Avaya Aura® Contact Center
&
Avaya Contact Center Select Enterprise Web Chat

Release 7.1.2

Issue 6.0

October 2021

Notice

While reasonable efforts have been made to ensure that the information in this document is complete and accurate at the time of printing, Avaya assumes no liability for any errors. Avaya reserves the right to make changes and corrections to the information in this document without the obligation to notify any person or organization of such changes.

Documentation disclaimer

“Documentation” means information published by Avaya in varying mediums which may include product information, operating instructions and performance specifications that Avaya may generally make available to users of its products and Hosted Services. Documentation does not include marketing materials. Avaya shall not be responsible for any modifications, additions, or deletions to the original Published version of documentation unless such modifications, additions, or deletions were performed by Avaya. End User agrees to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation, to the extent made by End User.

Link disclaimer

Avaya is not responsible for the contents or reliability of any linked websites referenced within this site or documentation provided by Avaya. Avaya is not responsible for the accuracy of any information, statement or content provided on these sites and does not necessarily endorse the products, services, or information described or offered within them. Avaya does not guarantee that these links will work all the time and has no control over the availability of the linked pages.

Warranty

Avaya provides a limited warranty on Avaya hardware and software. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya’s standard warranty language, as well as information regarding support for this product while under warranty is available to Avaya customers and other parties through the Avaya Support website: http://support.avaya.com or such successor site as designated by Avaya. Please note that if you acquired the product(s) from an authorized Avaya Channel

Partner outside of the United States and Canada, the warranty is provided to you by said Avaya Channel Partner and not by Avaya.

Licenses

THE SOFTWARE LICENSE TERMS AVAILABLE ON THE AVAYAWEBSITE, HTTP://SUPPORT.AVAYA.COM/LICENSEINFO

OR SUCH SUCCESSOR SITE AS DESIGNATED BY AVAYA, ARE APPLICABLE TO ANYONE WHO DOWNLOADS, USES AND/OR INSTALLS AVAYA SOFTWARE, PURCHASED FROM AVAYA INC., ANY AVAYA AFFILIATE, OR AN AVAYA CHANNEL PARTNER (AS APPLICABLE) UNDER A COMMERCIAL AGREEMENT WITH AVAYA OR AN AVAYA CHANNEL PARTNER. UNLESS OTHERWISE AGREED TO BY AVAYA IN WRITING, AVAYA DOES NOT EXTEND THIS LICENSE IF THE SOFTWARE WAS OBTAINED FROM ANYONE OTHER THAN AVAYA, AN AVAYA AFFILIATE OR AN AVAYA CHANNEL PARTNER; AVAYA RESERVES THE RIGHT TO TAKE LEGAL ACTION AGAINST YOU AND ANYONE ELSE USING OR SELLING THE SOFTWARE WITHOUT A LICENSE. BY INSTALLING, DOWNLOADING OR USING THE SOFTWARE, OR AUTHORIZING OTHERS TO DO SO, YOU, ON BEHALF OF YOURSELF AND THE ENTITY FOR WHOM YOU ARE INSTALLING, DOWNLOADING OR USING THE SOFTWARE (HEREINAFTER REFERRED TO INTERCHANGEABLY AS “YOU” AND “END USER”), AGREE TO THESE TERMS AND CONDITIONS AND CREATE A BINDING CONTRACT BETWEEN YOU AND AVAYA INC. OR THE APPLICABLE

AVAYA AFFILIATE (“AVAYA”).

Avaya grants you a license within the scope of the license types described below, with the exception of Heritage Nortel Software, for which the scope of the license is detailed below. Where the order documentation does not expressly identify a license type, the applicable license will be a Designated System License. The applicable number of licenses and units of capacity for which the license is granted will be one (1), unless a different number of licenses or units of capacity is specified in the documentation or other materials available to you. “Designated Processor” means a single stand-alone computing device. “Server” means a Designated Processor that hosts a software application to be accessed by multiple users.

License type(s)

Named User License (NU). You may: (i) install and use the Software on a single Designated Processor or Server per authorized Named User (defined below); or (ii) install and use the Software on a Server so long as only authorized Named Users access and use the Software. “Named User”, means a user or device that has been expressly authorized by Avaya to access and use the Software. At Avaya’s sole discretion, a “Named User” may be, without limitation, designated by name, corporate function (e.g., webmaster or helpdesk), an e-mail or voice mail account in the name of a person or corporate function, or a directory entry in the administrative database utilized by the Software that permits one user to interface with the Software.

Copyright

Except where expressly stated otherwise, no use should be made of materials on this site, the Documentation, Software, Hosted Service, or hardware provided by Avaya. All content on this site, the documentation, Hosted Service, and the Product provided by Avaya including the selection, arrangement and design of the content is owned either by Avaya or its licensors and is protected by copyright and other intellectual property laws including the sui generis rights relating to the protection of databases. You may

not modify, copy, reproduce, republish, upload, post, transmit or distribute in any way any content, in whole or in part, including any code and software unless expressly authorized by Avaya. Unauthorized reproduction, transmission, dissemination, storage, and or use without the express written consent of Avaya can be a criminal, as well as a civil offense under the applicable law.

Third Party Components

“Third Party Components” mean certain software programs or portions thereof included in the Software or Hosted Service may contain software (including open source software) distributed under third party agreements (“Third Party Components”), which contain terms regarding the rights to use certain portions of the Software (“Third Party Terms”). As required, information regarding distributed Linux OS source code (for those Products that have distributed Linux OS source code) and identifying the copyright holders of the Third Party Components and the Third Party Terms that apply is available in the Documentation or on Avaya’s website at:

http://support.avaya.com/Copyright or such successor site as designated by Avaya. You agree to the Third Party Terms for any such Third Party Components.

THIS PRODUCT IS LICENSED UNDER THE AVC PATENT PORTFOLIO LICENSE FOR THE PERSONAL USE OF A CONSUMER OR OTHER USES IN WHICH IT DOES NOT RECEIVE REMUNERATION TO (i) ENCODE VIDEO IN COMPLIANCE WITH THE AVC STANDARD (“AVC VIDEO”) AND/OR (ii) DECODE AVC VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSED TO PROVIDE AVC VIDEO. NO LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION MAY BE OBTAINED FROM MPEG LA, L.L.C. SEE HTTP://WWW.MPEGLA.COM.

Note to Service Provider

The Product or Hosted Service may use Third Party Components subject to Third Party Terms that do not allow hosting and require a Service Provider to be independently licensed for such purpose. It is your responsibility to obtain such licensing.

Preventing Toll Fraud

“Toll Fraud” is the unauthorized use of your telecommunications system by an unauthorized party (for example, a person who is not a corporate employee, agent, subcontractor, or is not working on your company's behalf). Be aware that there can be a risk of Toll Fraud associated with your system and that, if Toll Fraud occurs, