters	The aliasId field supports the same character sets as the contextId.

3.4.3. POST – Adding a Context

The interface allows clients to store context information using multiple key/value pairs. Moreover, clients will be able to define the unique ID to be used to retrieve that context information later

POST Context Information	
HTTP Method	POST
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/{id}
Variables	{lease}: see section Query parameter: lease {rid}: see section Query parameter: rid {alias}: see section Query parameter: alias {touchpoint}: see section Query parameter: touchpoint

JSON Structure Examples

Example 1: POST request with no optional parameters

```

=====
POST / HTTP/1.1
Host: <IP_ADDRESS>services/CSRest/cs/contexts/
Content Type: application/json
Content:
{"contextId":"ABC","data":{"key1_name":"value1_data","key2_name":"value2_data"}}
-----
Response Code: 200 OK

```

Example 2: POST request with all optional parameters

```

=====
POST / HTTP/1.1
Host: <IP_ADDRESS>services/CSRest/cs/contexts/?alias=X&lease=360&rid=X&touchpoint=Z
Content Type: application/json
Content:
{"contextId":"XXYYZZ","groupId":"AABB","persistToEDM":"true","tenantId":"DDEE",
"data":{"key1_name":" value1_data","key2_name":"value2_data"}}
-----
Response Code: 200 OK

```

3.4.4. UPSERT – Upserting a Context

The interface allows clients to create or update a whole context record associated with a specified contextId.

PUT Context Information	
HTTP Method	PUT
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/upsert/{id}
Variables	{lease}: see section Query parameter: lease {rid}: see section Query parameter: rid {touchpoint}: see section Query parameter: touchpoint

JSON Structure Example

PUT / HTTP/1.1

http://<IP_ADDRESS>/services/CSRest/cs/contexts/upsert/{id}/

Content Type: application/json; charset=UTF-8

Content: {"data":{"key1_name":"value1_data","key2_name":"value2_data"}}

Response Code: 200 OK

3.4.5. UPSERT – Upserting a Context which has optional identifier fields

The interface allows clients to create or update a whole context record associated with a specified contextId when has associated groupId and/or tenantId.

PUT Context Information	
HTTP Method	PUT
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/upsert/{id}
Variables	{lease}: see section Query parameter: lease {rid}: see section Query parameter: rid {touchpoint}: see section Query parameter: touchpoint

JSON Structure Example

PUT / HTTP/1.1

http://<IP_ADDRESS>/services/CSRest/cs/contexts/upsert/{id}/

Content Type: application/json; charset=UTF-8

Content: {"groupId":"exampleGroupId","tenantId":"exampleTenantId","data":{"key1_name":"value1_data","key2_name":"value2_data"}}

Response Code: 200 OK

3.4.6. UPSERT – Upserting a Context by aliasId

The interface allows clients to create or update a whole context record associated with a specified aliasId.

PUT Context Information	
HTTP Method	PUT
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/upsert/?alias={aliasId}
Variables	{lease}: see section Query parameter: lease {rid}: see section Query parameter: rid {touchpoint}: see section Query parameter: touchpoint

JSON Structure Example

PUT / HTTP/1.1

http://<IP_ADDRESS>/services/CSRest/cs/contexts/upsert/?alias={aliasId}

Content Type: application/json; charset=UTF-8

Content: {"data":{"key1_name":" value1_data","key2_name":"value2_data"}}

Response Code: 200 OK

3.4.7. UPSERT – Upserting a Context with aliasIds

The interface allows clients to create or update a whole context record, including aliasIds, associated with a specified contextId.

PUT Context Information	
HTTP Method	PUT
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/upsert/{id}
Variables	{lease}: see section Query parameter: lease {rid}: see section Query parameter: rid {touchpoint}: see section Query parameter: touchpoint

JSON Structure Example

PUT / HTTP/1.1

http://<IP_ADDRESS>/services/CSRest/cs/contexts/upsert/{id}/

Content Type: application/json; charset=UTF-8

Content: {"alias":["aliasId1","aliasId2"],"data":{"key1_name":" value1_data"}}

Response Code: 200 OK

3.4.8. UPSERT – Upserting a Context with aliasIds by aliasId

The interface allows clients to create or update a whole context record, including aliasIds, associated with a specified aliasId.

PUT Context Information	
HTTP Method	PUT
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/upsert/?alias={aliasId}
Variables	{lease}: see section Query parameter: lease {rid}: see section Query parameter: rid {touchpoint}: see section Query parameter: touchpoint

JSON Structure Example

PUT / HTTP/1.1

http://<IP_ADDRESS>/services/CSRest/cs/contexts/upsert/{id}/

Content Type: application/json; charset=UTF-8

Content: {"alias":["aliasId1","aliasId2"],"data":{"key1_name":" value1_data"}}

Response Code: 200 OK

3.4.9. GET – Getting a Context

The interface allows clients to retrieve all data associated with a specified contextId.

GET Context Information	
HTTP Method	GET
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/{id}
Variables	{rid}: see section Query parameter: rid {touchpoint}: see section Query parameter: touchpoint

JSON Structure

Example

GET / HTTP/1.1

http://<IP_ADDRESS>/services/CSRest/cs/contexts/{id}?rid={r1}&touchpoint={abc}

Response Code: 200 OK

HTTP Response: {"data":{"key1_name":"value1_data","key2_name":"value2_data"}}

3.4.10. GET – Getting a Context value

The interface allows clients to retrieve a single value from a context record.

GET Context Information	
HTTP Method	GET
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/{id}/keys/{key}
Variables	{rid}: see section Query parameter: rid {touchpoint}: see section Query parameter: touchpoint

JSON Structure

Example

```
GET / HTTP/1.1
```

```
http://<IP_ADDRESS>/services/CSRest/cs/contexts/{id}/keys/{key}
```

Response Code: 200 OK

HTTP Response: value1_data

3.4.11. GET – Getting a Context by aliasId

The interface allows clients to retrieve all data associated with a specified aliasId.

GET Context Information	
HTTP Method	GET
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/?alias={aliasId}
Variables	{alias}: see section Query parameter: alias {rid}: see section Query parameter: rid {touchpoint}: see section Query parameter: touchpoint

JSON Structure

Example

```
GET / HTTP/1.1
```

```
http://<IP_ADDRESS>/services/CSRest/cs/contexts/?alias={aliasId}&rid={r1}
```

Response Code: 200 OK

HTTP Response: {"data":{"key1_name":"value1_data","key2_name":"value2_data"}}

3.4.12. GET – Getting a Context value by aliasId

The interface allows clients to retrieve a single value from a context record associated with a specified aliasId.

GET Context Information	
HTTP Method	GET
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/keys/{key}/?alias={aliasId}
Variables	{alias}: see section Query parameter: alias {rid}: see section Query parameter: rid {touchpoint}: see section Query parameter: touchpoint

JSON Structure

Example

GET / HTTP/1.1

http://<IP_ADDRESS>/services/CSRest/cs/contexts/keys/{key}/?alias={aliasId}

Response Code: 200 OK

HTTP Response: value1_data

3.4.13. GET – Getting a Context's Audit Data

The interface allows clients to retrieve the Audit Data from a context record associated with a specified contextId.

GET Context Information	
HTTP Method	GET
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/audit/{id}
Variables	{rid}: see section Query parameter: rid

JSON Structure

Example

GET / HTTP/1.1

http://<IP_ADDRESS>/services/CSRest/cs/contexts/audit/{id}

Response Code: 200 OK

HTTP Response:

```
{"audit":[{"timeStamp":"1433320548693","touchPoint":"Postman","version":"1","operationId":"CR_C_C"}]}
```

3.4.14. GET – Getting a Context’s Audit Data by aliasId

The interface allows clients to retrieve the Audit Data from a context record associated with a specified aliasId.

GET Context Information	
HTTP Method	GET
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/audit/?alias={aliasId}
Variables	{alias}: see section Query parameter: alias {rid}: see section Query parameter: rid

JSON Structure

Example

```
GET / HTTP/1.1
```

```
http://<IP_ADDRESS>/services/CSRest/cs/contexts/audit/?alias={aliasId}
```

Response Code: 200 OK

HTTP Response:

```
{"audit":[{"timeStamp":"1433320548693","touchPoint":"Postman","version":"1","operationId":"CR_C_C"}]}
```

3.4.15. GET – Getting Contexts’ Metadata by groupId

The interface allows clients to retrieve the meta-data of Contexts associated with a specified groupId.

GET Context Information	
HTTP Method	GET
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/groups/{id}/
Variables	{rid}: see section Query parameter: rid {touchpoint}: see section Query parameter: touchpoint

JSON Structure

Example

```
GET / HTTP/1.1
```

```
http://<IP_ADDRESS>/services/CSRest/cs/contexts/group/{id}/
```

Response Code: 200 OK

HTTP Response: {"data":{"key1_name":"value1_data","key2_name":"value2_data"}}

3.4.16. UPDATE – Updating a Context

The interface allows clients to update a whole context record associated with a specified contextId.

PUT Context Information	
HTTP Method	PUT
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/{id}
Variables	{lease}: see section Query parameter: lease {rid}: see section Query parameter: rid {touchpoint}: see section Query parameter: touchpoint

JSON Structure Example

PUT / HTTP/1.1

http://<IP_ADDRESS>/services/CSRest/cs/contexts/{id}/?lease={9999}&rid={r1}

Content Type: application/json; charset=UTF-8

Content: {"data":{"key1_name":" value1_data","key2_name":"value2_data"}}

Response Code: 200 OK

3.4.17. UPDATE – Updating a Context value

The interface allows clients to update a single value in a context record associated with a specified contextId.

PUT Context Information	
HTTP Method	PUT
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/{id}/keys/{key}
Variables	{lease}: see section Query parameter: lease {rid}: see section Query parameter: rid {touchpoint}: see section Query parameter: touchpoint

JSON Structure Example

PUT / HTTP/1.1

http://<IP_ADDRESS>/services/CSRest/cs/contexts/{id}/keys/{key}

Content Type: application/json; charset=UTF-8

Content: value1_data

Response Code: 200 OK

3.4.18. UPDATE – Updating a Context by aliasId

The interface allows clients to update a whole context record associated with a specified aliasId.

PUT Context Information	
HTTP Method	PUT
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/?alias={aliasId}
Variables	{alias}: see section Query parameter: alias {lease}: see section Query parameter: lease {rid}: see section Query parameter: rid {touchpoint}: see section Query parameter: touchpoint

JSON Structure Example

PUT / HTTP/1.1

http://<IP_ADDRESS>/services/CSRest/cs/contexts/?alias={aliasId}&lease={1}&rid={1}

Content Type: application/json; charset=UTF-8

Content: {"data":{"key1_name":" value1_data","key2_name":"value2_data"}}

Response Code: 200 OK

3.4.19. UPDATE – Updating a Context value by aliasId

The interface allows clients to update a single value in a context record associated with a specified aliasId.

PUT Context Information	
HTTP Method	PUT
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/keys/{key}/?alias={aliasId}
Variables	{alias}: see section Query parameter: alias {lease}: see section Query parameter: lease {rid}: see section Query parameter: rid {touchpoint}: see section Query parameter: touchpoint

JSON Structure Example

PUT / HTTP/1.1

http://<IP_ADDRESS>/services/CSRest/cs/contexts/keys/{key}/?alias={aliasId}

Content Type: application/json; charset=UTF-8

Content: value1_data

Response Code: 200 OK

3.4.20. UPDATE – Adding an aliasId by contextId

The interface allows clients to add to the list of aliasIds associated with a context record using the contextId.

PUT Context Information	
HTTP Method	PUT
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/{id}/aliases
Variables	{lease}: see section Query parameter: lease {rid}: see section Query parameter: rid {touchpoint}: see section Query parameter: touchpoint

JSON Structure Example

PUT / HTTP/1.1

http://<IP_ADDRESS>/services/CSRest/cs/contexts/{id}/aliases

Content Type: application/json; charset=UTF-8

Content: ["aliasId1","aliasId2","aliasId3"]

Response Code: 200 OK

3.4.21. UPDATE – Adding an aliasId by aliasId

The interface allows clients to add to the list of aliasIds associated with a context record using existing aliasId.

PUT Context Information	
HTTP Method	PUT
Context Information	HTTP Response Body – JSON Structure
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/aliases?alias={aliasId}
Variables	{alias}: see section Query parameter: alias {lease}: see section Query parameter: lease {rid}: see section Query parameter: rid {touchpoint}: see section Query parameter: touchpoint

JSON Structure Example

PUT / HTTP/1.1

```
http://<IP_ADDRESS>/services/CSRest/cs/contexts/aliases?alias={aliasId1}
Content Type: application/json; charset=UTF-8
Content: [aliasId2", "aliasId3"]
```

Response Code: 200 OK

3.4.22. DELETE – Deleting a Context

The interface allows clients to delete a whole context record.

DELETE Context Information	
Http Method	DELETE
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/{id}
Variables	{rid}: see section Query parameter: rid

JSON Structure Example

```
DELETE / HTTP/1.1
http://<IP_ADDRESS>/services/CSRest/cs/contexts/{id}?rid={r1}
```

Response Code: 200 OK

3.4.23. DELETE – Deleting a Context value

The interface allows clients to delete a single value from a context record.

DELETE Context Information	
Http Method	DELETE
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/{id}/keys/{key}
Variables	{rid}: see section Query parameter: rid {touchpoint}: see section Query parameter: touchpoint

JSON Structure

Example

```
DELETE / HTTP/1.1
http://<IP_ADDRESS>/services/CSRest/cs/contexts/{id}/keys/{key}
```

Response Code: 200 OK

3.4.24. DELETE – Deleting a Context by aliasId

The interface allows clients to delete a whole context record.

DELETE Context Information	
Http Method	DELETE
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/?alias={aliasId}
Variables	{alias}: see section Query parameter: alias {rid}: see section Query parameter: rid

JSON Structure Example

DELETE / HTTP/1.1

http://<IP_ADDRESS>/services/CSRest/cs/contexts/?alias={aliasId1}&rid={r1}

Response Code: 200 OK

3.4.25. DELETE – Deleting a Context value by aliasId

The interface allows clients to delete a single value from a context record.

DELETE Context Information	
Http Method	DELETE
URI	<IP_ADDRESS>/services/CSRest/cs/contexts/keys/{key}/?alias={aliasId}
Variables	{alias}: see section Query parameter: alias {rid}: see section Query parameter: rid {touchpoint}: see section Query parameter: touchpoint

JSON Structure Example

DELETE / HTTP/1.1

http://<IP_ADDRESS>/services/CSRest/cs/contexts/keys/{key}/?alias={aliasId}

Response Code: 200 OK

3.4.26. DELETE – Deleting an aliasId

The interface allows clients to delete an existing aliasId from a context record.

DELETE aliasId from Context	
Http Method	DELETE
URI	<IP_ADDRESS>/services/CSRest/cs/aliases/?alias={aliasId}
Variables	{rid}: see section Query parameter: rid

	{touchpoint}: see section Query parameter: touchpoint
--	---

JSON Structure Example

DELETE / HTTP/1.1

http://<IP_ADDRESS>/services/CSRest/cs/aliases/?alias={aliasId}&rid={r1}

Response Code: 200 OK

4. Context Store SDK Tutorial

Context Store SDK provides two tutorials that developers can use as a template for creating their own applications and integrations. These example applications are provided solely as a guide to developing and do not contain the necessary hardening and fault tolerance to be considered as production applications.

4.1. Java Client

The Context Store SDK Java Swing GUI Client allows a user to verify that a Context Store instance has been set up correctly. All the operations that the Context Store supports can be run with the client provided here. The client is a standalone jar which can be opened by double clicking it.

4.1.1. Overview

If building with maven add the SDK dependency to the Java apps Maven POM file.

```
<dependency>
    <groupId>com.avaya.ingensg.cs</groupId>
    <artifactId>cs-sdk-api</artifactId>
    <version>version</version>
</dependency>
```

The following system properties need to be set before instantiating the Context Store Connection.

```
File keyStoreFile = new File("location of jks file");
System.setProperty("javax.net.ssl.keyStore", keyStoreFile.toString());
System.setProperty("javax.net.ssl.keyStorePassword", "keystorepassword");
System.setProperty("javax.net.ssl.trustStore",
System.getProperty("javax.net.ssl.keyStore"));
```

For example, to save a context to the Context Store, instantiate the Context Store Connection with following code:

```
ContextStore contextStore;
try {
contextStore = ContextStoreServiceFactory.getContextStoreWebService(HOST);
} catch (Exception e) {
    System.out.println("Error creating the context store instance "
+ e.toString());
    System.out.println("Exiting the program");
    System.exit(1);
}
List<String> aliasIds = new ArrayList<String>(Arrays.asList("alias1"));
String routingId = "0";
String touchpoint = "SDKClient";
ContextPojo context = new ContextPojo();
context.setcontextId("samplecontextId");
```

```
private Map<String, Object> data = new HashMap<String, Object>();
data.put("key1", "value1");
context.setData(data);

ServiceResponse response = contextStore.postContext(context, new
Long(12000), null, aliasIds, routingId, touchpoint);
    if (response.getStatus().equals(ContextStoreStatus.OK)) {
        System.out.println(String.format("postContext() - Added context data,
ID %s, data: %s", context.getContextId(), context.getData().toString()));
    } else {
        System.out.println("postContext() - Invalid request");
    }
}
```

4.1.2. SDK operations

The available operations and responses are as follows.

```
ServiceResponse response = contextStore.postContext(context, lease, sessionId);
```

(Post a Context object to Context Store. The Context object will expire according to the value supplied in the lease. A Response object is returned)

```
ServiceResponse response = contextStore.postContext(context, lease, sessionId,
aliasId(s), rid);
```

(Post a Context object to Context Store. The Context object will expire according to the value supplied in the lease. AliasId(s) will be given to the Context as well as a routingId. A Response object is returned.)

```
ServiceResponse response = contextStore.postContext(context, lease, sessionId,
aliasId(s), rid);
```

(Post a Context object to Context Store. The Context object will expire according to the value supplied in the lease. AliasId(s) will be given to the Context as well as a routingId. A Response object is returned. The touchpoint will be added to the Context's audit trail.)

```
PostWithNoContextIdResponse response = (PostWithNoContextIdResponse)
contextStore.postContextWithoutContextId(context, lease, sessionId);
```

(Posts Context object which does not have a contextId to the Context Store. The Context will expire after the lease period. The response contains the assigned contextId.)

```
PostWithNoContextIdResponse response = (PostWithNoContextIdResponse)
contextStore.postContextWithoutContextId(context, lease, sessionId, aliasId(s), rid)
```

(Posts Context object with no contextId to the Context Store. Accepts up to three aliasIds. The Context will expire after the lease period. The response contains the assigned contextId.)

```
PostWithNoContextIdResponse response = (PostWithNoContextIdResponse)
contextStore.postContextWithoutContextId(context, lease, sessionId, aliasId(s), rid,
touchpoint)
```

(Posts Context object with no contextId to the Context Store. Accepts up to three aliasIds. The Context will expire after the lease period. The Context will add the passed touchpoint string to its audit trail. The response contains the assigned contextId.)

```
GetContextDataResponse response = (GetContextDataResponse) contextStore.getData(id,
sessionId);
```

(Get data of a Context using contextId. The response contains the data returned.)

```
GetContextDataResponse response = (GetContextDataResponse) contextStore.getData(id,
sessionId, rid);
```

(Get data of a Context using contextId and its routingId. The response contains the data returned.)

```
GetContextDataResponse response = (GetContextDataResponse) contextStore.getData(id,
sessionId, rid, touchpoint);
```

(Get data of a Context using contextId and its routingId. The touchpoint will be added to the Context's audit entry. The response contains the data returned.)

```
ServiceResponse response = contextStore.getDataByAliasId(aliasId, sessionId, rid):
```

(Get data of a Context using its aliasId and its routingId. The response contains the data returned.)

```
ServiceResponse response = contextStore.getDataByAliasId(aliasId, sessionId, rid,
touchpoint):
```

(Get data of a Context using its aliasId and its routingId. The touchpoint will be added to the Contexts audit entry. The response contains the data returned.)

```
GetContextValueResponse response = (GetContextValueResponse) contextStore.getValue(id,
key, sessionId);
```

(Get value of Context with its contextId and the data key. The response contains the value returned.)

```
GetContextValueResponse response = (GetContextValueResponse) contextStore.getValue(id,
key, sessionId, rid);
```

(Get value of Context with its contextId and its routingId and the data key. The response contains the value returned.)

```
GetContextValueResponse response = (GetContextValueResponse) contextStore.getValue(id,
key, sessionId, rid, touchpoint);
```

(Get value of Context with its contextId and its routingId and the data key. The touchpoint will be added to the Context's audit entry. The response contains the value returned.)

```
ServiceResponse response = contextStore.getValueByAliasId(aliasId, key, sessionId, rid)
```

(Get value of Context with its aliasId and its routingId and the data key. The response contains the value returned.)

```
ServiceResponse response = contextStore.getValueByAliasId(aliasId, key, sessionId, rid, touchpoint)
```

(Get value of Context with its aliasId and its routingId and the data key. The touchpoint will be added to the Context's audit entry. The response contains the value returned.)

```
ContextStoreResponse response = (ContextStoreResponse) contextStore.putData(id, lease, data, sessionId);
```

(Update a Context's data entry and lease using contextId.)

```
ContextStoreResponse response = (ContextStoreResponse) contextStore.putData(id, lease, data, sessionId, rid);
```

(Update a Context's data entry and lease using contextId and its routingId.)

```
ContextStoreResponse response = (ContextStoreResponse) contextStore.putData(id, lease, data, sessionId, rid, touchpoint);
```

(Update a Context's data entry and lease using contextId and its routingId. The touchpoint is added to the Context's audit entry.)

```
ContextStoreResponse response = (ContextStoreResponse) contextStore.putDataByAliasId(aliasId, lease, data, sessionId, rid);
```

(Update a Context's data entry and lease using aliasId and its routingId.)

```
ContextStoreResponse response = (ContextStoreResponse) contextStore.putDataByAliasId(aliasId, lease, data, sessionId, rid, touchpoint);
```

(Update a Context's data entry and lease using aliasId and its routingId. The touchpoint is added to the Context's audit entry.)

```
ContextStoreResponse response = (ContextStoreResponse) contextStore.putValue(id, key, value, lease, sessionId);
```

(Update a Context's data value as identified by contextId, data key. Lease can also be modified in his action.)

```
ContextStoreResponse response = (ContextStoreResponse) contextStore.putValue(id, key, value, lease, sessionId, rid);
```

(Update a Context's data value as identified by contextId , data key and routingId. Lease can also be modified in his action.)

```
ContextStoreResponse response = (ContextStoreResponse) contextStore.putValue(id, key, value, lease, sessionId, rid, touchpoint);
```

(Update a Context's data value as identified by contextId , data key and routingId. Lease can also be modified in his action. The touchpoint is added to the Context's audit entry.)

```
ServiceResponse response = contextStore.putValueByAliasId(aliasId, key, value, lease, sessionId, rid);
```

(Update a Context's data value as identified by aliasId, data key and routingId. Lease can also be modified in his action.)

```
ServiceResponse response = contextStore.putValueByAliasId(aliasId, key, value, lease,
sessionId, rid, touchpoint);
```

(Update a Context's data value as identified by aliasId, data key and routingId. Lease can also be modified in his action. The touchpoint is added to the Context's audit entry.)

```
ContextStoreResponse response = (ContextStoreResponse) contextStore.deleteValue(id, key,
sessionId);
```

(Delete Context data value for given contextId and target key.)

```
ContextStoreResponse response = (ContextStoreResponse) contextStore.deleteValue(id, key,
sessionId, rid);
```

(Delete Context data value for given ID, target key and routingId.)

```
ContextStoreResponse response = (ContextStoreResponse) contextStore.deleteValue(id, key,
sessionId, rid, touchpoint);
```

(Delete Context value for given contextId, target key and routingId. The touchpoint is added to the Context's audit entry.)

```
ContextStoreResponse response = (ContextStoreResponse) contextStore.deleteValueByAliasID(aliasId,
key, sessionId, rid);
```

(Delete Context value for given aliasId, target key and routingId.)

```
ContextStoreResponse response = (ContextStoreResponse)
contextStore.deleteValueByAliasId(aliasId, key, sessionId, rid, touchpoint);
```

(Delete Context value for given aliasId, target key and routingId. The touchpoint is added to the Context's audit entry.)

```
ServiceResponse response = contextStore.deleteAliasId(aliasId, sessionId, rid);
```

(Deletes an alias object from the space associated with the passed aliasId and routingId.)

```
ServiceResponse response = contextStore.deleteAliasId(aliasId, sessionId, rid,
touchpoint);
```

(Deletes an alias object from the space associated with the passed aliasId and routingId. The touchpoint is added to the Context's audit trail.)

```
ServiceResponse response = contextStore.putAliasIdByAliasId(aliasId, lease, sessionId,
rid, newAliasId);
```

(Add a new aliasId to a Context based on its original aliasId and routingId.)

```
ServiceResponse response = contextStore.putAliasIdByAliasId(aliasId, lease, sessionId,
rid, newAliasId, touchpoint);
```

(Add a new aliasId to a Context based on its original aliasId and routingId. The touchpoint is added to the Context's audit trail.)

```
ServiceResponse response = contextStore.putAliasIdByContextId(id, lease, sessionId, rid, newAliasId, touchpoint);
```

(Add a new aliasId to a Context based on its contextId and routingId. The touchpoint is added to the Context's audit trail.)

```
ServiceResponse response = contextStore.putAliasIdByContextId(id, lease, sessionId, rid, newAliasId);
```

(Add a new aliasId to a Context based on its contextId and routingId.)

```
GetContextAuditResponse response = (GetContextAuditResponse) contextStore.getAuditDataByContextId(id, sessionId, rid);
```

(Returns the audit trail of the Context associated with the ContextId.)

```
GetContextAuditResponse response = (GetContextAuditResponse) contextStore.getAuditDataByAliasId(aliasId, sessionId, rid);
```

(Returns the audit trail of the Context associated with the aliasId.)

```
ContextStoreResponse response = (ContextStoreResponse) contextStore.deleteContext(id, sessionId);
```

(Delete Context associated with contextId.)

```
ContextStoreResponse response = (ContextStoreResponse) contextStore.deleteContext(id, sessionId, rid);
```

(Delete Context associated with contextId and routingId.)

```
ContextStoreResponse response = (ContextStoreResponse) contextStore.deleteContextByAliasId(aliasId, sessionId, rid);
```

(Delete Context associated with aliasId and routingId.)

```
GetContextIdsByGroupIdResponse response = (GetContextIdsByGroupIdResponse) contextStore.getContextIdsForGroupId(groupId, sessionId);
```

(Get Contexts by groupId, returns a list of contextIds.)

```
GetContextIdsByGroupIdResponse response = (GetContextIdsByGroupIdResponse) contextStore.getContextIdsForGroupId(groupId, sessionId, rid);
```

(Get Contexts by groupId and routingId, returns a list of contextIds.)

```
ServiceResponse response = contextStore.upsertContextWithContextID(contextID, lease, upsertPojo, sessionId, rid, touchpoint);
```

(Create a new context or update the context if it already exists via ContextID.)

```
ServiceResponse response = contextStore.upsertContextWithAliasID(aliasID, lease,
upsertPojo, sessionId, rid, touchpoint);
```

(Create a new context or update the context if it already exists via AliasID.)

4.1.3. Configuration

The IP address of the Context Store interface is entered on the **Connection Settings** tab of the Sample Client.

Context Store requires clients to identify themselves with a certificate before an operation can be run. For more information on configuration, see the [Certificate Based Authentication](#) section. Once the configuration has been completed on System Manager, the certs need to be loaded in to the client. This is configured in the **Key Store** section of the **Connection Settings** tab.

1. Click **Set** beside the **Key Store Location** to enter the Key Store File location.
2. Enter the password for this Key Store in the **Key Store Password** text box.
3. If the Trust Store is located in a different file to the Key Store, select the **No** radio button in the **Is the Trust Store file the same as the Key Store** option.

The system displays the **Trust Store Settings** section.

4. Enter the file location and password.

Once these configuration items are entered, users can proceed to the **Context Operations** tab to perform operations against the Context Store.

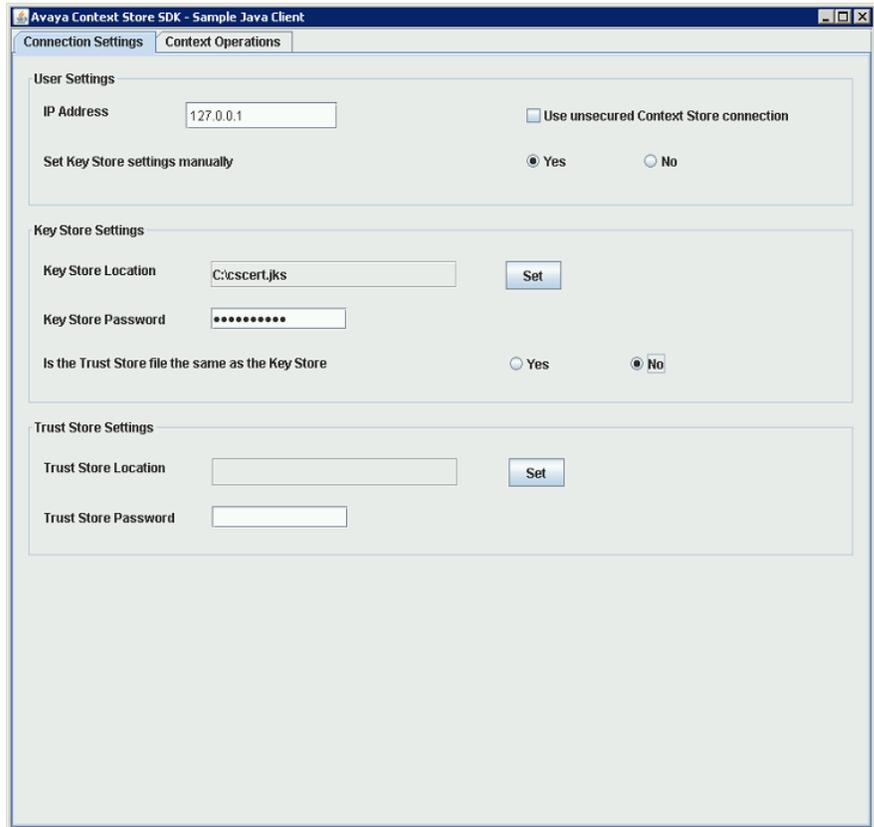


Figure 2 Sample Java Client Configuration

4.1.4. Usage

When running operations in the Context Operations tab only the relevant fields are used when populating the request. For example when adding a context, the only fields that need to be populated are the contextId, RoutingId, Touchpoint and Context Data the other fields are ignored for this operation.

The exception to this rule is the alias query parameter (**aliasIds** field); if this field is populated when any 'Add Context...' operation is submitted, the aliasIds in this field will be added to the context (if valid).

The format for the context data is `{"key1":"value1","key2":"value2"}`

The client does not support nested key value pairs such as `{"key1":"value1","key2":"key1":{"key3":"value3","key4":"value4"}}`

The format for the new alias data is `["newAlias1","newAlias2"]`

A sample from the program is below. An error is displayed if the configuration data is not available.

‡ This is a sample screen and might differ from final implementation

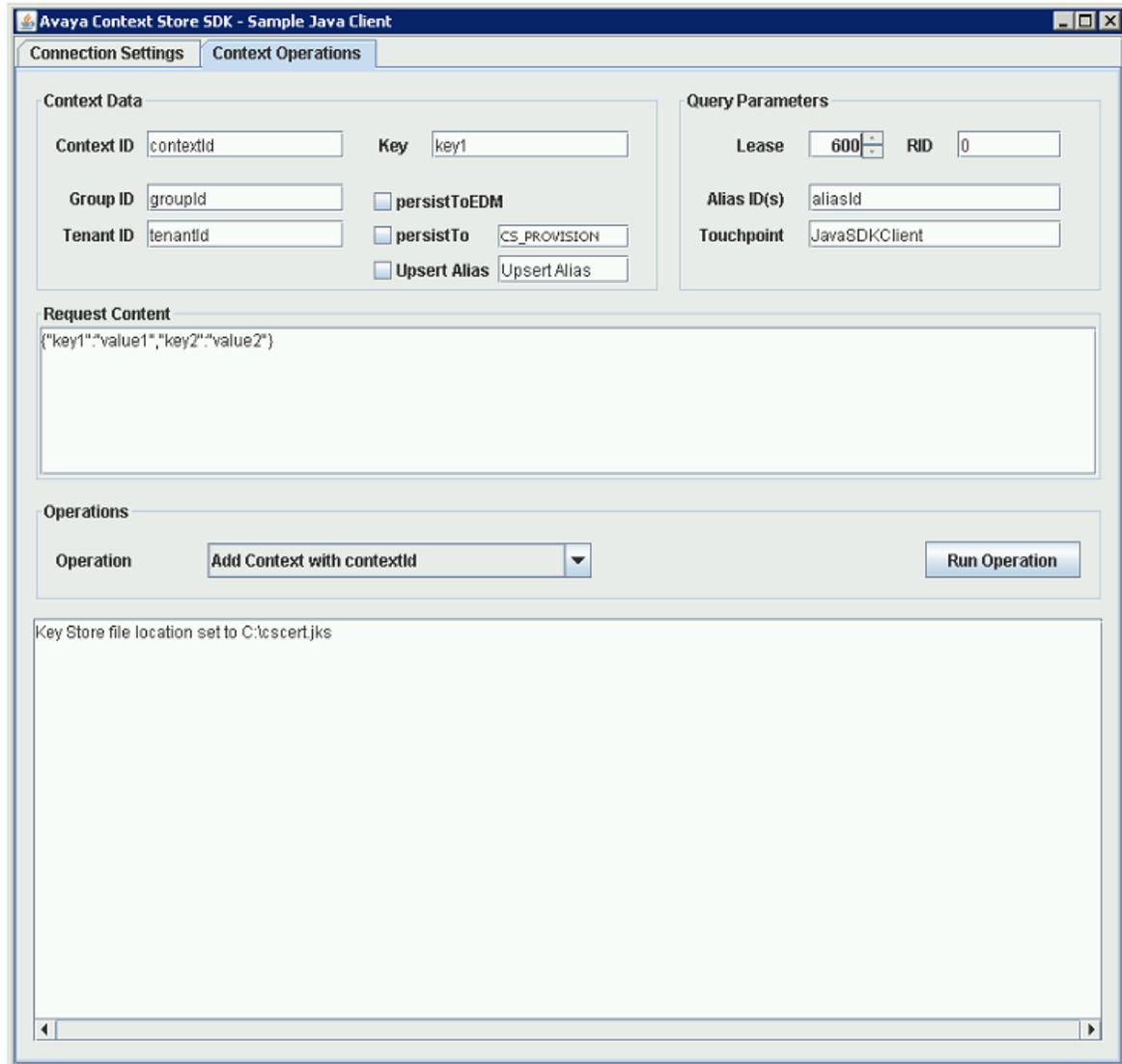


Figure 3 Java Sample Client

4.2. C# Client

This topic provides details for the C# GUI Sample Client. This Client is developed using [Visual Studio Express 2012](#) on .Net 4.5, and uses the latest recommended REST API exposed by the ASP.NET Web API. This Web API has to be imported into Express using NuGet.

4.2.1. GUI layout

The provided sample application is written in C# and it displays all the information required to create the request. The sample application also has a textbox which displays the server responses as well as the actual JSON which was transmitted.

Each section has a descriptive group box around it. The user and server details are configured in the **Connection Details** section.

The **Operations** menu contains all available operations for a context: Add, Update, Get and Delete. These operations take the information that is set in the **Context Details** section. Therefore, you must enter a value in either the **Context Id** textbox or **aliasId** textbox for the operations to succeed (the exception is Add

Context without contextId operation). Note, the **new aliasId** textbox is used for 'Update aliasIds...' operations only, this cannot be used for alias query parameter i.e. requests based on *aliasId* rather than *contextId*

The last section, **Display**, is used to display the request and response messages to and from the server. The response is highlighted in green if successful or red if not successful. All requests are in black.

The screenshot shows a window titled "Simple Client" with a menu bar containing "Configure" and "Operations". The main area features the Avaya logo and the text "Avaya Context Store Snap-in". Below this, there are three main sections:

- Context Details:** Contains four text input fields: "Context Id" (value: contextId), "Key" (value: key), "Group Id" (value: Demo), and "Value" (value: value). Below these is a "Context Data" field with the value: "key1":value1,"key2":value2,"key3":value3. A checkbox labeled "Persist to EDM" is also present.
- Query Parameters:** Contains a "Lease in sec" field with a spinner set to 7200, a "Routing Id" field with the value 0, an "aliasId" field with the value "alias1","alias2", and a "new aliasId" field with the value "alias1","alias2".
- Display:** A large text area for output, with a "Clear" button at the bottom right.

Figure 4 C# Sample Client

4.2.2. Application Layout:

This application is created as an example of how to consume a REST interface from a C# GUI Application. All code is contained in the default Form1.cs class. Each operation has a button and associated method. These operation methods (Add, Get, Update, Delete) are asynchronous. This is to prevent the main UI thread from blocking on any Rest API call. Used throughout the code is the "appendToTextDisplayWithColour()" method to write to the RichTextBox on the GUI. In a code, a simple log statement or sysout if a console application will suffice. A JSON datatype could have been used, but for simplicity, Strings were used throughout for the sending and receiving.

The following code can be changed to make the calling thread block by calling the "Result" property on the returned Task from the Rest method.

For example,

```
"HttpResponseMessage httpResponse = httpClient.PostAsync(url, httpContent).Result;"
```

in place of

```
"HttpResponseMessage httpResponse = await httpClient.PostAsync(url, httpContent);"
```

Code Snippet: addContext

The below snippet from the Form.cs files shows how to set the URL to be used and the content of the request to successfully add/POST a context to the Context Store.

```
private async void addContext(String id, String data, String lease, bool persistToEDM,
List< String > aliasIds, String rid)
{
    HttpClient httpClient = new HttpClient();
    String strResponse = String.Empty;

    String url = this.buildURL(FULL_CONTEXT_PATH);
    url = this.addUriQueryParam(url, ALIAS, aliasIds);
    url = this.addUriQueryParam(url, LEASE, lease);
    url = this.addUriQueryParam(url, RID, rid);

    //Content of request will be the User name and password
    String content = "{\"contextId\":\"" + id + "\",\"persistToEDM\":\""
        + persistToEDM.ToString().ToLower() + "\",\"data\":{\"" + data + "\"}}";

    try
    {
        appendToTextDisplayWithColour("\n" + DateTime.Now + " Add Context "
            + ((id.Length > 0) ? "with" : "without") + " contextId - Uri="
            + url, Color.Black);
        appendToTextDisplayWithColour("Content=" + content, Color.Black);

        HttpContent httpContent = new StringContent(content, Encoding.UTF8, "application/json");
        HttpResponseMessage httpResponse = await httpClient.PostAsync(url, httpContent);

        if (httpResponse.IsSuccessStatusCode)
        {
            if (String.Empty.Equals(id))
            {
                strResponse = await httpResponse.Content.ReadAsStringAsync();

                appendToTextDisplayWithColour("Success, Context was created, contextId=" + id
                    + strResponse + ", aliasId=" + aliasIds.ToString() + ", rid=" + rid + ",
                    lease=" + lease, Color.Green);
            }
            else
            {
                appendToTextDisplayWithColour("Success, Context was created, contextId=" + id
                    + ", aliasId=" + aliasIds.ToString() + ", rid=" + rid + ", lease=" + lease,
                    Color.Green);
            }
        }
    }
}
```

```
        }
    }
    else
    {
        appendToTextDisplayWithColour("Failed to create Context:Status Code="
        + httpResponse.StatusCode, Color.Red);
    }
}
catch (Exception e)
{
    appendToTextDisplayWithColour("Failed to post context to server e:" + e.Message, Color.Red);
}
finally
{
    httpClient.Dispose();
}
}
```

4.3. Security

Context Store can be configured to use [Certificate Based Authentication](#). This means, all clients must present a valid client certificate with their request. Not all REST client applications support the use of client certificates. If in doubt, use the clients provided in the Context Store SDK distribution as these clients support client certificates. When configuring applications to work with certificates, you have to provide a location to a keystore, containing a valid client certificate, and a password to access the keystore. For helpful information on configuring Certificate Based Authentication see the appendix of this document.

When using Certificate Based Authentication, it should be noted that it is not possible to make pure JavaScript (Ajax) calls to Context Store. This is due to disconnect between the browser and its JavaScript engine, meaning the client certificate is never sent. For this reason it is recommended to use a server side technology, e.g. servlets, to establish SSL connections with the Context Store cluster. The Context Store SDK has in built support for certificates.

5. Screen Pop

5.1. Overview

CSScreenPop is a service that users can use to view the data stored in Context Store. It uses a rules engine to determine the format and context of the data returned. It can use both the predefined rules that come with the engine or user defined rules to process the available data.

CSScreenPop supports the following formats for returning data: HTML, XML, JSON, URL, REDIRECT, MAILTO, WA and JSONARRAY. You can configure these output formats to view the context data either in a web browser or in other supported clients.

Configuring the rules in the engine, is just a matter of adding the rule in System Manager and the rules engine will update the rule set. The rules can change the output format and provides the ability to filter the results based on the keys in the context. For full details on how to create a user configurable rule or make use of the existing pre-defined rules, see the relevant sections below.

The URLs are designed to be easily integrated in to other applications, for example Avaya One-X Agent. For instructions on how to configure CSScreenPop and One-X agent, see the following sections.

5.2. Screen Pop Operations on the Context Store

- Operations
 - Create - Create a new context
 - Update - Update an existing context
 - View - View a context
 - Ucid - View a context by selecting the contextId from a UCID (or any other string)
 - Group - View a list of contexts stored in the same group
- Screen Pop operations are accessed via HTTP GET
 - Sample URL -
`http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/<OPERATION>/?id=<CONTEXT_ID>`
- Full details on the Operations and their URLs can be found in section [5.7 URLs & Operations](#).

5.3. Output Format

- Screen Pop can return context data in several formats, available formats are
 - HTML
 - XML
 - JSON
 - URL
 - REDIRECT
 - MAILTO
 - WA
 - JSONARRAY
- The format can be selected in two ways

- By entering pre-defined data in the context and letting the pre-defined or user configurable rules fire and then manipulate the context
- By adding a URL parameter 'format' to the Screen Pop URL and overriding the above pre-defined data set in the context
- Full details on what to configure and how to select the formats can be found in section [5.8 Response Formats](#).

5.4. Rules Engine

- A rules engine is provided to transform the context data stored in the Context Store into the selected format
- CSScreenPop uses a rule engine to take the rules stored in the space and apply them to the context data to create the response
- CSScreenPop comes with pre-defined rules that allow the customization of the context data based on certain key and value pairs being added to the context data
 - See the [5.9 Pre-Configured Rules](#) section for details of the pre-defined rules
- CSScreenPop also allows user configurable rules

See the section [5.10 Editable Rules](#) page for more details

5.5. Rest Service

- Screen Pop contains a REST service that is used to perform allowed operations on the Context Store

5.6. Data-grid

- Default rules are pre-configured in the Avaya Aura® Engagement Development Platform system through CSScreenPop service attributes.
- Screen Pop then loads the rules into the Context Store data-grid and these are accessed via the data-grid API.

5.7. URLs & Operations

This section explains the different URLs that are available and their functions as part of Context Store Screen Pop

- The CSScreenPop REST interface is found on this URL
 - `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/<OPERATION>/?id=<CONTEXT_ID>`
 - Where `<IP_ADDRESS>` is the external IP address of the Avaya Aura® Engagement Development Platform machine/cluster the CSScreenPop service is running on, `<OPERATION>` is the action being requested and where `<CONTEXT_ID>` is the id of the context that the CSScreenPop is using
- Each of the operations has its own URL (as described above) where the `<OPERATION>` defines the different operation

The URL operations are create and view a context, update and view a context, view a context, read contextId from UCID and view the context and finally view group's contextIds.

When creating a context with an aliasId, a contextId must be provided.

5.7.1. Create & View a Context

- This operation creates a context in the context store
- A minimum of one key/value pair must be added to the URL to add to the context
- Creates a new context if context with the requested id does not already exist
- Max length of URL can vary depending on the browser being used
- Default format returned is JSON
- **NOTE:** Counts as 2 requests on the Context Store system

URL

CREATE	<code>/services/CSScreenPop/cs/pop/create/</code>
---------------	---

URL Parameters

Key	Value	Notes
id	Any valid contextId that does not already exist	Mandatory Cannot be added as a key/value pair to the context
rid	Any valid rid	Optional Cannot be added as a key/value pair to the context
alias	Any valid aliasId	Optional Cannot be added as a key/value pair to the context
groupId	Any valid context groupId	Optional Cannot be added as a key/value pair to the context
tenantId	Any valid context tenantId	Optional Cannot be added as a key/value pair to the context
format	Any valid Context Store Screen Pop Format	Optional Cannot be added as a key/value pair to the context
lease	Any valid lease time	Optional Cannot be added as a key/value pair to the context
persistToEDM	true or false	Optional Cannot be added as a key/value pair to the context
persistTo	CS_PROVISION	Optional Cannot be added as a key/value pair to the context
<any other key>	<any valid value>	At least one is mandatory Name and value will be added to the context as the

		key/value pair All context data formatting and rules applied to the context data apply here also Can be any valid key/value pair Limited by length of URL
touchpoint	Any valid touchpoint string	Optional Cannot be added as a key/value pair to the context

Example URLs

- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/create/?id=12345format=html&foo=bar`
 - Creates a context with id of 12345 and a single key of foo with a value bar
 - Returns in html format
- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/create/?id=1&foo=bar&touchpoint=abc`
 - Creates a context with id of 1 and a single key of foo with a value bar
 - If audit trail is enabled, an entry will be created with touchpoint abc and operationId CR_C_C
- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/create/?id=1234567&format=html&foo=bar&alias=54321&groupId=6789&rid=routingId1&tenantId=abcd`
 - Creates a context with id of 1234567, an aliasId of 54321, a groupId of 6789, a routingId of routingId1, a tenantId of abcd and a single key of foo with a value bar
 - Returns in html format

5.7.2. Update & View a Context

- Updates a context in the context store
- Context must already exist to be updated
- Updates the context with the parameters on the URL (excluding id, alias, rid and format)
- Max length of URL is limited by the system and browser in use
- Default format to return in is JSON
- **NOTE:** Counts as 3 requests on the Context Store System

URL

Update	/services/CSScreenPop/cs/pop/update/
--------	--------------------------------------

URL Parameters

Key	Value	Notes
-----	-------	-------

id	Any valid contextId that already exists in the context store	Mandatory Cannot be added as a key/value pair to the context
rid	Any valid rid	Optional Cannot be added as a key/value pair to the context
alias	Any valid aliasId	Optional Cannot be added as a key/value pair to the context
format	Any valid Context Store Screen Pop Format	Optional Cannot be added as a key/value pair to the context
<any other key>	<any valid value>	Optional Name and value will be added to the context store as the key/value pair, should the key already exist then it is updated All context data formatting and rules applied to the context data apply here also Can be any valid key/value pair Limited by length of URL
touchpoint	Any valid touchpoint string	Optional Cannot be added as a key/value pair to the context

Example URLs

- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/update/?id=12345&format=html&foo=bar`
 - Updates the context which has contextId `12345`, with a single key of `foo` with a value `bar`
 - Returns in html format

- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/update/?alias=12345&format=html&foo=bar&rid=routingId2`
 - Updates the context which has aliasId `12345` and routingId of `routingId2` with a single key of `foo` with a value `bar`
 - Returns in html format

5.7.3. View a Context

- Views a context
- Default format returned is JSON
- **NOTE:** Counts as 2 requests on the Context Store System

URL

View	<code>/services/CSScreenPop/cs/pop/context/</code>
-------------	--

URL Parameters

Key	Value	Notes
id	Any valid contextId that already exists in the context store	Mandatory
rid	Any valid rid	Optional
alias	Any valid aliasId	Optional
format	Any valid Context Store Screen Pop Format	Optional
touchpoint	Any valid touchpoint string	Optional Cannot be added as a key/value pair to the context

Example URLs

- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/context/?id=12345&format=html`
 - Returns the context **12345** in html format
- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/context/?id=12345&alias=678&format=html`
 - Returns the context with aliasId **678** in html format

5.7.4. View a Context by Selecting contextId from Parameter (UCID)

- This operation takes any string and reads a contextId based on a start and stop position
- See the section [5.10.1 Configurable Properties](#) for instructions on how to set the range to select the contextId from
- Counts as a single request on the Context Store
- Not limited to UCID, can be any string where that contains a contextId that can be read from fixed locations within that string
- Default format returned is JSON
- **NOTE:** Counts as 2 requests on the Context Store System

URL

View	<code>/services/CSScreenPop/cs/pop/ucid/</code>
-------------	---

URL Parameters

Key	Value	Notes
ucid	Any valid UCID that contains a contextId that does already exist in the context store	Mandatory (if delimit is not set)
delimit	Any valid UCID that contains a contextId that does	Mandatory (if ucid is not set)

	already exists in the context store	
format	Any valid Context Store Screen Pop Format	Optional
touchpoint	Any valid touchpoint string	Optional Cannot be added as a key/value pair to the context

Example URLs

- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/ucid?ucid=000001234500000&format=html`
 - Assume configurable properties in snap-in properties page is set to '5,10'
 - Returns the context with id **12345** in html format
- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/ucid?delimit=00000.12345.00000&format=html`
 - Assume configurable properties in snap-in properties page is set to '\.' and 2
 - Returns the context with id **12345** in html format

5.7.5. View contextIds by groupId

- **NOTE:** Counts as 2 requests on the Context Store System
- Default format returned is JSON

URL

Group	<code>/services/CSScreenPop/cs/pop/group/</code>
--------------	--

URL Parameters

Key	Value	Notes
id	Any valid context groupId that already exists in the context store	Mandatory
format	Any valid Context Store Screen Pop Format	Optional

Example URLs

- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/group?id=6789&format=html`
 - Returns the all the contextId with groupId of **6789** in html format

5.7.6. Upsert & View a Context

- This operation creates or updates a context in the context store depending on its existence
- A minimum of one key/value pair must be added to the URL to add to the context
- Creates a new context if context with the requested id does not already exist otherwise the existing context will be updated

- Default format returned is JSON
- **NOTE:** Counts as 2 requests on the Context Store System

URL

Upsert by Context ID	/services/CSScreenPop/cs/pop/upsert/
Upsert by Alias ID	/services/CSScreenPop/cs/pop/upsert/alias/

URL Parameters

Key	Value	Notes
id	Any valid context Id. For the alias URL this will serve as the query alias for the request.	Mandatory Cannot be added as a key/value pair to the context
rid	Any valid rid	Optional Cannot be added as a key/value pair to the context
alias	Any valid aliasId	Optional Cannot be added as a key/value pair to the context
groupId	Any valid context groupId	Optional Cannot be added as a key/value pair to the context
tenantId	Any valid context tenantId	Optional Cannot be added as a key/value pair to the context
format	Any valid Context Store Screen Pop Format	Optional Cannot be added as a key/value pair to the context
lease	Any valid lease time	Optional Cannot be added as a key/value pair to the context
persistToEDM	true or false	Optional Cannot be added as a key/value pair to the context
persistTo	CS_PROVISION	Optional Cannot be added as a key/value pair to the context
<any other key>	<any valid value>	At least one is mandatory Name and value will be added to the context as the key/value pair All context data formatting and rules applied to the context data apply here also

		Can be any valid key/value pair Limited by length of URL
touchpoint	Any valid touchpoint string	Optional Cannot be added as a key/value pair to the context

Example URLs

- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/upsert/?id=12345&alias=678&format=html&foo=bar&rid=23`
 - Upserts a context with id of 12345, alias Id of 678, routing id of 23 and a single key of foo with a value bar
- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/upsert/alias/?id=12345&alias=678&groupId=6789&format=html&foo=bar&rid=23&tenantId=2468`
 - Uses alias of 12345 for the upsert query, alias Id of 678, routing id of 23, a group Id of 6789, a tenant id of 2468 and a single key of foo with a value bar

5.7.7. View a Context by Selecting contextId from Parameter (UCID)

- This operation takes any string and reads a contextId based on a start and stop position
- See the section [5.10.1 Configurable Properties](#) for instructions on how to set the range to select the contextId from
- Counts as a single request on the Context Store
- Not limited to UCID, can be any string where that contains a contextId that can be read from fixed locations within that string
- Default format returned is JSON
- **NOTE:** Counts as 2 requests on the Context Store System

URL

View	<code>/services/CSScreenPop/cs/pop/ucid/</code>
-------------	---

URL Parameters

Key	Value	Notes
ucid	Any valid UCID that contains a contextId that does already exist in the context store	Mandatory (if delimit is not set)
delimit	Any valid UCID that contains a contextId that does already exist in the context store	Mandatory (if ucid is not set)
format	Any valid Context Store Screen Pop Format	Optional
touchpoint	Any valid touchpoint string	Optional Cannot be added as a key/value pair to the context

Example URLs

- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/ucid?ucid=000001234500000&format=html`
 - Assume configurable properties in snap-in properties page is set to '5,10'
 - Returns the context with id 12345 in html format
- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/ucid?delimit=00000.12345.00000&format=html`
 - Assume configurable properties in snap-in properties page is set to '\.' and 2
 - Returns the context with id 12345 in html format

5.7.8. View contextIds by groupId

- **NOTE:** Counts as 2 requests on the Context Store System
- Default format returned is JSON

URL

Group	<code>/services/CSScreenPop/cs/pop/group/</code>
--------------	--

URL Parameters

Key	Value	Notes
id	Any valid context groupId that already exists in the context store	Mandatory
format	Any valid Context Store Screen Pop Format	Optional

Example URLs

- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/group?id=6789&format=html`
 - Returns the all the contextId with groupId of 6789 in html format

5.7.9. Older Browser Compatibility

Older versions of browsers are not supported by the HTML format of Context Store Screen Pop. There is a workaround in place to allow the html to render in older browsers, in particular IE8. To use this workaround, add '**pop.html**' to the end of each of the supported URL's. This will work for all 5 Screen Pop URLs that are listed in URLs and Operations.

For example, the following URLs:

`http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/create/?id=123&key1=value1` and
`http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/context/?id=123` will become:

- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/create/pop.html?id=123&key1=value1`
- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/context/pop.html?id=123`

There is no change in functionality in how these URLs work. This alternative URL will change the content type returned to allow older browsers to render the HTML. This should only be used where the fully

supported version of the URL's do not render as expected. This is known to affect IE 8 where the browser will not render the html and Firefox where the html renders but Firefox makes a second call to the webpage.

5.7.10. JSON Content Type

Certain consumers of JSON data need the content header in the response returned from the GET request to be set to `application/json`. If a consumer of the Screen Pop JSON data required this header to be set then append `'/json'` to the end of each of the supported URLs. This will work for all 5 Screen Pop URLs that are listed in URLs and Operations. If integrating with Engagement Designer then this URL should be used.

For example, the following URLs:

`http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/create/?id=12345&key1=value1` and
`http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/context/?id=12345` will become:

- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/create/json?id=12345&key1=value1`
- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/context/json?id=12345`

There is no change in functionality in how these URLs work. This alternative URL will change the content type returned to allow consumers that require it to be set to consume it.

5.8. Response Formats

There are 8 different formats and 3 different ways that the output format of the Screen Pop can be selected.

The 8 formats are

- JSON (default)
- HTML
- XML
- A URL
- A html page that redirects to a different URL
- A mailto link
- A Work Assignment format
- A JSON Array

The 3 different ways to select the format are as follows, all of which use the Screen Pop Rules Engine to generate the response

- By setting pre-defined data in the context
- By using the user configurable rules to detect specific keys
- By using the format override URL parameter
 - The format override function will override any of the above settings

5.8.1. Format Selection

These are the 3 ways the format can be set

1) Default Rules Format

- If the Context data contains a key named `'cs_screenpop_format'` then the format entered in the key will be selected provided it is one of the allowed values
- Additional context keys and values will need to be added to get certain formats to work or to expand the functionality, for example `'cs_screenpop_url'` and `'cs_screenpop_css'`
- These are explained further below and defined fully in section [5.9.2 Default Rules Provided.](#)

2) User Configurable Rules

- Users can also create their own rules which can select the format based on the engine matching contexts set in the rule
 - Further detail of how to configure the rules can be found here section [5.10 Editable Rules](#).

3) Format Override

- To avoid setting values in the Context the format override parameter can be used
- This will override the 'cs_screenpop_xxx'
- When setting the format to html a default css page is picked up using the updateable properties
- When setting the format to URL a default URL link is picked up using the updateable properties

5.8.2. Format Types

- The 8 formats that can be applied by the rules engine are listed below.
 - [HTML](#); [XML](#); [JSON](#); [URL](#); [REDIRECT](#); [MAILTO](#); [WA](#); [JSONARRAY](#)
 - The only values that will be accepted are these 6 strings, all other values will be rejected
- For full details on how the pre-defined rules are set up see in section [5.9 Pre-Configured Rules](#)

1) HTML

- This return the Context data requested in the HTML format shown below
- When requesting html format then a link to a css file can be added to the html returned to allow the html to be formatted
 - When not using the override in the URL then 'cs_screenpop_css' must be set to a valid, resolvable css file.
- CSS file provided is a demo css file, users should create and host their own css files
- As HTML is a viewable format the format will not return values of keys that start and end with a '*' character as they are considered non-viewable values.

Sample HTML response with optional CSS link included

```
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
    <link rel="stylesheet" type="text/css" href="<cs_screenpop_css"/>
  </head>
  <body>
    <div class="header">
      <div class="context-id">12345</div>
    </div>
    <div class="context" id="context">
      <div class="entry" id="key2">
        <div class="key">key2</div>
        <div class="value">value2</div>
      </div>
```

```
<div class="entry" id="key1">
  <div class="key">key1</div>
  <div class="value">value1</div>
</div>
</div>
</body>
</html>
```

Pre-defined

- Set value of key 'cs_screenpop_format' to 'html' in context
- To add a css link to the html then 'cs_screenpop_css' must be set correctly
- Set value of key 'cs_screenpop_include' to a pipe (|) separated list of key/value pairs to be added as parameters to the URL
- Set value of key 'cs_screenpop_exclude' to a pipe (|) separated list of key/value pairs that will not be added as parameters to the URL. All other contexts will be added.

User Configurable

- Create rule to set the format to html
- Example

```
o _if_contains_key1_then_format_html_priority_1_include_key1
```

Override Format

- To use set the URL parameter format to html (&format=html)
 - o Setting this allows the css to be read from the snap-in properties without the need to set 'cs_screenpop_css'
- Example
 - o `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/<OPERATION>/?id=<CONTEXT_ID>&format=html`

2) XML

- This return the Context data requested in XML format.
- As xml is not considered to be a user viewable format keys that start and end with a '*' will be returned so clients consuming the xml data returned will need to filter this data

Sample XML response

```
<?xml version="1.0" encoding="UTF-8"?>
<context id="12345">
  <entry name="key2" value="value2"/>
  <entry name="key1" value="value1"/>
</context>
```

Pre-defined

- Set value of key 'cs_screenpop_format' to 'xml' in context

- Set value of key 'cs_screenpop_include' to a pipe (|) separated list of key/value pairs to be added as parameters to the URL
- Set value of key 'cs_screenpop_exclude' to a pipe (|) separated list of key/value pairs that will not be added as parameters to the URL. All other contexts will be added.

User Configurable

- Create rule to set the format to xml
- Example
 - `_if_contains_key1_then_format_xml_priority_1_include_key1`

Override Format

- To use set the URL parameter format to xml (&format=xml)
- Example
 - `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/<OPERATION>/?id=<CONTEXT_ID>&format=xml`

3) JSON

- This return the Context data requested in JSON format.
- As json is not considered to be a user viewable format keys that start and end with a '*' will be returned so clients consuming the xml data returned will need to filter this data

```
{  
  "key2": "value2",  
  "key1": "value1"  
}
```

Pre-defined

- Set key 'cs_screenpop_format' to 'json' in context

User Configurable

- Create rule to set the format to json
- Example
 - `_if_contains_key1_then_format_json_priority_1_include_key1`

Override Format

- To use set the URL parameter format to json (&format=json)
- Example
 - `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/<OPERATION>/?id=<CONTEXT_ID>&format=json`

4) URL

- This return the Context data requested in URL format.
- All parameters in the context will be appended to the URL, unless include/exclude functionality is used

- If 'cs_screenpop_url' is not set in the context then the default url will be read from the Snap-in properties will be used.

```
<cs_screenpop_url?>key2=value2&key1=value1
```

Pre-defined

- Set value of key 'cs_screenpop_format' to 'url' in context
- Set value of key 'cs_screenpop_url' to be the start of the URL being redirected to
 - This is required and if not set the url output will not be generated
- Set value of key 'cs_screenpop_include' to a pipe (|) separated list of key/value pairs to be added as parameters to the URL
- Set value of key 'cs_screenpop_exclude' to a pipe (|) separated list of key/value pairs that will not be added as parameters to the URL. All other contexts will be added.

User Configurable

- Create rule to set the format to url
- Example
 - `_if_contains_key1_then_format_url_priority_1_include_key1`

Override Format

- To use set the URL parameter format to url (&format=url)
 - Setting this allows the URL start to be read from the snap-in properties without the need to set 'cs_screenpop_url'
- Example
 - `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/<OPERATION>/?id=<CONTEXT_ID>&format=url`

5) REDIRECT

- Redirects the browser to a configurable URL. Uses meta-refresh to redirect browser, see html returned below.
- All parameters in the context will be appended to the URL, unless include/exclude functionality is used
- If 'cs_screenpop_url' is not set in the context then the default url will be read from the Snap-in properties will be used.

```
<html>
  <head>
    <meta http-equiv="refresh" content="1;
url=<cs_screenpop_url>key2=value2&key1=value1" />
  </head>
</html>
```

Pre-defined

- Set value of key 'cs_screenpop_format' to 'redirect' in context
- Set value of key 'cs_screenpop_url' to be the start of the URL being redirected to

- This is required and if not set the url output will not be generated
- Set value of key 'cs_screenpop_include' to a pipe (|) separated list of key/value pairs to be added as parameters to the URL
- Set value of key 'cs_screenpop_exclude' to a pipe (|) separated list of key/value pairs that will not be added as parameters to the URL. All other contexts will be added.

User Configurable

- Create rule to set the format to xml
- Example
 - `_if_contains_key1_then_format_redirect_priority_1_include_key1`

Override Format

- To use set the URL parameter format to redirect (&format=redirect)
 - Setting this allows the URL start to be read from the snap-in properties without the need to set 'cs_screenpop_url'
- Example
 - `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/<OPERATION>/?id=<CONTEXT_ID>&format=redirect`

6) MAILTO

- Return a 'mailto' link
`mailto:<cs_screenpop_email>`

Pre-defined

- Set value of key 'cs_screenpop_format' to 'mailto' in context
- Set value of key 'cs_screenpop_email' to be the email address of the URL

User Configurable

- Create rule to set the format to mailto
- Example
 - `_if_contains_key1_then_format_mailto_priority_1_include_key1`

Override Format

- To use set the URL parameter format to mailto (&format=mailto) and ensure the context key 'cs_screenpop_email' has a valid email stored
 - The user must ensure this is a valid email address, no validation is performed
- Example
 - `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/<OPERATION>/?id=<CONTEXT_ID>&format=mailto`

7) WA

- Return json data in a format that can be directly input as attributes into the Work Assignment snap-in

- Format will combine both the key and the value into a list of attributes

```
{"attributes":["key1.value1","key2=value2","keyN=valueN"]}
```

Pre-defined

- Set value of key 'cs_screenpop_format' to 'wa' in contextUser Configurable
- Create rule to set the format to wa
- Example

- `_if_contains_ key1 _then_format_ wa _priority_ 1 _include_ key1 key2
KeyN`

Override Format

- To use set the URL parameter format to mailto (&format=wa)
- Example
 - http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/<OPERATION>/?id=<CONTEXT_ID>&format=wa

8) JSONARRAY

- Return context data in a json array
- Format will ignore the key and only use the value when converting into a list of attributes

```
["Value1","Value2","ValueN"]
```

Pre-defined

- Set value of key 'cs_screenpop_format' to 'jsonarray' in contextUser Configurable
- Create rule to set the format to jsonarray
- Example

- `_if_contains_ key1 _then_format_ jsonarray _priority_ 1 _include_
key1 key2 keyN`

Override Format

- To use set the URL parameter format to jsonarray (&format=jsonarray)
- Example
 - http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/<OPERATION>/?id=<CONTEXT_ID>&format=jsonarray

5.9. Pre-Configured Rules

Screen Pop comes with pre-configured rules that can be used to allow Context Store users to configure the output of the Screen Pop feature based on the pre-loaded rules.

This allows the rules engine to be used without the need for individual rules to be written, as the rules are accessed by adding certain key words to the context.

5.9.1. Rules Engine Overview

Rules Group

- There are three groups of rules, 'Format', 'Filter' and 'User'. Format and Filter are predefined and explained here, User rules are set by the user and are detailed on this page Editable Rules
- One rule from each group can be executed by the engine
- The 'Filter' rules filter the context data to either include or exclude certain selected keys
 - There are two rules in this group and they will be fired before
 - The rule is selected by adding a key 'cs_screenpop_format' to the context and setting its value to one of the six options below
- The 'Format' rules convert the context data into the requested format (see below)
 - There are 6 rules in this group and they are the lowest allowed priority so will be fired last
 - The rule is selected by adding either 'cs_screenpop_include' or 'cs_screenpop_exclude' to the context and setting its value to a list
 - Note that unlike the filter group there are two different keys in this rule so both can be added to the context. Users should take care to not add both 'cs_screenpop_include' and 'cs_screenpop_exclude' as the engine will not be able to determine which to select and may fire either

Rule Priority

- The 'Format' rules all have a priority of -2. This means they are always the last rules to be fired
- The 'Filter' rules all have a priority of -1. This means that they will always be fired second from last with the 'Format' rules being fired after.
- User rules will always have a priority greater than these two rules and are in a different rule group to all default rules.

5.9.2. Default Rules Provided

Format - JSON

Returns the context data in JSON format.

Key	Value	Notes
cs_screenpop_format	json	Required.

Format - HTML

Returns the context data in an HTML format.

If set, a link to a css page can be a part of the HTML returned. This is set using the parameter below or by using the value set in the Screen Pop properties.

Key	Value	Notes
cs_screenpop_format	html	Required.
cs_screenpop_css		Optional. Sample CSS file is included, to use set the value to '/services/CSScreenPop/demo/css/demo.css' Should html format be requested and no css page is set then the default page set in the snap-in attributes will be returned

Format - URL

Key	Value	Notes
cs_screenpop_format	url	Required.
cs_screenpop_url		Optional. Initial part of URL that context data is to be added to Should url format be requested and this value is not set then the default url set in the snap-in attributes will be returned

Format - REDIRECT

Key	Value	Notes
cs_screenpop_format	redirect	Required.
cs_screenpop_url		Optional. Initial part of URL that context data is to be added to Should url format be requested and this value is not set then the default url set in the snap-in attributes will be returned

Format - XML

Key	Value	Notes
cs_screenpop_format	xml	Required.

Format - WA

Key	Value	Notes
cs_screenpop_format	wa	Required.

Format - JSONARRAY

Key	Value	Notes
cs_screenpop_format	jsonarray	Required.

Format - MAILTO

Key	Value	Notes
cs_screenpop_format	mailto	Required.
cs_screenpop_email		Required. Email address to be returned as part of mailto

Filter - Include

Key	Value	Notes
cs_screenpop_include		Pipe separated list of keys to include, for example 'key1 key2 key3'

Filter - Exclude

Key	Value	Notes
cs_screenpop_exclude		Pipe separated list of keys to exclude, for example 'key1 key2 key3'

5.10. Editable Rules

Context Store Screen Pop includes the ability to create user configurable rules to allow Context Store users to customize the output of the Screen Pop feature based on the rules.

The user rules allow users to create a list of context keys so that if the keys exist in the context, the rules will be applied. All user configurable rules are considered to be in the same group so only one user configurable rule will fire. The default pre-defined rules can and will fire if they also match.

Where two sets of keys both match different rules, the higher priority will fire and all rules should have different priorities

Rules contain 5 pre-defined keywords that all start and end with '_'. Note that context keys that start or end with '_' are not allowed in the context store.

All of the user rules can be updated at run-time but must be edited one at a time.

Configurable Rule Keywords

1. `_if_contains_`
2. `_then_format_`
3. `_priority_`
4. `_include_`
5. `_exclude_`
6. `_update_`
7. `_delete_`

Configurable Rule Format

Rules are configured in the following format; the order of the keywords must not change

```
_if_contains_ <key(=value)> _then_format_ <html> <optional setting> _priority_
<1> _include/_exclude_ <key> _update <key=value> _delete_ <key>
```

Breakdown of Rule Format and Keywords

`_if_contains_`

- All rules must contain this value, the rule is considered to start where this text begins
- All text before this value is ignored
- Following this key and before the '`_then_format_`' key must be a space separated list of keys to match on. A logical AND operation is performed on all keys in this list and the rule will fire if all the keys exist provided no rule of higher priority has fired first.
- For a rule to be valid there must be a space separated list of keys between '`_if_contains_`' and '`_then_include_`'. A single key is also acceptable.
- The number of keys allowed in a user rule is limited to 10, if more than 10 keys are entered then the rule will not be added and a message will be added to the audit log
- Example:
 - `_if_contains_ key1 key2 key3 _then_contains_`

The rule will fire if a context contains keys named 'key1', 'key2' and 'key3'

_then_format_

- All rules must contain this value and it must appear after '_if_contains_'
- There are 6 valid values that can be after _then_format_ that the rules engine will accept. They are as listed below and follow the formats defined section [5.8.2 Format Types](#).

Allowed Formats

- **html**
 - A user rule that sets its format to 'html' can have an override that will allow each individual rule to have its own css page to link to
 - This is done by adding a 'css=' after the _then_format_ keyword
 - Example
 - `_if_contains_ key1 key2 _then_format_ html
css=http://www.avaya.com/demo.css _priority_ 5 _include_ key1`
- **xml**
 - No extensions to this format
- **json**
 - No extensions to this format
- **url**
- - A user rule that sets its format to 'url' can have an override that will allow each individual rule to have its own url to prepend to
 - This is done by adding a 'url=' after the _then_format_ keyword
 - Example
 - `_if_contains_ key1 key2 _then_format_ url
url=http://www.avaya.com/demo.html? _priority_ 5 _include_ key1`
- **redirect**
 - A user rule that sets its format to 'redirect' can have an override that will allow each individual rule to have its own url to prepend to
 - This is done by adding a 'url=' after the _then_format_ keyword
 - Example
 - `_if_contains_ key1 key2 _then_format_ redirect
url=http://www.avaya.com/demo.html? _priority_ 5 _include_ key1`
- **mailto**
 - No extensions to this format
- **wa**
 - No extensions to this format
- **jsonarray**
 - No extensions to this format
 -

priority

- Optional
- If not included a default value of 3 is selected automatically
- Must be a valid integer to be considered
- The higher the value the higher the priority

include or _exclude_

- Optional. Only one of **_include_** or **_exclude_** can be selected
 - If both are added '**_include_**' is selected over '**_exclude_**'
- Space separated list of keys to include or exclude in the Screen Pop response

update

- Optional
- Will update the space separated list of keys with the values set
- Space separated list of keys set to values to update the context in Context Store
- This counts as an operation against Context Store every time the rule is fired
- When using the **_update_** keyword in combination with the **_include_** keyword the data returned will be a combination of the keys listed in both lists. All keys listed in the **_update_** list will be returned whether or not they are in the **_include_** list.

delete

- Optional
- Will delete a single key. This is not a list and only a single key can be deleted
- This counts as an operation against Context Store every time the rule is fired

Examples

- All examples assume that no rule of higher priority matches unless otherwise stated.

Match on a Single Key - Only Return Key Matched

Rule

```
_if_contains_ key1 _then_format_ html _priority_ 1 _include_ key1
```

- If a context contains a key named 'key1' then the rules engine will fire this rule and return it in HTML format where the rule priority is 1 and the response will only return 'key1'

Match on 2 Keys - Only Return 2 Set Keys

Rule

```
_if_contains_ key1 key2 _then_format_ html _priority_ 2 _include_ key1 key2
```

- If a context contains a key named 'key1' and 'key2' then the rules engine will fire this rule and return it in HTML format where the rule priority is 2 and the response will only return 'key1' and 'key2'
- If combined with the rule above both would match but this rule would fire due to the higher priority

Match on 3 Keys - Return all Context Data except 2 Keys excluded

Rule

```
_if_contains_ key1 key2 key3 _then_format_ html _priority_ 3 _exclude_ key1 key2
```

- If a context contains a key named 'key1' and 'key2' and 'key3' then the rules engine will fire this rule and return it in HTML format where the rule priority is 3 and the response will return every key value pair except 'key1' and 'key2'
- If combined with the rule above both would match but this rule would fire due to the higher priority

Match on 4 Keys - Return all Context Data except 2 Keys excluded

Rule

```
_if_contains_ key1 key2 key3 key4 _then_format_ html _priority_ 4 _exclude_ key1 key2
```

- If a context contains a key named 'key1' and 'key2' and 'key3' and 'key4' then the rules engine will fire this rule and return it in HTML format where the rule priority is 4 and the response will return every key value pair except 'key1' and 'key2'
- If combined with the rule above both would match but this rule would fire due to the higher priority

Match on 5 Keys - Only Return 4 Set Keys

Rule

```
_if_contains_ key1 key2 key3 key4 key5 _then_format_ json _priority_ 5 _include_ key1 key4 key5 name
```

- If a context contains a key named 'key1' and 'key2' and 'key3' and 'key4' and 'key5' then the rules engine will fire this rule and return it in HTML format where the rule priority is 5 and the response will only return 'key1', 'key4', 'key5' and 'name'
 - Note that name is not matched on but is included in the response returned
- If combined with the rule above both would match but this rule would fire due to the higher priority

Format set to HTML with css page set in rule

Rule

```
_if_contains_ key1 key2 _then_format_ html css=http://www.ayaya.com/fake.css _priority_ 2 _include_ key1 key2
```

- If a context contains a key named 'key1' and 'key2' then the rules engine will fire this rule and return it in HTML format where the rule priority is 2 and the response will only return 'key1' and 'key2'
- The `_then_format_` is set to html and appended by 'css=<link to css page>' so the css used in the html returned by this rule will be 'http://www.ayaya.com/fake.css'

Format set to URL with url set in rule

Rule

```
_if_contains_ key1 key2 _then_format_ url css=http://www.ayaya.com/fake.html? _priority_ 2 _include_ key1 key2
```

- If a context contains a key named 'key1' and 'key2' then the rules engine will fire this rule and return it in HTML format where the rule priority is 2 and the response will only return 'key1' and 'key2'

- The `_then_` format is set to `url` and appended by `'url=<start of url>'` so the url returned will start with `'http://www.ayaya.com/fake.html?'`

Format set to REDIRECT with url set in rule

Rule

```
_if_contains_ key1 key2 _then_format_ redirect
css=http://www.ayaya.com/fake.html? _priority_ 2 _include_ key1 key2
```

- If a context contains a key named 'key1' and 'key2' then the rules engine will fire this rule and return it in HTML format where the rule priority is 2 and the response will only return 'key1' and 'key2'
- The `_then_` format is set to `redirect` and appended by `'url=<start of url>'` so the redirected to will start with `'http://www.ayaya.com/fake.html?'`

Format set to REDIRECT with url set in rule, update keys and delete keys

Rule

```
_if_contains_ key1 key2 _then_format_ redirect
css=http://www.ayaya.com/fake.html? _priority_ 2 _include_ key1 key2 key3
_update key3=value3 _delete_ key4
```

- If a context contains a key named 'key1' and 'key2' then the rules engine will fire this rule and return it in HTML format where the rule priority is 2 and the response will only return 'key1' and 'key2'. In addition 'key3' will be updated (or created) with 'value3' and 'key4' will be deleted.
- The `_then_` format is set to `redirect` and appended by `'url=<start of url>'` so the redirected to will start with `'http://www.ayaya.com/fake.html?'`

5.10.1. Configurable Properties

- Screen Pop contains a number of user configurable properties in total, these include 20 user configurable rules that are numbered individually from 1-20 but all work in the exact same fashion
- Certain attributes can be updated at runtime and others must be set before the service is installed.
- To configure these properties in System Manager go to
 - Home -> Elements -> Avaya Aura® Engagement Development Platform -> Configuration -> Attributes

Attributes Configuration

When a service is first installed, the factory default value picked by the service writer is used for each attribute for all service profiles. You may override the factory default value by using the Service Globals tab below. If you need to set specific values for attributes in a service profile, then use the Service Profiles tab below.

Cluster: Automation
Service: CSScreenPop

Name	Override Default	Effective Value	Description
Base for URL	<input type="checkbox"/>	http://127.0.0.1/7	Base URL used when selecting a format of uri or redirect. Will automatically update rules engine on changing.
Context Store Rest Version	<input checked="" type="checkbox"/>	3.1.0.0.2600	Set this to be the version of CSRest installed and selected on the cluster for example 3.0.0.0.xxx
CSS for HTML	<input type="checkbox"/>	/services/CSScreenPop/demo/css/demo.css	CSS page loaded when selecting a format of html. Will automatically update rules engine on changing.
Identifier Delimit Character	<input type="checkbox"/>	\.	Value for identifier delimit character. The chosen delimiter must be preceded by an escape character, for example \. sets . as the delimiter
Identifier Delimit Position	<input type="checkbox"/>	2	Value for identifier delimit position. Example if delimiter attribute is set to \. then set position to 3 to retrieve a context id of 333333 from the identifier 111111.222222.333333.444444
Identifier Parsing Position	<input type="checkbox"/>	0,10	Used to detect a context id from a string that contains the id. Comma separated list that must only contain two numbers, the first contains the start point of the context id and the second contains the end position. Will automatically update rules engine on changing.
JavaScript for HTML	<input type="checkbox"/>		JavaScript file to include when selecting a format of html. Provide http link to where the JavaScript file is hosted.
Supplier Id	<input type="checkbox"/>	10000000	Avaya provided supplier id
User Rule 01	<input type="checkbox"/>		User Rule 01
User Rule 02	<input type="checkbox"/>		User Rule 02
User Rule 03	<input type="checkbox"/>		User Rule 03
User Rule 04	<input type="checkbox"/>		User Rule 04
User Rule 05	<input type="checkbox"/>		User Rule 05
User Rule 06	<input type="checkbox"/>		User Rule 06
User Rule 07	<input type="checkbox"/>		User Rule 07
User Rule 08	<input type="checkbox"/>		User Rule 08
User Rule 09	<input type="checkbox"/>		User Rule 09
User Rule 10	<input type="checkbox"/>		User Rule 10
User Rule 11	<input type="checkbox"/>		User Rule 11
User Rule 12	<input type="checkbox"/>		User Rule 12
User Rule 13	<input type="checkbox"/>		User Rule 13
User Rule 14	<input type="checkbox"/>		User Rule 14
User Rule 15	<input type="checkbox"/>		User Rule 15
User Rule 16	<input type="checkbox"/>		User Rule 16
User Rule 17	<input type="checkbox"/>		User Rule 17

Name	Override Default	Effective Value	Description
User Rule 18	<input type="checkbox"/>		User Rule 18
User Rule 19	<input type="checkbox"/>		User Rule 19
User Rule 20	<input type="checkbox"/>		User Rule 20

Name	Override Default	Effective Value	Description
FEAT_CS_EXPIRATION	<input type="checkbox"/>	on	Context Store Expiration Feature
VALUE_CS_AEP_CONNECTOR	<input type="checkbox"/>	1000	Context Store AEP Connector
VALUE_CS_API	<input type="checkbox"/>	1000	Context Store API
VALUE_CS_GEO_REDUNDANT_1	<input type="checkbox"/>	10	Context Store GEO Redundancy
VALUE_CS_SERVER	<input type="checkbox"/>	100	Context Store Server
VALUE_CS_USERS	<input type="checkbox"/>	1000	Context Store Users

Figure 5 SMGR Attributes Configuration

5.10.2. Base for URL

- This is the path added when selecting the override format of URL.

- It must be set to a valid url path that parameters can be passed to (if passing parameters).
- The default value shows a demonstration css file provided by Context Store, any users using this css file must create their own css file and use this property to point to its URL.
- Only used when format override is set to URL.
- Can be updated at run-time.

5.10.3. Context Store Rest Version

Set this to be the version of CSRest installed and selected on the cluster for example 3.0.0.0.xxx.

This is a mandatory setting. CSScreenPop will fail to connect to CSRest if this parameter is not set.

5.10.4. CSS for HTML

- This is the path added when selecting the override format of HTML.
- It must point to a valid css path.
- The default value shows a demonstration css file provided by Context Store, any users using this css file must create their own css file and use this property to point to its URL.
- Only used when format override is set to HTML
- Can be updated at run-time

5.10.5. Identifier Delimit Character

- This is a regex value used to delimit a string to select a contextId
- Used in conjunction with 'Default setting for ucid delimit position'
- Example is if 'Default setting for ucid delimit character' is set to '\' then set to 'Default setting for ucid delimit position' 3 to retrieve a contextId of 333333 from 111111.222222.333333.444444

5.10.6. Identifier Delimit Position

- This is the position in decided on by the character above
- Used in conjunction with 'Default setting for ucid delimit character'
- Example is if 'Default setting for ucid delimit character' is set to '\' then set to 'Default setting for ucid delimit position' 3 to retrieve a contextId of 333333 from 111111.222222.333333.444444

5.10.7. Identifier Parsing Position

- Value must be a comma separated list of 2 positive integers where the first number is the start location and the second is the end location.
- Used in conjunction with the Ucid Operation.
- Example:
 - Set value of property to '5,10'
 - Use URL to request the following '?ucid=000001234500000'
 - Will return the context data for the context with the id '12345'
- Can be updated at run-time

JavaScript for HTML

- This is the path added when selecting the override format of HTML.
- It must point to a valid .js file hosted externally to the Context Store cluster.
- Only used when format override is set to HTML. File is added as a link to the script in the HTML page.
- Can be updated at run-time

5.10.8. User Rules 01-20

- This appears 20 times from 01 to 20 and stores the 20 user configurable rules
- The format of these rules is covered in more detail section [5.10 Editable Rules](#)
- All of these rules can be updated at run-time

5.10.9. Example URLs & Their Functions

This section provides some sample usages of Context Store Screen Pop and explains how to use the URL's to do so. It is provided as an example of possibilities and is by no means an exhaustive list.

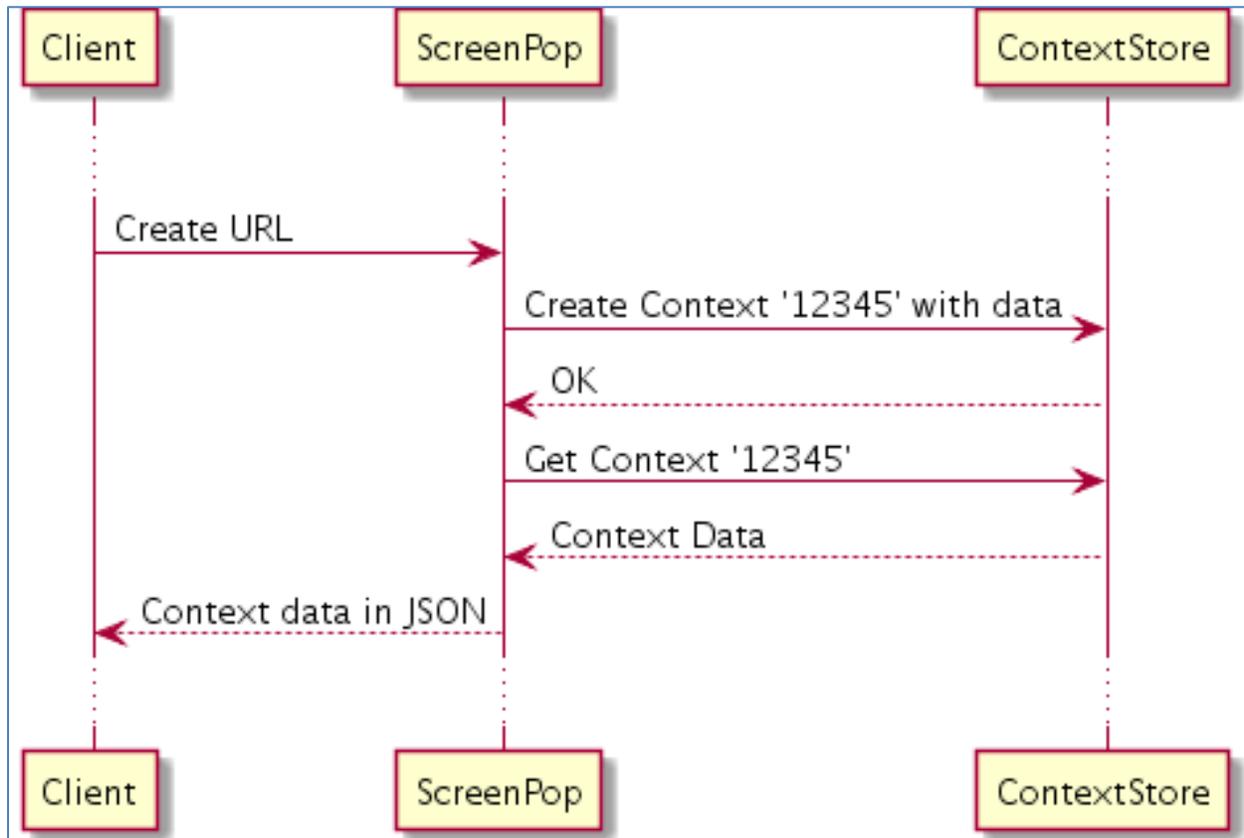
<IP_ADDRESS> Refers to the IP address of the Avaya Aura® Engagement Development Platform cluster (or security module IP of a single-node deployment) that Context Store (including Screen Pop) is running on. See Avaya Aura® Engagement Development Platform documentation for further information on the IP address.

Create a Context with a Single Entry and Return Context

This URL will create a context with an id of '12345' and return the context with the single entry in the default JSON format

- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/create/?id=12345&key1=value1`

Figure 6 ScreenPop Create Context Flow

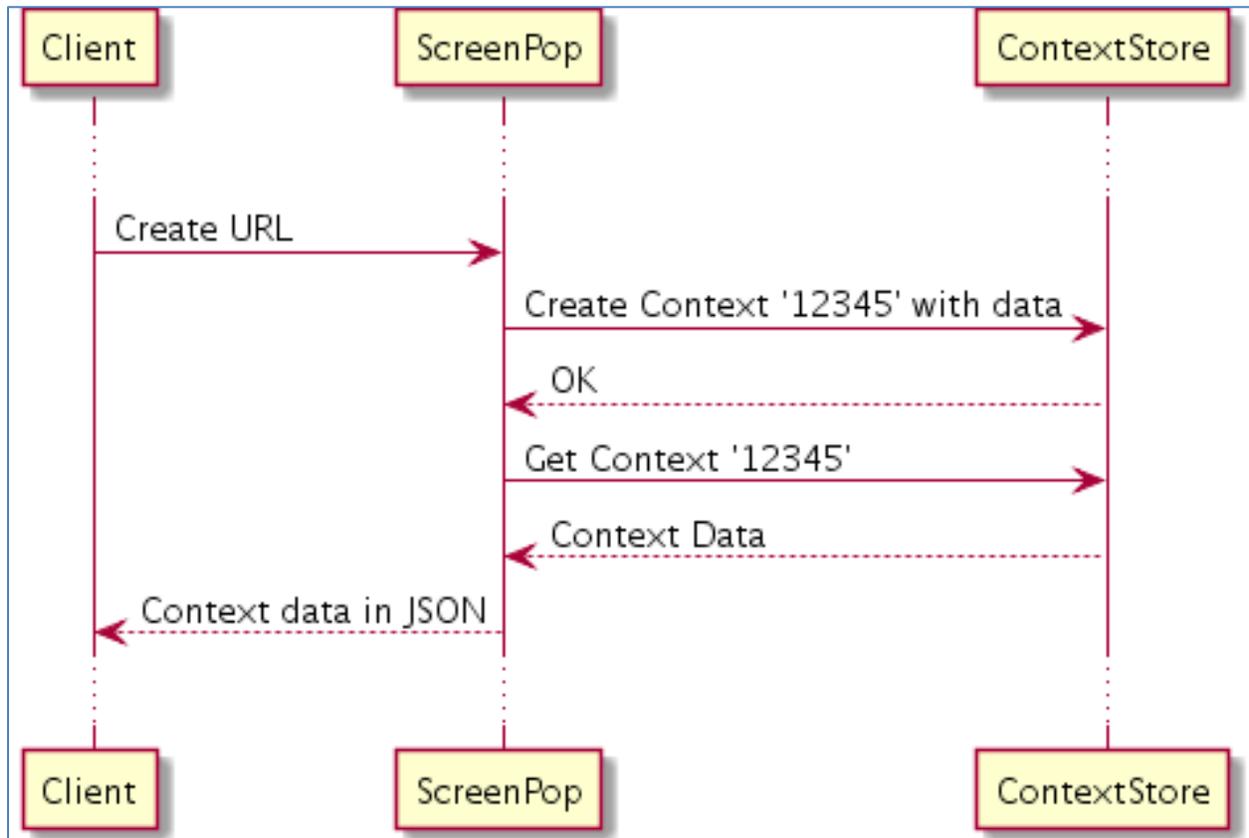


Create a Context with Multiple Entries and Return the Context

This URL will create a context with an id of '12345' and return the context with the entries in the default JSON format

- http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/create/?id=12345&key1=value1&key2=value2&keyN=valueN

Figure 7 ScreenPop Create Context with Multiple Entries Flow



Update a Single Entry in an Existing Context

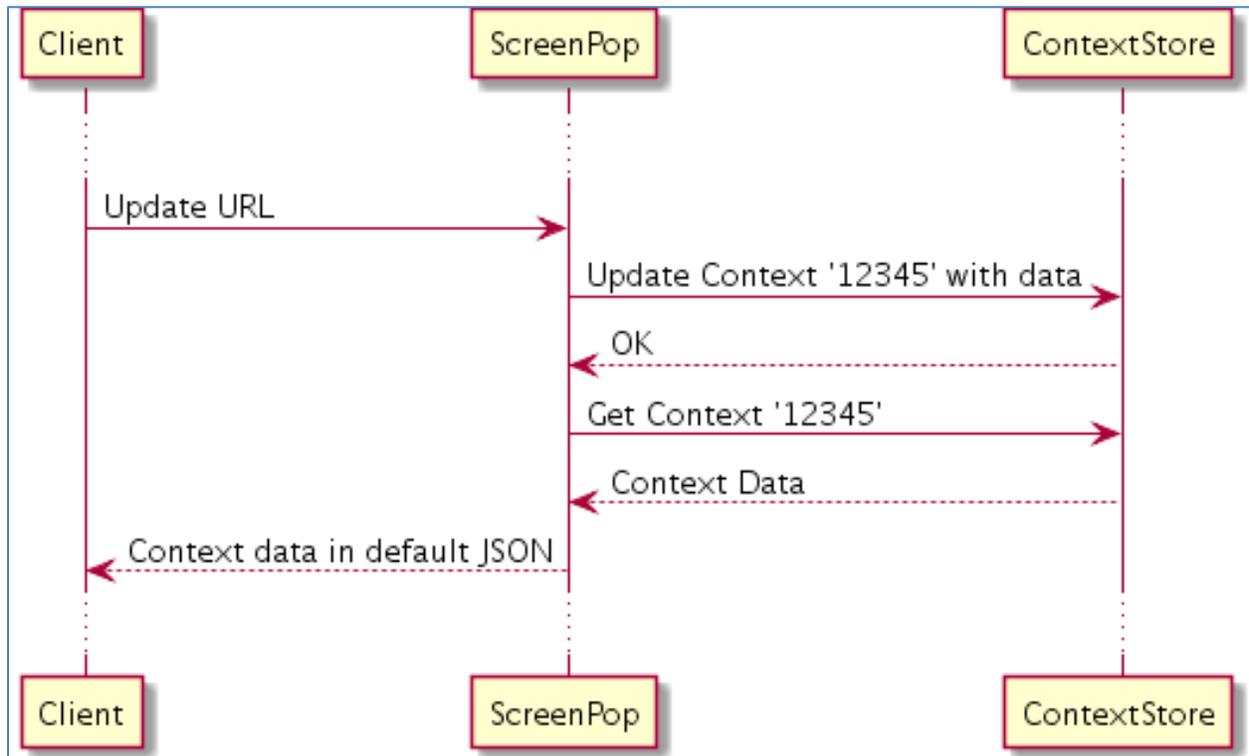
This URL will update an existing context with contextId '12345'. It will either add a new key or update an existing key named 'updateKey' with the value of 'updateValue'

`http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/update/?id=12345&updateKey=updateValue`

The same operation can be requested using an aliasId for the context

`http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/update/?alias=12345&updateKey=updateValue`

Figure 8 ScreenPop Update Single Value Flow

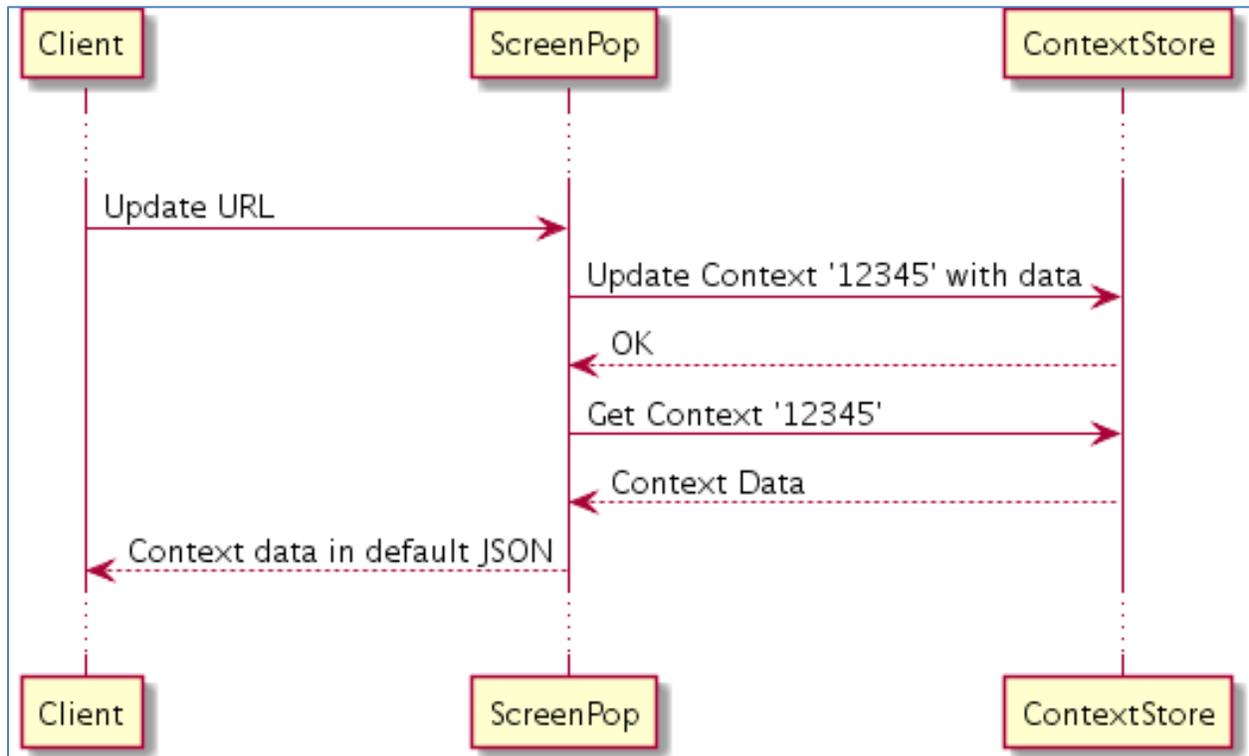


Update Multiple Entries in an Existing Context

This URL will update an existing context '12345'. It will update the keys 'updateKey1' and 'updateKeyN' with the values 'updateValue1' and 'updateValueN'. Should either of the keys exist they will be updated, should either not exist they will be created.

`http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/update/?id=12345&updateKey1=updateValue1&updateKeyN=updateValueN`

Figure 9 ScreenPop Update Multiple Values Flow

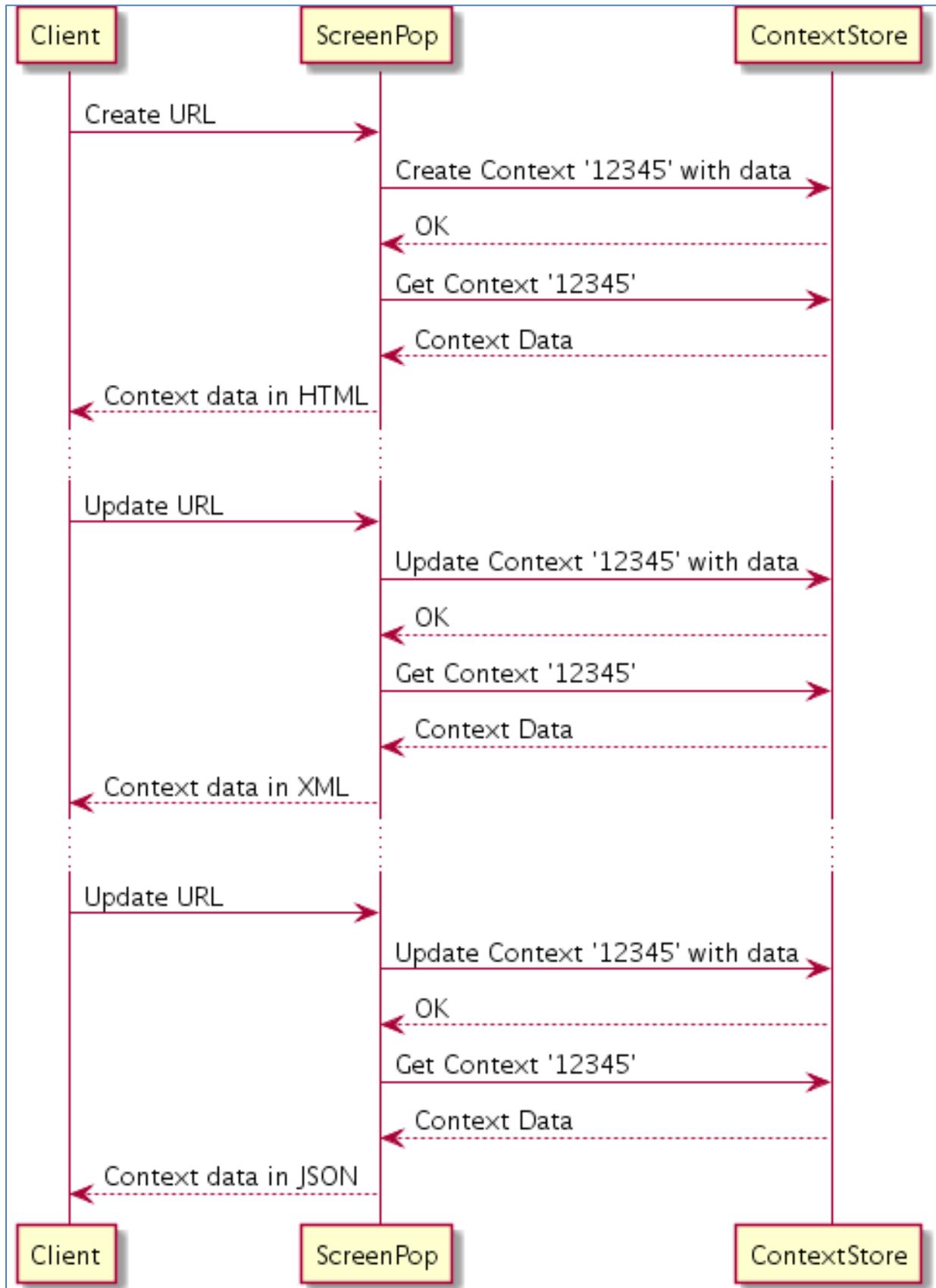


Create a Context with Multiple Entries and Update Multiple Times Changing the Response Format

This URL will create a context with an id of '12345' and return the context with the entries in the formats highlighted in bold below

- http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/create/?id=12345&key1=value1&key2=value2&format=html
- http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/update/?id=12345&key3=value3&keyN=valueN&format=xml
- http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/update/?id=12345&key4=value4&format=json

Figure 10 ScreenPop Create Context and Update for Multiple EntriesFlow



View an Existing Context

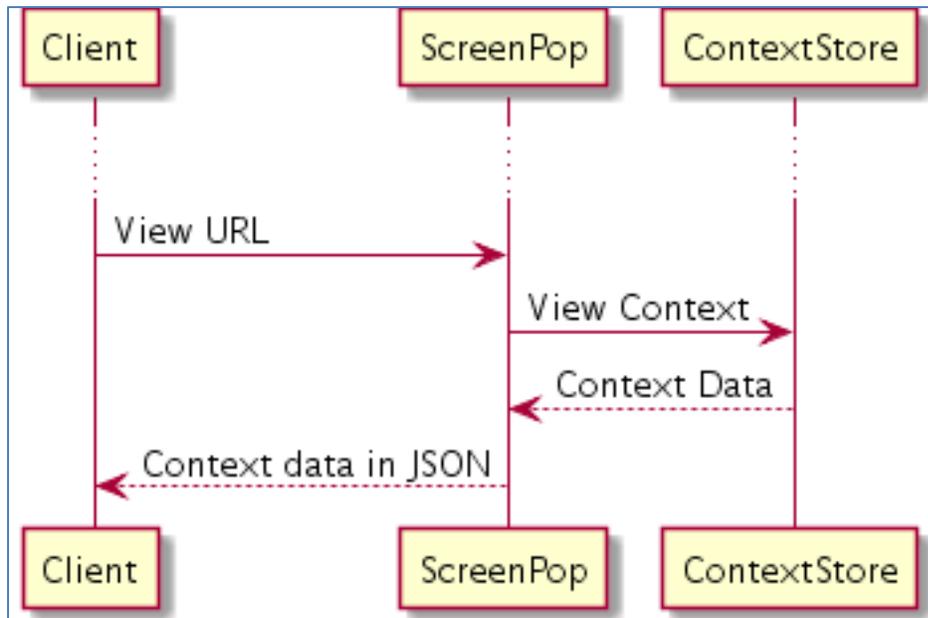
This URL will return a view of a Context with contextId '12345'

- http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/context/?id=12345

This URL will return a view of a Context with an aliasId '12345'

- http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/context/?alias=12345

Figure 11 ScreenPop View Existing Context Flow

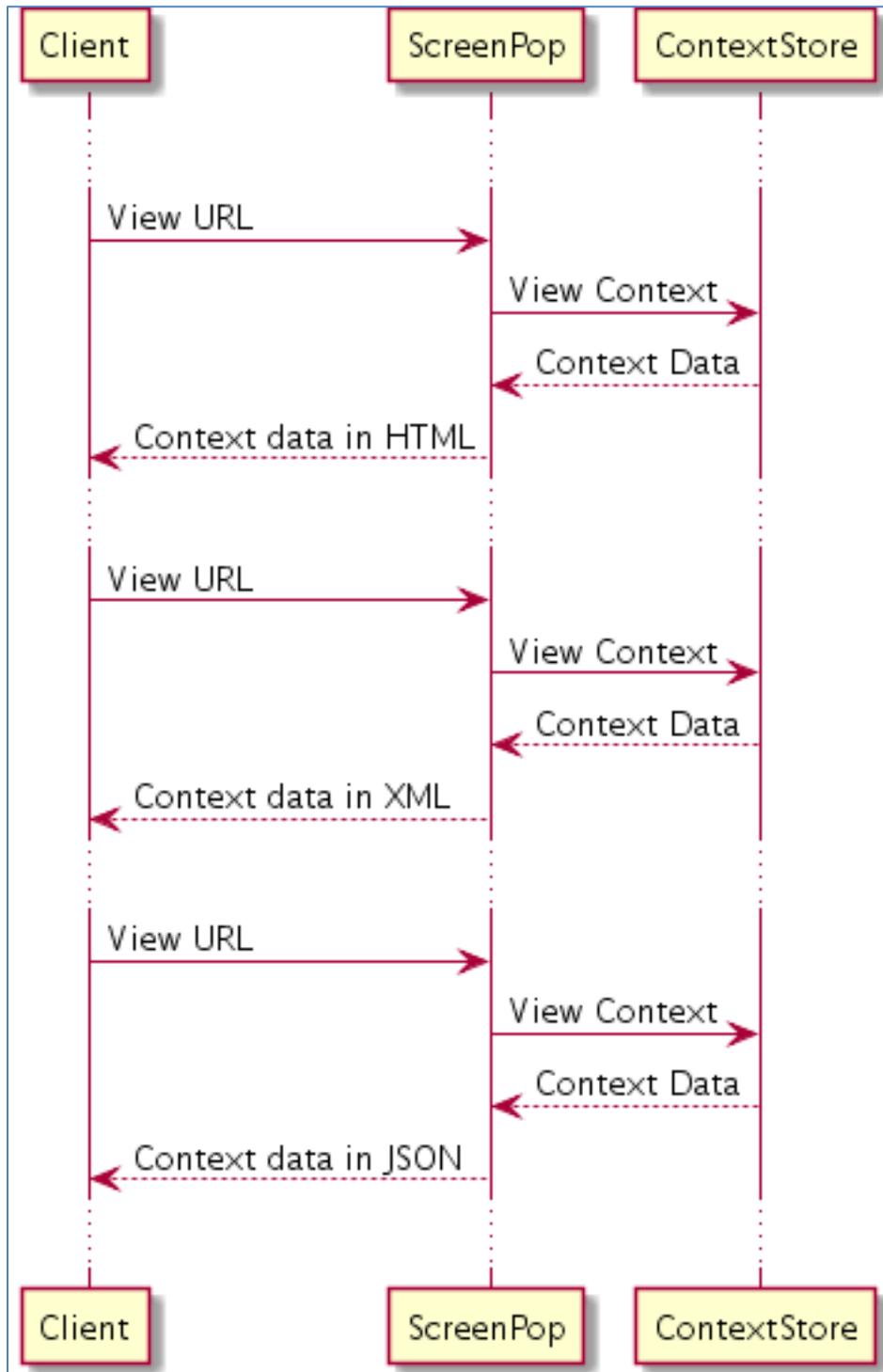


View an Existing Context in HTML/XML/JSON

This URL will return a view of a Context of contextId '12345' in the format highlighted in bold

- http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/context/?id=12345&format=html
- http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/context/?id=12345&format=xml
- http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/context/?id=12345&format=json

Figure 12 ScreenPop View Existing Context different Formats Flow

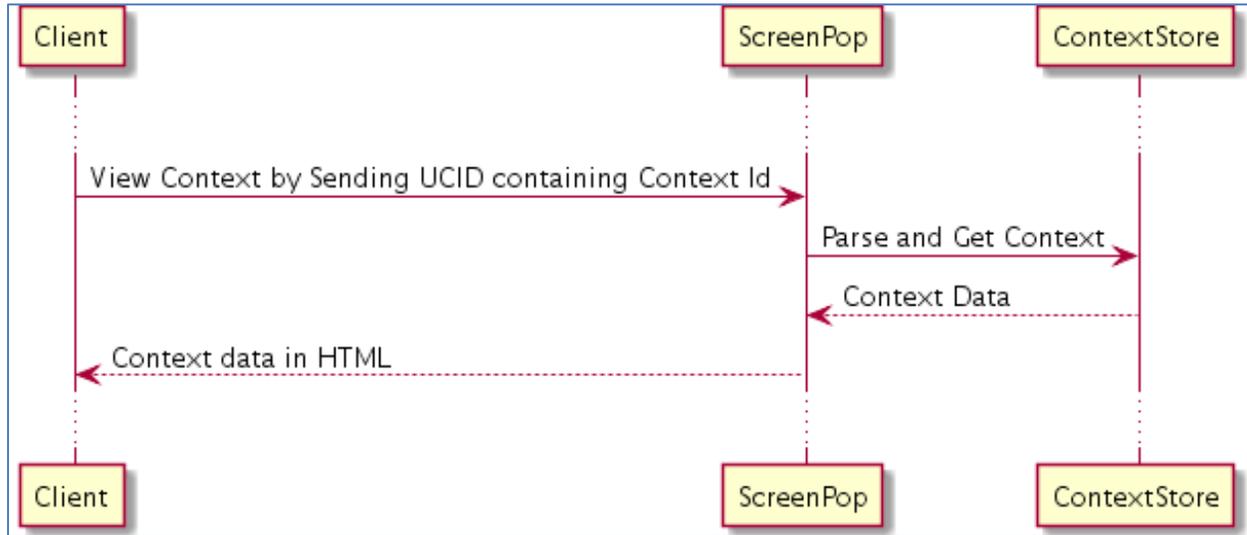


View an Existing Context by Parsing a UCID in HTML

This URL will return a view of a Context of Id '12345' based on parsing a UCID. This assumes the range is set to '5,10'. This is configured by setting the 'Identifier Parsing Position' property; see the section [5.10.7 Identifier Parsing Position](#) for more information.

- `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/ucid?ucid=000001234500000&format=html`

Figure 13 Parsing UCID Flow



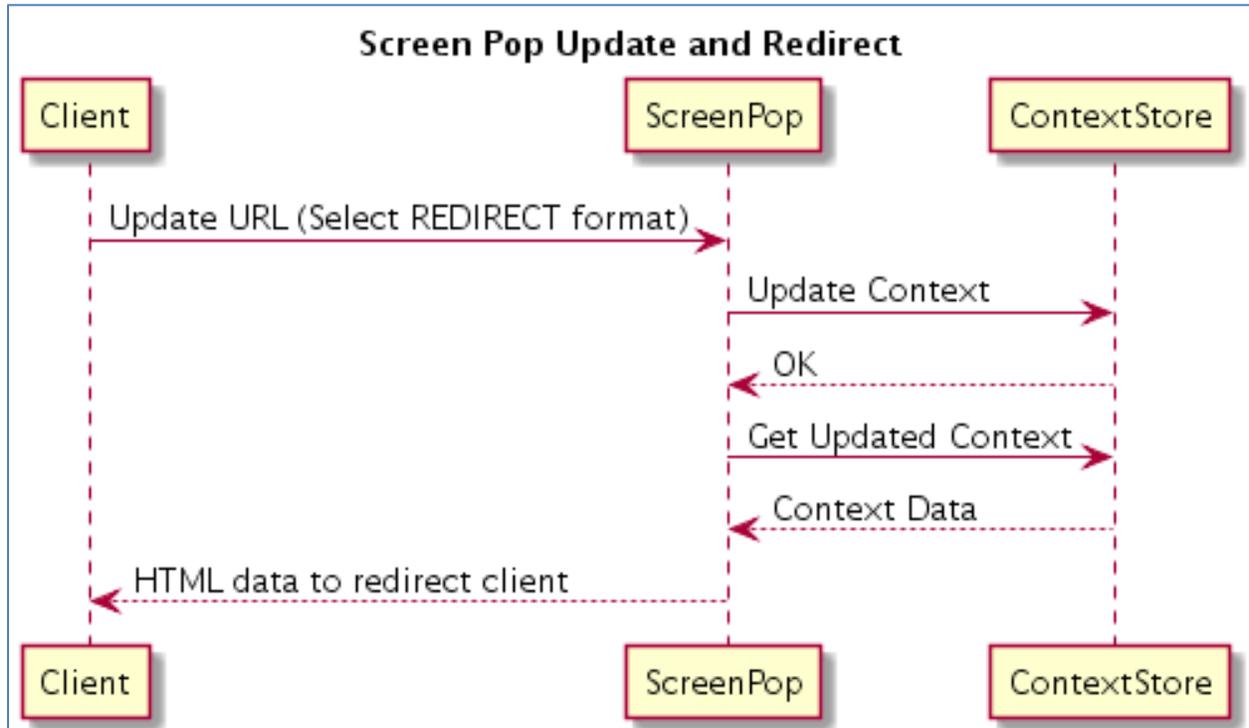
Update a Context and Redirect to an External Web Page Passing Parameters

This URL will update a context '12345' and then redirect to an external web page as configured in the section [5.10.2 Base for URL](#). Only certain parameters (keyToInclude1 and keyToInclude2) from the context will be passed onto the redirecting URL, 'keyToInclude1' and 'keyToInclude2'.

```

http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/update/?id=12345&keyToInclude1=valueToInclude1&keyToInclude2=valueToInclude2&updateKey1=updateValue1&format=redirect&cs_screenpop_include=keyToInclude1|keyToInclude2
  
```

Figure 14 ScreenPop Redirect Flow



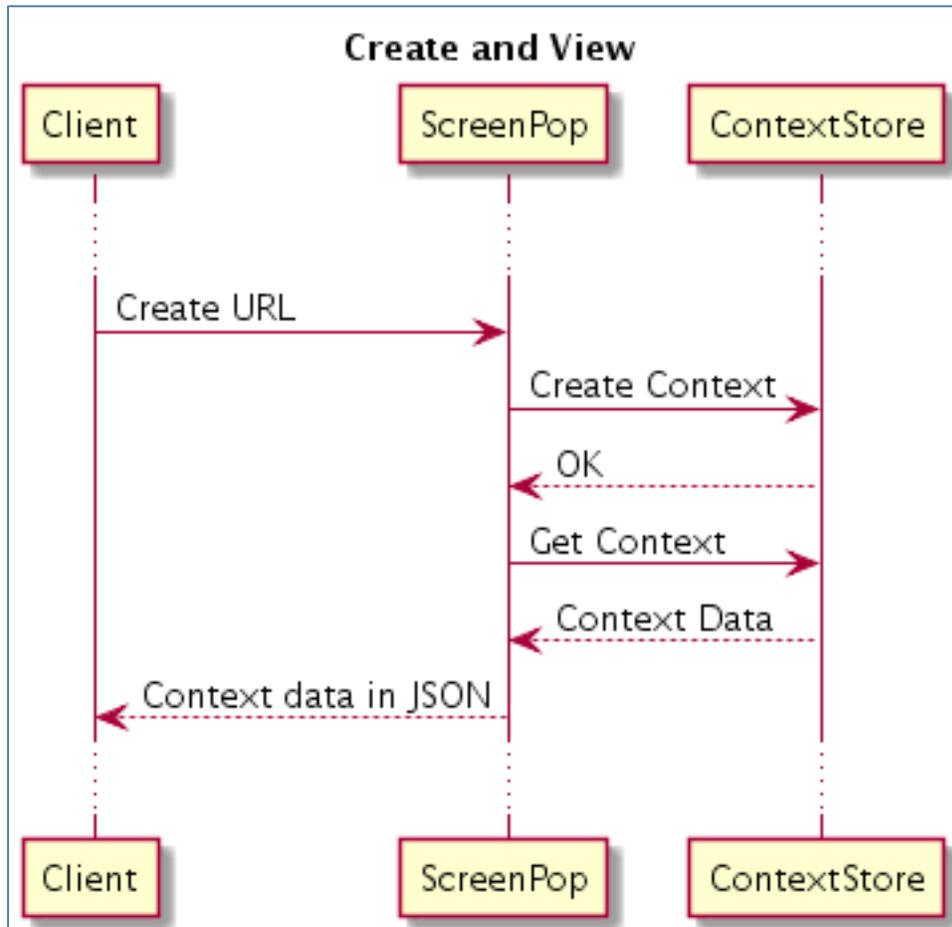
5.11. Sample Usages

This page gives some sample usages for Context Store Screen Pop.

Examples of the URLs referred to are available in the section [5.10.9 Example URLs & Their Functions](#).

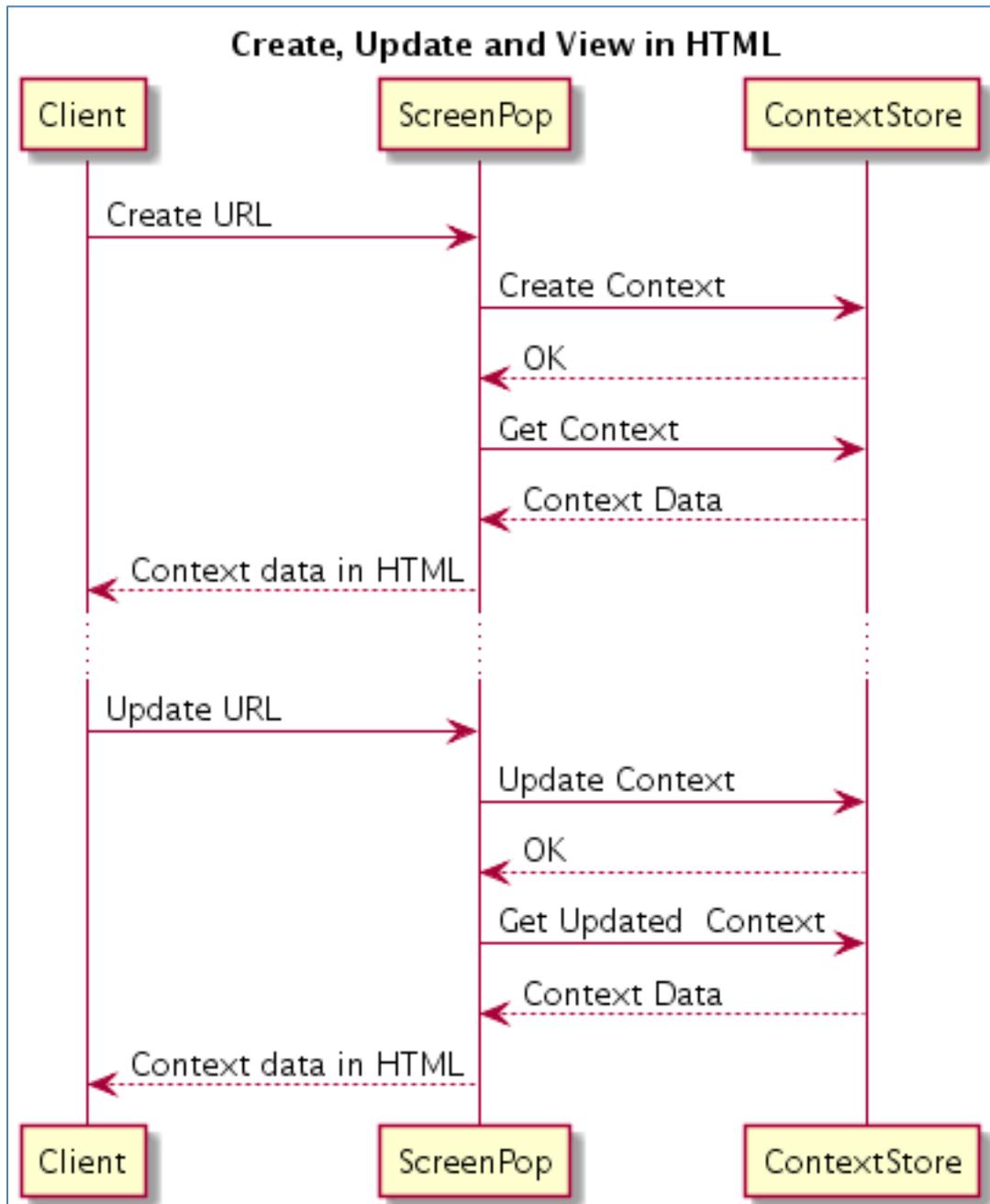
5.11.1. Create and View

Figure 15 ScreenPop Create & View Flow



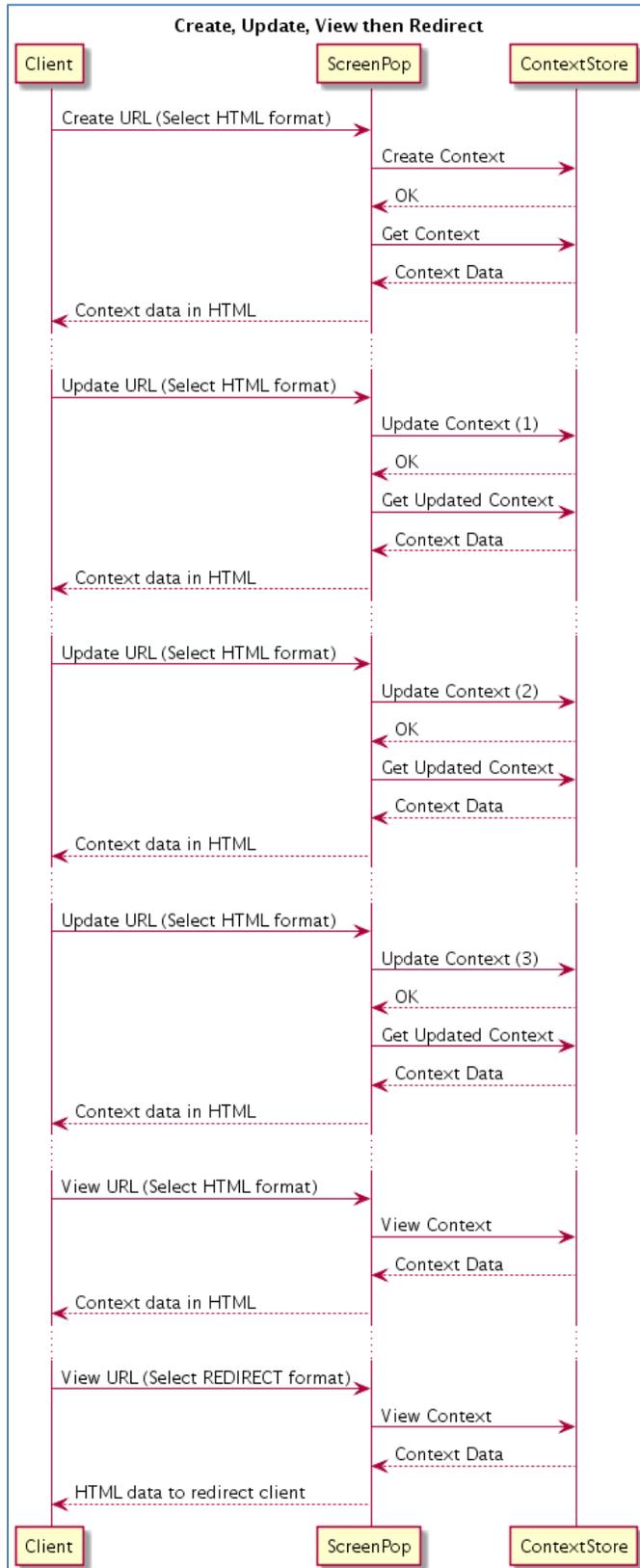
5.11.2. Create, Update and View

Figure 16 ScreenPop Create, Update & View Flow



5.11.3. Create, Update Multiple Times, View then Redirect

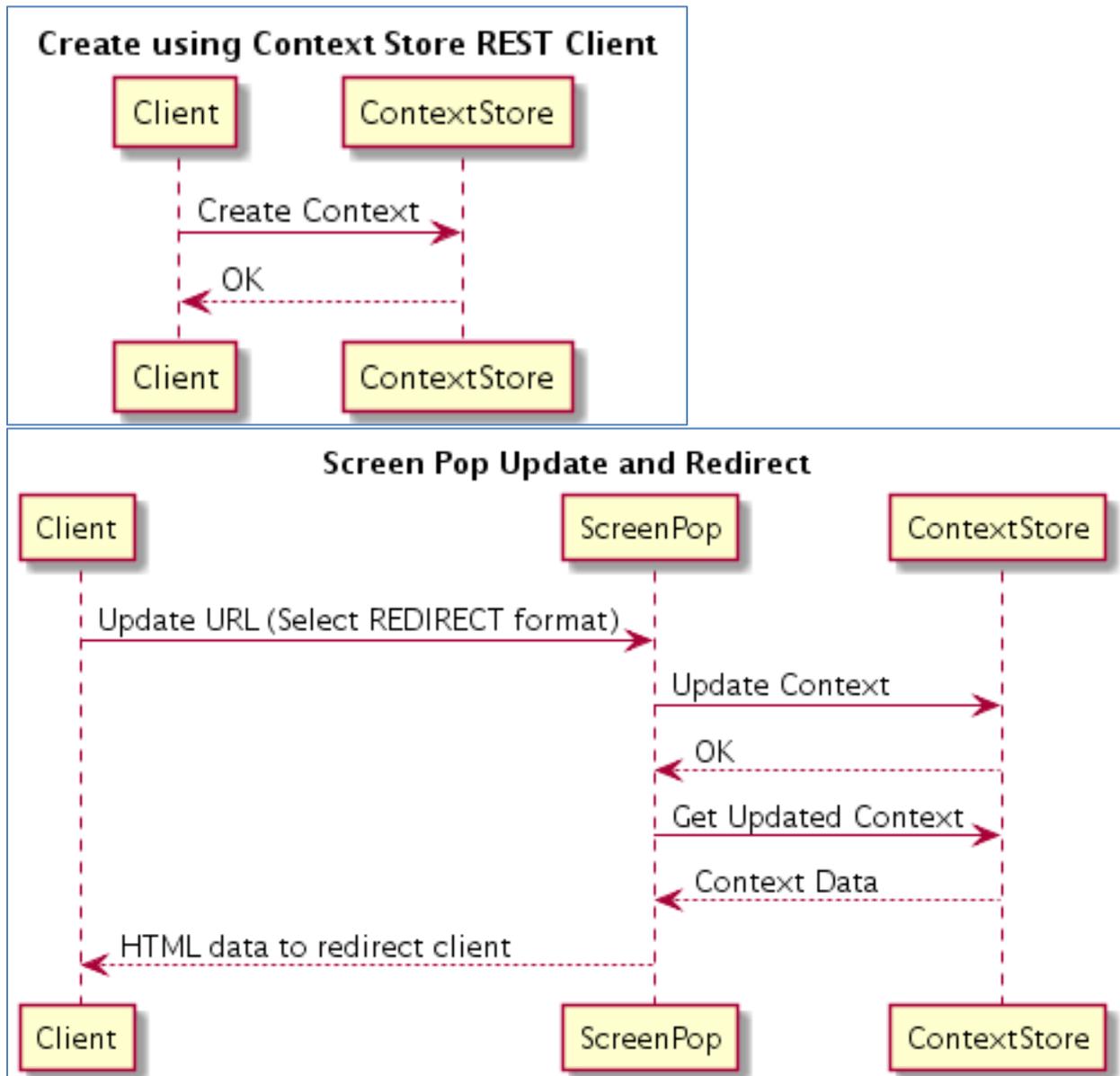
Figure 17 ScreenPop Create, Update & View Flow



5.11.4. Create Context in CSRest, Update and Redirect

Create Context Outside of Screen Pop then Update and Redirect in one step

Figure 18 ScreenPop Create, Update & Redirect Flow



5.12. Example Configuration for Communication Manager

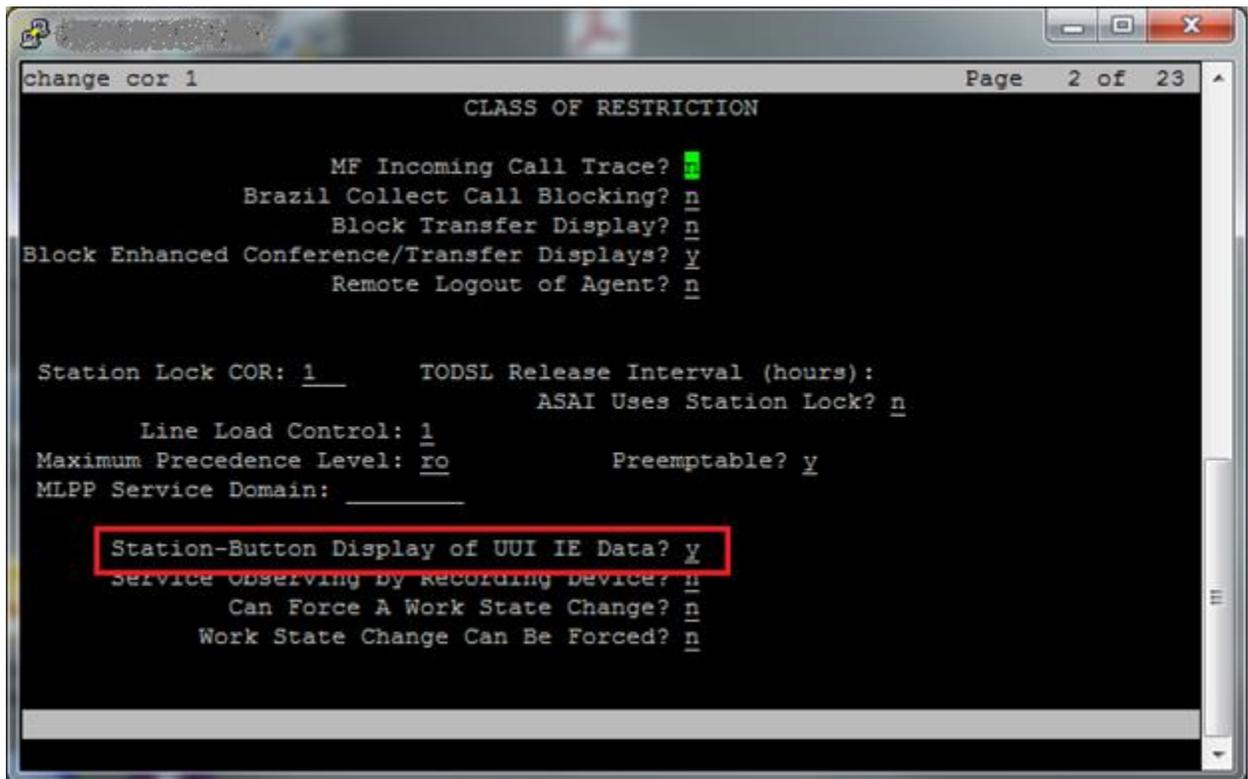
5.12.1. Configure CM to support Screen Pop

For the Screen Pop to function on an agent desktop, the call data sent to the agent's station by CM must include the UserToUser (UUI) field.

This configuration must be done on the station of every agent that needs to use the Screen Pop feature.

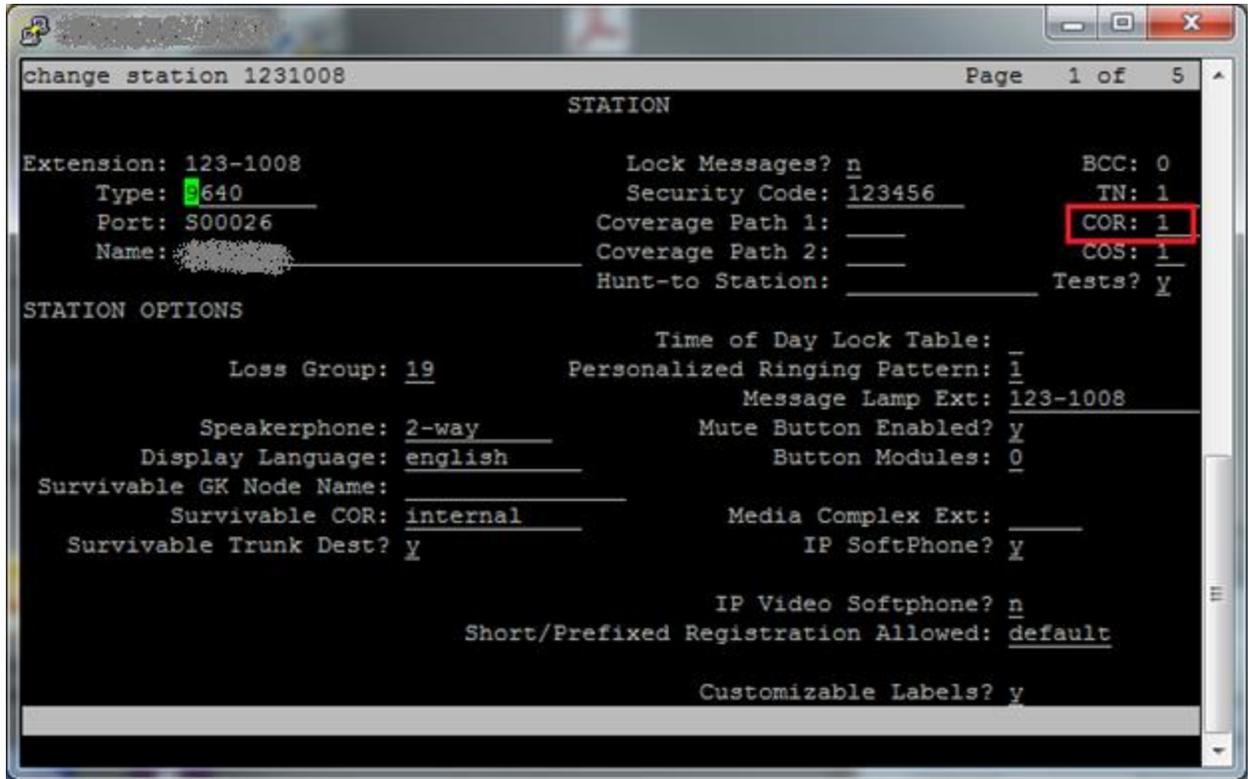
- (1) Update *Class Of Restriction* (COR) configuration to enable '**Station-Button Display of UUI IE Data**' as shown below (set to 'y')

Figure 19 Screen Pop CM Config



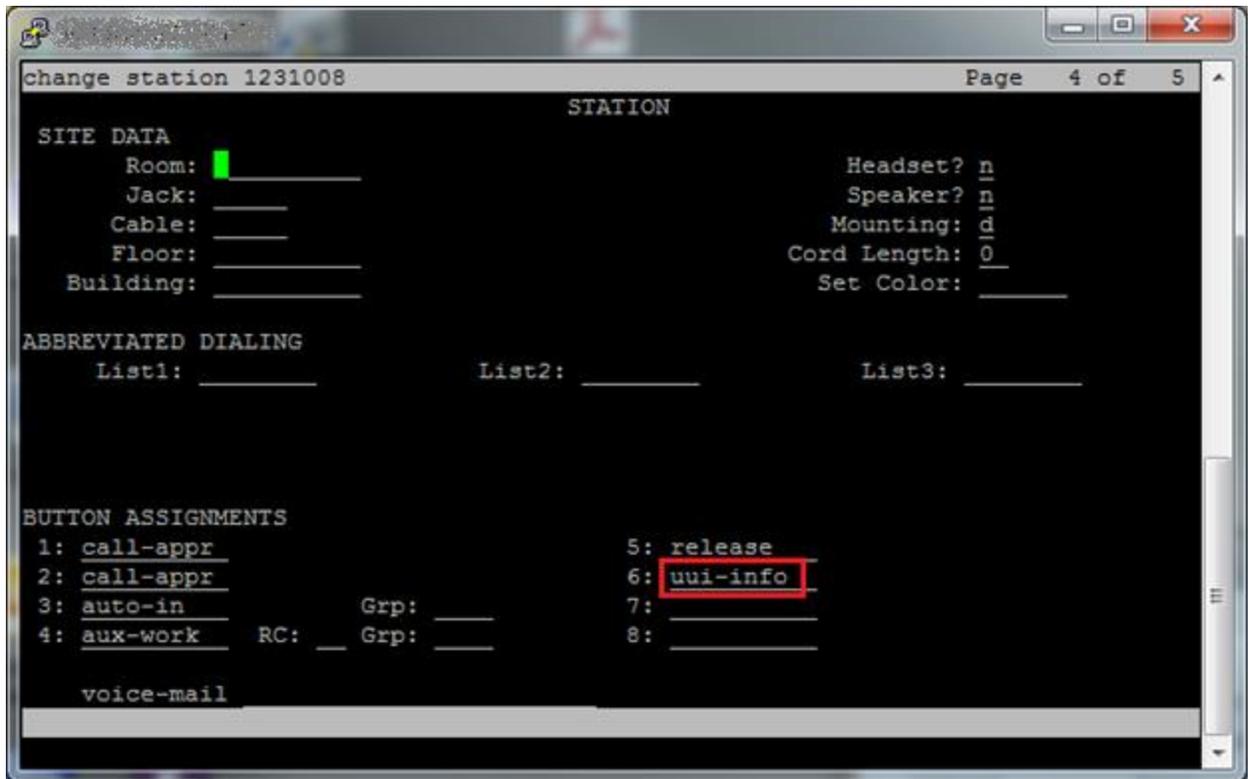
- (1.1) If not known, The *COR* id number can be found in station information as shown in the screenshot below.

Figure 20 ScreenPop CM Config Station



(2) Update the station configuration to include *uui-info* in **BUTTON ASSIGNMENTS** as shown below

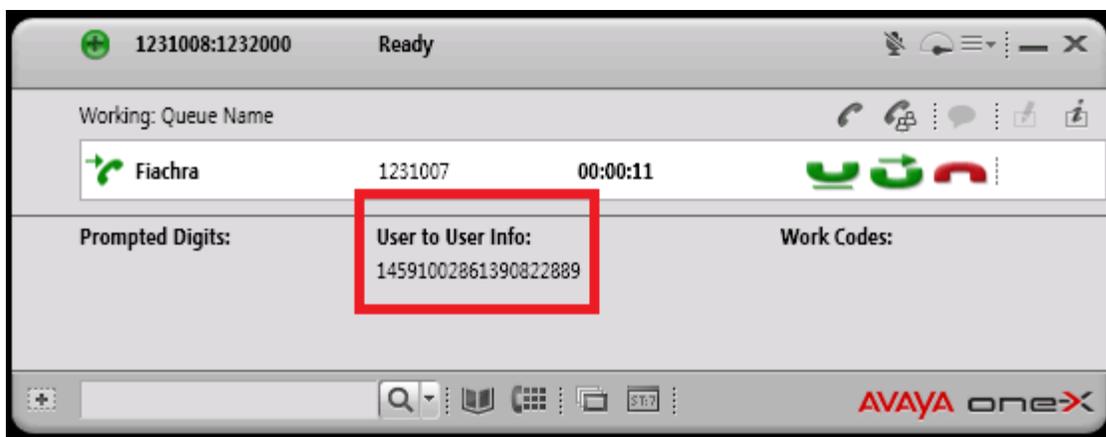
Figure 21 ScreenPop CM Config Station UUI



(3) Disconnect the station (if agent is already logged into agent desktop) and then reconnect the station and login the agent.

(4) Place a test call to the agent and click on the information button ('i' icon) to show call details. The **User to User Info** field should now be populated.

Figure 22 ScreenPop Desktop UI



5.13. Example Configuration for One-X Screen Pop

5.13.1. Configuring Security Certificates for Agent Desktops

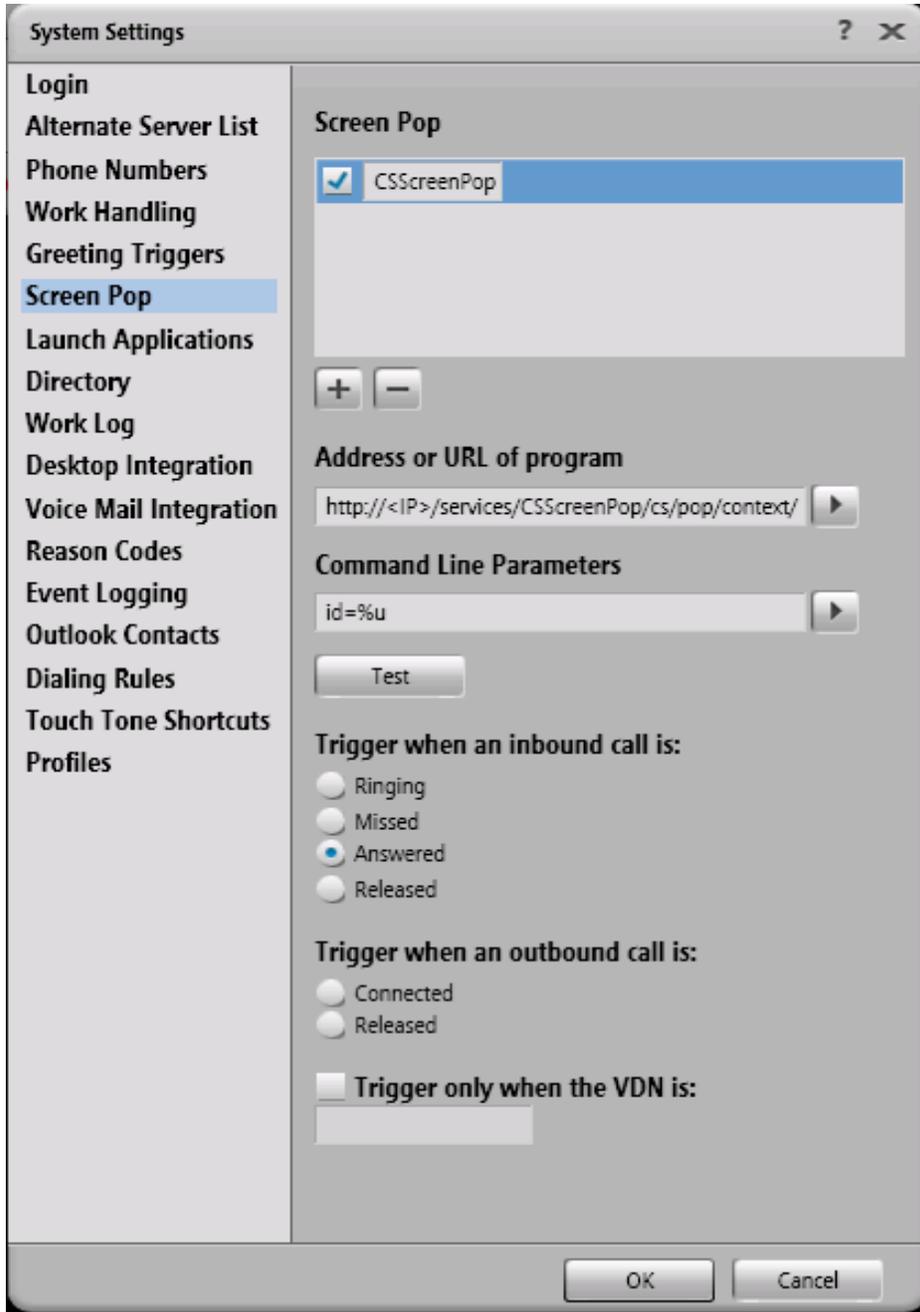
Context Store uses certs to authenticate users of the context store. Before an agent's client can call the screen pop, it must first be configured with the relevant security certs for Context Store.

The section [11.2 Certificate Based Authentication](#) contains details on how to configure certs.

5.13.2. Configuration of the Screen Pop feature in the One-X Agent Desktop client

1. Open the **System Settings** from System **Options** menu in OneX desktop client.
 1. Select **Screen Pop** in the menu bar on the left
 2. Click the + symbol button to create a new screen pop and give it a suitable name.
 3. For '**Address or URL of program**' field, enter the applicable screen pop URL.
 - E.g. `http://<IP_ADDRESS>/services/CSScreenPop/cs/pop/context/` or `https://<IP_ADDRESS>/services/CSScreenPop/cs/pop/context/`
2. For **Command Line Parameters** of this screen pop, the call's UUI field is used. The applicable identifier must be populated in the UUI filed by the call flow application.
 - Example using contextId: `id=%u`
 - Example using aliasId: `alias=%u`
3. For **Trigger when an inbound is:** configuration, select the **Answered** radio button.

Figure 23 ScreenPop Desktop Screen Pop



6. CRM Integration

6.1. Overview

This new feature available as a Context Store snap-in provides the ability to trigger an Engagement Designer Workflow thus giving the ability to integrate with CRM systems through the ED Workflow.

A rules engine is provided to determine whether or not the workflow should be triggered

6.1.1. Features

- Provided as a Context Store Snap-in and loaded using standard SMGR SVAR loading mechanism
- User defined business rules provide the ability to determine if a rule should be triggered and which EDP event it should raise
- Rules can be triggered using 2 Context Store REST API Method calls provided the parameter 'rules=true' is added to the URL
- Create Context (POST)
- Update Context (PUT)
- 5 independent workflows can be triggered based on the user defined rules

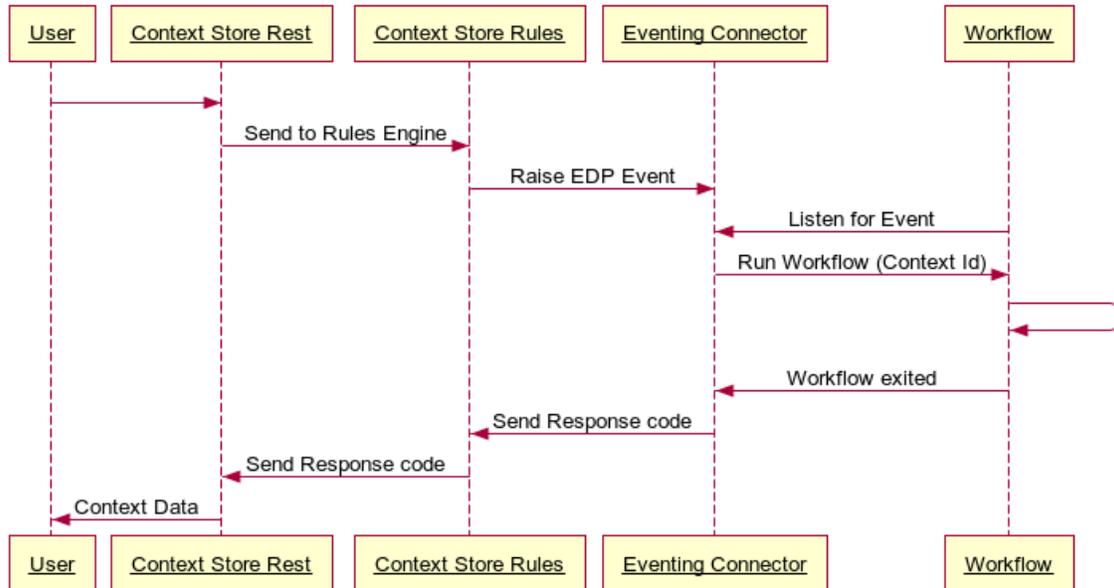
6.1.2. Caveats

- Requires Context Store and Engagement Designer to be installed on the same SMGR
- This functionality is not dependent on any OSGi task type nor any Context Store blocks as it is designed to raise an EDP event which will in turn cause any flow to start
- Once the workflow has started and passed in the Context Id then this piece of functionality is complete, the completion of the workflow is dependent on the ED itself.

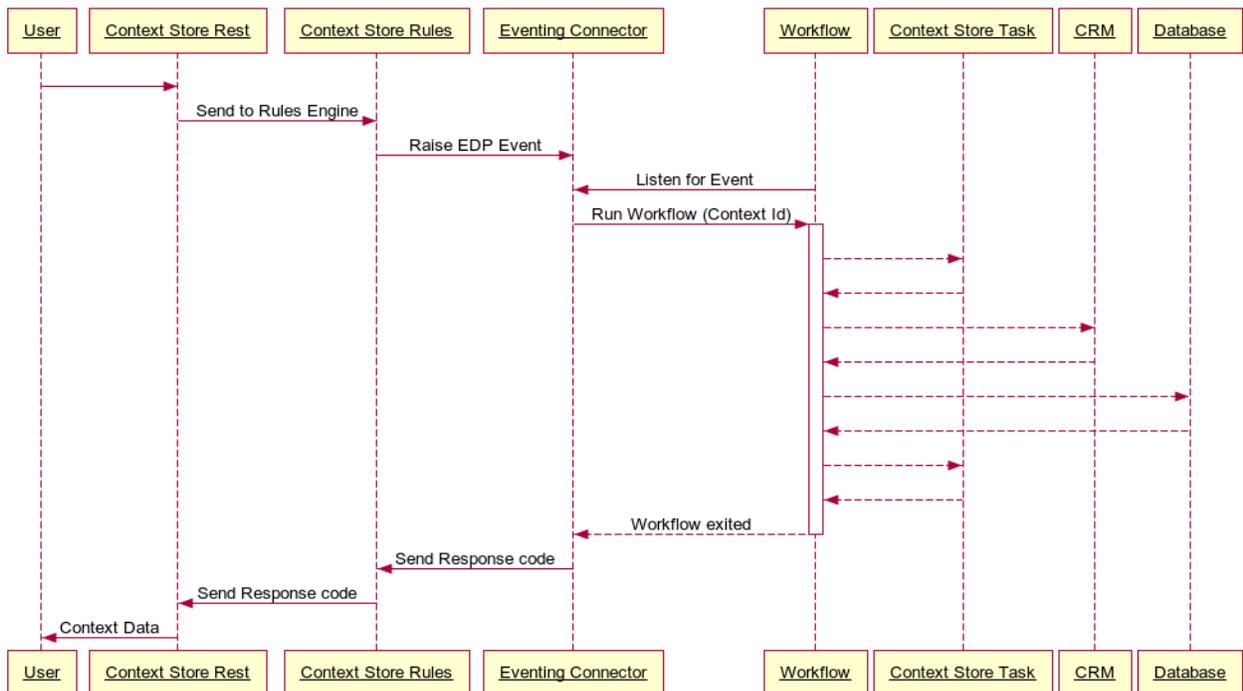
6.2. High Level Design

Below is a sequence diagram of how the Context Store Rules integrates between Context Store and an EDP workflow

6.2.1. Basic Flow



6.2.2. Example CRM & CS Integration



6.3. Configuration

There are 4 steps required to use this feature

1. Create an EDP event unique to the workflow being created
2. Create an ED Workflow that listens for the event
3. Configure the rules in the SMGR attributes for CSRules

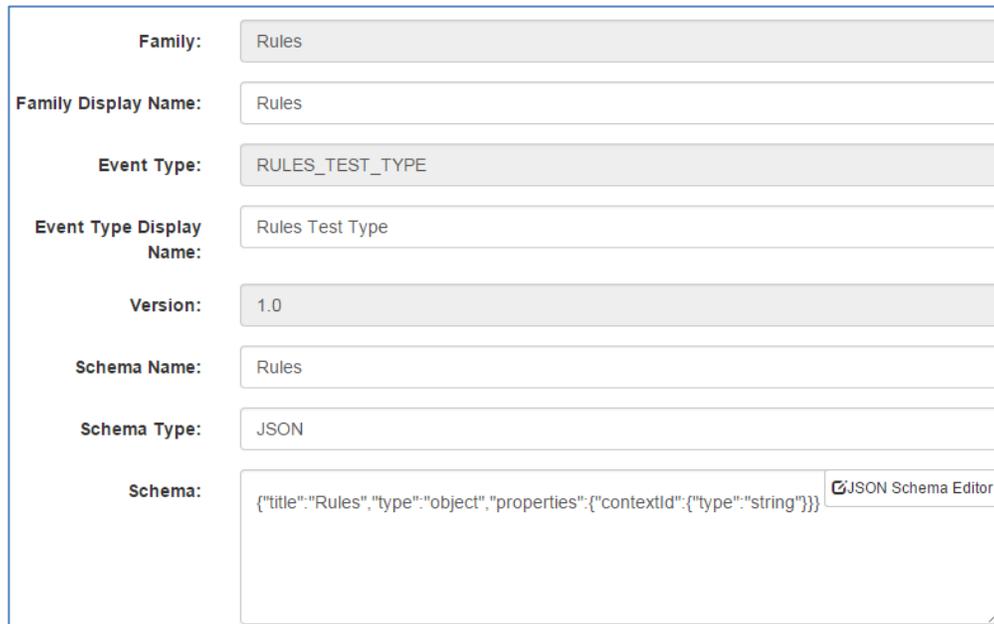
4. Create or Update the context with the rules=true parameter set.

6.3.1. Create EDP Event

This is designed as a quick start guide, for full details consult the ED and EDP documentation.

In ED/EDP 3.1 you must use the admin console in ED; in older versions this step was done in the Element Manager

Create an event as follows



The screenshot shows a configuration form for creating an EDP event. The fields are as follows:

Family:	Rules
Family Display Name:	Rules
Event Type:	RULES_TEST_TYPE
Event Type Display Name:	Rules Test Type
Version:	1.0
Schema Name:	Rules
Schema Type:	JSON
Schema:	<pre>{ "title": "Rules", "type": "object", "properties": { "contextId": { "type": "string" } } }</pre> JSON Schema Editor

- **Family**
 - This is to identify the family of the event being created, convention is that this is in camel case.
 - This example uses 'Rules'
- **Type**
 - This is to identify the type of the event being created, convention is that this is all upper case and is identical to the **Schema Name**.
 - This example uses 'RULES_TEST_TYPE'
- **Version**
 - This is to identify the version of the event being created.
 - This example uses '1.0'
- **Schema Name**
 - This is to identify the schema name of the event being created, convention is that this is all upper case and is identical to the **Type**.
 - This example uses 'Rules'
- **Schema Type**
 - Must be set to 'JSON'
- **Schema**

- Must be exactly as below, changing '**Family Name**' to match Type and **Schema Name**.
- Creating the schema in ED is best done via the 'JSON Schema Editor'
- Set the title to 'Rules'
- Add a String and set the name to 'contextStore' (please note the case)

Schema

```
{
  "title": "Rules",
  "type": "object",
  "properties": {
    "contextId": {
      "type": "string"
    }
  }
}
```

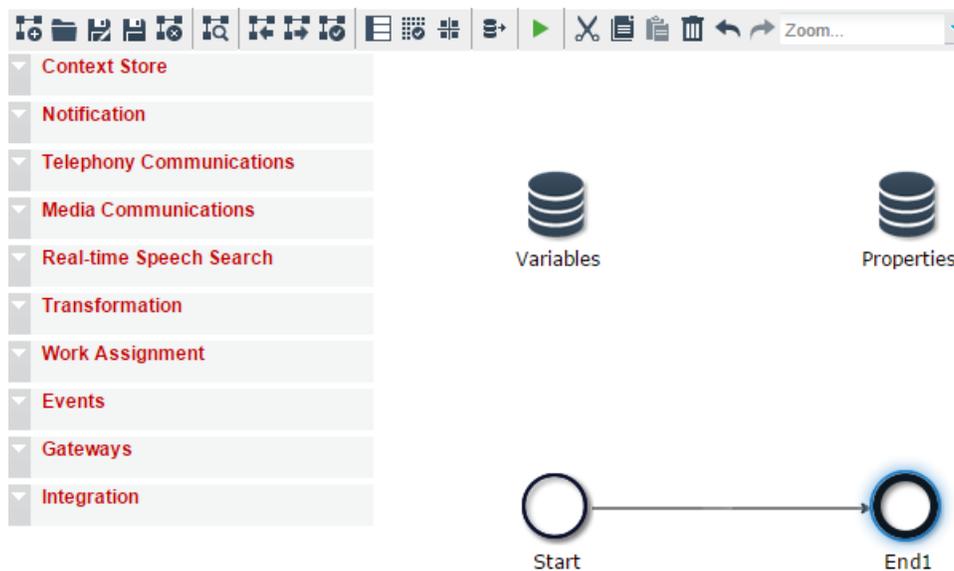
6.3.2. Create Workflow

This step requires existing knowledge of how to create an Engagement Designer workflow and is intended as a quick start guide to get the minimum possible workflow that can be used to highlight this feature configured.

For full details consult the ED and EDP documentation.

It is not intended as a working solution but is provided as a starting point for full integrated solutions.

- In ED the minimum valid workflow is looks as follows



- The start block is configured as follows

- The output mapping of the start block is as follows



- The example workflow can be imported by downloading the sample workflow, `MinimumWorkflow.xml`, from the [Avaya Context Store DevConnect](#) site and importing it into Engagement Designer

6.3.3. Configure Rule

- There are 9 items that need to be configured for each rule to work, 4 relate to the Eventing Connector where the workflow is installed and 5 relate to the configuration of the rule itself
 - Eventing Connector Attributes
 - Eventing Connector Family
 - This should match the Family value entered when creating the event
 - Eventing Connector Type
 - This should match the Type value entered when creating the event
 - Eventing Connector URL
 - This is the HTTP location of the Eventing Connector where the ED Workflow is installed

- HTTP only
- Eventing Connector Version
 - This should match the Family value entered when creating the event
- Rule Attributes
 - Eventing Rule Priority
 - Number between 1 and 5
 - Only one rule can fire and the priority will determine which rule will fire when the context matches multiple rules
 - Name
 - Alphanumeric string indicating the unique name a user wishes to assign to a rule
 - Equation
 - String value representing the key in the context the rules engine will match against
 - Can only be set to '=='
 - Example:
 - KeyToMatchAgainst == "ValueToMatchAgainst"
 - Compares value to the string "ValueToMatchAgainst"
 - Double quotation marks are required for string comparison
 - KeyToMatchAgainst == 2
 - Compares value to the numeric 2
 - No quotation marks for numeric comparison
 - KeyToMatchAgainst == "2"
 - Compares value to the string "2"
 - Double quotation marks are required for string comparison

Name	Override Default	Effective Value	Description
Rule 01: Eventing Connector Family	<input checked="" type="checkbox"/>	Rules	Rules Eventing Connector Family for rule 01
Rule 01: Eventing Connector Type	<input checked="" type="checkbox"/>	RULES_TEST_TYPE	Rules Eventing Connector Type for rule 01
Rule 01: Eventing Connector URL	<input checked="" type="checkbox"/>	http://<CLUSTER_IP>/services/EventingConnector/events	Rules Eventing Connector URL for rule 01
Rule 01: Eventing Connector Version	<input checked="" type="checkbox"/>	1.0	Rules Eventing Connector Version for rule 01
Rule 01: Rule Equation	<input checked="" type="checkbox"/>	KeyToMatchAgainst == "ValueOfKeyToMatchAgainst"	Enter equation for rule 01
Rule 01: Rule Name	<input checked="" type="checkbox"/>	UniqueRuleName	Enter name for rule 01
Rule 01: Rule Priority	<input checked="" type="checkbox"/>	5	Enter priority for rule 01

6.3.4. Firing a Rule

- 2 Context Store Rest operations will trigger a rule
 - Create Context
 - Update Context

- The attribute 'rules=true' must be added to the URL to send the data to the rules engine, after this the engine will assess which (if any) rule will be fired and event raised
 - This attribute can ONLY be set if the CSRules snap-in is installed
- Example Request
 - Create context (POST)

```
http://<IP_ADDRESS>/services/CSRest/cs/contexts/?rules=true
```

```
{
  "contextId": "<CONTEXT_ID>",
  "data": {
    "KeyToMatchAgainst": "ValueOfKeyToMatchAgainst"
  }
}
```

Context Store Rest Response

The following additional entry will appear in the CSRest log files to indicate that the create or update request will send to the rules engine

```
2015-06-30 15:00:31,317 [WebContainer : 1] util.ServiceHelper INFO -
[M:processPostContext][T:null]. Sending context with id <CONTEXT_ID> to rules engine
```

Eventing Connector Response

This is the expected response seen in the Eventing connector's logs. Note that full logging must be enabled to see these log statements

Thing to note

- doPost body : {"contextId":"<CONTEXT_ID>"}
- This shows the context id sent in the event
- PublicationImpl [family=Rules, type=RULES_TEST_TYPE, eventVersion=1.0]
- This shows the family, the type and the version of the event raised

Sample log output

```
2015-06-30 14:18:29,308 [WebContainer : 2] EventingConnector FINEST - EventingConnector-
3.1.0.0.43009 - doPost ENTER
2015-06-30 14:18:29,309 [WebContainer : 2] EventingConnector FINEST - EventingConnector-
3.1.0.0.43009 - populateEventMetadata metadata: EventMetaDataImpl [user=null, userAsMatched=null,
serviceProfile=null, correlationId=null, producerName=null, producerVersion=null, valueMap={},
isImmutable=false]
```

```
2015-06-30 14:18:29,309 [WebContainer : 2] EventingConnector FINER - EventingConnector-3.1.0.0.43009 - getCall: interactionId: null
2015-06-30 14:18:29,309 [WebContainer : 2] EventingConnector FINEST - EventingConnector-3.1.0.0.43009 - doPost: It is not NOT_HOST_INTERACTION
2015-06-30 14:18:29,309 [WebContainer : 2] EventingConnector FINEST - EventingConnector-3.1.0.0.43009 - doPost body : {"contextId":"DemoContextRules06"}
2015-06-30 14:18:29,309 [WebContainer : 2] EventingConnector FINER - EventingConnector-3.1.0.0.43009 - createEventProducer ENTER eventBody.length=34
2015-06-30 14:18:29,309 [WebContainer : 2] EventingConnector FINEST - EventingConnector-3.1.0.0.43009 - populateUserHandleAndDomain user=null
2015-06-30 14:18:29,309 [WebContainer : 2] EventingConnector FINER - EventingConnector-3.1.0.0.43009 - publish ENTER publication=PublicationImpl [family=Rules, type=RULES_TEST_TYPE, eventMetaData=EventMetaDataImpl [user=null, userAsMatched=null, serviceProfile=null, correlationId=null, producerName=null, producerVersion=null, valueMap={}, isImmutable=true], effectiveUser=null, userMinusDefaultDomain=null, userHandle=null, userDomain=null, eventBody=<not shown>, eventVersion=1.0, actualProducerName=EventingConnector, actualProducerVersion=3.1.0.0.43009, publicationId=c3-EventingConnect-3.1.0.0.43009-23aa5b68-da5d-48bf-b2c6-ed9df368b6be, publicationTimestamp=1435670309309, metaDataBitSet=CeBitSet [val=0x2], valueCount=0]
2015-06-30 14:18:29,310 [WebContainer : 2] EventingConnector FINER - EventingConnector-3.1.0.0.43009 - setInteractionResponseBody Response status is set to 200
```

Engagement Designer Response

The following log message indicates the workflow has intercepted the event raised by the rules engine

- Enabling full logging will show the payload delivered

Sample log output

```
2015-06-30 14:18:29,312 [pool-121-thread-1] EngagementDesigner INFO - EngagementDesigner-3.1.0.0.05004 - Received event: Rules:RULES_TEST_TYPE Payload: EventImpl [family=Rules, type=RULES_TEST_TYPE, payload=<not shown>, version=1.0, publicationId=c3-EventingConnect-3.1.0.0.43009-23aa5b68-da5d-48bf-b2c6-ed9df368b6be, subscriptionId=EngagementDesig-3.1.0.0.05004-2731cd87a8db3953d88f5d1621aa8bfb29d3d34a3624829cb5f274450c049a08, consumerName=JMGTTest01, consumerVersion=3, metadata=EventMetaDataImpl [user=null, userAsMatched=null, serviceProfile=null, correlationId=null, producerName=null, producerVersion=null, valueMap={}, isImmutable=true], consumerPrivateData=null, style=ASYN, publicationTimestamp=1435670309309]
```

7. Context Store PDC

7.1. Overview

The Context Store PDC is a plugin for Eclipse/Orchestration Designer that facilitates the CRUD operations that can be run against Context Store from a flow that runs in the Avaya Aura® Experience Portal environment. In this way it acts as an interface to the Context Store.

7.2. Installation Prerequisites of the Context Store PDC in Eclipse/Orchestration Designer

This section explains how to install the Context Store PDC plugin in to Avaya Orchestration Designer (Orchestration Designer) and how to run the sample call flow projects (which are provided through [DevConnect](#)) that uses the Context Store PDC to interact with Context Store. The sample applications demonstrate basic flows which show how to add, retrieve and mutate information context information. A number of applications need to be installed and configured before this is possible.

7.2.1. Software requirements

- Orchestration Designer 6.0, 7.0 or 7.0.1
- Apache Tomcat 6.0 or 7.0
- Latest Context Store PDC.
- A sample application that uses the Context Store PDC in its call flow.

7.2.2. Prerequisites

- Download a supported GA build of Orchestration Designer e.g. OD 7.0.
 - The link to the OD 7.0 ISO is [here](#). Note this file is 1.9 GB.
 - For reference, the General Orchestration Designer DevConnect site is located [here](#).
- Note a **32** bit JRE is needed to run the Eclipse version supplied with Orchestration Designer.
- Extract the ISO to a directory.
- Follow the instructions in the "Installing Orchestration Designer using pre-packaged installation" section (Chapter Two) from the AAOD_GettingStarted.pdf guide. This file will be located in the directory AAOD_7_0\AAOD7.0\eclipse which was created when the ISO was extracted. The steps to configure tomcat in Eclipse do not need to be run at this point.
- Download the appropriate version of [Apache Tomcat](#) for the system that you will be running Orchestration Designer on. Unzip in to a directory and take note of same.
- Also download the latest Context Store PDC jar to the machine that will be running Orchestration Designer. This can be downloaded from the [Avaya Context Store DevConnect](#) site.
- If DNS is not set up on the machine running Orchestration Designer, an entry should be inserted in the hosts file for the Rest Interface. This is the same IP as used in the Context Store PDC configuration screen.
- Note only one Context Store cluster can be configured per application. This means that each Context Store node in a flow has to use the same Context Store cluster. Each application can have a different Context Store cluster configured for it.

7.3. Installing/Upgrading the Context Store PDC plugin

The instructions below should be followed exactly and especially with regards the starting and stopping of Orchestration Designer.

1. The speech perspective needs to be open in order to configure the Context Store PDC.

Select **Window > Open Perspective > Speech** in Orchestration Designer.

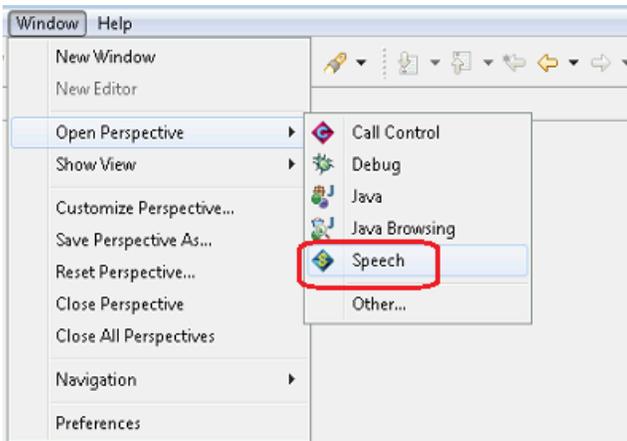


Figure 24 OD: Speech Projects

2. Open the properties for each current project open in Orchestration Designer by right clicking the project inside Orchestration Designer and select **Properties**.

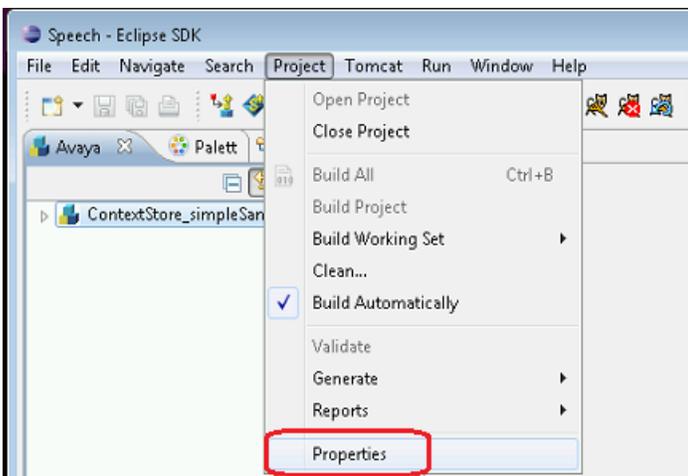


Figure 25 OD: Speech Project Properties

3. In the next screen, choose the **Orchestration Designer** item in the list on the left hand side
4. In the Orchestration Designer dialog select the **Pluggable Connectors** tab;
 - **If there is a Context Store PDC already deployed**, un-deploy the old plugin by un-checking the Context Store Connector check box and selecting **OK**. See Figure 26 below.
 - **If there is no Context Store Connector in the list**, proceed to the step of copying the Context Store PDC in to the plugin folder below.

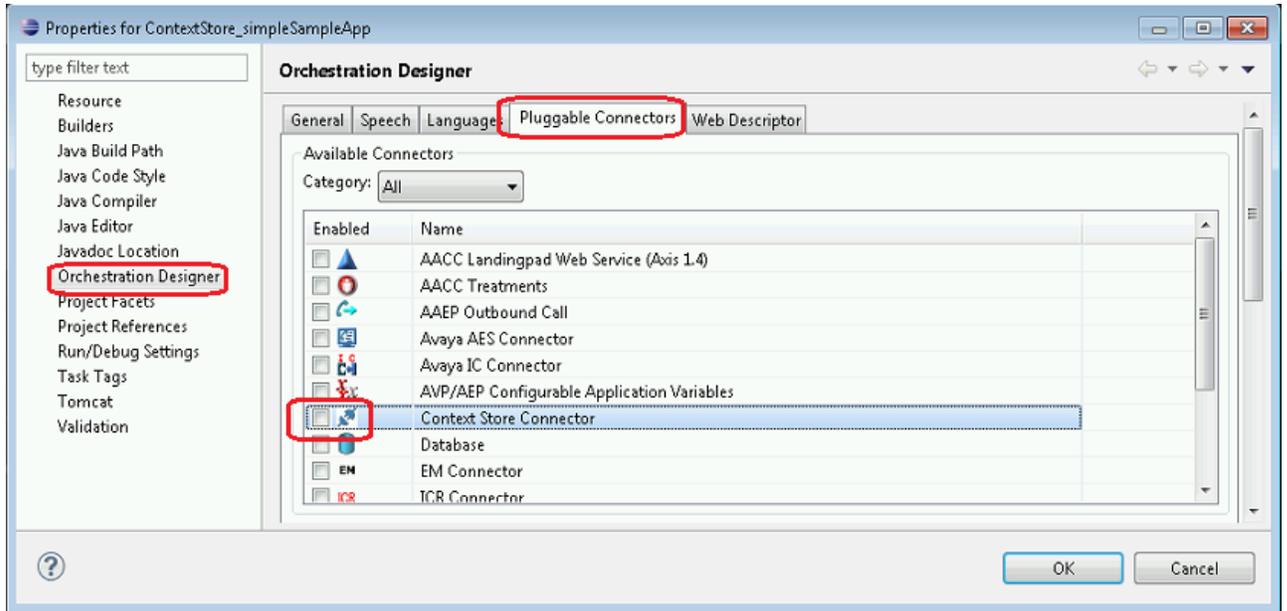


Figure 26 OD: Un-deploy Existing Context Store Connector

7. Close Orchestration Designer.
8. Delete the old plugin jar from the `... \AAOD<version> \eclipse \plugins` folder.

NB: if there was a Context Store PDC in the list of **Pluggable Connectors** but not deployed, the existing plugin must still to be deleted from the plugins folder.
9. Start Orchestration Designer.
10. Copy the Context Store PDC plugin jar into the plugin folder.
11. Restart Orchestration Designer.
12. Deploy the plugin by checking the Context Store Connector check box and clicking **OK**.

7.4. Tomcat configuration in Orchestration Designer

1. Open the Avaya Aura Orchestration Designer version installed above.
2. In Eclipse, select the menu **Window** and choose the **Preferences** option.

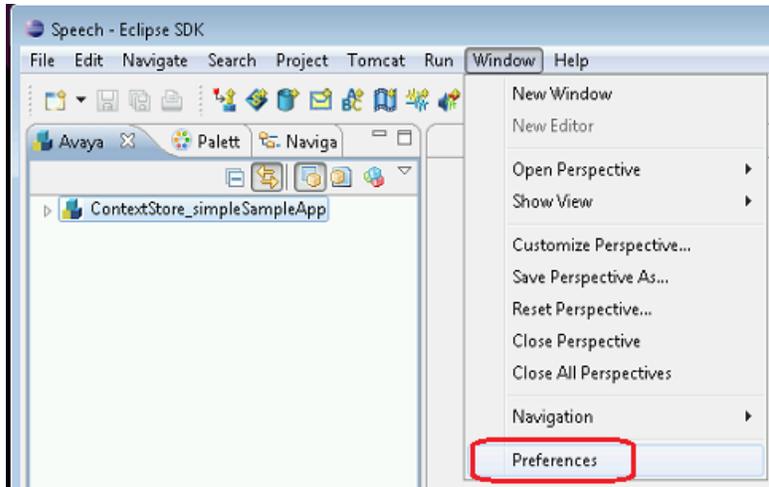


Figure 27 OD: Preferences

3. In the next screen, choose **Tomcat** item in the list on the left hand side.
4. Mark **Version 6.X** in the **Tomcat version** section and configure the **Tomcat home** directory.

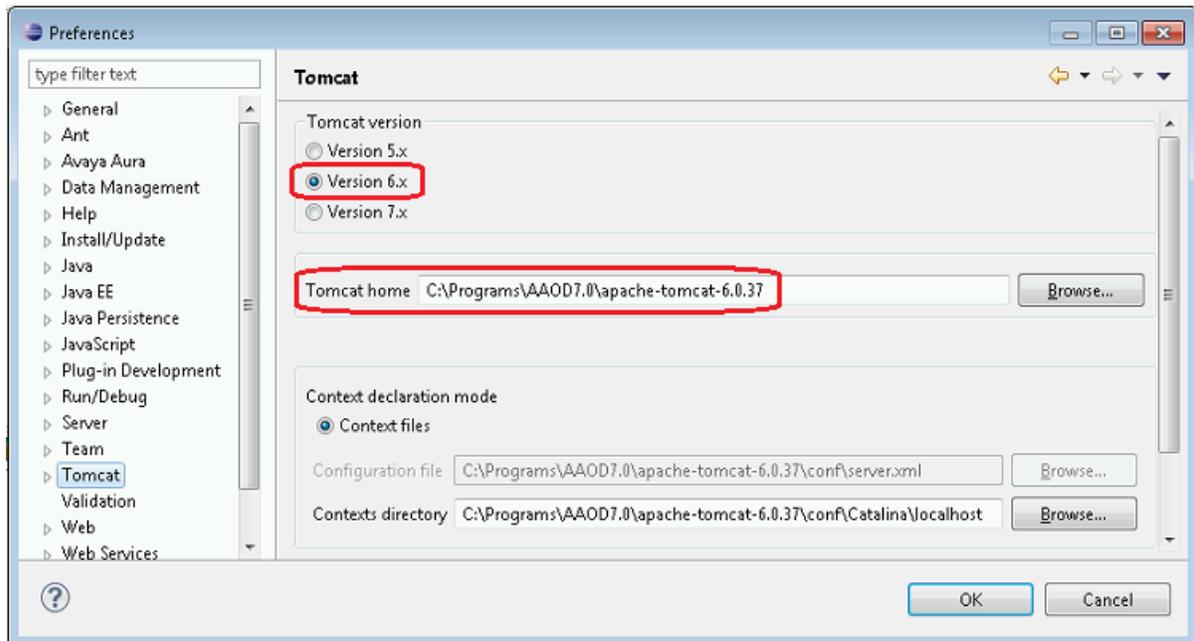


Figure 28 OD: Tomcat Version Preference

7.5. Configuring certificates in Orchestration Designer

- The Context Store PDC uses certificates to authenticate itself with Context Store.
- The application runtimeconfig, which is packaged with Orchestration Designer, is used to configure these certs.
- Context Store requires clients to identify themselves with a cert before an operation is run. Please follow the instructions in the [11.2 Certificate based authentication](#) section. Once the configuration has been completed on System Manager, the certs need to be made available for the PDC to use.
- In Eclipse start Tomcat, see screen capture below.



Figure 29 OD Start Tomcat

1. Log in to <IP running the Orchestration Designer>:8080/runtimeconfig/ using the username and password ddadmin:ddadmin
2. Click **Certificates** from the left column.
3. Select the Use other radio button if is not already and click the Change button.
4. The Keystore Path may need to be entered manually. E.g. "C:\canBeDeleted\test3.jks" the file needs to be located on the system that is running Tomcat.
5. Enter the password that you configured in System Manager for the cert in the **Password:** and **Confirm:** text boxes.
6. Click **Validate** to confirm the password entered is correct for the cert.

The system displays the message `Keystore location and password validate OK`. Click **Save** to apply changes.
7. Click **Save** (on **Home > Certificates > Change Keystore**).

The system displays the **Home > Certificates** page.
8. Click **Save** (on **Home > Certificates**) and confirm that the system displays the message ***Certificate changes have been successfully applied.***

7.6. Using the Context Store Connector

7.6.1. Configure the Context Store plugin

Before using the **Context Store PDC** in a project, you must configure the global parameters for the project. These can be accessed from the **Project -> Properties** menu as shown in the screen capture below.

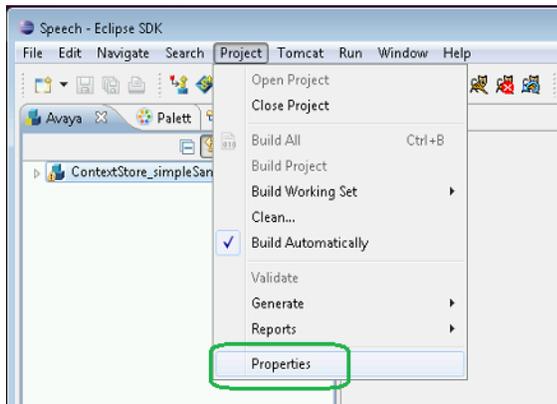


Figure 30 OD: Project Properties

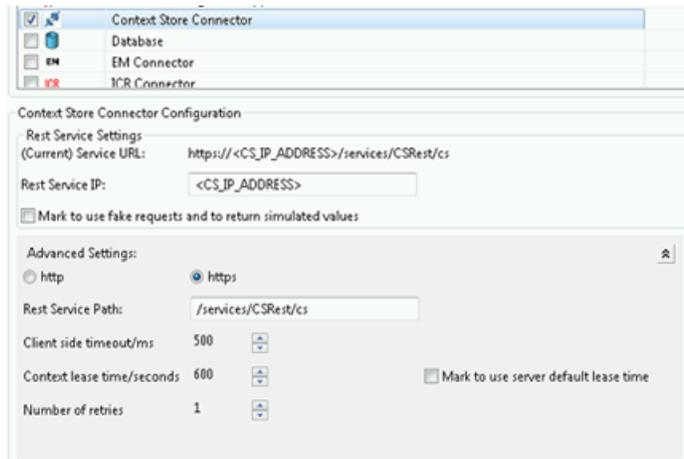


Figure 31 Configuring the CS Connector

After following the instructions in the [Installing/Upgrading the Context Store PDC plugin](#) section, navigate to **Context Store Connector** in the **Available Connector** dialog box, and enter the following properties.

- **Rest Service IP:** The IP of the Context Store Cluster. This is the only configuration item that will need to be changed.
- **Rest Service Path:** This will only need to be changed if the path changes. You will be notified if this is necessary.
- **Client Side Timeout:** The timeout, in milliseconds before the Context Store PDC will timeout and retry.
- **Context Lease Time:** The time, in seconds, the information will be stored in the Context Store. Enabling the corresponding tick box will select the Context Store default least time for the context.
- **Number of Retries:** The number of retries that will occur after a client side timeout is detected.

7.6.2. Add the Context Store Connector in the workflow

1. To add the connector to the workflow, locate **Palette** in the **Data** object that is in the **Application Items**.
2. Drag the **Data** object and drop it in the workflow.

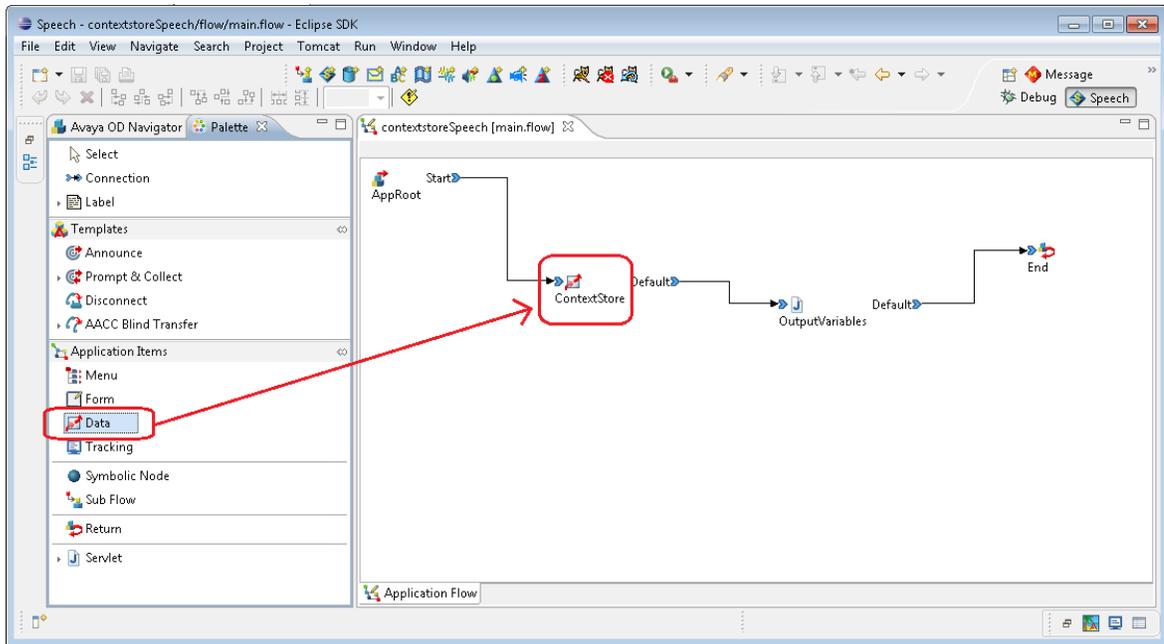
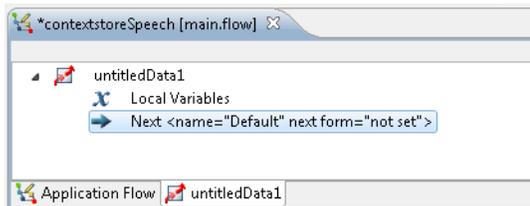


Figure 32 OD: Context Store 'Data' node

3. Double click on the Data node to view its properties. A new Data node has two empty properties by default – *Local Variables* and a *Next* node pointer as shown below.



4. In the **Palette**, find the **Get/Set Context in Context Store** item in the **Context Store Connector** section.
5. Drag the **Get/Set Context in Context Store** object and drop inside the **Data** object as shown below.

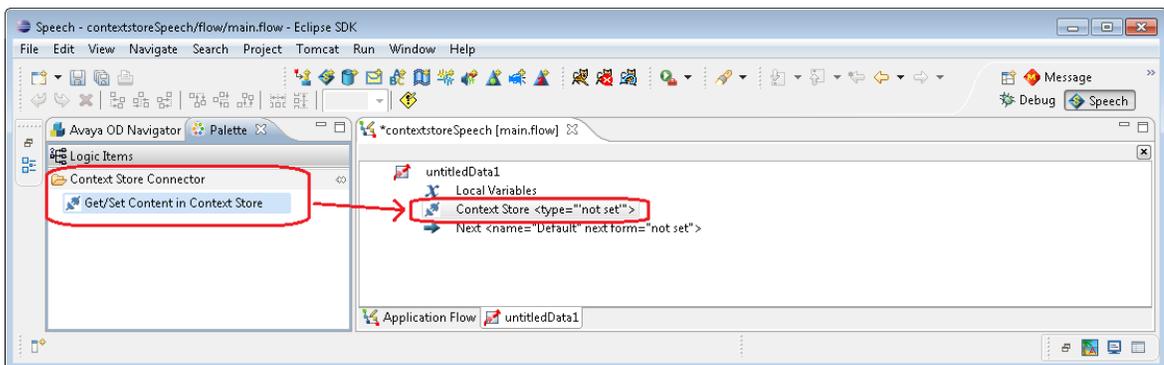


Figure 33 OD: Context Store Connector

6. Click the **Context Store Connector** that you have added to the **Data** object. The system displays the **Property** window where you can configure the Connector.

The screenshot below shows the variable configuration of the sample Context Store project

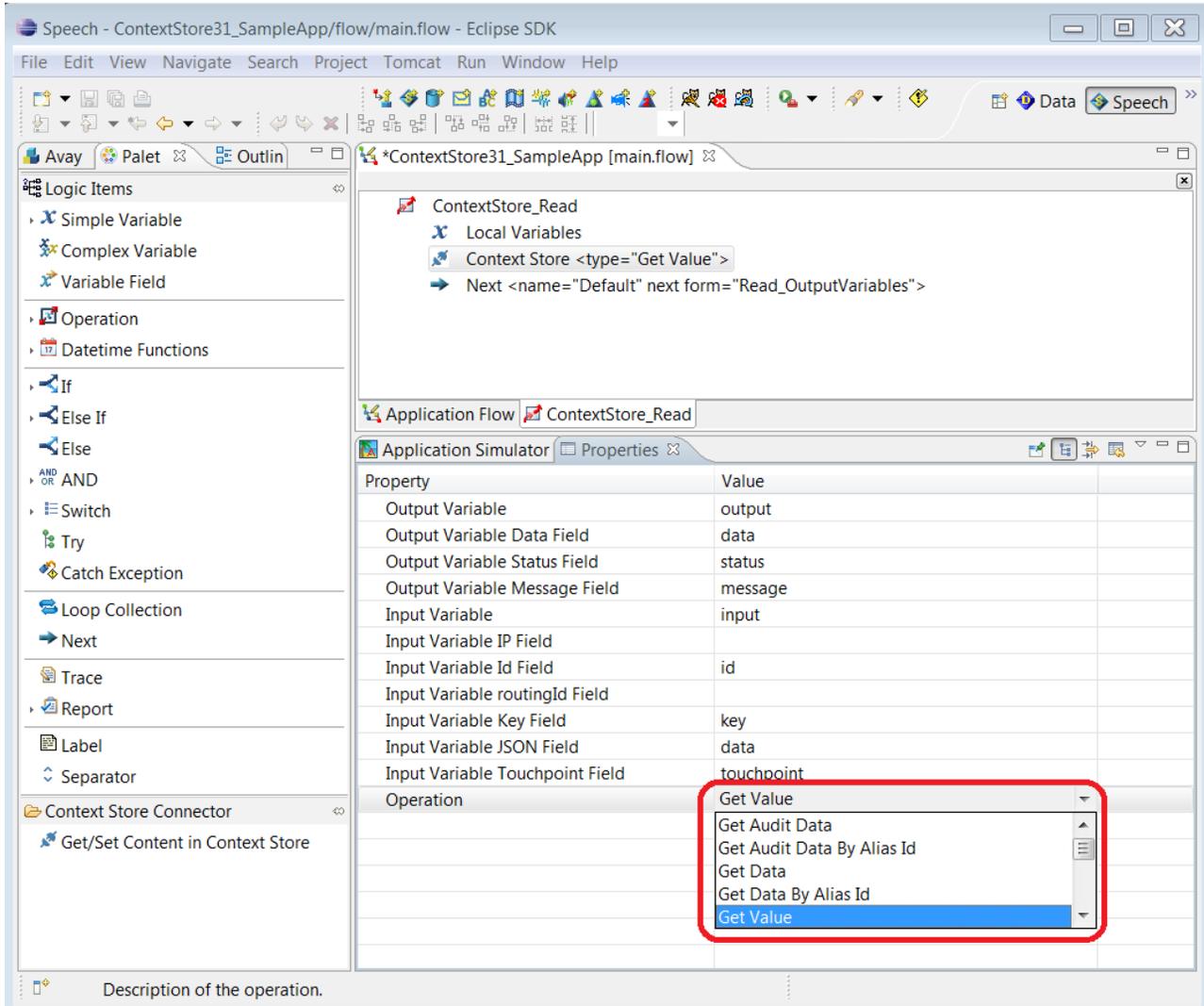


Figure 34 OD: Setting CS Connector Input and Output variables

7.6.3. Context Store PDC input/output variables

You must create the **input/output** variables to be used by the **Context Store Connector**.

1. To create the variables, go to the **Speech Navigator** and open the folder `flow` inside the project.
2. Double click the **project.variables** and create two **Complex Variables** as shown below - one for **input** and other for **output**:

input must have the variables to represent:

- **data**: the JSON data that needs to be passed as parameter to the Context Store request.
- **id**: the identifier that needs to be passed as parameter to the Context Store request .
- **key**: the key of the context that needs to be passed as parameter to the Context Store request.

In a geo-redundant CS deployment, **input** variable must have a variable to represent the routingId

- **rid**: a routingId for the context needs to be passed as parameter to the Context Store request. If a routingId not supplied, Context Store will apply the default when processing the request.

When using the Audit Trail feature, an additional input variable is required for the touchpoint parameter

- **touchpoint**: the touchpoint identifier that can be passed as a parameter to most requests to the Context Store rest interface (doesn't apply for 'Get Audit Data', 'Get Audit Data', 'Get Context Ids' or 'Delete Context' requests)

An optional IP address variable can also be used to override that configured in the OD settings

- **ip**: Allows for the default context store IP configured in the service settings to be overridden with a new IP for individual requests. While the field is left blank the default IP configured in the setting will be used for all requests.

output must have the variables to represent:

- **data**: the data or error message description that is returned as response to the Context Store request.
- **message**: the message (OK or message error) that is returned as response to the Context Store request.
- **status**: the status code (200 is OK or other code that represents the error) that is returned as response to the Context Store request.

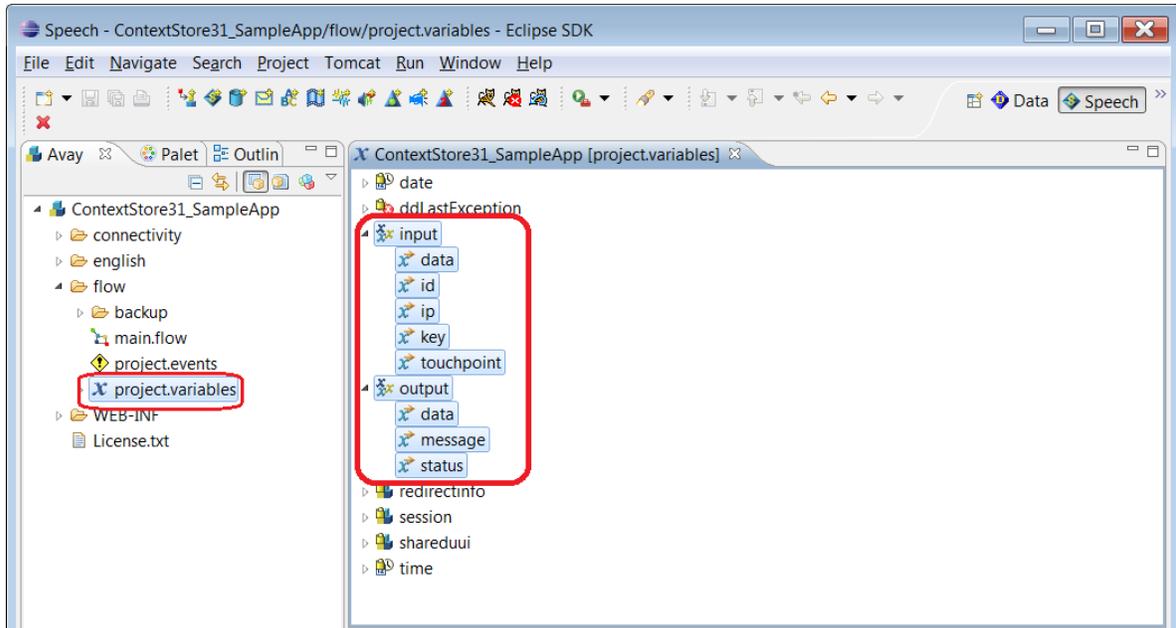


Figure 35 OD: Context Store Project Variables

3. In the **Property** window, enter the value to be passed to the **Context Store Connector**. In a real application, these values must be passed in a dynamic way by the workflow.

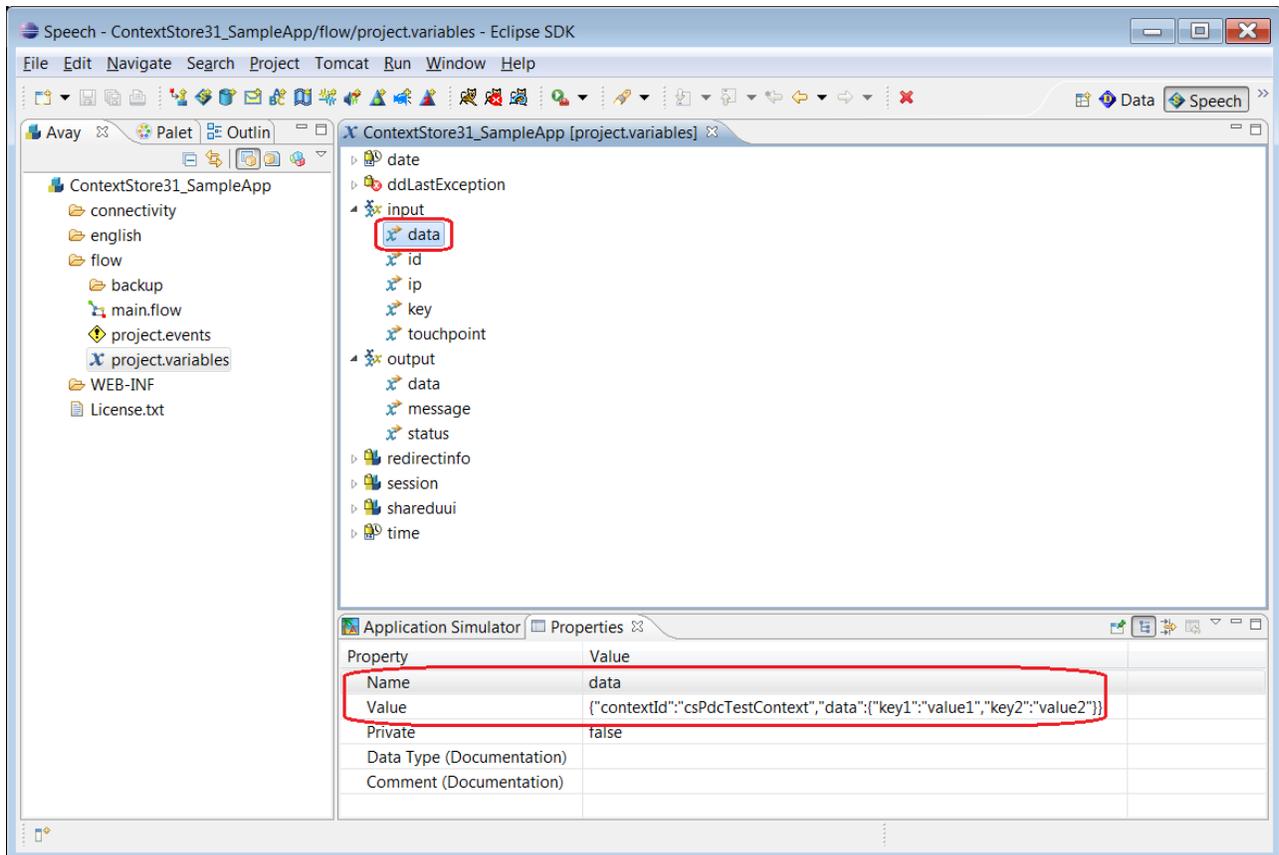


Figure 36 OD: Configure Properties

4. Return to the **Context Store Connector Property** window, as shown in Figure 34, and enter the properties with the variables created in the step above.

5. In the **Operation property**, enter the operation with one of the values:
 - 1) **Get Data**: Get context information. Input Parameters:
 - id, example `testcontextId`
 - rid, (optional) example `12`
 - ip, (optional) example `127.0.0.1`
 - touchpoint, (optional) example `pdcTouchpoint`
 - 2) **Get Data by aliasId**: Get context information using aliasId. Input Parameters:
 - id, example `testaliasId`
 - rid, (optional) example `12`
 - ip, (optional) example `127.0.0.1`
 - touchpoint, (optional) example `pdcTouchpoint`
 - 3) **Get Value**: Get context key's value. Input Parameters:
 - id, example `testcontextId`
 - key, example `key1_name`
 - rid, (optional) example `12`
 - ip, (optional) example `127.0.0.1`
 - touchpoint, (optional) example `pdcTouchpoint`
 - 4) **Get Value by aliasId**: Get context key's value using aliasId. Input Parameters:
 - id, example `testaliasId`
 - key, example `key1_name`
 - rid, (optional) example `12`
 - ip, (optional) example `127.0.0.1`
 - touchpoint, (optional) example `pdcTouchpoint`
 - 5) **Get Context Ids**: Get metadata for a group of contexts. Input Parameters:
 - id, example `XYZ`
 - rid, (optional) example `12`
 - ip, (optional) example `127.0.0.1`
 - 6) **Get Audit Data**: Get audit data for a context. Input Parameters:
 - id, example `testcontextId`
 - rid, (optional) example `12`
 - ip, (optional) example `127.0.0.1`
 - 7) **Get Audit Data by Alias Id**: Get audit data for a context using aliasId. Input Parameters:

- id, example `testaliasId`
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
- 8) **Put Data:** Update context information. Input Parameters:
- id, example `testcontextId`
 - data in JSON format, example

```
{"data":{"key1_name":"value1_updated","key3_name":"value3_new_value"}}
```
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdcTouchpoint`
- 9) **Put Data by aliasId:** Update context information using aliasId. Input Parameters:
- id, example `testaliasId`
 - data in JSON format, example

```
{"data":{"key1_name":"value1_updated","key3_name":"value3_new_value"}}
```
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdcTouchpoint`
- 10) **Put Value:** Update context data. Input Parameters::
- id, example `testcontextId`
 - key, example `key1_name`
 - data in JSON, example `value1_updated`
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdcTouchpoint`
- 11) **Put Value by aliasId:** Update context data using aliasId. Input Parameters:
- id, example `testaliasId`
 - key, example `key1_name`
 - data in JSON, example `value1_updated`
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdcTouchpoint`
- 12) **Put aliasId by contextId:** Update aliasId list using aliasId. Input Parameters:
- id, example `testcontextId`
 - data in JSON, example `["aliasId2","aliasId3"]`

- rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdCTouchpoint`
- 13) **Put aliasId by aliasId:** Update aliasId list using aliasId. Input Parameters:
- id, example `testaliasId`
 - data in JSON, example `["aliasId2", "aliasId3"]`
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdCTouchpoint`
- 14) **Upsert Context:** Create a new context object with provided contextId or update an existing context if there is a matching contextId already existing in the space. Input Parameters:
- id, example `testcontextId`
 - data in JSON, example

```
{"groupId": "XYZ", "data": {"key1_name": "value1_data", "key2_name": "value2_data"}}
```
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdCTouchpoint`
- 15) **Upsert Context by aliasId:** Create a new context object with provided alias Id (and return auto generated contextId) or update an existing context if there is a matching contextId already existing in the space (will return contextId of associated context). Input Parameters:
- id, example `testaliasId`
 - data in JSON, example

```
{"groupId": "XYZ", "data": {"key1_name": "value1_data", "key2_name": "value2_data"}}
```
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdCTouchpoint`
- 16) **Post Context:** Post context Information. Input Parameters:
- data in JSON format, example

```
{"contextId": "optional_contextId", "groupId": "XYZ", "data": {"key1_name": "value1_data", "key2_name": "value2_data"}}
```
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdCTouchpoint`
- 17) **Post Context with aliasId:** Post context Information. Input Parameters:
- id, example `testaliasId` (or to create multiple aliasIds, `aliasId1, aliasId2`)

- data in JSON format, example

```
{"contextId": "optional_contextId", "groupId": "XYZ", "data": {"key1_name": "value1_data", "key2_name": "value2_data"}}
```
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdcTouchpoint`
- 18) **Delete Context:** Delete context information. Input Parameters:
- id, example `testcontextId`
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
- 19) **Delete Context by aliasId:** Delete context using aliasId. Input Parameters:
- id, example `testaliasId`
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
- 20) **Delete Value:** Delete context data. Input Parameters:
- id, example `testcontextId`
 - key, example `key1_name`
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdcTouchpoint`
- 21) **Delete Value by aliasId:** Delete context data using aliasId. Input Parameters:
- id, example `testaliasId`
 - key, example `key1_name`
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdcTouchpoint`
- 22) **Delete aliasId:** Delete aliasId associated with a context. Input Parameters:
- id, example `testaliasId`
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdcTouchpoint`

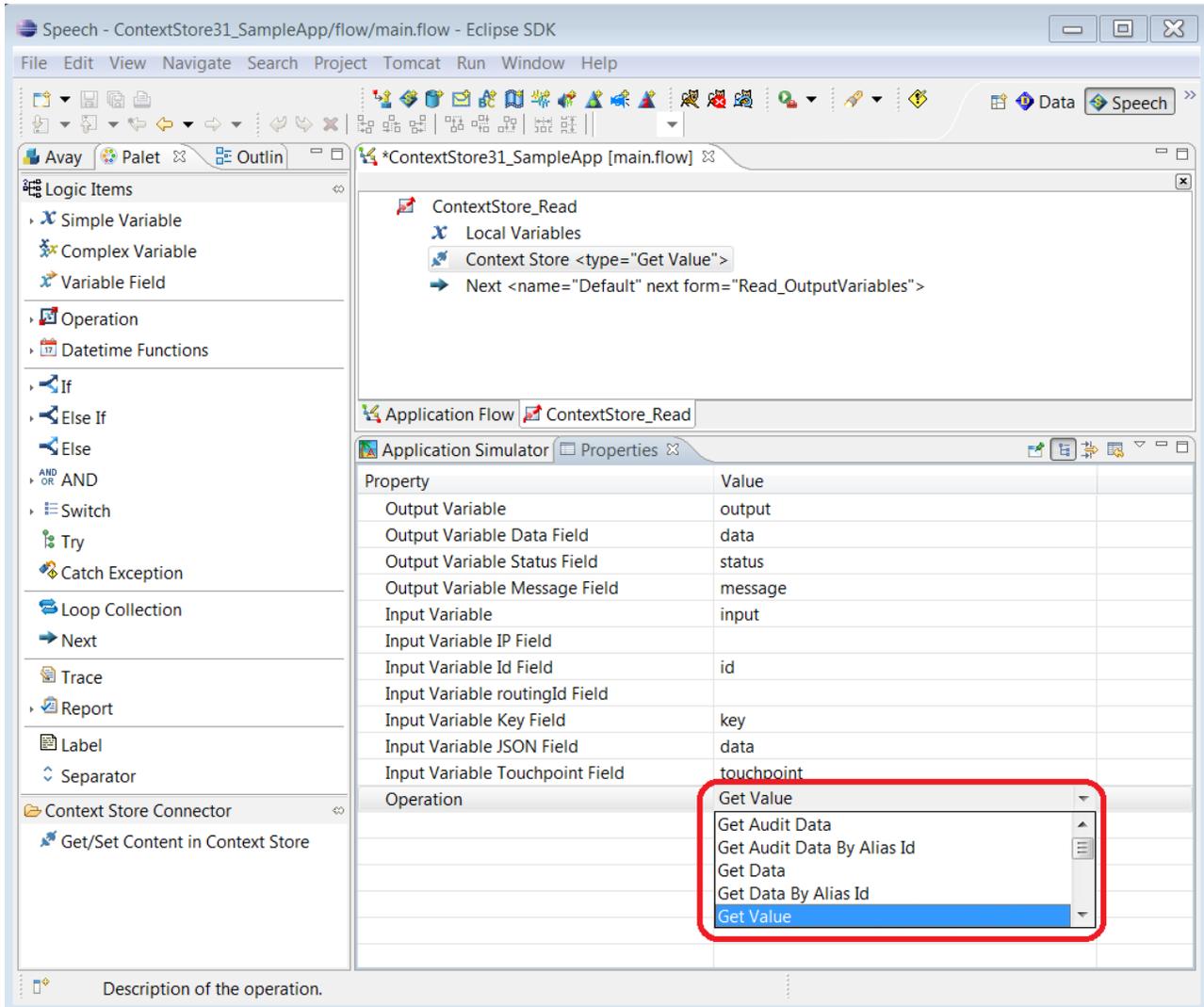


Figure 37 OD: Context Store PDC Operations

7.6.4. Retrieving output variable values

1. To get the output values, you can plug a **Servlet** component after the **Data** component in the workflow.
2. Write a code that seems to the code below in the **Servlet class** associated with the **Servlet component** plugged in the previous step.

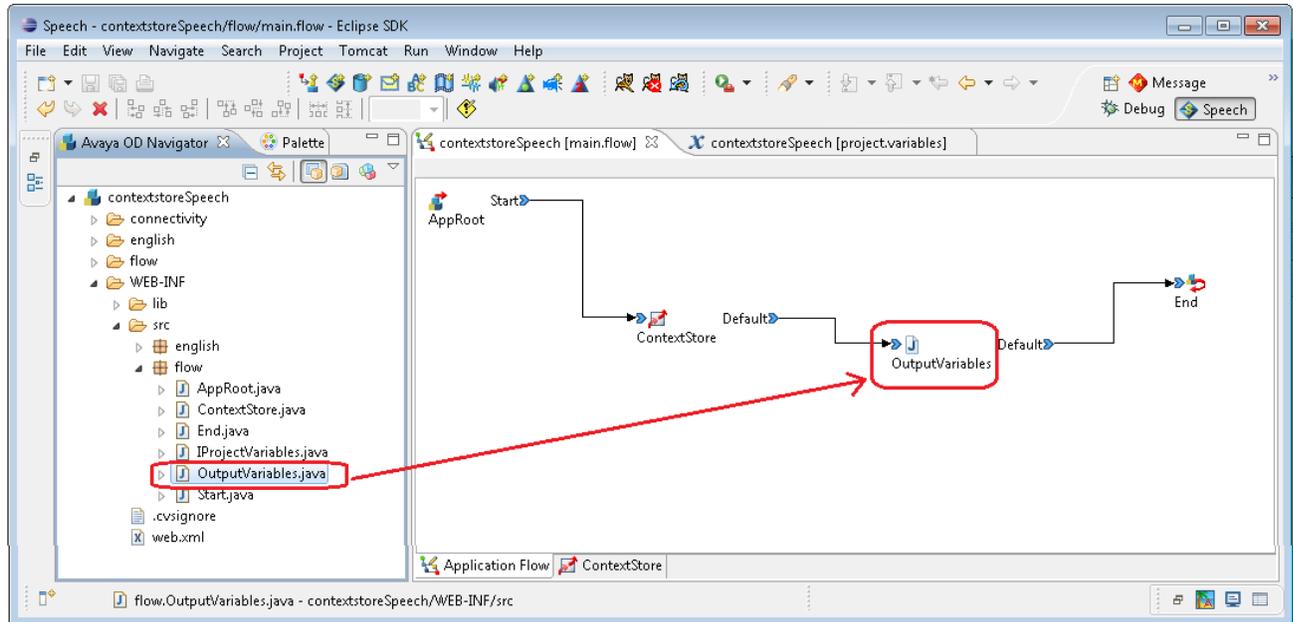


Figure 38 OD: Output Variables

All GET operations (as well as POST operation for auto-generated *contextId*) send back data in the request response under normal circumstances: below are some examples

Operation	Object Returned	Example code to retrieve data
Get Value Get Audit Data Post Context	String	<pre>String outputValue = outvariable.getComplexVariable().getField(IProjectVariables. OUTPUT_FIELD_DATA).getStringValue();</pre>
Get Data	Map <String, Object>	<pre>Map<String, Object> dataMap = (Map<String, Object>) outvariable.getComplexVariable().getField(IProjectVariables. OUTPUT_FIELD_DATA).getObjectValue();</pre>
Get Context Ids	List <Map<String, Object>>	<pre>List<Map<String, Object>> listOfContextData = List<Map<String, Object>>() outvariable.getComplexVariable().getField(IProjectVariables. OUTPUT_FIELD_DATA).getObjectValue();</pre>

Figure 39 OD: Working with Output Variables

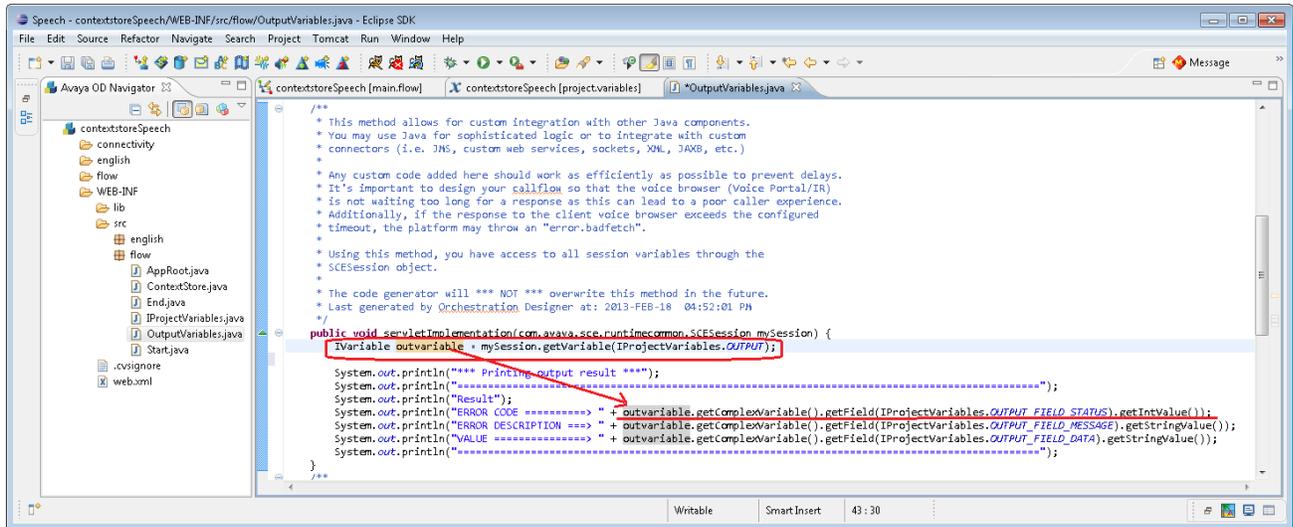


Figure 40 OD: OutputVariables.java

7.7. Context Store PDC – Running Sample Projects

1. Create a new directory on the machine that is running Orchestration Designer.
2. Download the Context Store sample applications from the [Avaya Context Store DevConnect](#) site; this resource is listed as **Context Store PDC Sample Applications** on the *Downloads* page.
3. Unzip the sample projects from the downloaded resource into the directory created in step 1.
4. Open Eclipse and switch the Eclipse workspace to the directory created above. If the select workspace dialog is not shown when Eclipse is opened, select **File > Switch > Workspace > Other**, and Browse to the newly created directory's location and select **OK**.
5. In the Eclipse, select the **File menu** and choose **Import**.
6. In the next screen, choose **Existing Project into Workspace** option in the **General** folder and click **Next**.

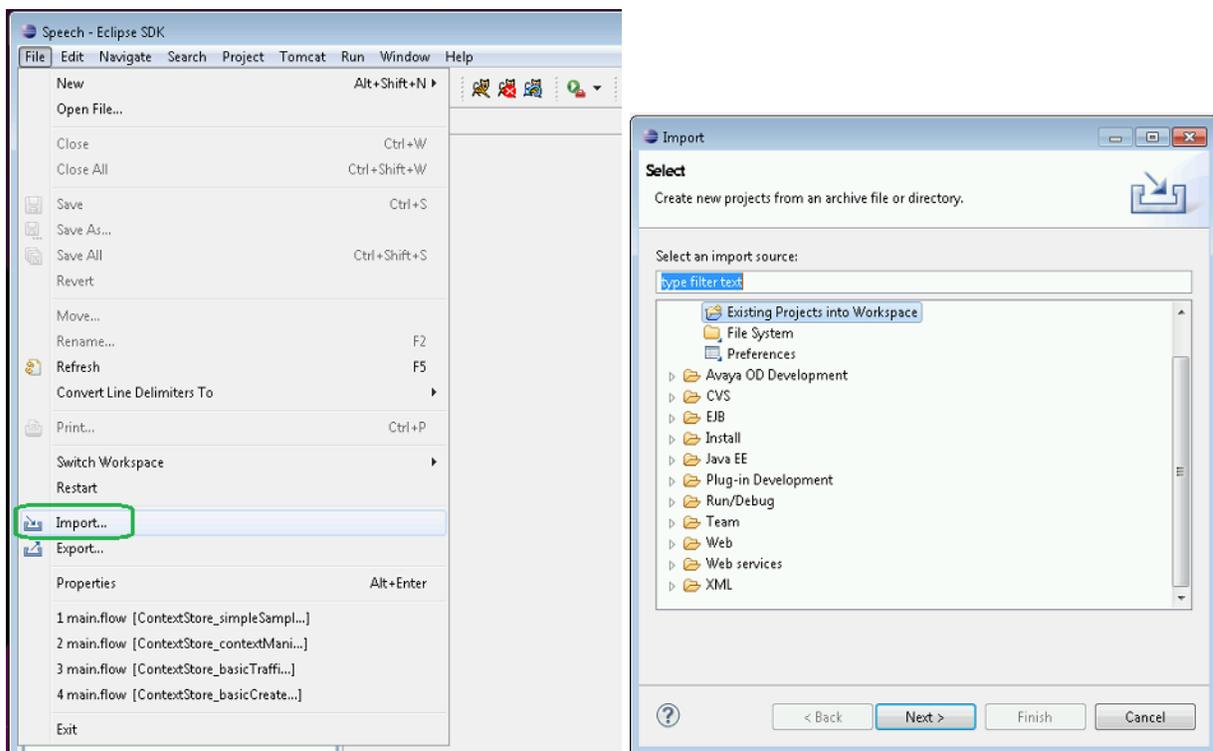


Figure 41 OD: Import Sample Project

7. In **Select root directory**, **Browse** to the workspace directory where you unzipped the sample Context Store PDC applications and press **OK**.
8. Check the box beside *ContextStore_simpleSampleApp* in the **Project list** and click **Finish**.

Note: there are a number of sample applications provided in the archive available from [DevConnect](#). This document gives instructions for setting up and running the most basic application only (*ContextStore_simpleSampleApp*); all of the other sample applications can be imported, setup and tested in the OD simulator by following the exact same steps.

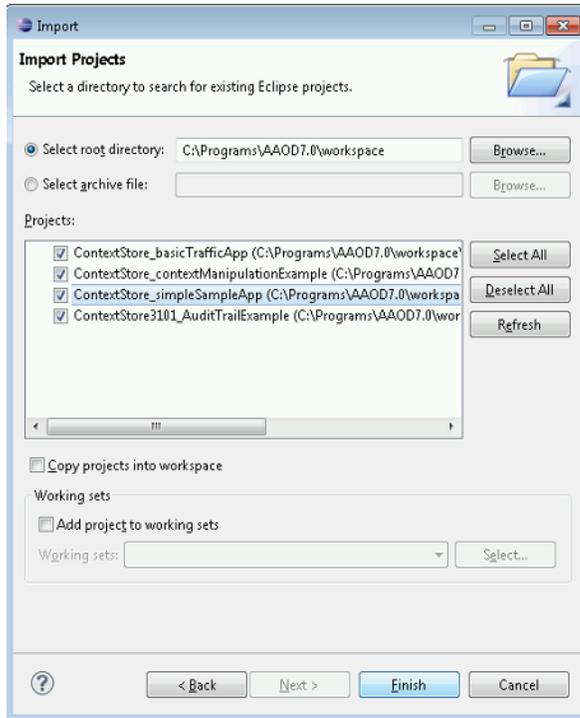


Figure 42 OD: Import Project Dialog

7.7.1. Configuring the project to use the Context Store PDC

1. Select the **Window** menu, choose **Open Perspective** option and choose **Speech**.
2. In the Avaya Orchestration Designer Navigator view, select *ContextStore_simpleSampleApp*.
3. Select the **Project** menu and choose **Properties** option;

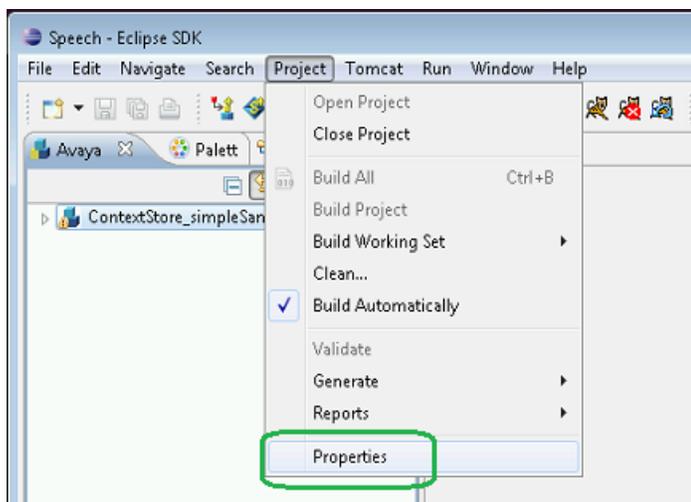


Figure 43 OD: Project Properties

4. In the next screen, choose **Orchestration Designer** item in the list and in the right side go to the **Pluggable Connector** tab;
5. Check the **Context Store Connector** box and configure the settings for your Context Store cluster i.e. enters the IP address of your Context Store interface, see the diagram below.

- The Client side timeout unit is milliseconds, this is the time the Context Store PDC will wait for a response from the interface.
- The Context Lease Time unit is seconds, this is the time the context will be stored in the Context Store.

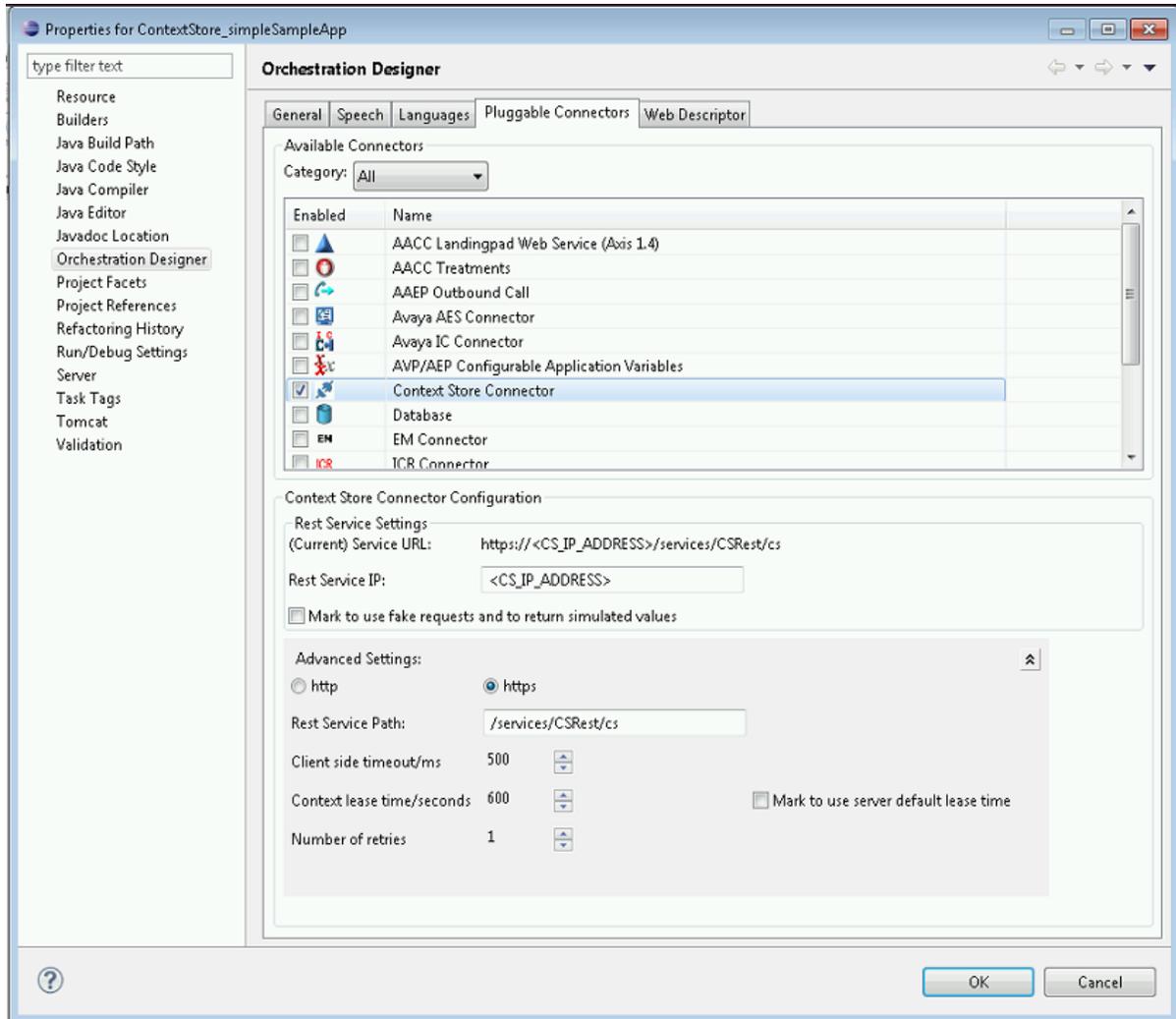


Figure 44 OD: Configure Context Store PDC

- Click **OK** to set the configuration

7.7.2. Testing Context Store PDC Sample Applications

With all the above configured and with the Context Store server up and running, you can run the test project.

- In Orchestration Designer, open the **Application Simulator** view, select the sample application to run in the **Available Projects** list (*ContextStore_simpleSampleApp*) and press the **Run Application** button.

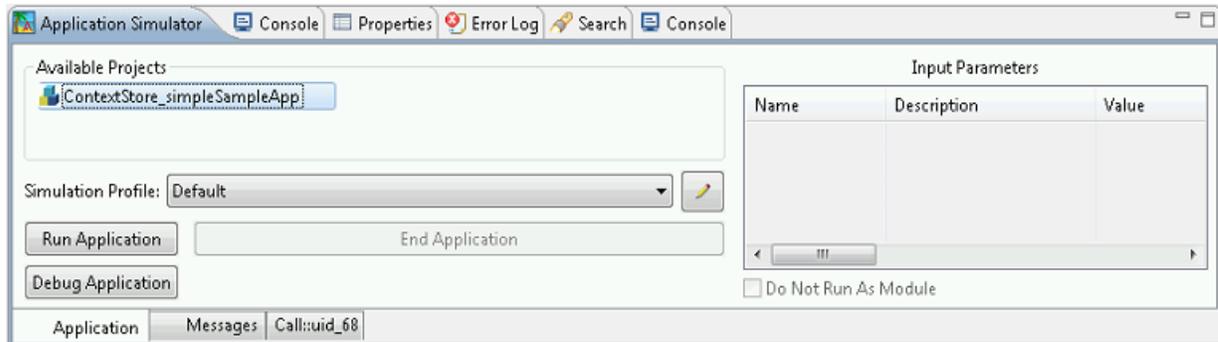


Figure 45 OD: Application Simulator

The result should look like the screenshot below if the test context is successfully created.

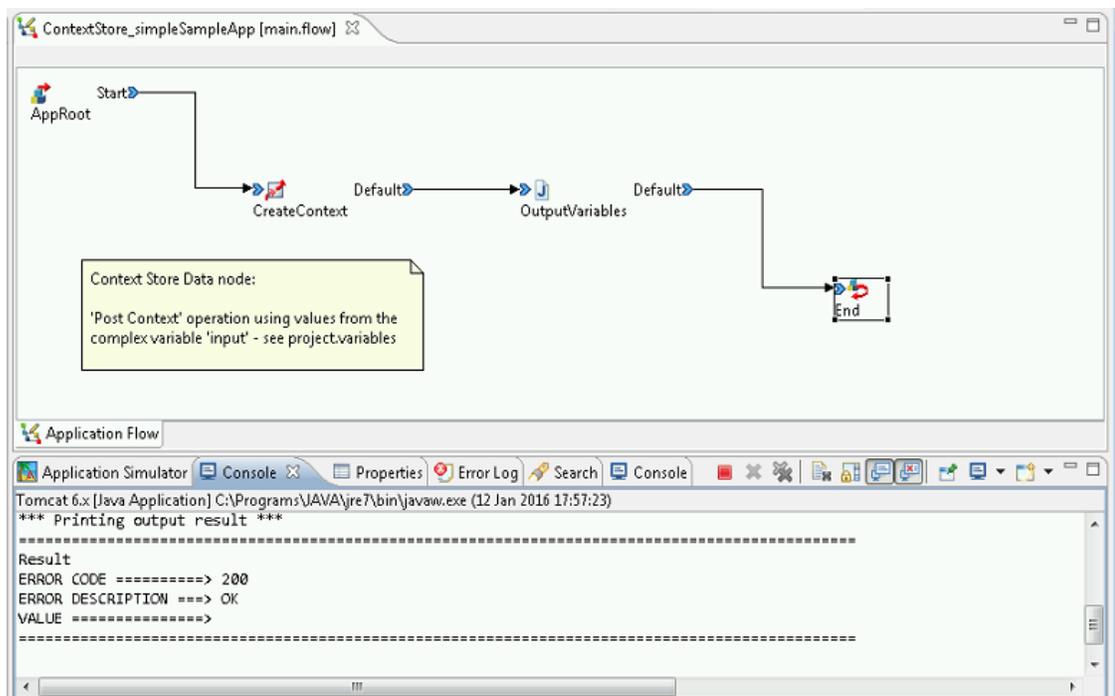


Figure 46 OD: Test Context Successfully Created

If the context cannot be created, the applicable reason/error will be displayed.

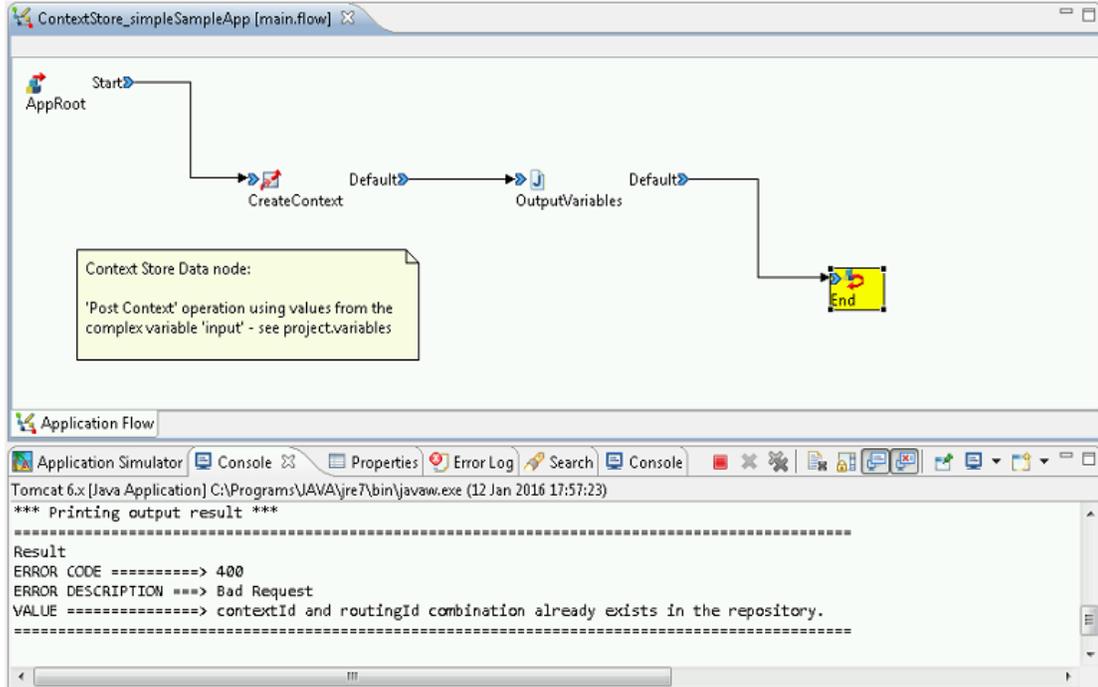


Figure 47 OD: Create Context Request Failed - ContextId Is Not Unique

7.7.3. Sample Audit Trail Application

In the Context Store 3.1.0.1 release, audit trail functionality was integrated with the Context Store Pluggable Data Connector. For more information about this audit feature see section 3.2 Audit Trail Feature (since CS 3.1).

A sample application which demonstrates the use of the *touchpoint* parameter, as well as retrieval of a context's audit trail information, is provided in the **Context Store PDC Sample Applications** resource - *ContextStore3101_AuditTrailExample*.

This sample application can be imported and tested in the Orchestration Designer simulator following the same instructions above.

For this feature to function correctly, there are two configuration requirements:

1. Enable audit trail feature on Context Store through the CSManager attribute *CS Audit: Event Limit*
2. One must create/populate the **touchpoint** field in the **input** variable and the assign this field as the value for **Input Variable Touchpoint Field** for the Context Store data node as show below.

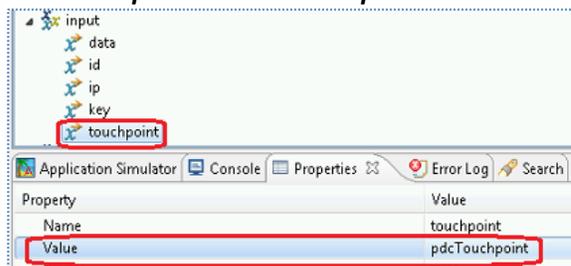




Figure 48 OD: Assign Touchpoint Parameter Value

7.8. PDC - Experience Portal Test Setup

7.8.1. Overview

Context Store supports call flow applications that are developed in Orchestration Designer 6.0 and 7.0 and deployed in Experience Portal 6.0 and 7.0 using the Context Store PDC.

Orchestration Designer, which is based on Eclipse, is used to create call applications, and Experience Portal facilitates the deployment and running of these applications.

NB: Experience Portal only supports Tomcat 6.0 and this is why even though Orchestration Designer supports both Tomcat 6.0 and 7.0, the applications must be tested with Tomcat 6.0.

Note, only one Context Store Cluster can be configured per application. This means that each Context Store node in a flow has to use the same Context Store. Each application can have a different Context Store configured for it.

7.8.2. What's needed

- Orchestration Designer
- One of the Sample flows
- Runtimeconfig application, which is supplied with the Orchestration Designer
- Context Store system deployed and running
- Experience Portal Tomcat configured on the system

Download the sample call flows archive from the Context Store [DevConnect](#) site; this resource is listed as **Context Store PDC Sample Applications** on the *Downloads* page. Unzip this resource; it contains multiple sample Context Store call flow projects for Orchestration Designer and Avaya Aura Experience Portal. The project discussed below; *ContextStore_basicTrafficApp* and *ContextStore_advancedTrafficApp*.

7.8.3. Sample Call Flow One.

ContextStore_basicTrafficApp: A Sample Call Flow with the Context Store PDC included is provided the **Context Store PDC Sample Applications** resource as an example of how to use the PDC in a flow. This saves a context with a contextId of the UCID from the incoming call and prints the response from the Context Store.

This project can be opened using the same instructions as in section [7.3 Installing/Upgrading the Context Store PDC plugin](#)

The instructions below should be followed exactly and especially with regards the starting and stopping of Orchestration Designer.

1. The speech perspective needs to be open in order to configure the Context Store PDC.

Select **Window > Open Perspective > Speech** in Orchestration Designer.

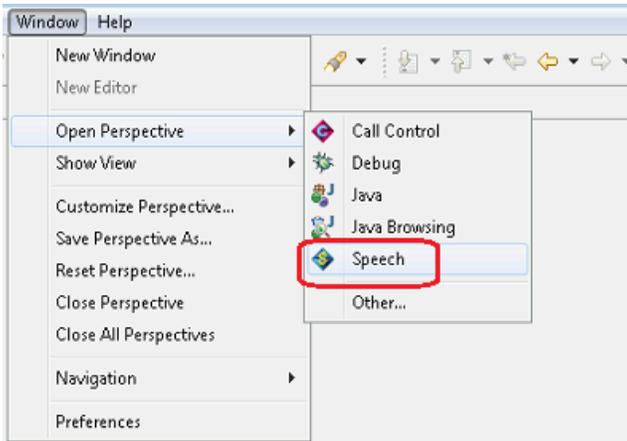


Figure 24 OD: Speech Projects

2. Open the properties for each current project open in Orchestration Designer by right clicking the project inside Orchestration Designer and select **Properties**.

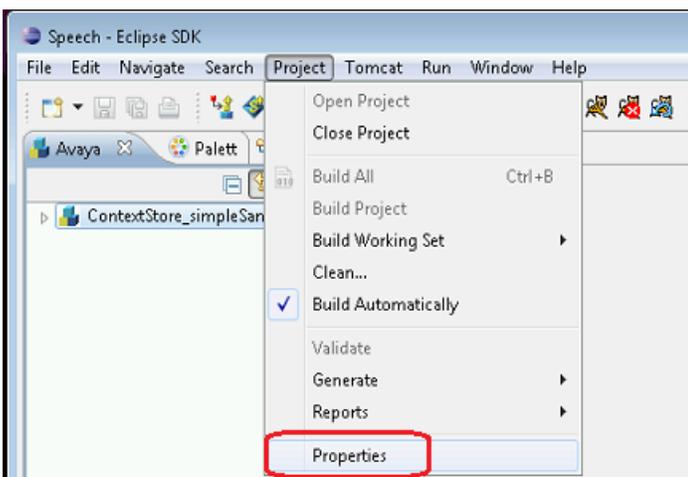


Figure 25 OD: Speech Project Properties

3. In the next screen, choose the **Orchestration Designer** item in the list on the left hand side
4. In the Orchestration Designer dialog select the **Pluggable Connectors** tab;
 - **If there is a Context Store PDC already deployed**, un-deploy the old plugin by un-checking the Context Store Connector check box and selecting **OK**. See Figure 26 below.
 - **If there is no Context Store Connector in the list**, proceed to the step of copying the Context Store PDC in to the plugin folder below.

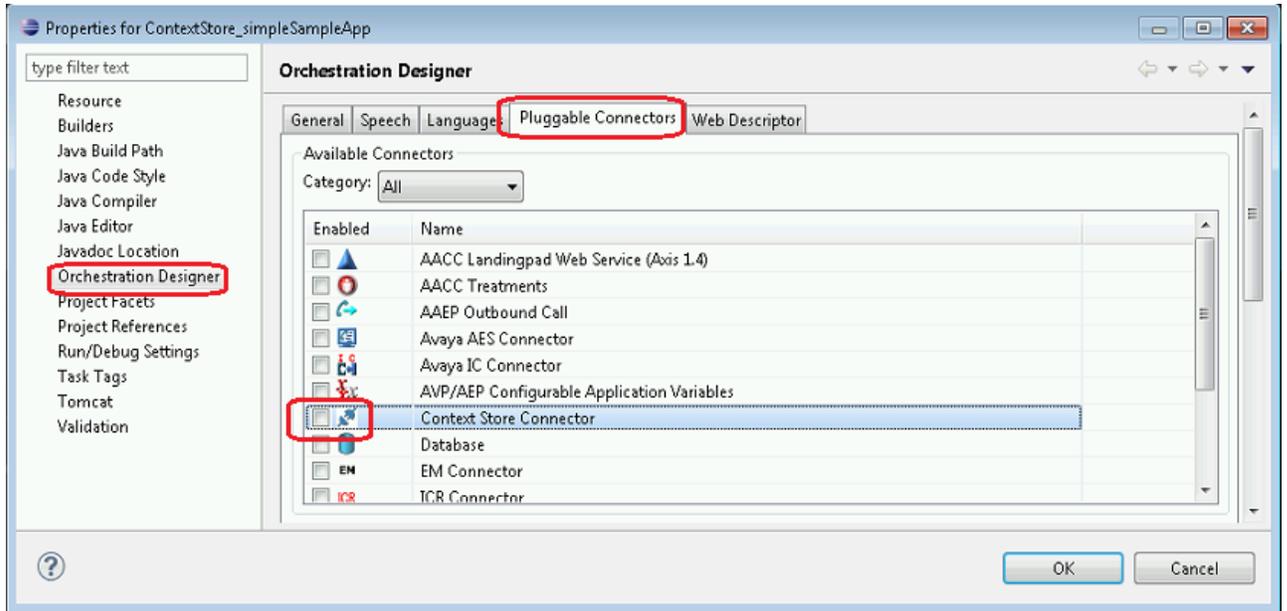


Figure 26 OD: Un-deploy Existing Context Store Connector

7. Close Orchestration Designer.
8. Delete the old plugin jar from the `... \AAOD<version> \eclipse \plugins` folder.

NB: if there was a Context Store PDC in the list of **Pluggable Connectors** but not deployed, the existing plugin must still to be deleted from the plugins folder.
9. Start Orchestration Designer.
10. Copy the Context Store PDC plugin jar into the plugin folder.
11. Restart Orchestration Designer.
12. Deploy the plugin by checking the Context Store Connector check box and clicking **OK**.

7.9. Tomcat configuration in Orchestration Designer

5. Open the Avaya Aura Orchestration Designer version installed above.
6. In Eclipse, select the menu **Window** and choose the **Preferences** option.

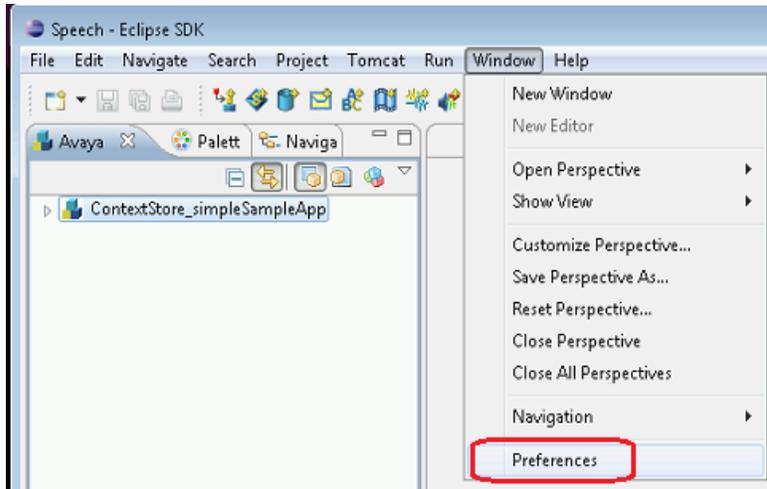


Figure 27 OD: Preferences

7. In the next screen, choose **Tomcat** item in the list on the left hand side.
8. Mark **Version 6.X** in the **Tomcat version** section and configure the **Tomcat home** directory.

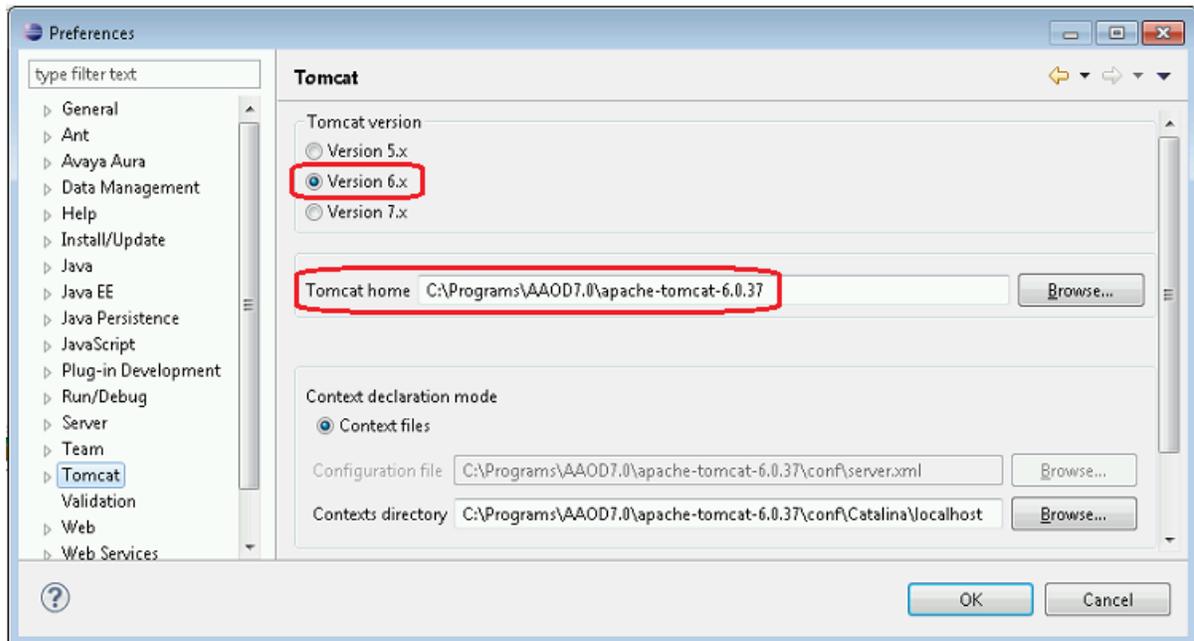


Figure 28 OD: Tomcat Version Preference

7.10. Configuring certificates in Orchestration Designer

- The Context Store PDC uses certificates to authenticate itself with Context Store.
- The application runtimeconfig, which is packaged with Orchestration Designer, is used to configure these certs.
- Context Store requires clients to identify themselves with a cert before an operation is run. Please follow the instructions in the [11.2 Certificate based authentication](#) section. Once the configuration has been completed on System Manager, the certs need to be made available for the PDC to use.
- In Eclipse start Tomcat, see screen capture below.

9. Log in to <IP running the Orchestration Designer>:8080/runtimeconfig/ using the username and password ddadmin:ddadmin

10. Click **Certificates** from the left column.

11. Select the Use other radio button if is not already and click the Change button.

12. The Keystore Path may need to be entered manually. E.g. "C:\canBeDeleted\test3.jks" the file needs to be located on the system that is running Tomcat.

13. Enter the password that you configured in System Manager for the cert in the **Password:** and **Confirm:** text boxes.

14. Click **Validate** to confirm the password entered is correct for the cert.

The system displays the message `Keystore location and password validate OK`. Click **Save** to apply changes.

15. Click **Save** (on Home > Certificates > Change Keystore).

The system displays the **Home > Certificates** page.

16. Click **Save** (on Home > Certificates) and confirm that the system displays the message ***Certificate changes have been successfully applied.***

7.11. Using the Context Store Connector

7.11.1. Configure the Context Store plugin

Before using the **Context Store PDC** in a project, you must configure the global parameters for the project. These can be accessed from the **Project -> Properties** menu as shown in the screen capture below.

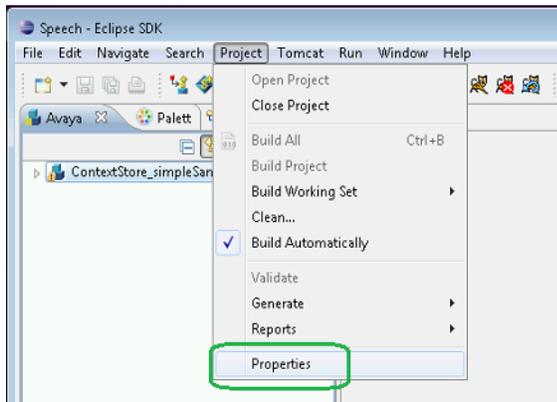


Figure 30 OD: Project Properties

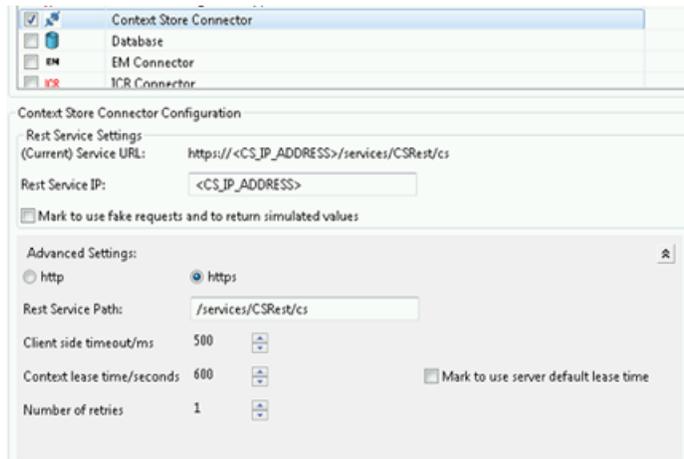


Figure 31 Configuring the CS Connector

After following the instructions in the Installing/Upgrading the Context Store PDC plugin section, navigate to **Context Store Connector** in the **Available Connector** dialog box, and enter the following properties.

- **Rest Service IP:** The IP of the Context Store Cluster. This is the only configuration item that will need to be changed.
- **Rest Service Path:** This will only need to be changed if the path changes. You will be notified if this is necessary.
- **Client Side Timeout:** The timeout, in milliseconds before the Context Store PDC will timeout and retry.
- **Context Lease Time:** The time, in seconds, the information will be stored in the Context Store. Enabling the corresponding tick box will select the Context Store default least time for the context.
- **Number of Retries:** The number of retries that will occur after a client side timeout is detected.

7.11.2. Add the Context Store Connector in the workflow

1. To add the connector to the workflow, locate **Palette** in the **Data** object that is in the **Application Items**.
2. Drag the **Data** object and drop it in the workflow.

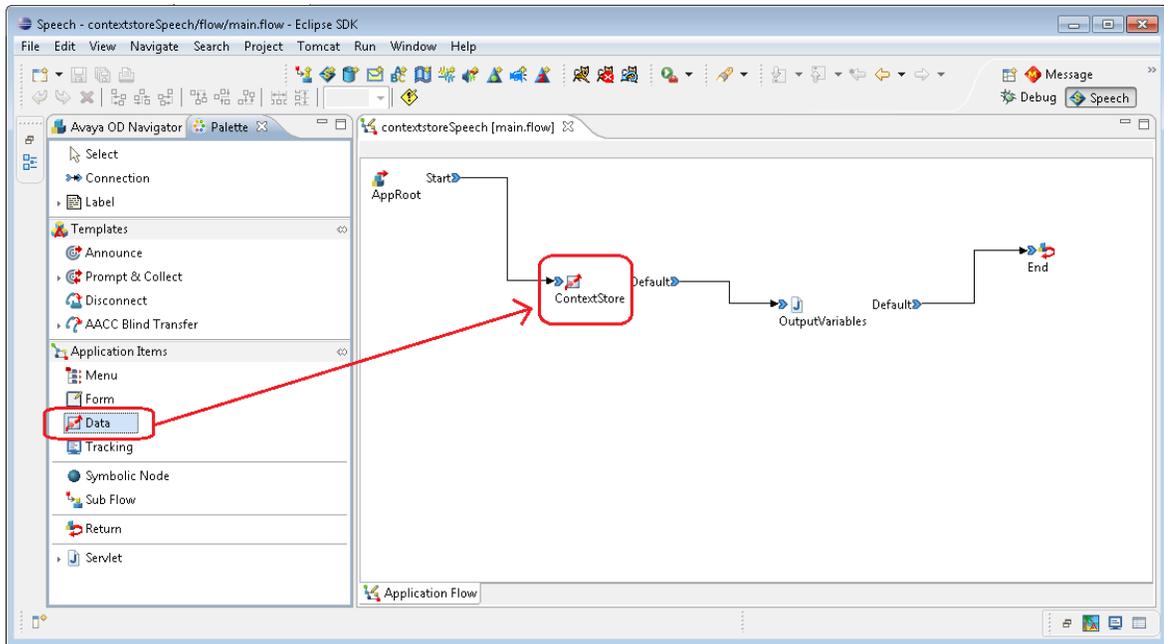
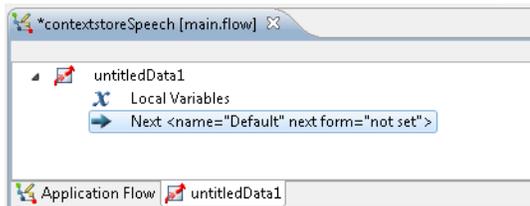


Figure 32 OD: Context Store 'Data' node

3. Double click on the Data node to view its properties. A new Data node has two empty properties by default – *Local Variables* and a *Next* node pointer as shown below.



4. In the **Palette**, find the **Get/Set Context in Context Store** item in the **Context Store Connector** section.
5. Drag the **Get/Set Context in Context Store** object and drop inside the **Data** object as shown below.

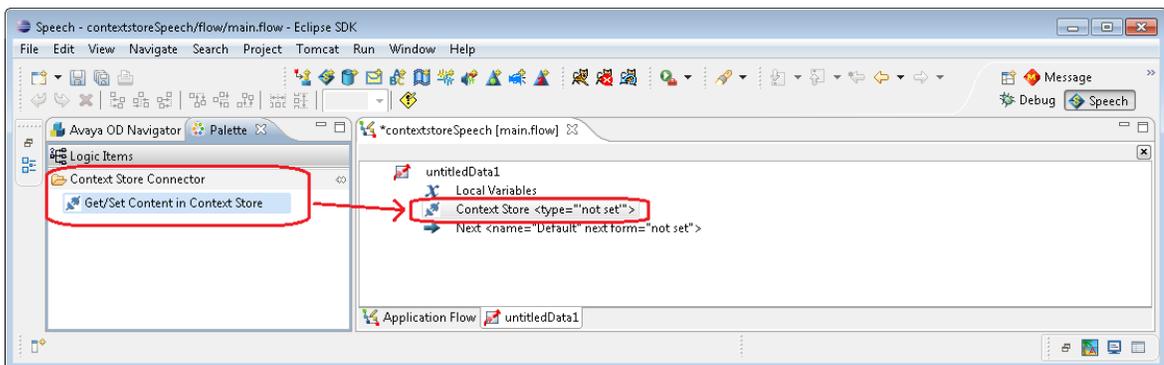


Figure 33 OD: Context Store Connector

6. Click the **Context Store Connector** that you have added to the **Data** object. The system displays the **Property** window where you can configure the Connector.

The screenshot below shows the variable configuration of the sample Context Store project

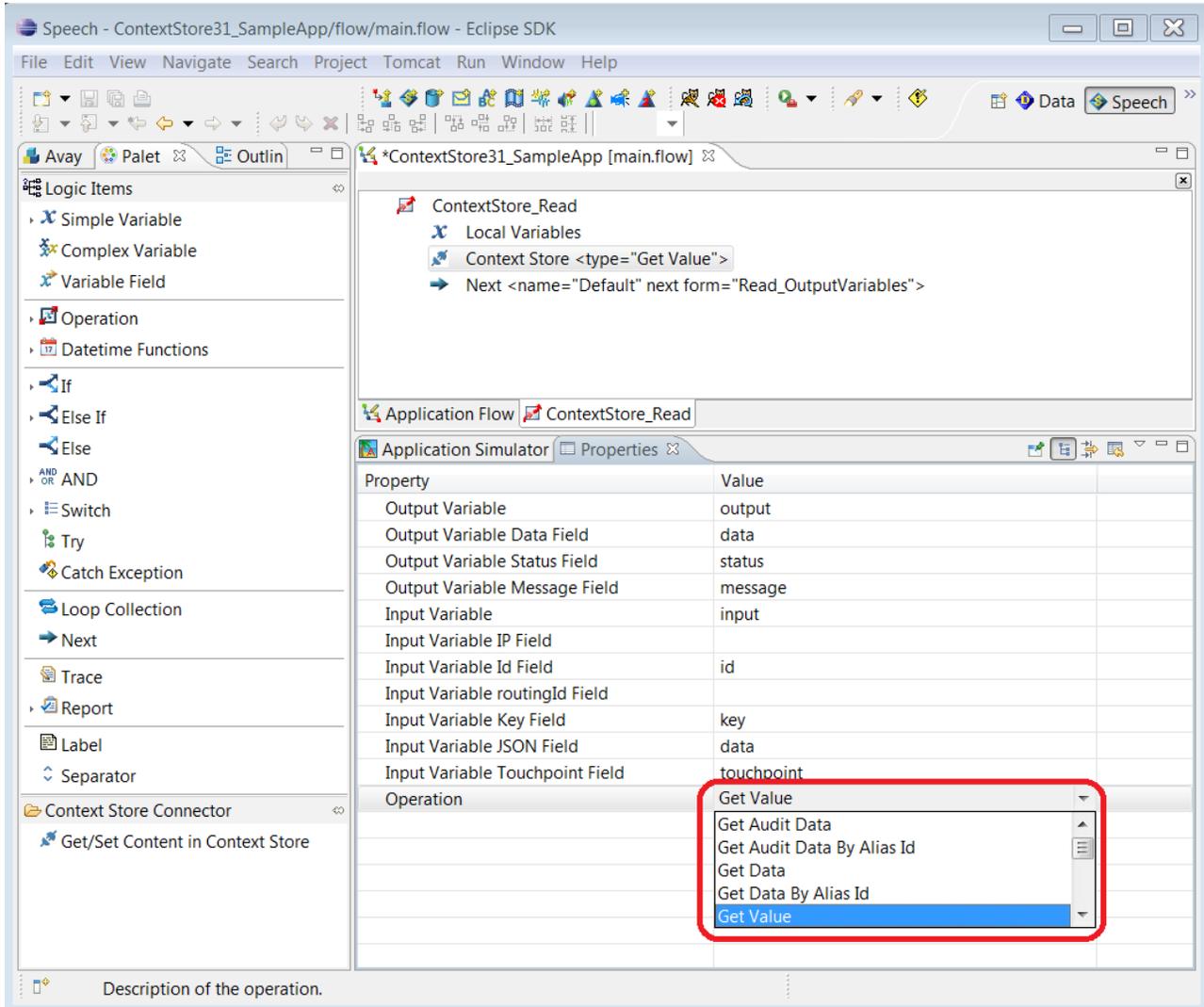


Figure 34 OD: Setting CS Connector Input and Output variables

7.11.3. Context Store PDC input/output variables

You must create the **input/output** variables to be used by the **Context Store Connector**.

3. To create the variables, go to the **Speech Navigator** and open the folder `flow` inside the project.
4. Double click the **project.variables** and create two **Complex Variables** as shown below - one for **input** and other for **output**:

input must have the variables to represent:

- **data**: the JSON data that needs to be passed as parameter to the Context Store request.
- **id**: the identifier that needs to be passed as parameter to the Context Store request .
- **key**: the key of the context that needs to be passed as parameter to the Context Store request.

In a geo-redundant CS deployment, **input** variable must have a variable to represent the routingId

- **rid**: a routingId for the context needs to be passed as parameter to the Context Store request. If a routingId not supplied, Context Store will apply the default when processing the request.

When using the Audit Trail feature, an additional input variable is required for the touchpoint parameter

- **touchpoint**: the touchpoint identifier that can be passed as a parameter to most requests to the Context Store rest interface (doesn't apply for 'Get Audit Data', 'Get Audit Data', 'Get Context Ids' or 'Delete Context' requests)

An optional IP address variable can also be used to override that configured in the OD settings

- **ip**: Allows for the default context store IP configured in the service settings to be overridden with a new IP for individual requests. While the field is left blank the default IP configured in the setting will be used for all requests.

output must have the variables to represent:

- **data**: the data or error message description that is returned as response to the Context Store request.
- **message**: the message (OK or message error) that is returned as response to the Context Store request.
- **status**: the status code (200 is OK or other code that represents the error) that is returned as response to the Context Store request.

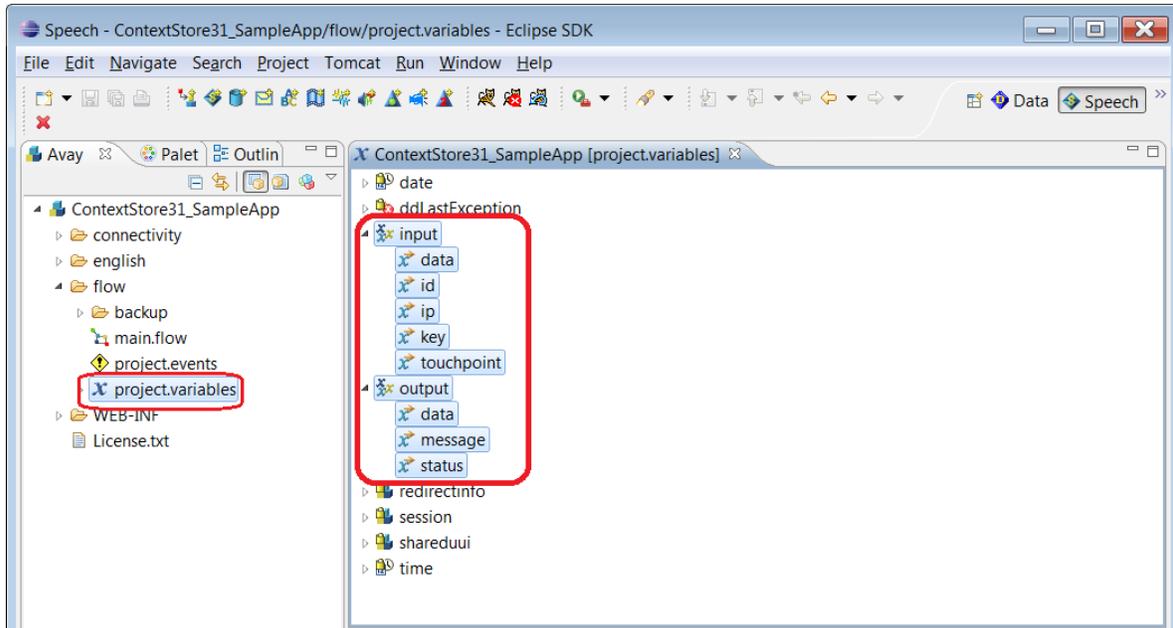


Figure 35 OD: Context Store Project Variables

3. In the **Property** window, enter the value to be passed to the **Context Store Connector**. In a real application, these values must be passed in a dynamic way by the workflow.

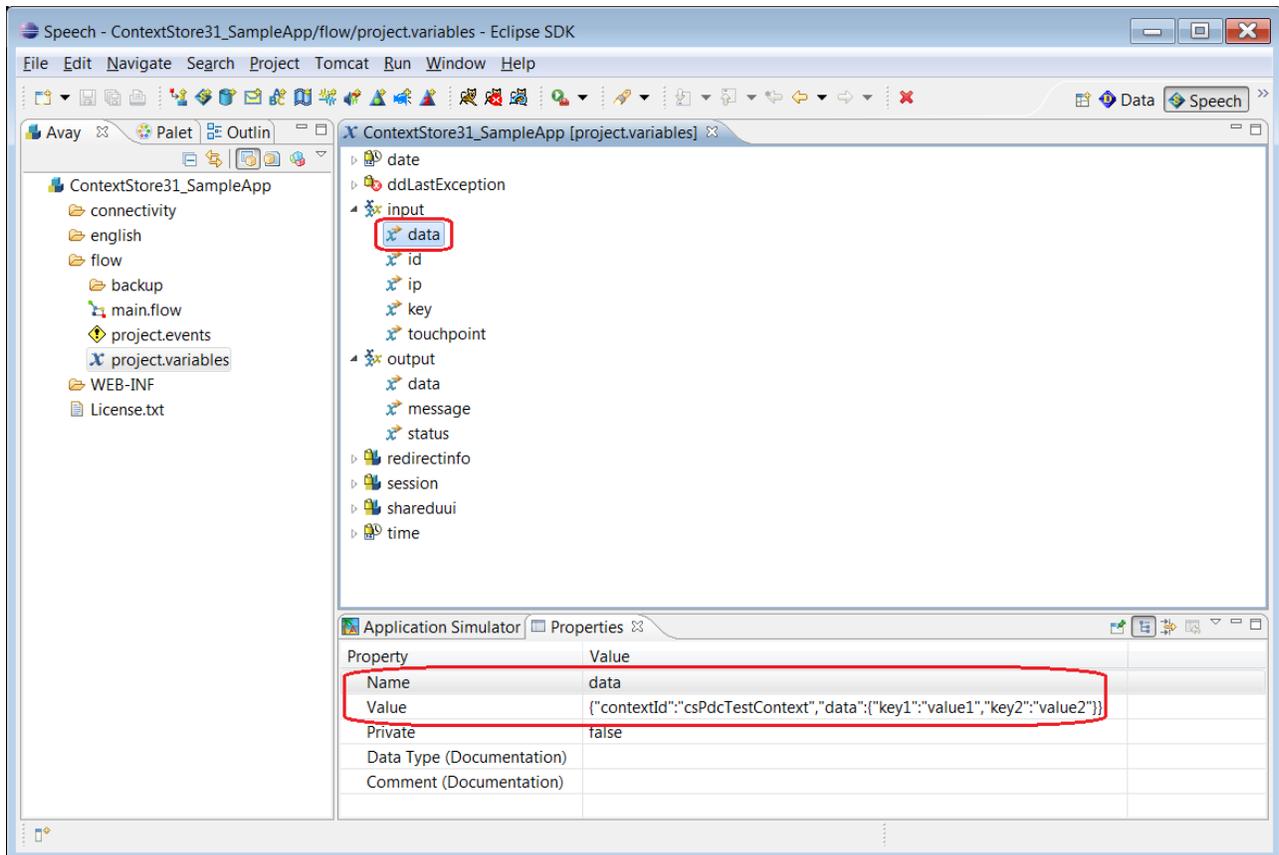


Figure 36 OD: Configure Properties

6. Return to the **Context Store Connector Property** window, as shown in Figure 34, and enter the properties with the variables created in the step above.

7. In the **Operation property**, enter the operation with one of the values:
 - 23) **Get Data**: Get context information. Input Parameters:
 - id, example `testcontextId`
 - rid, (optional) example `12`
 - ip, (optional) example `127.0.0.1`
 - touchpoint, (optional) example `pdCTouchpoint`
 - 24) **Get Data by aliasId**: Get context information using aliasId. Input Parameters:
 - id, example `testaliasId`
 - rid, (optional) example `12`
 - ip, (optional) example `127.0.0.1`
 - touchpoint, (optional) example `pdCTouchpoint`
 - 25) **Get Value**: Get context key's value. Input Parameters:
 - id, example `testcontextId`
 - key, example `key1_name`
 - rid, (optional) example `12`
 - ip, (optional) example `127.0.0.1`
 - touchpoint, (optional) example `pdCTouchpoint`
 - 26) **Get Value by aliasId**: Get context key's value using aliasId. Input Parameters:
 - id, example `testaliasId`
 - key, example `key1_name`
 - rid, (optional) example `12`
 - ip, (optional) example `127.0.0.1`
 - touchpoint, (optional) example `pdCTouchpoint`
 - 27) **Get Context Ids**: Get metadata for a group of contexts. Input Parameters:
 - id, example `XYZ`
 - rid, (optional) example `12`
 - ip, (optional) example `127.0.0.1`
 - 28) **Get Audit Data**: Get audit data for a context. Input Parameters:
 - id, example `testcontextId`
 - rid, (optional) example `12`
 - ip, (optional) example `127.0.0.1`
 - 29) **Get Audit Data by Alias Id**: Get audit data for a context using aliasId. Input Parameters:

- id, example `testaliasId`
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
- 30) **Put Data:** Update context information. Input Parameters:
- id, example `testcontextId`
 - data in JSON format, example

```
{"data": {"key1_name": "value1_updated", "key3_name": "value3_new_value"}}
```
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdCTouchpoint`
- 31) **Put Data by aliasId:** Update context information using aliasId. Input Parameters:
- id, example `testaliasId`
 - data in JSON format, example

```
{"data": {"key1_name": "value1_updated", "key3_name": "value3_new_value"}}
```
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdCTouchpoint`
- 32) **Put Value:** Update context data. Input Parameters::
- id, example `testcontextId`
 - key, example `key1_name`
 - data in JSON, example `value1_updated`
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdCTouchpoint`
- 33) **Put Value by aliasId:** Update context data using aliasId. Input Parameters:
- id, example `testaliasId`
 - key, example `key1_name`
 - data in JSON, example `value1_updated`
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdCTouchpoint`
- 34) **Put aliasId by contextId:** Update aliasId list using aliasId. Input Parameters:
- id, example `testcontextId`
 - data in JSON, example `["aliasId2", "aliasId3"]`

- rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdCTouchpoint`
- 35) **Put aliasId by aliasId:** Update aliasId list using aliasId. Input Parameters:
- id, example `testaliasId`
 - data in JSON, example `["aliasId2", "aliasId3"]`
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdCTouchpoint`
- 36) **Upsert Context:** Create a new context object with provided contextId or update an existing context if there is a matching contextId already existing in the space. Input Parameters:
- id, example `testcontextId`
 - data in JSON, example

```
{"groupId": "XYZ", "data": {"key1_name": "value1_data", "key2_name": "value2_data"}}
```
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdCTouchpoint`
- 37) **Upsert Context by aliasId:** Create a new context object with provided alias Id (and return auto generated contextId) or update an existing context if there is a matching contextId already existing in the space (will return contextId of associated context). Input Parameters:
- id, example `testaliasId`
 - data in JSON, example

```
{"groupId": "XYZ", "data": {"key1_name": "value1_data", "key2_name": "value2_data"}}
```
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdCTouchpoint`
- 38) **Post Context:** Post context Information. Input Parameters:
- data in JSON format, example

```
{"contextId": "optional_contextId", "groupId": "XYZ", "data": {"key1_name": "value1_data", "key2_name": "value2_data"}}
```
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdCTouchpoint`
- 39) **Post Context with aliasId:** Post context Information. Input Parameters:
- id, example `testaliasId` (or to create multiple aliasIds, `aliasId1, aliasId2`)

- data in JSON format, example

```
{"contextId": "optional_contextId", "groupId": "XYZ", "data": {"key1_name": "value1_data", "key2_name": "value2_data"}}
```
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdcTouchpoint`
- 40) **Delete Context:** Delete context information. Input Parameters:
- id, example `testcontextId`
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
- 41) **Delete Context by aliasId:** Delete context using aliasId. Input Parameters:
- id, example `testaliasId`
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
- 42) **Delete Value:** Delete context data. Input Parameters:
- id, example `testcontextId`
 - key, example `key1_name`
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdcTouchpoint`
- 43) **Delete Value by aliasId:** Delete context data using aliasId. Input Parameters:
- id, example `testaliasId`
 - key, example `key1_name`
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdcTouchpoint`
- 44) **Delete aliasId:** Delete aliasId associated with a context. Input Parameters:
- id, example `testaliasId`
 - rid, (optional) example 12
 - ip, (optional) example 127.0.0.1
 - touchpoint, (optional) example `pdcTouchpoint`

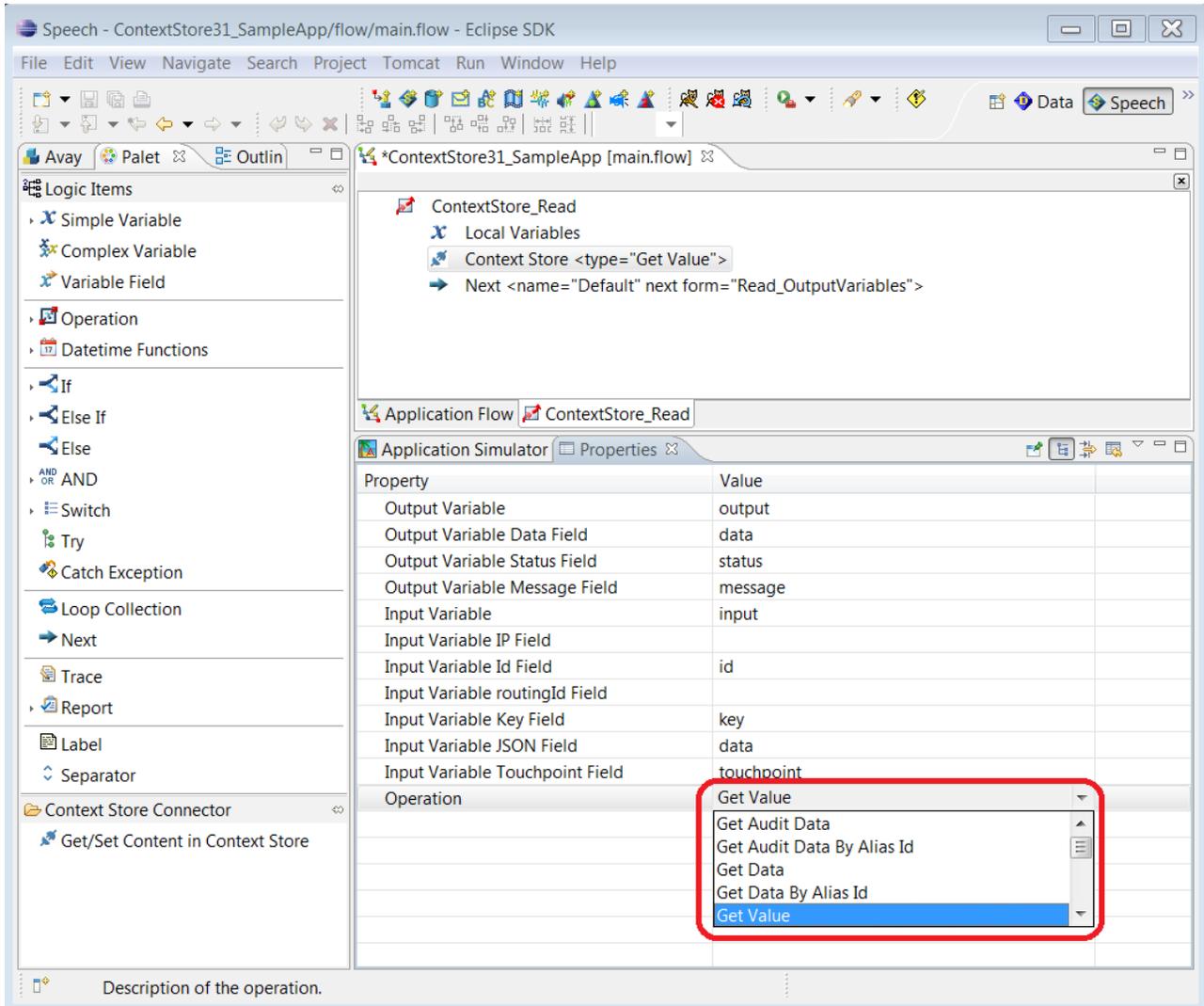


Figure 37 OD: Context Store PDC Operations

7.11.4. Retrieving output variable values

1. To get the output values, you can plug a **Servlet** component after the **Data** component in the workflow.
2. Write a code that seems to the code below in the **Servlet class** associated with the **Servlet component** plugged in the previous step.

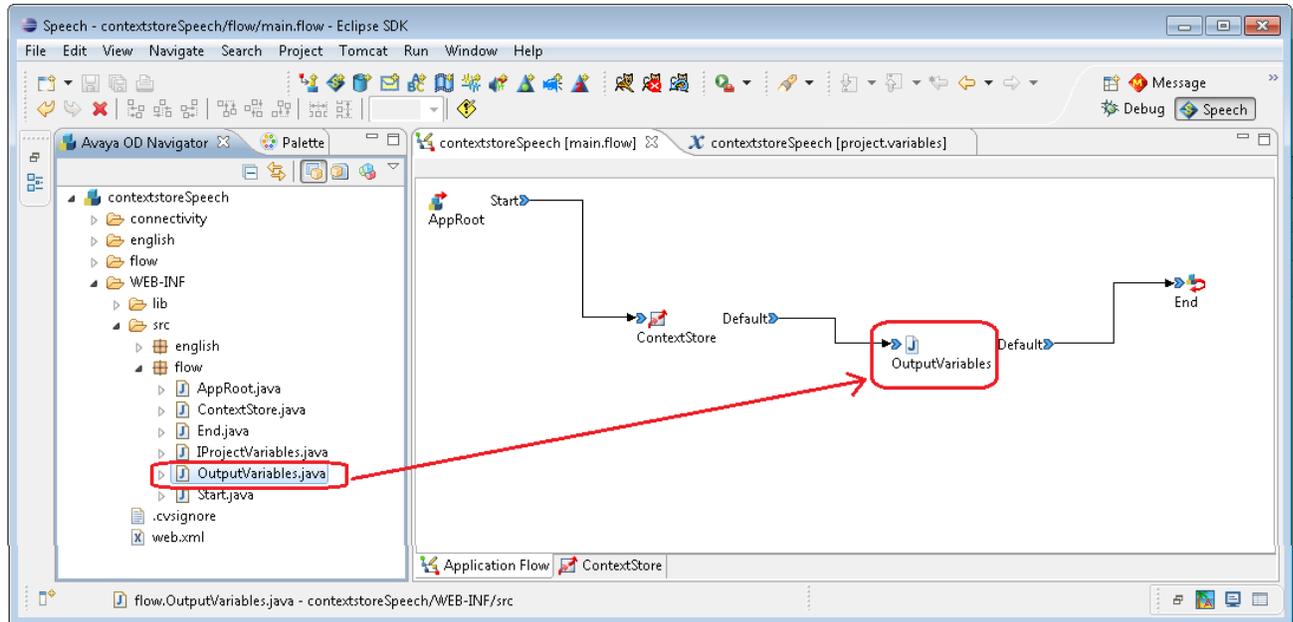


Figure 38 OD: Output Variables

All GET operations (as well as POST operation for auto-generated *contextId*) send back data in the request response under normal circumstances: below are some examples

Operation	Object Returned	Example code to retrieve data
Get Value Get Audit Data Post Context	String	<pre>String outputValue = outvariable.getComplexVariable().getField(IProjectVariables. OUTPUT_FIELD_DATA).getStringValue();</pre>
Get Data	Map <String, Object>	<pre>Map<String, Object> dataMap = (Map<String, Object>) outvariable.getComplexVariable().getField(IProjectVariables. OUTPUT_FIELD_DATA).getObjectValue();</pre>
Get Context Ids	List <Map<String, Object>>	<pre>List<Map<String, Object>> listOfContextData = List<Map<String, Object>>() outvariable.getComplexVariable().getField(IProjectVariables. OUTPUT_FIELD_DATA).getObjectValue();</pre>

Figure 39 OD: Working with Output Variables

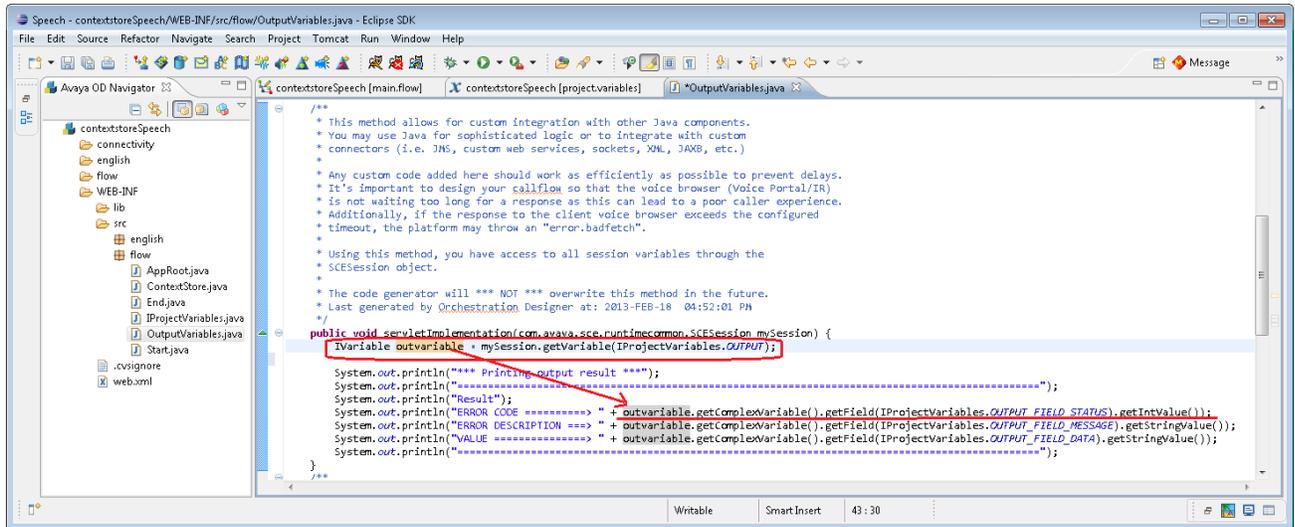


Figure 40 OD: OutputVariables.java

Context Store PDC – Running Sample Project. The only difference is that this project is not wrapped in a zip archive and therefore can be copied directly to your workspace directory.

After the project has been configured and tested in Orchestration Designer, it needs to be exported as a war to run in the Experience Portal environment.

The Orchestration Designer software bundle has documentation packaged with it. The base eclipse folder contains the Avaya Aura Orchestration Designer Developer's Guide pdf. You must read Chapter 25 Application Deployment before attempting to export an application from Orchestration Designer and deploying it on an application server.

Instructions on how to Export the sample application are covered in the section **Exporting an Orchestration Designer project** in the Avaya Aura Orchestration Designer Developer's Guide. For step three, choose Export Orchestration Designer Speech Project. For step six (Specify Platform Details) choose Experience Portal for Platform. The defaults can be chosen for all other steps.

7.11.5. Sample Call Flow Two.

ContextStore_advancedTrafficApp: A second sample flow is provided in the **Context Store PDC Sample Applications** resource. This saves a context without a contextId, retrieves the contextId sent back by the Context Store and saves this to the UCID field of the call. The context is then updated, retrieved, and finally deleted from the Context Store. This project should be imported, configured and tested as above.

Runtimeconfig application

The Orchestration Designer 7.0 pdf Avaya Aura Orchestration Designer Developer's Guide, chapter 25 Application Deployment outlines how to export the files required for the specific application server that will be running your application. The section that deals with this is "Run-time support file export"

Run the steps in the section "Exporting the run-time support files". For step four "On the Export Runtime Support page" select the "Export Orchestration Designer runtime configuration application" tick box

This will output two file runtimeSupportTomcat6.zip and runtimeconfig.war.

7.11.6. Installing the sample application plus runtimeconfig on an Experience Portal system

The section **Deploying the run-time support files** needs to be followed to deploy the runtimeSupportTomcat6 files. These steps can be summarized as:

1. Stop the tomcat service, make a backup of the tomcat lib folder on the tomcat application server. Extract the runtimeSupportTomcat6.zip to the tomcat/lib folder and delete the older versions of any jars in that folder. Copy the runtimeconfig.war into the tomcat/webapps folder.
2. Follow the procedures in the Certificate management in a run-time environment section to configure the certs on the application server. The Certificate Based Authentication section in this document outlines how to configure certs for a Context Store deployment. The jks file should be copied into the tomcat/lib/ folder on the tomcat server.
3. One additional step needs to be run for the certs in addition to what is in the doc. On the tomcat server, edit the file tomcat/lib/trustedcert.properties and change the following two lines to the correct entries for the jks file that was generated:
 - WebLM.trustStore=/ - WebLM.trustStorePassword=<cert password>
4. Copy the <sample application>.war that was exported from Orchestration Designer into the tomcat/webapps folder.
5. Start the tomcat service.

8. Context Store Task Type for Engagement Designer

8.1. Overview

The Context Store Task Type is the node in the Engagement Designer (ED) that interfaces with Context Store. The CS Task Type enables Context Store operations to be run from within a workflow in ED.

Using the CS Task Type node; users can perform Add, Update, Retrieve, Upsert and Delete operations for both contexts and values within an ED flow.

The CSTasks service requires the Engagement Designer platform hence the CSTasks SVAR can only be installed on an Engagement Designer Cluster.

For installation instructions see the Avaya Context Store Snap-In 3.1 Reference Guide.

8.2. Usage

8.2.1. Input and Output

Engagement Designer nodes pass data into the CS Task node via six input parameters csIdInput, csRoutingIdInput, csAliasIdInput, csTouchpointInput, csKeyInput and csDataInput. csIdInput will only ever contain a Context Id or Group Id.

- csRoutingIdInput will only ever contain the routing Id of the for the context. This parameter is used for the Context Store Geo-Redundancy feature
- csAliasIdInput will only ever contain the aliasIds for a context. The data should be a comma separated string. e.g. alias1,alias2.
- csTouchpointInput will only ever contain a string to define a touchpoint. This parameter is used for the Context Store Audit Feature.
- csKeyInput will only ever contain a key name.
- csDataInput can contain slightly different data depending on the operation required.

Not every operation requires all of the six input values to contain data.

Required Input Mapping/Parameters

Operation	csIdInput	csRoutingIdInput	csAliasIdInput	csTouchpointInput	csKeyInput	csDataInput
Add Context	No	Optional	Optional	Optional	No	Yes
Add Context without Context Id	No	Optional	Optional	Optional	No	Yes
Delete Alias Id	No	Optional	Yes	Optional	No	No
Delete Context	Yes	Optional	No	No	No	No
Delete Context by	No	Optional	Yes	No	No	No

Context Store Task Type for Engagement Designer

Alias Id						
Delete Key	Yes	Optional	No	Optional	Yes	No
Delete Key by Alias Id	No	Optional	Yes	Optional	Yes	No
Get Context	Yes	Optional	No	Optional	No	No
Get Context Audit Data	Yes	Optional	No	No	No	No
Get Context Audit Data by Alias Id	No	Optional	Yes	No	No	No
Get Context by Alias Id	No	Optional	Yes	Optional	No	No
Get Context Ids for Group Id	Yes	Optional	No	No	No	No
Get Key	Yes	Optional	No	Optional	Yes	No
Get Key by Alias Id	No	Optional	Yes	Optional	Yes	No
Update Alias Ids by Alias Id	No	Optional	Yes	Optional	No	Yes
Update Alias Ids by Context Id	Yes	Optional	No	Optional	No	Yes
Update Context	Yes	Optional	No	Optional	No	Yes
Update Context by Alias Id	No	Optional	Yes	Optional	No	Yes
Update Key	Yes	Optional	No	Optional	Yes	Yes
Update Key by Alias Id	No	Optional	Yes	Optional	Yes	Yes
Upsert Context	Yes	Optional	No	Optional	No	Yes
Upsert Context by Alias Id	No	Optional	Yes	Optional	No	Yes

Output Parameters

When an operation returns successfully, the items in the table below are populated. When an error occurs, the `csStatusOutput` and `csMessageOutput` parameters will be populated. The `csObjectOutput` can be populated with an object, the format which is specified by the user via an output schema.

Operation	csStatusOutput	csMessageOutput	csDataOutput	csObjectOutput
Add Context	Yes	No	No	
Add Context without Context Id	Yes	No	Yes	
Get Context	Yes	No	Yes	Yes
Get Context by Alias Id	Yes	No	Yes	Yes
Get Context Ids for Group Id	Yes	No	Yes	
Get Context Audit Data	Yes	No	Yes	
Get Context Audit Data by Alias Id	Yes	No	Yes	
Get Key	Yes	No	Yes	Yes
Get Key by Alias Id	Yes	No	Yes	Yes
Update Context	Yes	No	No	
Update Context by Alias Id	Yes	No	No	
Update Key	Yes	No	No	
Update Key by Alias Id	Yes	No	No	
Update Alias Ids by Context Id	Yes	No	No	
Update Alias Ids by Alias Id	Yes	No	No	
Upsert Context	Yes	No	No	
Upsert Context by Alias Id	Yes	No	Yes	
Delete Context	Yes	No	No	
Delete Context by Alias Id	Yes	No	No	
Delete Alias	Yes	No	No	

Delete Key	Yes	No	No	
Delete Key by Alias Id	Yes	No	No	

8.2.2. CS Task Type Operations

All operations which can be executed using the CS Task Type in ED 3.1 are listed below with required inputs, optional inputs and sample JSON for the specific request. Example workflows for each of these operations can be downloaded from the [Avaya Context Store DevConnect](#) site

Add Context

Using the Add Context operation, a new Context can be added to the Context Store.

- The required input: is csDataInput
- The optional inputs: lease, csRoutingIdInput, csAliasIdInput, csTouchpointInput

```
{
  "workflowversion": "1",
  "alias": "null",
  "imessage": {
    "csIdInput": "",
    "csRoutingIdInput": "1",
    "csTouchpointInput": "EDFlowTest",
    "csAliasIdInput": "DemoAlias1, DemoAlias2",
    "csKeyInput": "",
    "csDataInput": {
      "contextId": "DemoContext1",
      "persistToEDM": "true",
      "persistTo": "CS_PROVISION",
      "tenantId": "demo",
      "groupId": "demo",
      "data": {
        "key1_name": "value1_data",
        "key2_name": "value2_data"
      }
    }
  },
  "iprocess": "AddContext",
  "customer": "customer"
}
```

Add Context without Context Id

Using the Add Context without Context Id operation, a new Context with an auto generated id can be added to the Context Store.

- The required input: csDataInput
- The optional inputs: lease, csRoutingIdInput, csAliasIdInput, csTouchpointInput

```
{
  "workflowversion": "1",
  "alias": "null",
  "imessage": {
    "csIdInput": "",
    "csRoutingIdInput": "1",
    "csTouchpointInput": "EDFlowTest",
    "csAliasIdInput": "DemoAlias3, DemoAlias4",
    "csKeyInput": "",
    "csDataInput": {
      "contextId": "",
      "persistToEDM": "true",
      "tenantId": "demo",
      "groupId": "demo",
      "data": {
        "key1_name": "value1_data",
        "key2_name": "value2_data"
      }
    }
  },
  "iprocess": "AddContextWithouContextId",
  "customer": "customer"
}
```

Get Context

Using the Get Context operation, a previously stored Context can be retrieved from the Context Store using Context Id associated to that Context.

- The required input: csIdInput
- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{
  "workflowversion": "1",
  "alias": "null",
  "imessage": {
    "csIdInput": "DemoContext1",
    "csRoutingIdInput": "1",
    "csTouchpointInput": "EDFlowTest"
  },
  "iprocess": "GetContext",
  "customer": "customer"
}
```

Get Context Ids with Alias Id

Using the Get Context with Alias Id operation, a previously stored Context can be retrieved from the Context Store using Alias Id associated to that Context.

- The required input: csAliasIdInput

- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{"workflowversion":"1","alias":"null","imessage":{"csRoutingIdInput":"1","csTouchpointInput":"EDFlowTest","csAliasIdInput":"DemoAlias1"},"iprocess":"GetContextByAliasId","customer":"customer"}
```

Get Context Ids for Group Id

Using the Get Context Ids for Group Id operation, the contextIds of all Contexts belonging to a group can be retrieved from a Context Store.

- The required input: csIdInput
- The optional inputs: csRoutingIdInput

```
{"workflowversion":"1","alias":"null","imessage":{"csIdInput":"demo","csRoutingIdInput":"1"},"iprocess":"GetContextIdsForGroupId","customer":"customer"}
```

Get Context Audit Data

Using the Get Context Audit Data operation, the audit data of a Context in a Context Store can be retrieved using Context Id associated to that Context.

- The required input: csIdInput
- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{"workflowversion":"1","alias":"null","imessage":{"csIdInput":"DemoContext1","csRoutingIdInput":"1"},"iprocess":"GetContextAuditData","customer":"customer"}
```

Get Context Audit Data by Alias Id

Using the Get Context Audit Data by Alias Id operation, the audit data of a Context in a Context Store can be retrieved using Alias Id associated to that Context.

- The required input: csAliasIdInput
- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{"workflowversion":"1","alias":"null","imessage":{"csRoutingIdInput":"1","csAliasIdInput":"DemoAlias1"},"iprocess":"GetContextAuditDataByAliasId","customer":"customer"}
```

Get Key

Using the Get Key operation, the data associated to a key in the Context can be retrieved using context id from a Context Store.

- The required input: csIdInput, csKeyInput
- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{"workflowversion":"1","alias":"null","imessage":{"csIdInput":"DemoContext1","csRoutingIdInput":"1","csTouchpointInput":"EDFlow","csKeyInput":"key1_name"},"iprocess":"GetKey","customer":"customer"}
```

Get Key by Alias Id

Using the Get Key by Alias Id operation, the data associated to a key in the Context can be retrieved using alias Id associated to that context from a Context Store.

- The required input: csAliasIdInput, csKeyInput
- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{"workflowversion":"1","alias":"null","imessage":{"csAliasIdInput":"DemoAlias1","csRoutingIdInput":"1","csTouchpointInput":"EDFlow","csKeyInput":"key1_name"},"iprocess":"GetKeyByAliasId","customer":"customer"}
```

Update Context

Using the Update Context operation, a Context in Context Store can be updated using Context Id associated to that Context.

- The required input: csIdInput, csDataInput
- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{"workflowversion":"1","alias":"null","imessage":{"csIdInput":"DemoContext1","csRoutingIdInput":"1","csTouchpointInput":"EDFlow","csDataInput":{"data":{"key3_name":"value3_data","key4_name":"value4_data"}}},"iprocess":"UpdateContext","customer":"customer"}
```

Update Context by Alias Id

Using the Update Context operation, a Context in Context Store can be updated using Alias Id associated to that Context.

- The required input: csAliasIdInput, csDataInput
- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{"workflowversion":"1","alias":"null","imessage":{"csAliasIdInput":"DemoAlias1","csRoutingIdInput":"1","csTouchpointInput":"EDFlow","csDataInput":{"data":{"key5_name":"value5_data","key6_name":"value6_data"}}},"iprocess":"UpdateContextByAliasId","customer":"customer"}
```

Update Key

Using the Update Key operation, the value of a key associated to a Context in Context Store can be updated using Context Id associated to that Context.

- The required input: csIdInput, csKeyInput, csDataInput
- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{"workflowversion":"1","alias":"null","imessage":{"csIdInput":"DemoContext1","csRoutingIdInput":"1","csKeyInput":"key1_name","csTouchpointInput":"EDFlow","csDataInput":"updated_value1"},"iprocess":"UpdateKey","customer":"customer"}
```

Update Key by Alias Id

Using the Update Key by Alias Id operation, the value of a key associated to a Context in Context Store can be updated using Alias Id associated to that Context.

- The required input: csAliasIdInput, csKeyInput, csDataInput
- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{"workflowversion":"1","alias":"null","imessage":{"csAliasIdInput":"DemoAlias1","csRoutingIdInput":"1","csKeyInput":"key2_name","csTouchpointInput":"EDFlow","csDataInput":"update_d_value2"},"iprocess":"UpdateKeyByAliasId","customer":"customer"}
```

Update Alias Ids by Context Id

Using the Update Alias Ids by Context Id operation, the Aliases associated to a Context in Context Store can be updated using Context Id associated to that Context.

- The required input: csIdInput, csDataInput
- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{"workflowversion":"1","alias":"null","imessage":{"csIdInput":"DemoContext1","csRoutingIdInput":"1","csTouchpointInput":"EDFlow","csDataInput":["\\"DemoAlias6\\""]},"iprocess":"UpdateAliasIdsByContextId","customer":"customer"}
```

Update Alias Ids by Alias Id

Using the Update Alias Ids by Alias Id operation, the Aliases associated to a Context in Context Store can be updated using Alias Id associated to that Context.

- The required input: csAliasIdInput, csDataInput
- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{"workflowversion":"1","alias":"null","imessage":{"csAliasIdInput":"DemoAlias1","csRoutingIdInput":"1","csTouchpointInput":"EDFlow","csDataInput":["\\"DemoAlias7\\""]},"iprocess":"UpdateAliasIdsByAliasId","customer":"customer"}
```

Upsert Context

Using the Upsert Context operation, a Context in Context Store can be created or updated using Context Id associated to that Context.

- The required input: csIdInput, csDataInput
- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{"workflowversion":"1","alias":"null","imessage":{"csIdInput":"DemoContext1","csRoutingIdInput":"1","csTouchpointInput":"EDFlow","csDataInput":{"\\"data\":"\\"key1_name\":"\\"value1_data\","key2_name\":"\\"value2_data\\""}}},"iprocess":"UpsertContextByContextId","customer":"customer"}
```

Upsert Context by Alias Id

Using the Upsert Context operation, a Context in Context Store can be created or updated using Alias Id associated to that Context.

- The required input: csAliasIdInput, csDataInput
- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{"workflowversion":"1","alias":"null","imessage":{"csAliasIdInput":"DemoAlias1","csRoutingIdInput":"1","csTouchpointInput":"EDFlow","csDataInput":{"\\"data\":"\\"key1_name\":"\\"value1_data\","key2_name\":"\\"value2_data\\""}}},"iprocess":"UpsertContextByAliasId","customer":"customer"}
```

Delete Context

Using the Delete Context operation, the Context in Context Store can be deleted using Context Id associated to that Context.

- The required input: csIdInput
- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{"workflowversion":"1","alias":"null","imessage":{"csIdInput":"DemoContext1","csRoutingIdInput":"1","csTouchpointInput":"EDFlow"},"iprocess":"DeleteContext","customer":"customer"}
```

Delete Context by Alias Id

Using the Delete Context by Alias Id operation, the Context in Context Store can be deleted using Alias Id associated to that Context.

- The required input: csAliasIdInput
- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{"workflowversion":"1","alias":"null","imessage":{"csAliasIdInput":"DemoAlias1","csRoutingIdInput":"1","csTouchpointInput":"EDFlow"},"iprocess":"DeleteContextByAliasId","customer":"customer"}
```

Delete Alias Id

Using the Delete Alias Id operation, the Alias Id of a Context in Context Store can be deleted.

- The required input: csAliasIdInput
- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{"workflowversion":"1","alias":"null","imessage":{"csAliasIdInput":"DemoAlias1","csRoutingIdInput":"1","csTouchpointInput":"EDFlow"},"iprocess":"DeleteAlias","customer":"customer"}
```

Delete Key

Using the Delete Key operation, the Key in a Context in Context Store can be deleted using Context Id associated to that Context

- The required input: csIdInput, csKeyInput
- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{"workflowversion":"1","alias":"null","imessage":{"csIdInput":"DemoContext1","csRoutingIdInput":"1","csTouchpointInput":"EDFlow","csKeyInput":"key1_name"},"iprocess":"DeleteKey","customer":"customer"}
```

Delete Key by Alias Id

Using the Delete Key by Alias Id operation, the Key in a Context in Context Store can be deleted using Alias Id associated to that Context

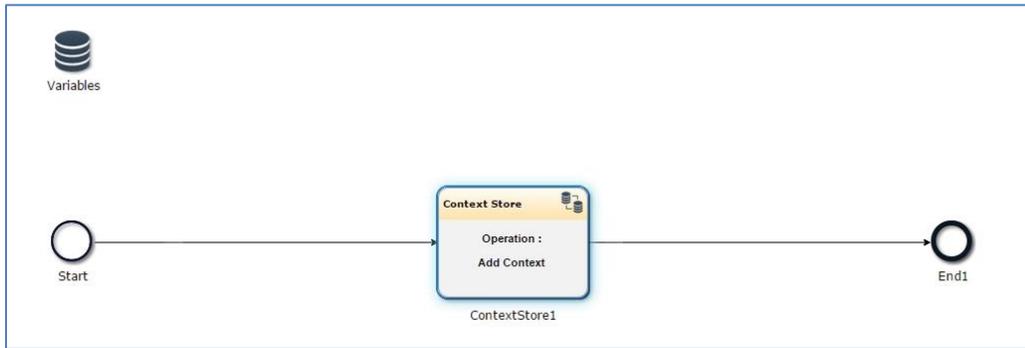
- The required input: csAliasIdInput, csKeyInput
- The optional inputs: csRoutingIdInput, csTouchpointInput

```
{"workflowversion":"1","alias":"null","imessage":{"csAliasIdInput":"DemoAlias1","csRoutingIdInput":"1","csTouchpointInput":"EDFlow","csKeyInput":"key1_name"},"iprocess":"DeleteKeyByAliasId","customer":"customer"}
```

8.2.3. Creating a workflow in ED

This section serves as a quick start guide on how to create a workflow in ED to test the Context Store Engagement Designer task type. For more detailed information about this product, please refer to Avaya Engagement Designer documentation.

The following is an example to create a flow of "Add Context" operation of CS Task Type. This flow will save a Context with a Context Id to the context store. Note the Variable does not need to be connected to the other nodes in the flow, it can be thought of as a global variable that the nodes can read and write data from.



Location of the nodes that make up this flow



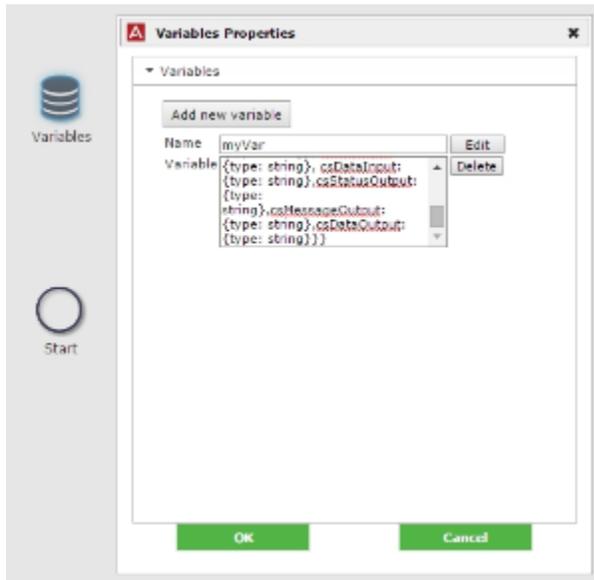
Drag the nodes into the flow and connect up as in the first image above. Please note the Variables node, which should already exist on the workflow, does not need to be connected to other node.

Configuring the Variables Node

Right click on the Variables node and select properties, enter myVar in the name text box and enter the following json in the schema box. See diagram below.

```
{title: ContextStoreSchema,type: object, properties: {csIdInput: {type: string},  
csRoutingIdInput: {type: string}, csAliasIdInput: {type: string},  
csTouchpointInput: {type: string}, csKeyInput: {type: string}, csDataInput:
```

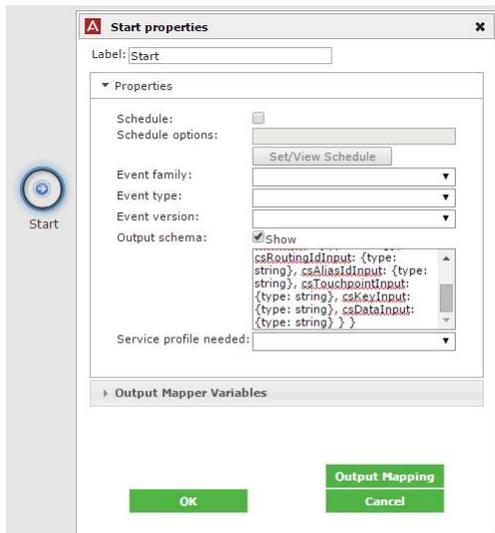
```
{type: string},csStatusOutput: {type: string},csMessageOutput: {type: string},csDataOutput: {type: string}}}
```



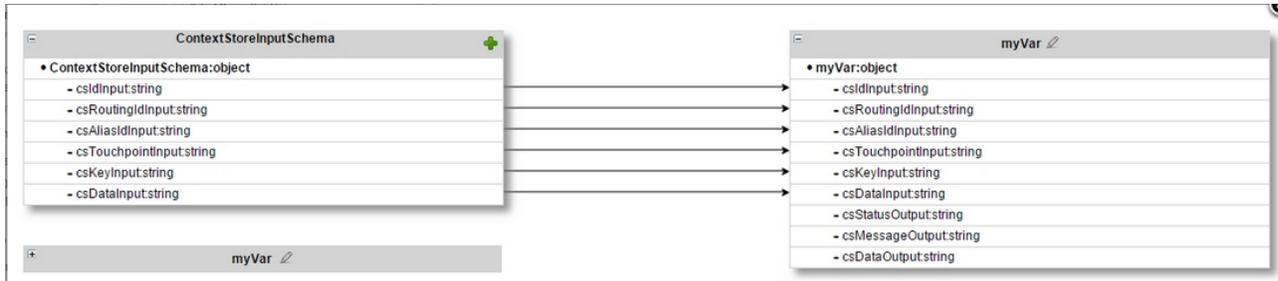
Configuring the Start Node

Right click on the Start node and select properties, enter the following json in the OutputSchema box. See diagram below.

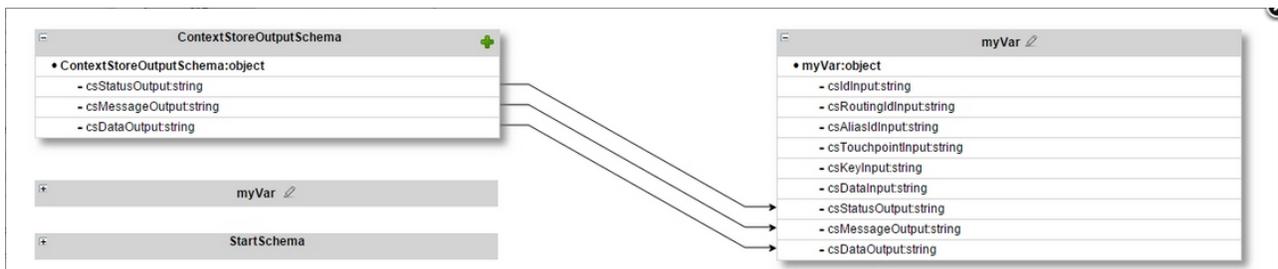
```
{title: ContextStoreInputSchema,type: object, properties: { csIdInput: {type: string}, csRoutingIdInput: {type: string}, csAliasIdInput: {type: string}, csTouchpointInput: {type: string}, csKeyInput: {type: string}, csDataInput: {type: string} } }
```



In addition to adding the output schema for the start node the output mapping also needs to be configured. Right click on the Start node and select properties click the output mapping button. On the diagram that is displayed link the properties as follows. See the diagram below.



To set the output mappings right click on the Context Store Node and select properties click the Output mapping button. On the diagram that is displayed link the properties as follows. See the diagram below.



Configuring the End Node

You can configure the End node the same way as the start node and provide a schema and mapping to intercept the incoming data.

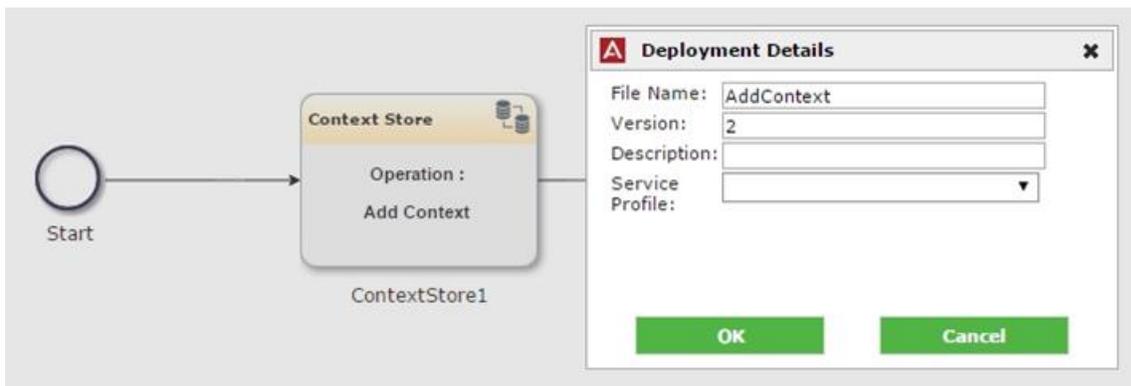
Deploying and testing the Workflow

In order for an individual workflow to be available it needs to be deployed.

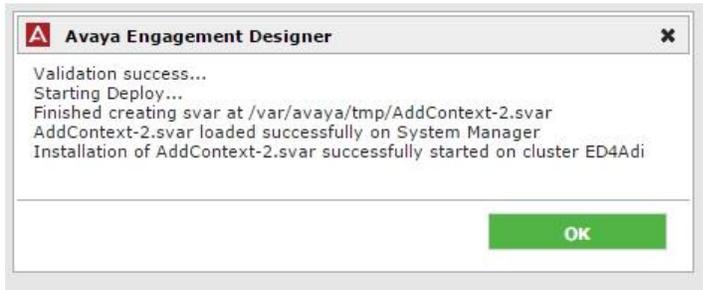


To deploy the flow created using above steps, first validate the workflow and then save it using a meaningful name. After saving click on the "Deploy Workflow" button on the menu toolbar.

Enter in a Name and version number for the flow and click OK. Verify that workflow was deployed without any errors.



Please take note of the Name and the Version number as you would need it to use the flow. A confirmation similar to below will appear.

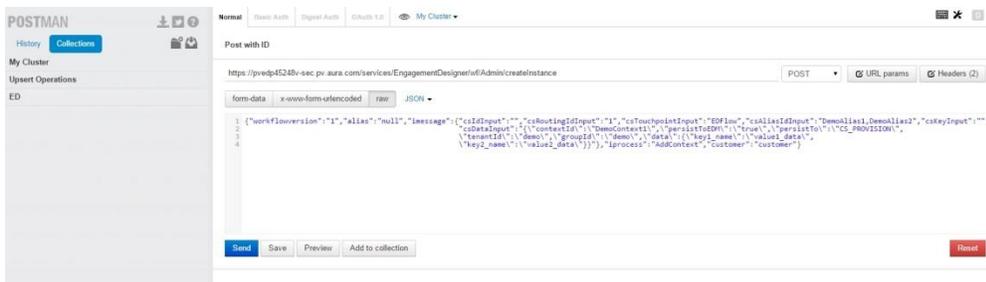


Check the *Service Management* tab in the *Engagement Development Platform* Page of SMGR to confirm if deployment has completed before using the flow.

Verifying The Output From A Workflow.

For a simple workflow like the example above the only current way to verify the output from the CS node apart from checking a context via some other client is to view the log files for the output.

The log files for Context Store are located in the folder `/var/log/Avaya/services/CSRest/` on the EDP server.



Using PostMan (a Google Chrome Extension), send in the request to the ED Cluster.

After the request has been completed, check Context Store logs or retrieve the Context with "contextId=DemoContext1" and "rid=1" using some other client to confirm the success of the request.

8.2.4. Importing the Sample Workflow into the Designer

A sample workflow similar to described above can be downloaded from <http://www.avaya.com/devconnect>.

The downloaded workflow can be imported into the Engagement Designer with the following steps.

Open the download file in workflow.



Click on the "Import Workflow from file" icon.

8.2.5. Validation

Each workflow should be validated before deployment. A workflow can be validated using the Validation icon in menu bar.

The screenshot shows the Araya Engagement Designer interface. The top menu bar includes a 'Validate Workflow' icon. The main workspace displays a workflow diagram with nodes: 'Start', 'ContextStore1' (containing 'Operation: Add Context'), and 'End1'. A 'Zoom view' window is open at the bottom, displaying the following validation error log:

ContextStore1: A value for the Context Store Server must be supplied
ContextStore1: No mapping has been provided. Please provide mapping for required fields.
ContextStore1: No end found after the node ContextStore1
Start: No end found after the node Start
1 Warning(s)
End1: There is an orphan node: End1

If the parameters in CS task have been filled, then the preference is given to these values during the execution of the Tasks. The provided value must conform to the existing validations rules. If a parameter is not provided in the properties then a mapping has to be provided for the specific input. e.g. if the parameter value for "Key Name" in properties is left blank and it is needed for the requested operation then the validation will fail. In such events the required fields, which are missing to complete the operation will be listed.

The screenshot shows the Araya Engagement Designer interface. The main workspace displays a workflow diagram with nodes: 'Start' and 'End1'. A 'Zoom view' window is open at the bottom, displaying the following validation error log:

1 Error(s)
ContextStore1: An input mapping for following required field(s) is missing, csDataInput.
0 Warning(s)

8.2.6. Re-use of Collaboration Designer 3.0 Workflows

Flows which were created with Collaboration Designer 3.0 require re-mapping if imported into Engagement Designer 3.1 which can be time consuming. Therefore, if possible, it is recommended that flows are recreated in ED 3.1 instead.

If flow created with Collaboration Designer 3.0 must be re-used, please follow the steps below.

1. First create a script called *portFlowToED31.sh* on the ED 3.1 node containing the following code:

```
if [ $# -eq 0 ]
then
echo "Usage: portFlowToOSGi.sh <path to ED flow xml file>"
```

```
        exit
    fi

    inputFile="$1"

    if [ ! -e $inputFile ]
    then
        echo "Can't find file" $inputFile
        exit
    fi

    newFileName=`echo "$inputFile" | sed s/.xml/-OSGi.xml/`
    cat "$inputFile" | sed s/com.avaya.designer.task.cs/com.avaya.ingensg.tasks.cs/g >
    $newFileName

    echo "New flow file is " $newFileName
```

2. Copy the old flow (<flow-name>.xml) to the ED 3.1 node on which the script was created_
3. Change to the directory which contains the old flow
4. Run the script using the following command: *portFlowToED31.sh* <flow-name>.xml
5. This will produce a new xml file which is compatible with ED 3.1. This new flow would have to be re-mapped before it can be used
6. Import the flow into the ED and open each individual CS Task Type node and remap the missing mapping links.

9. Notifications

9.1. Overview

The Context Store Notifications feature allows users to configure up to five subscriptions. Each subscription must have a valid endpoint to consume the notification stream. The consumer endpoint must be capable of supporting the generated notification traffic. For every read, create, update and delete operation on a Context, a notification will be sent to registered clients.

The notification sent to clients will contain a complete copy of the Context state so the payload could be KB's in size. Care must be taken when developing applications against this feature to account for throughput both rate and capacity. At the maximum supported traffic rate of 1240 requests/second with 2KB of data this will generate ~600 notifications/second with 2KB for each subscription. The system as a whole will be generating over 3000 notifications/second if all five subscriptions are registered.

Note the supported traffic rate varies with the Context Store deployment size; configuration and supported rates are documented in the Context Store Reference Guide.

9.2. Security Configuration

Clients of CSNotifications must be configured to use the keystore generated and downloaded from the SMGR managing the Context Store cluster. There are several options available when configuring certificates using SMGR. To help getting started, one particular approach is documented in the Appendix of this document - see section [11.2.2 Create Client Keystore](#).

1. Configure the CSNotifications client(s) with the required security certificate.

E.g. For client running in Jetty container; the keystore, `<CERT-NAME>.jks` must be copied to the `/etc/` folder in the Jetty instance. Configure the Jetty instance for SSL as follows.

1. In `start.ini` file, After line

```
jetty.dump.stop=false
```

add the following lines:

```
etc/jetty-ssl.xml
```

```
etc/jetty-https.xml
```

2. In Jetty `/etc/jetty-ssl.xml` add or re-configure `KeyStorePath` and `KeyStorePassword`

```
<Set name="KeyStorePath"><Property name="jetty.base" default="." /></Property
name="jetty.keystore" default="etc/<CERT-NAME>.jks"/></Set>
<Set name="KeyStorePassword"><Property name="jetty.keystore.password"
default="<PASSWORD>"/></Set>
```

Note: `TrustStore` and `KeyManagerPassword` configuration are not required

2. After installing the CSNotifications snap-in, all subscriptions are configured via the services attributes.

E.g. `{"endpointURI":"https://<IP_ADDRESS>:<PORT>/NotificationsConsumer/notifications/consumer/", "tenantId":"customer123", "groupId":"productABC", "enabled":"false"}`

Note: All configurations must be done at runtime; any attribute changes made before the service is installed will not be processed. The service can't be deployed with subscriptions already enabled; the subscription can be created and simply enabled once the service has been installed in the cluster (all nodes reporting service installed).

9.3. Usage

This feature provides a notification stream for client to consume all operations on the Context Store Rest interface will result in a notification being sent. Client must write a consumer Rest application which will accept a Post operation with the JSON structure outlined below.

```
{ "subscriptionId": "cs.subscription.1", "contextId": "testId", "aliasIds": "abc,123", "tenantId": "someId", "groupId": "testGroupId", "operation": "WRITE", "versionId": "1", "createTimestamp": "1406793988683", "updatedTimestamp": "1406793991234", "data": { "key1": "value1", "key2": "value2", "key3": "value3" } }
```

- "subscriptionId" The subscription Id which this notification was generate for, mandatory.
- "contextId" The Id of the context which triggered this notification, mandatory.
- "aliasIds" The aliasIds associated with the context, optional*.
- "groupId" The groupId specified when the context was created, optional*.
- "tenantId" The tenantId specified when the context was created, optional*.
- "operation" The operation which triggered this notification, values are WRITE,UPDATE and DELETE.
- "versionId" The version of the context which triggered this notification. This value is an integer which is incremented on every interaction with the context, value on create context is 1 so subsequent update would be 2 etc. This field can be used to distinguish between multiple interactions. The versionId will not be incremented by the following requests – get audit data using contextId, get audit data using aliasId and get contexts using groupId.
- "createTimestamp" The time the context was created, it is a long, milliseconds since Epoch.
- "updatedTimestamp" The time the context was updated, it is a long, milliseconds since Epoch.
- "data" The complete data which is associated with the context. Note this could be a large amount of data.

*optional fields may not be populated.

The following are code snippets of a sample application which consumes the notification stream. Sample client code is available on the [Avaya Context Store DevConnect](#) site.

StateEvent class will take in JSON and create a java object which can be passed around in the application.

```
@XmlElement(name = "StateEvent")
public class StateEvent {
    @XmlElement(name = "contextId")
    private String contextId;
    @XmlElement(name = "aliasIds")
    private String aliasIds;
    @XmlElement(name = "groupId")
    private String groupId;
    @XmlElement(name = "tenantId")
    private String tenantId;
    @XmlElement(name = "versionId")
    private String versionId;
    @XmlElement(name = "data")
```

```
private Map<String, Object> data = new HashMap<>();
@XmlElement(name = "createTimestamp")
private String createTimestamp;
@XmlElement(name = "updatedTimestamp")
private String updatedTimestamp;
@XmlElement(name = "operation")
private String operation;
@XmlElement(name = "subscriptionId")
private String subscriptionId;
public StateEvent() {
}

public StateEvent(String contextId, String aliasIds, String groupId, String tenantId,
String versionId, Map<String, Object> data, String createTimestamp, String
updatedTimestamp) {
    this.contextId = contextId;
    this.aliasIds = aliasIds;
    this.groupId = groupId..... Continue for all fields
}

Getters and setters would be here
} //end of class
```

The application itself, JAX-RS

```
@Path("/consumer/")
@ApplicationPath("notifications")
public class NotificationsConsumerApp extends Application {

    public NotificationsConsumerApp() {
    }

    @POST
    @Consumes(MediaType.APPLICATION_JSON)
    public Response postStateEvent(StateEvent stateEvent) {

        // Printing out the contents of the of the Notification
        //200 Ok response must be sent back to the server otherwise it will resend the
notification,
        //delays in sending the response could impact performance on the server.
        return Response.ok().entity("StateEvent Received, contextId:"
            + stateEvent.getContextId()).build();
    }
}
```

9.4. Performance/Capacity

When running on the supported hardware configuration, please see the CPU section. CSNotifications was designed to service up to five subscriptions at the rate of 620 notifications per second per subscription. This is based on a maximum incoming traffic rate of 1240 requests/second (broken down into 50% Post/Put requests and 50% Get requests) supported by Context Store's ReST interface. Note that this throughput is not possible using smaller Context Store deployments (less than 3 EDP nodes with 64GB of memory each). The above was engineered for a Context data size of 2KB's.

Note that every notification has a complete copy of the Context state so on an update the complete data is sent not just what has changed. The final state is always sent not the delta and in this release there is no way to mark the deltas.

If the External DataMart feature is enabled the supported throughput for notifications is reduced by half. For example, in a Context Store deployment which supports maximum throughput (i.e. 1,240 requests per second) with these features disabled, half this rate (i.e. 620 requests per second) is supported with up to five subscriptions plus EDM enabled. As stated above, this traffic breaks down to 50% Post/Put and 50% Get requests.

The consumer endpoint associated with each subscription must be engineered to support the required throughput. For max traffic this is just over 600 notifications per second with a payload matching the Context which triggered it. Care must be taken in how the notifications are handled so as to send a response back to Context Store as soon as possible on receipt of the notification. If the notification is not acknowledged within a certain time duplicate notifications may be received and also the liveliness of the system will be impacted as the notifications for that subscription back up. It is advised that the Rest application hosting the endpoint should off load the notification to another thread/threadpool for processing to limit the build-up of notification requests on the web container hosting the client Rest application.

9.5. High Availability and Fault Tolerance

The Context Store Notifications feature utilizes the underlying HA support of Context Store and so supports the same failure cases, i.e. critical component failure, complete server failure.

To handle problems on the notifications receiver/consumer end, if a notification attempt fails it will retry to send this notification to that subscriber endpoint twice more in the next 40 second period. If after retrying we are still unable to send the notification the subscription will be disabled. The disabling of the subscription is to protect the Context Store. As mentioned in the previous sections each Subscription is pushed a notification stream which matches the inputs into the context store.

For a full traffic system this equate to just over 600 notifications a second. If an endpoint was not available for a long period of time, 30/60 seconds, then this would create a large buffer of notifications. These notifications would impact the memory off the Context Store and also possible affect its liveliness. The retry mechanism does allow for temporary outages due to network issues/ client side issues. If a subscription has to be disabled a corresponding Alarm is generated and is raised to the System Manager. The log viewer will show the actual details of the alarm including which Subscription has been disabled.

The alarm can only be cleared by a user updating the subscription in question from Element Manager -> Attributes -> Service Clusters -> CSNotifications. It is the user who must verify the status of the consumer endpoint before re-enabling the subscription. To re-enable a subscription after it has been disabled automatically, the user must toggle the subscription. This is achieved by first disabling the subscription, "enable": "false" in the attribute, clicking Commit, and waiting a few moments for the Element manager to update, the enabling again, "enable": "true" and clicking Commit. It may take a short while for the system to propagate the change to all machines before the subscription becomes active.

Event Tracker (Agent Notifications)

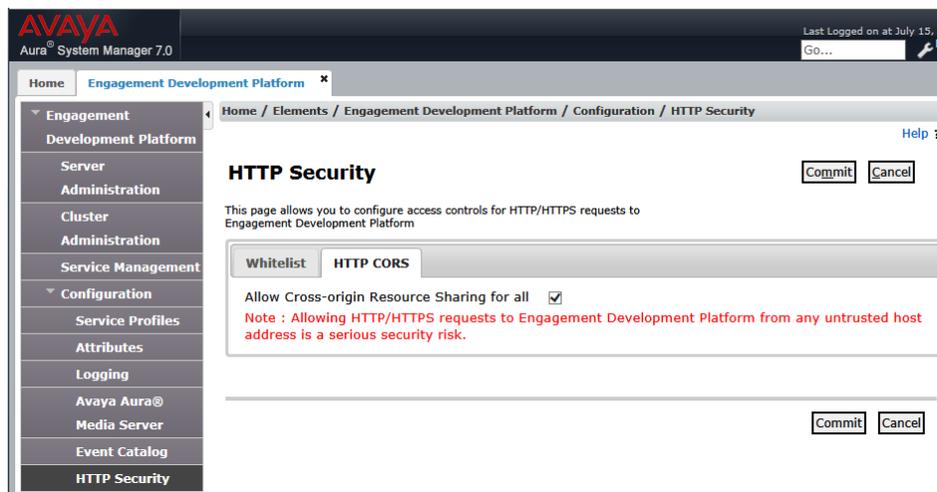
9.6. Overview

This feature services a need for flexible, customizable, fine-grained event notifications. Individual users, such as an agent or agent supervisor, can register for personalized event notification streams. A sample JavaScript web interface is provided to demonstrate the usage of this feature. Event streams are registered using this web interface and the applicable event notifications are delivered directly to the registered user through that same interface

9.7. Configuration

9.7.1. Configure EDP and Context Store

1. Enable Context event streaming, which facilitates this feature, through CSManger attribute *Event Stream: Enable Event Streaming* as described in the Context Store Snap-In Reference Guide.
2. Install the Streams SVAR which processes the Context events in relation to all currently registered user event streams.
3. Configure EDP to allow cross-origin resource sharing (CORS). The client registration relies on an AJAX call to the ReST service at `<server>/endpoint/cometd` and CORS must be allowed for this call to be successful.
 - a. Browse to **Configuration->HTTP Security** in the EDP Element Manager section of SMGR
 - b. Select the HTTP CORS tab.
 - c. Check the **Allow Cross-origin Resource Sharing for all** checkbox as shown below:



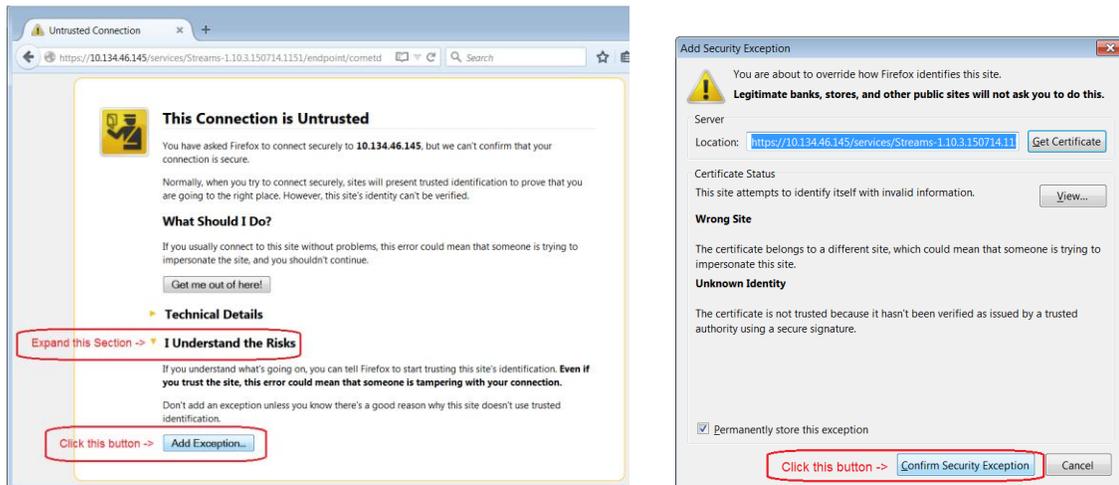
Streams Service Endpoint

If you wish to use an unsecured cometD connection to the streams service, skip the **Install Security Certificates** section that follows and use URL http://<IP_ADDRESS>/services/Streams-<version>/endpoint/cometd in your client.

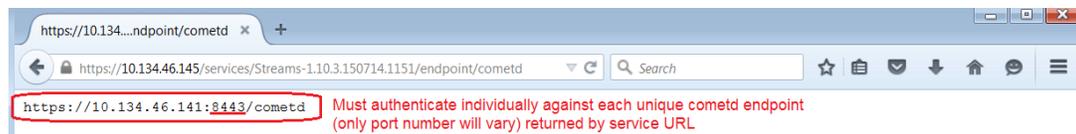
9.7.2. Install Security Certificates

If you wish to use a secure connection to the streams service you must install all the required, EDP-generated SSL certificates in your browser, as described in this step, before proceeding.

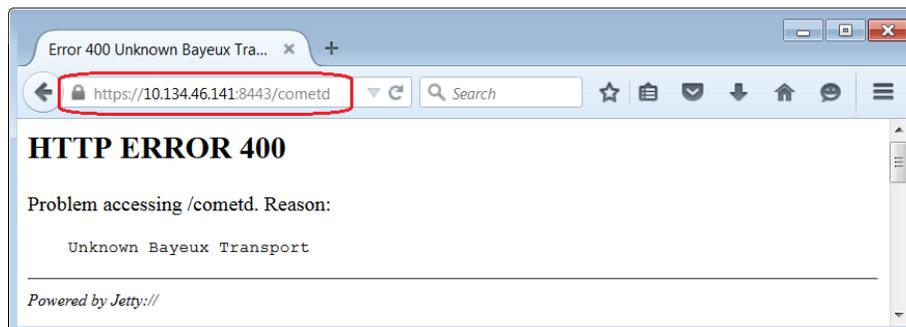
1. Browse to the Streams service's secured cometd endpoint URL in your browser and accept the certificate - https://<IP_ADDRESS>/services/Streams-<version>/endpoint/cometd



2. You should now see a short-form of the cometd endpoint displayed on the page - e.g. `https://IP_ADDRESS:<PORT>/cometd`. By refreshing this page multiple times you will be able to see each of the unique cometD endpoints created by the service. Take note of each of these unique, secure endpoints as you must authenticate for each one separately.



3. Browse to each unique secure endpoint and accept all certificates offered (as shown in step a)
4. After accepting the certificate for each endpoint you will see a page like the one shown below



5. **NB: You must accept the security certificates for all the unique endpoints returned by the service's cometD endpoint in step 1.**

9.8. Usage

A sample JavaScript client is provided (through the [Avaya Context Store DevConnect](#) site) for testing this feature. To trial the Event Tracker feature - use the test client as documented below.

9.8.1. Setup the JavaScript Test Client

1. Download the JavaScript test client archive (EventTrackerTestClient.zip) which is distributed through the [Avaya Context Store DevConnect](#) site.
2. Unzip the file to directory on the test machine and launch the index.html page in the root directory of the extracted folder
3. In the **Server:** textbox, enter either the secure or non-secure cometd endpoint URL for the service
 - `http://<IP_ADDRESS>/services/Streams-<VERSION>/endpoint/cometd`
 - `https://<IP_ADDRESS>/services/Streams-<VERSION>/endpoint/cometd`
4. Click the **Connect** button
5. If the steps above have been followed correctly and the connection is successful, a green connected status indicator () will appear at bottom of page.

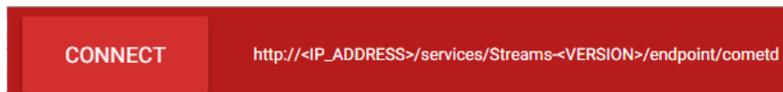


Figure 49: Streams endpoint connection

9.8.2. Register and Verify an Event Stream

NB: all input is case-sensitive.

1. Click on **Settings** icon on top right to open stream configuration panel. This panel will only be enabled if the client has connected successfully.
2. Choose the **Context Store** profile option in drop down.
3. Enter a stream name which will help the user identify the stream.
4. Set the **Instances** to specify a value for any of the **Dimensions** above e.g. [{"GroupId":"test"}]
5. Click the **Register** button.

Stream Configuration

Context Store

Stream Name
Agent Bob

Instances
[{"GroupId": "test"}]

Measures
{}

Push Rate

REGISTER

Figure 50: Event stream configuration

6. Verify the Stream got registered successfully by creating (or updating) a Context with the groupid value 'test'.
7. Expand sub grid of created stream, a notification should appear in the expanded sub grid under columns "Instances" and "Fields".

Stream Name	Channel ID	Unregister
Agent Bob	/com.avaya.cc.contextstore:0:1871620499/PT55/A1*1438094417934*1155	<input type="button" value="Unregister"/>
Instances		Fields
{"UUID":"164ae950-f409-4aba-a662-d23768602a89","ContextId":"2493791c68","GroupId":"test","TenantId":"undefined","RoutingId":...		[{"n":"Company","v":"Acme","t":1438869379826}, {"n":"Agent","v":"Bob","t":1438869379...}

Figure 51: Event stream notification received

9.8.3. Advanced Event Stream Configuration

Instances must be specified in a JSON Object Array. More than one instance of a Dimension may be specified. Additionally, instances of multiple Dimensions may also be specified

e.g. [{"ContextId": "A"}, {"ContextId": "B"}]

e.g. [{"ContextId": "A", "GroupId": "A"}, {"ContextId": "B", "GroupId": "B"}]

Measures (key/value) must be specified in a JSON Map

e.g. {"Key1": "Value1", "Key2": "Value2"}

10. Performance and Scalability Considerations

10.1. Overview

Context Store is an in memory data-grid that stores the contexts in the machines physical memory as opposed to the physical disk. Storing context in memory allows Context Store to deliver its service in the most efficient method possible without the overhead of disk operations. As a consequence Context Store pre-allocates a dedicated block of memory to store contexts to ensure the grid is not competing with other system resources at run time for memory to meet its advertised capacity.

In addition to the original context being stored, a live in-memory replica of the context store is also stored as a backup. In the event of a failure this backup is promoted to be the active memory block and the system can continue without interruption. This active backup setup means that the memory footprint of a context is effectively doubled with a copy of the context present on two nodes.

The Context Store cluster is engineered to sustain one complete node failure without any service interruption. To facilitate this robustness an extra node exists that can sustain the capacity of the other nodes in the cluster. During runtime each of these nodes work in an active/active manner partitioning Contexts across all the nodes, in the event of a single node failure the active and backups are restored to the remaining nodes with no effect to performance or HA. The exception to this behaviour being a single-node deployment – if that single node fails, data cannot be recovered.

10.2. Capacity Planning

When planning a system the engineer will need to estimate the amount and type of operations that will be made on Context Store in their enterprise. Context Store has a maximum capacity of 1240 requests per second (not supported on all deployment sizes), in a very simplistic system where we assume every piece of Context that is added to Context store is read by at least one client we can expect these requests to be broken into a ratio of 1:1. A more realistic system will probably have a ratio of 1:4 creates to reads where several applications will want to read the Contexts as the contact flows through the system.

Once the engineer has determined the number of operations they will need to know the type of data and the duration that this data is required. The creates, updates, data size and duration all have an interdependent relationship where raising the value on one of these three will lower the value of the other two i.e. increasing the lease time will mean more memory is consumed for longer in the in memory cache and hence we can't support as many creates per second.

A capacity planning tool that demonstrates the relationship between these parameters is available on the [Avaya Context Store DevConnect](#) site.

The following tables show the relationship between size, lease and memory for different setups. This data is based on standard Context Store traffic with no optional features deployed. For information on the capacity and performance implications of enabled additional features, see [10.2 Capacity Planning](#) subsections.

Table 2- Cluster containing 3 nodes with 64GB of memory each, Contexts stored with a 2 hour lease

Context Size in KB	Max new Contexts per Second	Number of Contexts	Memory Consumption
2	1240 (max supported)	8928000	17GB
8	630	4536000	35GB
16	316	2275200	35GB

32	159	1144800	35GB
64	79	568800	35GB

Table 3 - Cluster containing 3 nodes with 64GB of memory each, 2KB Contexts stored

Lease in Hour(s)	Max new Contexts per Second	Number of Contexts	Memory Consumption
12	415	17928000	34GB
24	207	17884800	34GB
48	102	17625600	34GB
168 (1 Week)	29	17539200	33GB
744 (1 Month)	6	16092000	31GB

10.2.1. Enabling Optional Features

2KB is the standard context object size used for all performance testing and certification. The supported 'requests per second' rate and lease time vary depending on the Context Store cluster size and the resources allocated to the EDP nodes within that cluster. If additional optional features are enabled, the following adjustments must be taken into consideration.

Audit Feature

Enabling audit trail requires a substantial amount of space in the data-grid, particularly for longer audit trails (> 10 entries). This increased memory requirement will impact the performance which is based on 2KB of Context data. If enabling the audit feature, the size of the Context data must be reduced in order to achieve the performance levels certified for a cluster.

CS Audit: Event Limit (length of Audit Trail)	Reduction in Context size	Notes
21+	1 KB	NB: Additionally the throughput and/or lease must be reduced when storing very long audit trails
11 - 20	1 KB	Throughput and lease can remain at certified levels if Context object size is reduced by 1KB. If 2KB Context data (or larger) is required, throughput and/or lease must be reduced
1 - 10	0.5 KB	Throughput and lease remain at certified levels if Context object size is reduced by 0.5KB. If 2KB Context data (or larger) is required, throughput and/or lease must be reduced

Alias Feature

Associating additional aliasIds with Context objects requires a substantial amount of space in the data-grid; each aliasId is written to the data-grid as a separate object which is linked to the main Context object. This increased memory requirement will impact the performance which is based on 2KB of Context data. If enabling the alias feature, the standard 2KB Context object must be reduced to 1.5KB in order to achieve the performance levels certified for a cluster.

Geo Redundancy

The traffic rate supported in a geo-redundant deployment is the same as that supported by single cluster of the given size; this is because data created in each cluster is replicated to the other.

CSNotifications

If both the CSNotifications and External DataMart features are enabled on a Context Store cluster, the supported throughput for CSNotifications is reduced by half. For example, in a Context Store deployment which supports maximum throughput (i.e. 1,240 requests per second) with these features disabled, half this rate (i.e. 620 requests per second) is supported with up to five subscriptions plus EDM enabled.

Event Tracker

A significant amount of power is required to process all the Context Store data events with respect to the streams which are currently registered. As the number of registered streams, the complexity of those streams' filters and the differences between the streams increases – the processing required increases greatly therefore the throughput or size of data must be reduced in this scenario.

10.3. Hardware and Network

10.3.1. CPU

Context Store defines a CPU as is being part of a single or multi-core processor that has a speed equivalent to a 2.9GHz Intel Xeon processor. Context Store has been engineered and certified to support Capacity and High Availability based on nodes which have dedicated non Hyper Threaded CPU's. Context Store does not support VM features that sub divide the processing power of the core as it does not expect to have to compete with external processes during max capacity or failover where Context Store requires additional processing overhead to ensure no service interruption. The memory and CPU requirements for the various certified Context Store deployments are available in the Reference Guide.

10.3.2. Network

All Nodes in the Context Store Cluster must be located within the same LAN with high speed connectivity and high capacity bandwidth to allow communication between the Nodes to work as efficiently as possible. A slow LAN will adversely impact the latency of the REST requests.

10.3.3. VMware

Context Store is a Snap-In that is running on the Avaya Aura® Engagement Development Platform which in turn is running on VMware infrastructure. For definitive questions on support VWware versions please refer to the Avaya Aura® Engagement Development Platform documentation.

VMware has many power and resource management features that can be used to optimize machine usage in a shared environment. Context Store is engineered to support all features running at max capacity and tolerate a node failure with minimal service impact. To accomplish this, it assumes it has 100% access to all resources that are allocated to it. If it has to compete with VMware optimization features, it may impact its ability to support the advertised capacity and high availability. Hence it is recommended that these features be disabled on the environment that Context Store is running on.

VMware uses the vMotion technology to migrate a running virtual machine from one ESX host to another without incurring downtime. Context Store has not certified this feature and hence it is recommended that this feature not be used during the running of Context Store.

10.4. High Availability

Context Store is a cluster consisting of 1-5 nodes with each of the nodes having the identical resources. Context Store is engineered to be highly available and to survive the failure of a single node while maintaining the ability to continue service and maintain throughput and quoted capacities without loss of data. If through a planned or unplanned action a node is no longer available to a cluster the cluster will compress onto the remaining nodes until the node becomes available again.

Single-node deployments are the exception; if the only node is lost, the data is not recoverable.

Context Store HA guarantees data is preserved after a single node failure only. If a subsequent node fails, the service may continue but the integrity of the data will be compromised as there may no longer be sufficient resources (CPU, memory, disk) remaining to store existing contexts or maintain engineered service level. Context Store does not support two concurrent node failures. In the event of multiple nodes failing it is highly recommended the customer should stop Context Store and restart the cluster clear out any unused contexts and have an established baseline of the data in the Contexts in the enterprise solution.

To minimize the risk of two concurrent node failures, Context Store requires that each of the Nodes be installed on individual VM hosts so that in the event of a VM host failing or being in accessible form the network the two remaining nodes are still available. Each of these VM hosts will need to be on the same high quality LAN so no latencies are incurred in the solution.

As a lot of data can be transmitted from and between the nodes in the Context Store cluster it is recommended that the Avaya Aura® Engagement Development Platform Management IP and Asset IP are assigned their own network port.

10.5. Geo Redundancy

Context Store geo redundancy is an architecture that can be utilized to enhance the high availability of Context Store. Using geo redundancy, two Context Store clusters can be paired to operate in an active/active mode. The traffic rate supported in a geo-redundant deployment is the same as that supported by single cluster of the given size; this is because data created in each cluster is replicated to the other.

Service preservation across the two clusters is achieved through the use of a highly available load balancer. The load balancer provides a common address through which both clusters can be reached. For routing requests and distributing the traffic load between the clusters, Context Store uses an optional field, **rid** query parameter, in each REST request. If the rid query parameter is not specified, the load balancer routes to a default Context Store cluster.

Session preservation between two clusters is achieved by deploying Context Store with data-grid replication channel configured between the two clusters. The state of the ContextStoreSpace in both clusters in the geo setup is replicated across this link. The network should have a dedicated 1 GB channel with a minimum of 300 MB bandwidth being available for Session Replication between clusters.

If the EDM feature is configured on a cluster; Contexts added/updated via the replication channel with a *persistToEDM* or *persistTo* flag preset will be replicated to the database.

Geo redundancy supports a failover of all Context Store operations to a single cluster in the event that a cluster, in the geo setup, is compromised. Because the state of a compromised cluster's data-grid cannot be guaranteed during this period, GEO and EDM replication may still be maintained. In Context Store, after a failover only the operations of the surviving cluster are supported from this point, as data integrity between the clusters cannot be guaranteed.

If the data-grid replication link is disconnected, a replication redo-log is used to contain the backlog of data to replicate to the other cluster should this link is re-established. In a prolonged outage if the redo-log exceeds its capacity it drops the oldest data and collects the newest. After failover, when the compromised cluster is

restored and reintroduced to the geo setup, the redo-log will replicate its contents to the other cluster. Because the contents of this redo-log are not guaranteed, Context Store does not support the integrity of this replicated data.

The status of the replication channel is logged by the CSManager service.

10.6. External DataMart

The External Data Mart (EDM) allows persistence of contexts from the Context Store into an external database via a JDBC connection. It is the responsibility of the customer to consume this output.

The following should be used as a guide to the specification required for the external database;

The table below indicates the disk size of contexts persisted per hour. The database must be able to consume this amount of data throughput. Exact details may vary from vendor to vendor but the following table is a rough guide to the amount of disk space consumed by the EDM attached to a system running at high capacity with a large percentage of create and update operations.

Context Size (KB)	Create&Updates/Sec	GB/Hour
2	1240	8.25
16	1000	55
32	800	88
64	500	110

Management of the EDM tables is very important. Size and throughput must be carefully considered to ensure the tables are available to the EDM feature.

Note: in a geo-redundant Context Store deployment, each configured External DataMart will be populated with Context data from both Context Store clusters in the active/active deployment

All database testing performed against an 8 core 2.9GHz, 32GB memory server located on the same LAN as the Context Store cluster.

10.6.1. Provisioning from External DataMart

In addition to persisting flagged context data to the CS_OPERATION table in the DataMart; contexts can also be written to the CS_PROVISION table in the DataMart – this table is reserved for provisioned context data which is required when a system starts up and should therefore be limited to essential data only.

When a Context Store cluster is restarted, all contexts which exist in the CS_PROVISION table will be written back into the data-grid with infinite leases. Context Store will not accept any new CSRest requests until the data-grid is fully established and system provisioning is complete. If the *EDM Persistence* feature is enabled on the cluster, provisioned Context data will be written into the CS_OPERATION table every time the cluster is restarted because it is written into the data-grid after each restart.

10.7. Throttling

Context Store employs a throttling feature to prevent all new operations when the system is in risk of becoming overloaded as a result of overly aggressive or eventually unsustainable traffic rates.

The system is considered to be in this compromised state when the traffic rate exceeds a high traffic rate threshold configurable in System Manager.

The throttling feature also utilises a low traffic rate threshold to avoid oscillation between the overload states and to provide a customisable buffer (between the high traffic rate thresholds) below which normal operation returns. This buffer safeguards the integrity of the data and customer flows against frequent throttle state toggling.

All throttling thresholds are configurable as CSManager attributes.

10.8. Sequenced Apps

Context Store is optimized to be a highly performant system that exists in a sequenced application environment. To avoid latencies typically associated with transactional locking Context Store avoids locking across all new creates and hence does not guarantee that the same customer specified contextId cannot be used in the same millisecond interval, to avoid this scenario use the Context Store generated contextId.

Context Store guarantees that all updates are written securely and transactionally to the data-grid, but it is possible in a highly concurrent environment that the data in the Context can be quickly updated and hence developers should implement the necessary handling in their client applications to ensure that this situation is handled correctly.

11. Appendix

11.1. Context Store API Documentation

11.1.1. Overview

Context Store uses Unicode, UTF-8 encoding exclusively

contextId may be defined using any of the following ASCII characters:

- Uppercase and lowercase English letters (a to z, A to Z) (ASCII: 65 to 90, 97 to 122)
- Digits 0 to 9 (ASCII: 48 to 57)
- Characters - ~ (ASCII: 45, 95, 126). *Provided that they are not the first or last character.*
- Character . (dot, period, full stop) (ASCII: 46) *Provided that it is not the first or last character, and provided also that it does not appear two or more times consecutively.*

Context data keys may be defined using any of the following ASCII characters:

- Uppercase and lowercase English letters (a to z, A to Z) (ASCII: 65 to 90, 97 to 122)
- Digits 0 to 9 (ASCII: 48 to 57)
- *Characters * - ~* (ASCII: 42, 45, 95, 126). *Provided that they are not the first or last character.*
- Character . (dot, period, full stop) (ASCII: 46) *Provided that it is not the first or last character, and provided also that it does not appear two or more times consecutively.*
- An asterisk character * - is allowed at the start and end of a keyname, this identifies the value as sensitive and means this value will not appear in any logs.

Note 1: Characters with special meaning in URLs i.e. ; # % cannot be detected by the Context Store interface and should not be used in the contextId.

Note 2: Extra care must be taken when submitting very long strings as identifiers (contextId, aliasId, groupId, tenantId or key) as the engine used for validation is not well optimized for long strings. If an invalid character is found deep into a long string it can cause the thread to hang, EDP will automatically return a '504 Gateway timeout' error after 10 seconds.

Long, invalid input strings cause significant spikes in CPU usage. If several of these are submitted, the EDP server trying to process them may automatically change to 'Denying' state due to CPU overload. If this occurs the affected EDP server must be restarted and then manually changed back to 'Accepting' state.

BasePath:

```
http://<IP_ADDRESS>/services/CSRest/cs/contexts/
```

11.1.2. Data Types

There are two Data Types - Context and Data. The Data is contained within the Context.

Example Context data type

```
{
  "contextId": "samplecontextId",
  "routingId": "0",
  "data": {"mykey1": "myvalue1", "mykey2": "myvalue2"}
```

```

    "metaData": {
      "groupId": " sampleGroupId ",
      "versionId": "1",
      "aliasIds": "samplealiasId1,samplealiasId2",
      "createdTime": "1438795335742",
      "lastUpdatedTime": "",
      "persistToEDM": "true",
      "persistTo": "CS_PROVISION",
      "tenantId": " sampleTenantId "
    }
  }

```

Example Data data type

```

{
  {"mykey1": "myvalue1", "mykey2": "myvalue2"}
}

```

Context

field	type	required
groupId	string	optional
tenantId	string	optional
aliasIds	list	optional
routingId	string	optional
persistToEDM	boolean	optional
persistTo	string	optional
data	Map[string,Object]	required
contextId	string	required

Data

field	type	required
data	Map[string,Object]	required

11.1.3. Context Operations

postContext

Creates a new context entry. A context entry contains key/value pairs.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/`

Operation: POST

Parameters:

- body

Parameter	Required	Description	Data Type
body	true	The Context to Add	Context
persistToEDM	false	Persistence flag	string
persistTo	false	Provisioning flag	string

- query

Parameter	Required	Description	Data Type
lease	false	Context will be stored for period specified in lease	string
alias	false	Alternative identifier for the context. Multiple permitted	string
touchpoint	false	User/application identifier for Audit Trail	string
rid	false	Routing identifier in geo-redundant deployment	string

Response Errors:

Status Code	Reason
400	Context with data key blank or null
404	No Data Found

upsertContext by contextId

Adds new context entry or updates an existing one using contextId as identifier. A context entry contains key value pairs.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/upsert/`

Operation: PUT

Parameters:

- body

Parameter	Required	Description	Data Type
body	true	The Context to Add	Context
persistToEDM	false	Persistence flag	string
persistTo	false	Provisioning flag	string
alias	false	aliasIds to be created/updated	string
groupId	false	If this doesn't match groupId for existing Context to be updated, the request will fail	string
tenantId	false	If this doesn't match the tenantId for existing Context to be updated, the request will fail	string

- path

Parameter	Required	Description	Data Type
id	true	The contextId	string

- query

Parameter	Required	Description	Data Type
lease	false	Context will be stored for period specified in lease	string
touchpoint	false	User/application identifier for Audit Trail	string
rid	false	Routing identifier in geo-redundant deployment	string

upsertContext by aliasId

Adds new context entry or updates an existing one using aliasId as identifier. A context entry contains key value pairs.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/upsert/`

Operation: PUT

Parameters:

- body

Parameter	Required	Description	Data Type
body	true	The Context to Add	Context

alias	true	aliasId required to identify the context	string
persistToEDM	false	Persistence flag	string
persistTo	false	Provisioning flag	string
groupId	false	If this doesn't match groupId for existing Context to be updated, the request will fail	string
tenantId	false	If this doesn't match the tenantId for existing Context to be updated, the request will fail	string

- query

Parameter	Required	Description	Data Type
lease	false	Context will be stored for period specified in lease	string
alias	false	Alternative identifier for the context. Multiple permitted	string
touchpoint	false	User/application identifier for Audit Trail	string
rid	false	Routing identifier in geo-redundant deployment	string

getData by contextId

Returns the data associated with the specified contextId.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/{id}/`

Operation: GET

Parameters:

- path

Parameter	Required	Description	Data Type
id	true	The contextId	string

- query

Parameter	Required	Description	Data Type
touchpoint	false	User/application identifier for Audit Trail	string
rid	false	Routing identifier in geo-redundant deployment	string

Response Errors:

Status Code	Reason
404	No Data Found
400	contextId or aliasId is blank or null

getData by aliasId

Returns the data associated with the specified.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/`

Operation: GET

Parameters

- query

Parameter	Required	Description	Data Type
alias	true	The aliasId	string
touchpoint	false	User/application identifier for Audit Trail	string
rid	false	Routing identifier in geo-redundant deployment	string

Response Errors

Status Code	Reason
404	No Data Found
400	contextId or aliasId is blank or null

getValue by contextId

Get the value of a key using contextId as identifier.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/{id}/keys/{key}/`

Operation: GET

Parameters

- path

Parameter	Required	Description	Data Type
id	true	The contextId	string

key	true	The Key Name	string
-----	------	--------------	--------

- query

Parameter	Required	Description	Data Type
touchpoint	false	User/application identifier for Audit Trail	string
rid	false	Routing identifier in geo-redundant deployment	string

Response Errors

Status Code	Reason
404	No data found
400	contextId or aliasId is blank or null

getValue by aliasId

Get the value of a key using aliasId as identifier.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/keys/{key}/`

Operation: GET

Parameters:

- path

Parameter	Required	Description	Data Type
key	true	The Key Name	string

- query

Parameter	Required	Description	Data Type
alias	true	The aliasId	string
touchpoint	false	User/application identifier for Audit Trail	string
rid	false	Routing identifier in geo-redundant deployment	string

Response Errors:

Status Code	Reason
404	No data found

400	contextId or aliasId is blank or null
-----	---------------------------------------

getContexts by groupId

Get metadata for all contexts associated with the specified groupId

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/groups/{id}/`

Operation: GET

Parameters:

- path

Parameter	Required	Description	Data Type
id	true	The groupId	string

- query

Parameter	Required	Description	Data Type
touchpoint	false	User/application identifier for Audit Trail	string
rid	false	Routing identifier in geo-redundant deployment	string

Response Errors:

Status Code	Reason
404	No Data Found

getAuditData by contextId

Get Audit Trail data for the context entry using contextId as the identifier.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/audit/{id}/`

Operation: GET

Parameters

- path

Parameter	Required	Description	Data Type
id	true	The contextId	string

- query

Parameter	Required	Description	Data Type
rid	false	Routing identifier in geo-redundant deployment	string

Response Errors:

Status Code	Reason
404	No Data Found
400	contextId or aliasId is blank or null

getAuditData by aliasId

Get Audit Trail data for the context entry using aliasId as the identifier.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/audit/`

Operation: GET

Parameters

- query

Parameter	Required	Description	Data Type
alias	true	The aliasId	string

- query

Parameter	Required	Description	Data Type
rid	false	Routing identifier in geo-redundant deployment	string

Response Errors

Status Code	Reason
404	No Data Found
400	contextId or aliasId is blank or null

putData by contextId

Update the data associated with the specified contextId.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/{id}/`

Operation: PUT

Parameters

- path

Parameter	Required	Description	Data Type
id	true	The contextId	string

- body

Parameter	Required	Description	Data Type
body	true	The data to add	Data

- query

Parameter	Required	Description	Data Type
lease	true	Context will be stored for period specified in lease. Updates restart the lease period.	string
touchpoint	false	User/application identifier for Audit Trail	string
rid	false	Routing identifier in geo-redundant deployment	string

Response Errors

Status Code	Reason
404	No Data Found

putData by aliasId

Update the data associated with the specified aliasId.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/`

Operation: PUT

Parameters

- body

Parameter	Required	Description	Data Type
body	true	The data to add	Data

- query

Parameter	Required	Description	Data Type
lease	true	Context will be stored for period specified in lease. Updates restart the lease period.	string
alias	true	The aliasId	string
touchpoint	false	User/application identifier for Audit Trail	string
rid	false	Routing identifier in geo-redundant deployment	string

Response Errors

Status Code	Reason
404	No Data Found

putValue by contextId

Update value of a key associated with the specified contextId.

URL: http://<IP_ADDRESS>/services/CSRest/cs/contexts/{id}/

Operation: PUT

Parameters

- body

Parameter	Required	Description	Data Type
body	true	The value to add	string

- query

Parameter	Required	Description	Data Type
lease	false	Context will be stored for period specified in lease. Updates restart the lease period.	string
touchpoint	false	User/application identifier for Audit Trail	string
rid	false	Routing identifier in geo-redundant deployment	string

Response Errors

Status Code	Reason
-------------	--------

400	Context has a null or empty data key
-----	--------------------------------------

putValue by aliasId

Update value of a key associated with the specified aliasId.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/`

Operation: PUT

Parameters

- body

Parameter	Required	Description	Data Type
body	true	The value to add	string

- query

Parameter	Required	Description	Data Type
lease	false	Context will be stored for period specified in lease. Updates restart the lease period.	string
alias	true	The aliasId	string
touchpoint	false	User/application identifier for Audit Trail	string
rid	false	Routing identifier in geo-redundant deployment	string

Response Errors

Status Code	Reason
400	Context has a null or empty data key

put aliasId by contextId

Update the list of aliasIds associated with this context entry. Identify the context using the contextId.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/{id}/aliases/`

Operation: PUT

Parameters

- path

Parameter	Required	Description	Data Type
-----------	----------	-------------	-----------

id	true	The contextId	string
----	------	---------------	--------

- body

Parameter	Required	Description	Data Type
body	true	The list of aliasIds to be added e.g. ["aliasId1","aliasId2"]	JSON

- query

Parameter	Required	Description	Data Type
lease	true	Context will be stored for period specified in lease. Updates restart the lease period.	string
alias	true	The aliasId	string
touchpoint	false	User/application identifier for Audit Trail	string
rid	false	Routing identifier in geo-redundant deployment	string

Response Errors

Status Code	Reason
404	No Data Found

put aliasId by aliasId

Update the list of aliasIds associated with this context entry. Identify the context using an existing aliasId.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/aliases/`

Operation: PUT

Parameters

- body

Parameter	Required	Description	Data Type
body	true	The list of aliasIds to be added e.g. ["aliasId1","aliasId2"]	JSON

- query

Parameter	Required	Description	Data Type
lease	true	Context will be stored for period specified in lease. Updates restart the lease period.	string
alias	true	The aliasId	string

touchpoint	false	User/application identifier for Audit Trail	string
rid	false	Routing identifier in geo-redundant deployment	string

Response Errors

Status Code	Reason
404	No Data Found

deleteContext by contextId

Deletes the context entry identified by the given contextId.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/{id}/`

Operation: `DELETE`

Parameters

- path

Parameter	Required	Description	Data Type
id	true	The contextId	string

- query

Parameter	Required	Description	Data Type
rid	false	Routing identifier in geo-redundant deployment	string

Response Errors

Status Code	Reason
404	No Data Found

deleteContext by aliasId

Deletes the context entry identified by the given aliasId.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/`

Operation: `DELETE`

Parameters

- query

Parameter	Required	Description	Data Type
alias	true	The aliasId	string

- query

Parameter	Required	Description	Data Type
rid	false	Routing identifier in geo-redundant deployment	string

Response Errors

Status Code	Reason
404	No Data Found

deleteValue by contextId

Delete Value : delete value of a key, if no keys exist context will still be present with empty data.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/{id}/keys/{key}/`

Operation: DELETE

Parameters

- path

Parameter	Required	Description	Data Type
id	true	The contextId	string
key	true	The Key Name	string

- query

Parameter	Required	Description	Data Type
touchpoint	false	User/application identifier for Audit Trail	string
rid	false	Routing identifier in geo-redundant deployment	string

Response Errors

Status Code	Reason
404	No Value was associated with specified Key

deleteValue by aliasId

Delete Value : delete value of a key, if no keys exist context will still be present with empty data.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/keys/{key}/`

Operation: DELETE

Parameters

- path

Parameter	Required	Description	Data Type
key	true	The Key Name	string

- query

Parameter	Required	Description	Data Type
alias	true	The aliasId	string
touchpoint	false	User/application identifier for Audit Trail	string
rid	false	Routing identifier in geo-redundant deployment	string

Response Errors

Status Code	Reason
404	No Value was associated with specified Key

delete aliasId

Delete the specified aliasId and remove any association from the context entry it belonged to.

URL: `http://<IP_ADDRESS>/services/CSRest/cs/contexts/aliases/`

Operation: DELETE

Parameters

- query

Parameter	Required	Description	Data Type
alias	true	The aliasId	string
touchpoint	false	User/application identifier for Audit Trail	string

rid	false	Routing identifier in geo-redundant deployment	string
-----	-------	--	--------

Response Errors

Status Code	Reason
404	No Data Found

11.1.4. Interface Error Codes

The following section contains the Error codes that are generated by each of the interfaces.

REST Status Codes

Code	Description	Scenario	Message	Note
OK	OK		Any operation successfully executed	n/a
BAD_REQUEST	Any HTTP 400+ response	Post Context: context contains an empty groupId tenantId parameter	"Context groupId tenantId is empty."	
		Post Context: context contains an empty or invalid persistTo parameter	"Context persistTo flag contains invalid value."	
		Post Context: missing/blank key	"Context data contains a null or empty data key."	
		Post Context: Duplicate identifier	"contextId aliasId routingId combination already exists in the repository." "Context aliasIds are not unique."	
		Post/Put/Get/Delete identifier containing invalid characters	"contextId groupId tenantId aliasId rid touchpoint contains invalid characters."	
		Post/Put/Get/Delete key containing invalid characters	"Key contains invalid characters."	
		Post/Put/Get/Delete: Identifier contains more characters than permitted count of 255.	"contextId aliasId groupId tenantId key aliasId touchpoint rid cannot be longer than 255 characters."	
		Post/Put Alias: maximum number of aliasIds reached/no-one mentioned this	"Cannot create more than 3 aliasIds per Context."	
		Put Data: missing/blank key	"Data key blank or null."	
		Put/Get/Delete: contextId/aliasId/group	"No data found."	

		Id does not exist		
		Put/Get/Delete: key does not exist	"No Value was associated with specified Key [keyname]"	
FORBIDDEN	403	Requests per second threshold has been breached. Interface is throttled.	"Request Throttled:Response sent"	
ERROR	Any HTTP 500+ response	Put/Get/Delete value/data: missing/blank contextId or groupId parameter Put/Get/Delete value: missing/blank key parameter	n/a	JSON Parse Exception occurs internally.
		An SDKException is thrown for any certificate related issues.	Error creating the context store instance : (Any of below messages follow) The required security system property (one of keyStore, keyStorePassword, trustStore or trustStorePassword) is empty – cannot create connection as a result. IOException loading the cert file into the key store. NoSuchAlgorithmException loading the cert file into the key store. CertificateException loading the cert file into the key store.	

PDC Status Codes

Code	Description	Scenario	Message	Note
200	OK	Any operation successfully executed	n/a	
400	Bad Request	Post Context (with groupId parameter) Post Context: missing/blank key Put Data: missing/blank key	"Context groupId is empty." "Context data contains a null or empty data key." "Data key blank or null."	

		Post/Put/Get/Delete with identifier containing invalid characters Post/Put/Get/Delete key containing invalid characters	"contextId/groupId/aliasId contains invalid characters." "Key contains invalid characters."	
403	Forbidden	Requests per second threshold has been breached. Interface is throttled.	"Request Throttled:Response sent"	
404	Not Found	Put/Get/Delete: contextId/groupId does not exist Put/Get/Delete: key does not exist	"No data found." "No Value was associated with specified Key [keyname]"	
408	Client side timeout	Timeout if the operation takes longer than the configured timeout period		
500	Validation Error	Put/Get/Delete value/data: missing/blank contextId or groupId parameter Put/Get/Delete value: missing/blank key parameter	"The id parameter must not be empty" "The key parameter must not be empty"	
500	SSL context may not be null	Client is not authenticated or is using an invalid or expired certificate.		
503	Service Temporarily Unavailable	Service unavailable at the configured IP address		
500	Internal Server Error	Invalid JSON data submitted to context store		Submitting invalid JSON causes major issues for JAX-RS and WAS. This can even cripple the system eventually.

Screen Pop Error Codes

Code	Description	Scenario	Message	Note
404	Not found	Context not found or response throttled	Context '<CONTEXT_ID>' not found or response throttled	

11.2. Certificate Based Authentication

For comprehensive documentation on configuring HTTP Security and Certificates please refer to the Avaya Aura® Engagement Development Platform and System Manager documentation.

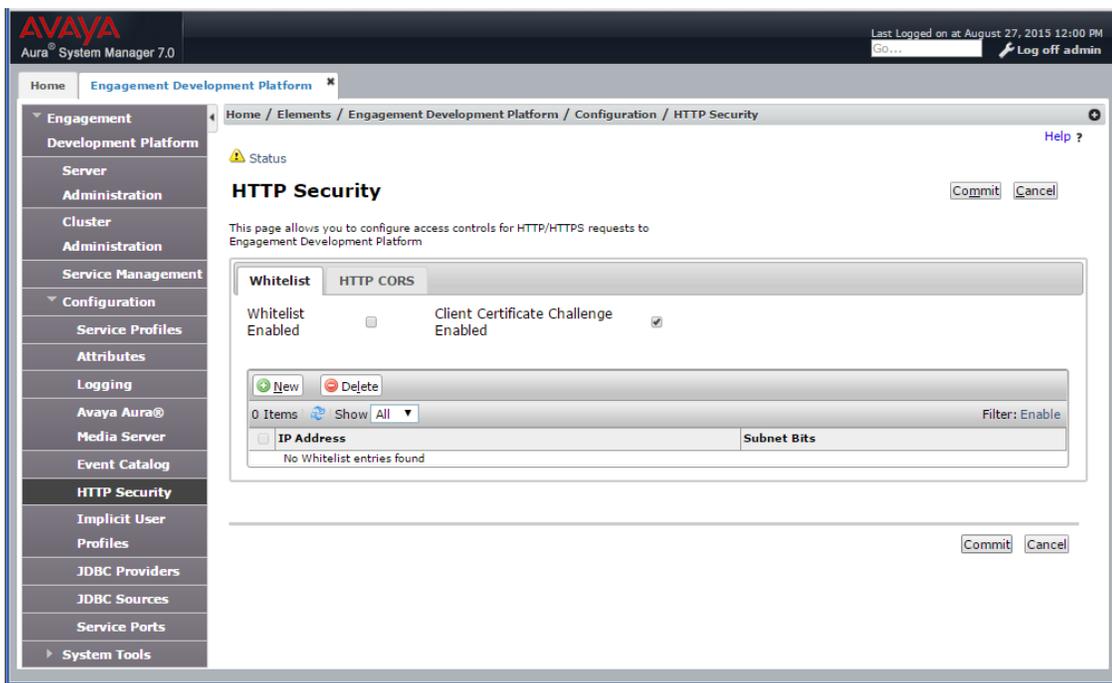
This section is intended to assist developers in getting started quickly with configuring Certificate Based Authentication for use with Context Store. There are several options available for configuring certificates; the steps below represent one particular approach which has been verified during development.

11.2.1. Configuring Client Certificate Challenge

1. Login in to System Manager as Administrator.
2. Navigate to **Elements -> Engagement Development Platform**.
3. Click **Configuration** and select **HTTP Security**.
4. Check the **Client Certificate Challenge Enabled** checkbox and click **Commit**.

With this configuration any client trying to access Context Store must provide a trusted certificate with requests.

Figure 52 System Manager HTTP Security



11.2.2. Create Client Keystore

1. Login in to System Manager as Administrator.
2. Navigate to **Security > Certificates > Authority**.
3. Select **Add End Entity** from the side menu.
4. Select profile INBOUND_OUTBOUND_TLS in the End Entity Profile drop down. Enter values for Username, Password, CN, OU, O and C fields. (see below image). **Note the password needs to be at least six characters long.**
5. For the Token field, select P12 file, JKS file or PEM file depending on the requirements of the Context Store client.

6. Click **Add** button and wait for successful completion.
7. Next, select **Public Web** from the side menu.
8. Click **Create Keystore**.
9. Enter the username and password used on the **Add End Entity** form and click **OK**.
10. On subsequent page (*Token Certificate Enrollment*), accept default options and click **Enroll**.
11. Download keystore on to client machine.

Figure 53 System Manager Certificate Authority

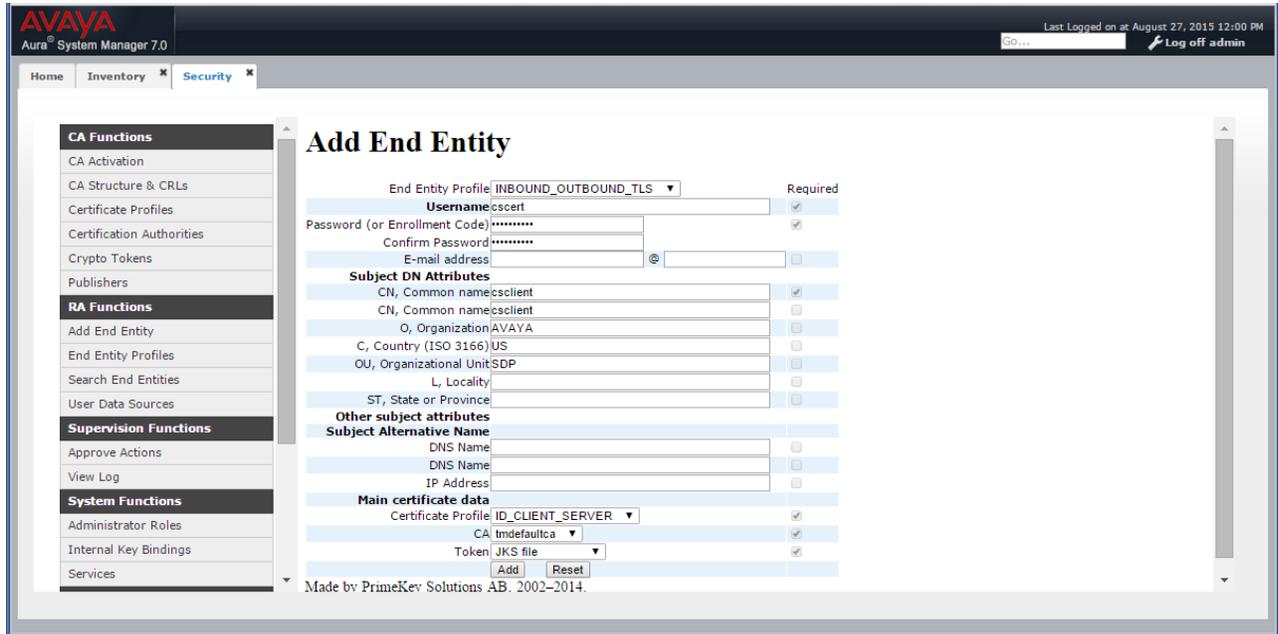


Figure 54 EJBCA

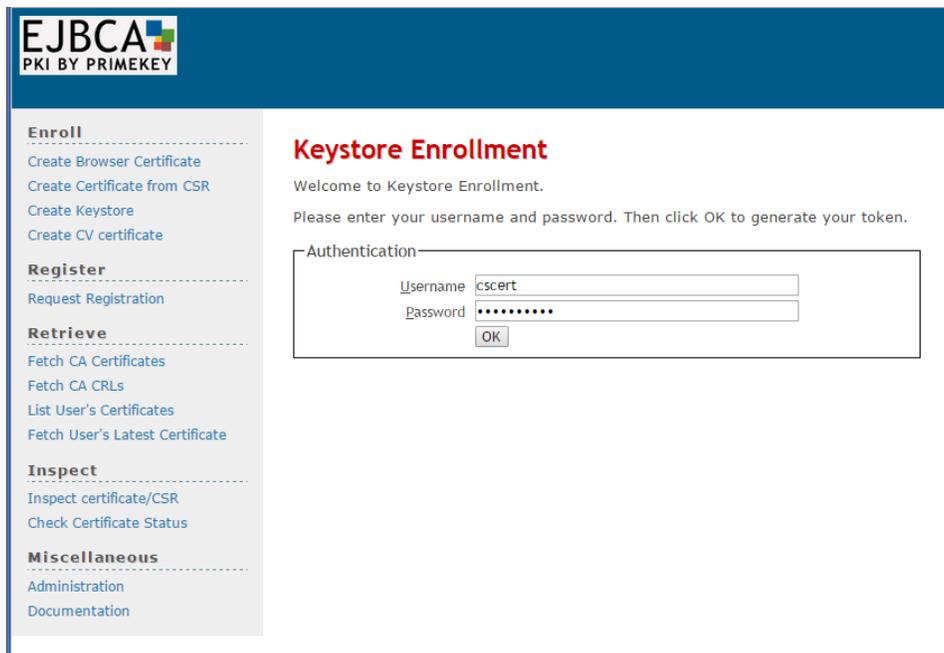


Figure 55 EJBCA Token Certificate Enrollment

EJBCA
PKI BY PRIMEKEY

Enroll

- Create Browser Certificate
- Create Certificate from CSR
- Create Keystore
- Create CV certificate

Register

- Request Registration

Retrieve

- Fetch CA Certificates
- Fetch CA CRLs
- List User's Certificates
- Fetch User's Latest Certificate

Inspect

- Inspect certificate/CSR
- Check Certificate Status

Miscellaneous

- Administration
- Documentation

EJBCA Token Certificate Enrollment

Welcome to keystore enrollment.

If you want to, you can manually install the CA certificate(s) in your browser, otherwise this will be done automatically when your certificate is retrieved.

Install CA certificates:

[Certificate chain](#)

Please choose a key length, then click OK to fetch your certificate.

Options

Leave values as default if unsure.

Key length: ▾

11.2.3. Download Avaya Aura® Engagement Development Platform Trusted Certificate from System Manager

1. Login to System Manager Console as a System Administrator.
2. Navigate to **Inventory** -> **Manage Elements**
3. Select checkbox for a single Avaya Aura® Engagement Development Platform server
4. From the **More Actions** drop down, select **Configure Trusted Certificate**.
5. Find and select checkbox for: **Store Type** = `SECURITY_MODULE_HTTP` with **Subject Name** = `O=Avaya, OU=MGMT, CN=System Manager CA`
6. Click **Export** and save file.

Figure 56 System Manager Manage Elements - Configure Trusted Certificates

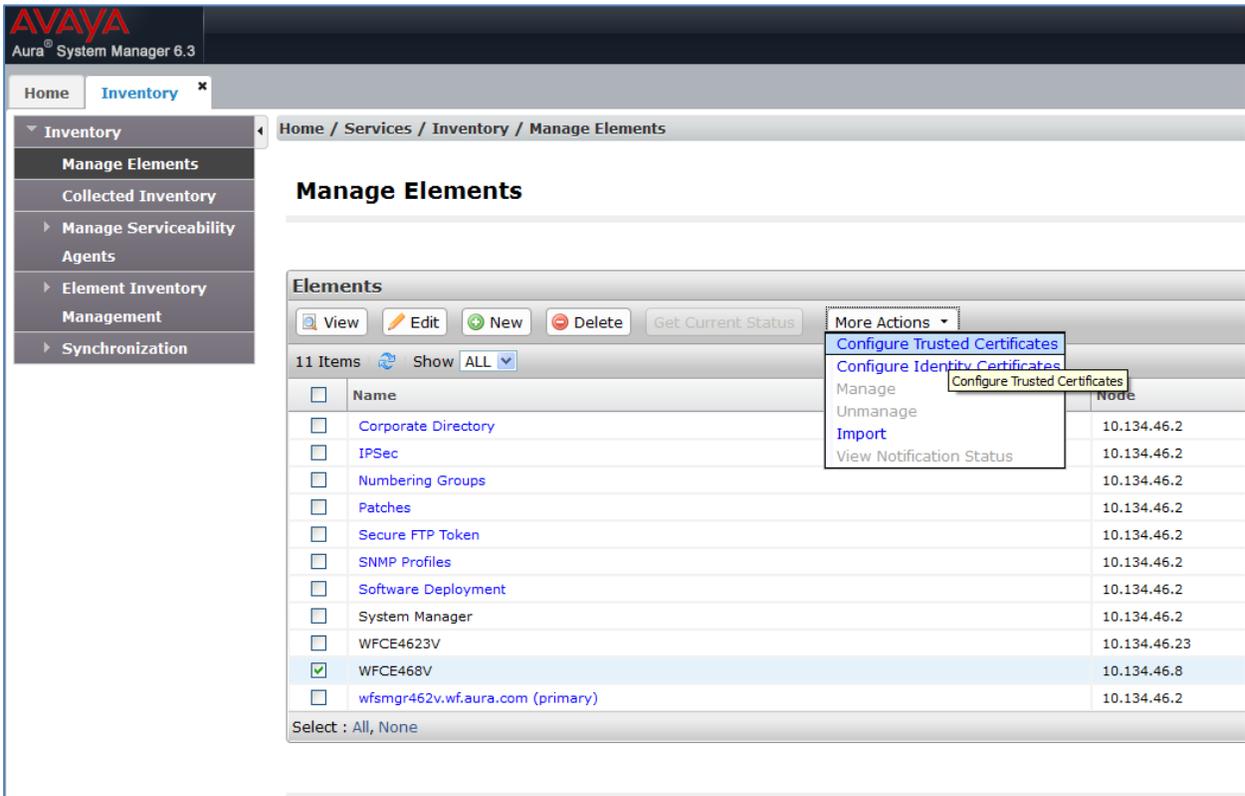
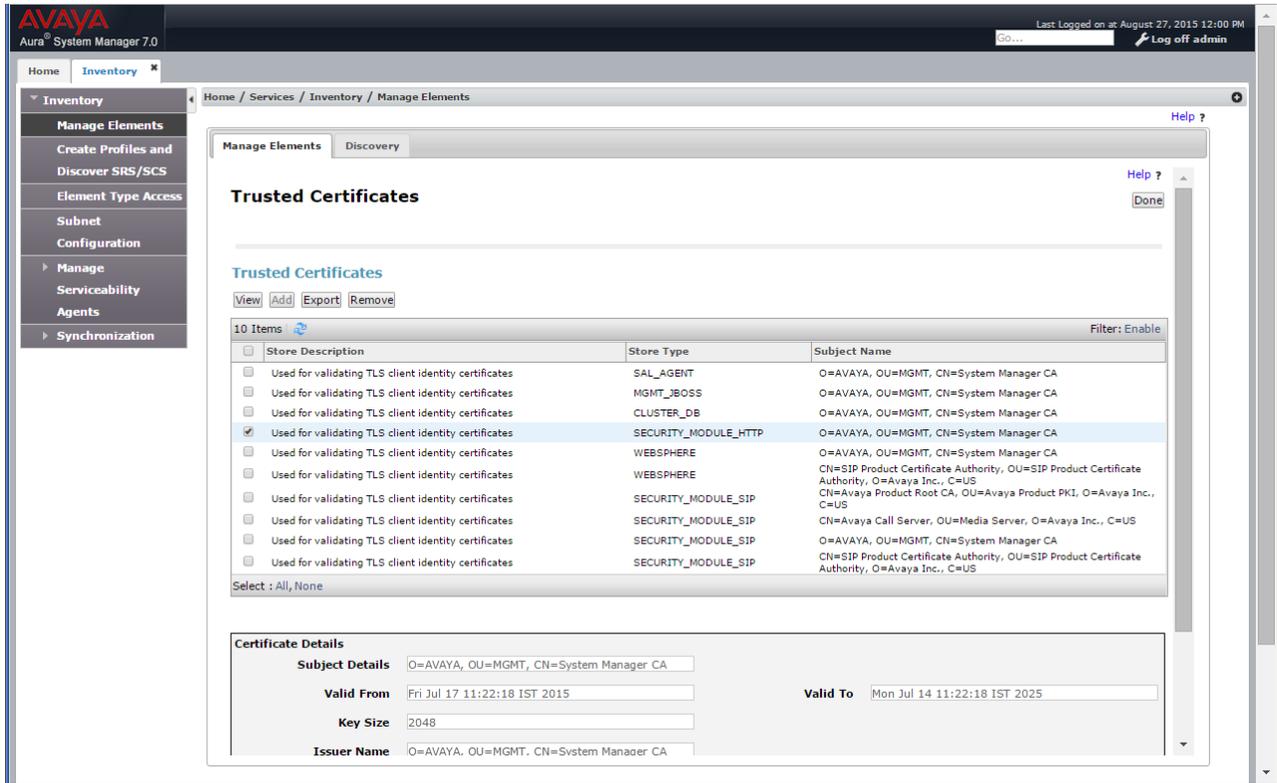


Figure 57 System Manager Trusted Certificates



11.2.4. Import Trusted Certificate into Keystore

In command line, navigate to target *jre bin directory* (if keytool is not in system path)

You must supply the location of the downloaded keystore and the location of the trusted certificate to import into the keystore to command line.

e.g.: "keytool -import -file C:\trustcert.pem -keystore C:\keystore.jks"

You will be prompted for keystore password and will receive status of import on command line.

Once the keystore is created it can be shared with other clients.

11.2.5. Verifying Successful Authentication

Verification will vary for each client. If using Java swing UI you can browse to the location of your keystore. Once the keystore is loaded requests to Context Store will be allowed.

11.2.6. General Information Regarding Java SSL

The above process is not the only way to achieve Mutual TLS (Client Certificate Challenge), but is the simplest to explain as both keystore and truststore are in one .jks file. The truststore and keystore can be separate entities (and in practice should be), the truststore will contain the trusted certificates and public keys and the keystore will contain the client certificate and private keys.

KeyStore - used to provide client credential to server

TrustStore - used by client to verify credential received from server

It is up to the clients to decide how they wish to store their certificates.

11.2.7. Thin Client Access

Some customers may want to access Context Store via a Thin Web Client. This section covers the current findings on what can be supported in this release.

Access from client browser directly to Context Store can be achieved, where Mutual TLS is configured, by following the below steps:

- create a pk12 certificate via System Manager (this is the same process as above but choosing pk12 instead of jks file type)
- Install downloaded pk12 certificate into browser.
- make requests directly via https://<CLUSTER_IP>/services/CSRest/cs/contexts/someid in browser

At time of writing, Mutual TLS can't be supported via AJAX. As the **Client certificate authentication is only performed upon request by the server and done transparently by the browser, as far as HTTP and javascript layers are concerned.**

Thin Client Options:

- HTTPS requiring trust certificates.
- HTTPS requiring trust certificates and whitelist enabled for known/trusted clients.
- HTTP only, assuming customer is satisfied with data center security.

11.2.8. Troubleshooting SSL Connections

OpenSSL

The `s_client` command implements a generic SSL/TLS client which connects to a remote host using SSL/TLS. It is a very useful diagnostic tool for SSL servers.

https://www.openssl.org/docs/apps/s_client.html

example: `openssl s_client -connect <ip>:<port>`

Portecle

Portecle is a user friendly GUI application for creating, managing and examining keystores, keys, certificates, certificate requests, certificate revocation lists and more.

<http://portecle.sourceforge.net/>

On Avaya Aura® Engagement Development Platform Instance

Enable debug logging on nginx to see what is being received on server side:

```
/var/log/nginx/error.log
```

Set nginx logging to debug

```
vi /etc/nginx/nginx.conf
```

```
change
```

```
error_log /var/log/nginx/error.log debug
```

```
to
```

```
error_log /var/log/nginx/error.log error
```

If Nginx debug shows error:

"Self signed certificate in certificate chain"

It indicates that the certificate being used is not linked with the Avaya Aura® Engagement Development Platform. Ensure the correct System Manager is used to create the certificate and that the trusted certificate is taken from the linked CEs.

Client Side

If Java, run client with java property `-Djavax.net.debug="ssl"`. This will show useful debug information about the SSL connection.

11.3. A10 Load Balancer Configuration

Context Store, by default, will try to fulfil requests if only one operational node remains in a cluster. In order for Context Store to fail over to the backup cluster it is recommended that the CSManger attribute "Cluster Deny Service on two node outage" is set to true for each of the Context Store clusters in a geo environment. This will ensure that if two nodes in a cluster are down requests will be sent to the other cluster. In a single-node deployment, this attribute is not applicable.

11.3.1. A10 Installation

These configuration steps assume that a working A10 system is installed and that the person configuring the system has prior knowledge of it. The install steps for the A10 ova provided here are only for reference and should not be taken as a definitive guide to installing an A10 load balancer. The A10 documentation should be referred to before attempting an install of the A10 load balancer.

The steps outlined here first configure HTTP routing in the A10 load balancer then when this is successfully routing traffic, the configuration for HTTPS is then attempted. It is suggested that this is the order in which the steps are configured although both HTTP and HTTPS configuration can take place at the same time.

The following steps outline a software only install of the A10 product.

Prerequisites

- Two Context Store Clusters that have replication enabled between them.
- A10 VMware image.
- IP address as specified in the A10 install doc.
- Register on the A10 website to get receive the software license, this can only be retrieved after the OVA image has been installed as a unique vThunder Host ID (UID) from the installed A10 image is needed to receive the license.
- The full instructions for installing the license are in the installation guide for the A10 product itself.

Installation

1. Download the VMware image.
2. Read the vThunder Install Guide pdf.
3. Install the OVA as per the instructions in the Installation Using vSphere Client chapter of the vThunder Install Guide pdf.
4. Follow the steps in the "Initial vThunder Configuration" chapter.
5. Install the License as per the steps in the "License Installation" chapter.

11.3.2. HTTP Traffic Configuration

1. Login to the A10 webui by entering the maintenance IP into a browser.

There are two modes when viewing the A10 website. Monitor and Config. **All the configuration steps below should be run in Config mode.**

2. In the following steps entities are required to be added or configured. Each of the steps will be described in detail below. The terms Health Monitor, Servers, Service Group, Template, Virtual Server and Virtual Service in this document are A10 specific and refer to concepts within the A10 product. The following is required to enable A10 to route HTTP traffic through two Context Store Clusters.

- a. Add a Health Monitor.
- b. Add two Servers, one for each CS cluster.
- c. Create Two Service Groups. Each Service Group will contain the two CS servers one as the main and the other as the backup.
- d. Configure a Template. A template is what the traffic gets routed by in A10.
- e. Add a Virtual Server.
- f. Configure a Virtual Service.
- g. Configure the IP Source NAT if required.

Health Monitor

In A10 a health monitor profile is used to periodically check if a server is available. The default health monitor profile uses pings to evaluate if a server is available. This will not work when only one of the active/standby machines is available and the other EDP is down as well. As a result a new health monitor will need to be configured based on the Context Stores rest interface availability rather than pings to verify if a CS deployment is operational.

1. Click SLB -> Health Monitor -> Health Monitor -> Add.
2. Enter a name for the health monitor, suggested "restCheckHTTP", Choose "HTTP" for the Type, choose "GET" for the URL type and enter `"/services/CSRest/cs/contexts/HttpHealthCheckFromTheLoadBalancerShouldNormallyReturnA404"` as the actual URL.
3. Enter "404" in the Expect text area. The Screen should look as below. Note the text area for the URL is truncating the full URL.
4. Press OK at the end of the Health Check screen.

Health Monitor	
Name: *	restCheck
Retry:	3
Consec Pass Req'd:	1
Interval:	5 Seconds
Timeout:	5 Seconds
Strictly Retry:	<input type="checkbox"/>
Disable After Down:	<input type="checkbox"/>

Method	
Override IPv4:	
Override IPv6:	
Override Port:	
Method:	<input checked="" type="radio"/> Internal <input type="radio"/> External
Type:	HTTP
Port:	80
Host:	
URL:	GET <input type="text" value="/services/CSRest/cs/"/>
User:	
Password:	
Expect:	404 <input type="radio"/> Text <input checked="" type="radio"/> Code
Maintenance Code:	
Passive Status:	<input type="checkbox"/>

Figure 58: A10 Health Monitor configuration

Server

Two Servers need to be added in this section, one for each CS Cluster. Click `SLB -> Service -> Server -> Add`. Enter a name (should identify the Cluster) and the IP of the first CS deployment. In the Port section enter 80 in the first port text field, change the Health Monitor drop down menu to the blank option and click Add then Click OK. Repeat for the second Cluster by clicking `SLB -> Service -> Server -> Add`. After these steps have been run when you click `SLB -> Service -> Server`, there should be two servers in the list.

Service Group

Two Service Groups need to be added. Each service group will contain two server entries, one which will be the default for the service group and another which is the backup. Click `SLB -> Service -> Service Group -> Add`. Enter a name (Should specify the CS cluster it will be the default Service Group for), for the Health Monitor drop down choose the HTTP health monitor created above. In the Server section select a server (Which will be the default for this service group) from the Server drop down menu, enter 80 as the port, select **16** in the Priority drop down and Press Add. In the Server section select a server (Which will be the backup for this service group) from the Server drop down menu, enter 80 as the port, select **1** in the Priority drop down and Press Add Finally click OK at the bottom of the screen. **Note the priorities entered for the servers in the service group are important for the correct routing of traffic.**

Add the Service Group for the second server by clicking `SLB -> Service -> Service Group -> Add`. The two servers are entered again but the priorities are switched.

Below, Figure 59, is an example service group, the default server for this service group is 4685 and the backup is 4664. When the Service Group configuration is finished there will be two Service Groups, with the default and backup servers switched over.

Service Group

Name: * csccluster-two

Type: TCP

Algorithm: Fastest Response Time

Auto Stateless Method:

Traffic Replication:

Health Monitor: restCheck

Server Template: default

Server Port Template: default

Policy Template:

Min Active Members:

Priority Affinity:

Send client reset when server selection fails

Send log information on backup server events

Stats Data: Enabled Disabled

Extended Stats: Enabled Disabled

Priority: Action: Proceed

Priority	Action
<input type="checkbox"/> 1	Proceed
<input type="checkbox"/> 2	Proceed
<input type="checkbox"/> 3	Proceed
<input type="checkbox"/> 4	Proceed

Description:

Server

IPv4/IPv6: IPv4 IPv6

Server: * csccluster4664 Port: *

Server Port Template(SPT): default Priority: 1

Stats Data: Enabled Disabled

Server	Port	SPT	Priority	Stats Data
<input checked="" type="checkbox"/> csccluster4664	80	default	1	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/> csccluster4685	80	default	16	<input checked="" type="checkbox"/>

Figure 59: A10 Service Group configuration

Template

A template is what the traffic gets routed by in A10. This section will configure traffic with the rid parameter = 1 to go to the non-default cluster. Choose, SLB -> Template -> Application -> HTTP > Add. Enter a name and click on the App Switching. Enter "rid=1" in the URL field, Choose the non-default Cluster from the Service Group drop down menu, choose "Ends With" from the Match Type drop down menu and click Add in the App Switching section. Enter "rid=1&" in the URL field, Choose the non-default Cluster from the Service Group drop down menu, choose "Contains" from the Match Type drop down menu and click Add in the App Switching section. Finally click OK at the end of the screen to save the Template. After the above is finished the App Switching screen should look as in the screen capture below.

App Switching

By: URL Host

Case Insensitive URL:

URL Hits: Enabled Disabled

URL Switching:

URL:

Service Group:

Match Type:

<input type="checkbox"/>	URL	Service Group	Match Type
<input type="checkbox"/>	rid=1	cscluster-one	Ends With
<input type="checkbox"/>	rid=1&	cscluster-one	Contains

URL Hash:

First Last (4-128)Bytes Use Server Status

Offset:

Figure 60: A10 Template configuration

Virtual Server

Click on the Config Mode tab. Click SLB -> Service -> Virtual Server -> Add. Enter a name and the IP of the LB. In the port section press the Add button. Select the type as HTTP and enter 80 in the first port text box. Press Okay on the Virtual Server Port screen and Okay again on the Virtual Server page.

Virtual Service

Click SLB -> Service -> Virtual Service -> Click on the HTTP_80 service. In the drop down list for the "Service Group" option choose the service group that will be the default CS cluster. In the drop down list for the HTTP template choose the template created earlier. Press Okay on the Virtual Service page.

IP Source NAT

An, IP Source NAT -> IPv4 Pool, may be required if the LB IP and the CS clusters are on a different network. To add click, IP Source NAT -> IPv4 Pool -> Add, and enter the details appropriate to the network.

If a IP Source NAT is required, it needs to be added to each Virtual Service, Click SLB -> Service -> Virtual Server -> <Virtual Service Name>, in the Source NAT Pool select the IP Source NAT created earlier.

After all the above is configured switch to Monitor Mode and navigate to SLB -> Service -> Virtual Server -> Virtual Server. Expand out each of the servers. There should be a green up arrow beside each server. This signifies that each server is contactable via the health check.

Verify the A10 LB is able to route HTTP traffic through each cluster when both clusters are operational based on the rid parameter. Failover tests should be run as well for when a cluster is completely down (Both Active and Standby nodes in a cluster are offline) and when the cluster IP is still active due to only one EDP from the CS cluster been active. In single-node deployment where there is only one IP address available (the security module IP of the node), it is considered down once this single node is offline.

11.3.3. HTTPS Traffic Configuration

Make sure the configuration checked in the section [11.2.1 Configuring Client Certificate Challenge](#) is enabled.

Retrieve the keystore Certificate from SMGR by following the instructions in the Create Client Keystore section and download the keystore to your local machine. Note the keystore cert needs to be downloaded from SMGR in the PEM file format; this differs from the format stated in the 11.2.2 Create Client Keystore section. The other sections from the [11.2 Certificate Based Authentication](#) chapter do not need to be run.

Login to the A10 webui by entering the maintenance IP into a browser.

There are two modes when viewing the A10 website. Monitor and Config. **All the configuration steps below should be run in Config mode.**

In the following steps entities are required to be added or configured. Each of the steps will be described in detail below. The terms Health Monitor, Servers, Service Group, Template, Virtual Server and Virtual Service in this document are A10 specific and refer to concepts within the A10 product. The following is required to enable A10 to route traffic to two Context Store Clusters.

- Upload SMGR Certificate.
- Add two SSL Templates, Client and Server.
- Add a HTTPS Health Monitor.
- Modify the two Servers by adding the 443 port to each.
- Create two HTTPS Service Groups. Each Service Group will contain the two CS servers one as the default and the other as the backup.
- Modify the Virtual Server, add the 443 port.
- Add a Template to route the traffic through the HTTPS Service Group.
- Modify the HTTPS Virtual Service.

Upload Certificate

Click SLB -> SSL Management -> Certificate and click Import. Enter a name then click the Browse button beside the Certificate Source and select the keystore pem file that was downloaded from SMGR previously. Click Browse beside the Private Key Source and select the same keystore pem file downloaded from SMGR previously. Click OK to save the import of the certs.

Create Client SSL Template

Choose, SLB -> Template -> SSL -> Client SSL and click Add. Enter a name with client and an identifier for the SMGR it originated from in it. Choose the cert created in the upload cert step for the drop downs beside "Certificate Name", "Chain Cert Name" and "Key Name" enter the certs password in the "Pass Phrase" and "Confirm Pass Phrase" fields and click OK.

Create Server SSL Template

Choose, SLB -> Template -> SSL -> Server SSL and click Add. Enter a name with server and an identifier for the SMGR it originated from in it. Choose the cert created in the upload cert step for the drop downs beside "Certificate Name" and "Key Name" enter the certs password in the "Pass Phrase" and "Confirm Pass Phrase" fields and click OK.

Health Monitor

In A10 a health monitor profile is used to check if a server is available. The default health monitor profile uses pings to evaluate if a server is available. This will not work when only one of the active/standby

machines is available and the other EDP is down as well. As a result a new health monitor will need to be configured based on the rest interfaces availability rather than pings to verify if a CS cluster is operational.

Click SLB -> Health Monitor -> Health Monitor -> Add. Enter a name for the health monitor, suggested "restCheckHTTPS", Choose "HTTPS" for the Type, choose "GET" for the URL type and enter "/services/CSRest/cs/contexts/HttpsHealthCheckFromTheLoadBalancerShouldNormallyReturnA404" as the actual URL. Enter "404" in the Expect text area. Choose the cert created in the upload cert step for the drop down list beside "Certificate Name" and "Key Name", enter the cert password in the "Pass Phrase" and "Confirm Pass Phrase" fields and click OK at the end of the Health Check screen.

Server

The Two Servers added in the HTTP section need to be updated with the 443 port. Click SLB -> Service -> Server. Click on one of the servers. In the Port section enter 443 in the first port text field, select the Server SSL Template created above for the Server-SSL Template (SST) drop down and click Add, then Click OK. Repeat for the second Cluster by clicking SLB -> Service -> Server and click on the other server. After these steps have been run when you click SLB -> Service -> Server, there should be two servers in the list, both which have the ports 80 and 443 configured.

Service Group

Two Service Groups need to be added that are HTTPS specific. Each service group will contain two server entries, one which will be the default for the service group and another which is the backup. Click SLB -> Service -> Service Group -> Add. Enter a name (Should specify the CS cluster it will be the default Service Group for and have HTTPS also in the name), for the Health Monitor drop down choose the HTTPS health monitor created above. In the Server section select a server (Which will be the default for this service group) from the Server drop down menu, enter 443 as the port, select 16 in the Priority drop down and Press Add. In the Server section select a server (Which will be the backup for this service group) from the Server drop down menu, enter 443 as the port, select 1 in the Priority drop down and Press Add Finally click OK at the bottom of the screen. Add the Service Group for the second server by clicking SLB -> Service -> Service Group -> Add.

Virtual Server

Click SLB -> Service -> Virtual Server. Select the preexisting virtual server. In the port section press the Add button. Select the type as HTTPS and enter 443 in the first port text box. In the Service Group drop down choose the default HTTPS service group created previously. For the "Client-SSL Template" choose the Client SSL template created previously and for the "Server-SSL Template" drop down choose the Server-SSL Template created previously. Press Okay on the Virtual Server Port screen and Okay again on the Virtual Server page.

Template

This section will configure traffic with the rid parameter = 1 to go to the non-default cluster. Choose, SLB -> Template -> Application -> HTTP > Add. Enter a name (It should contain HTTPS somewhere in the name) and click on the App Switching. Enter "rid=1" in the URL field, Choose the non-default HTTPS Cluster from the Service Group drop down menu, choose "Ends With" from the Match Type drop down menu and click Add in the App Switching section. Enter "rid=1&" in the URL field, Choose the non-default HTTPS Cluster from the Service Group drop down menu, choose "Contains" from the Match Type drop down menu and click Add in the App Switching section. Finally click OK at the end of the screen to save the Template.

Virtual Service

Click SLB -> Service -> Virtual Service. Click on the HTTP_443 service. In the drop down list for the "Service Group" option select (If it is not selected already) the HTTPS service group that will be the default CS cluster. In the drop down list for the HTTP template choose the HTTPS template created earlier in

the HTTPS configuration. Verify the "Client-SSL Template" and "Server-SSL Template" options have the correct template selected. Press Okay on the Virtual Service page.

After all the above is configured switch to Monitor Mode and navigate to `SLB -> Service -> Virtual Server -> Virtual Server`. Expand out each of the servers. There should be a green up arrow beside each server. This signifies that each server is contactable via the health check.

Verify the A10 LB is able to route HTTPS traffic through each cluster when both clusters are operational based on the rid parameter. Failover tests should be run as well for when the cluster IP is completely down (Both Active and Standby nodes are offline) and when the cluster IP is still active due to only one EDP from the CS cluster been active. In single-node deployment where there is only one IP address available (the security module IP of the node), it is considered down once this single node is offline.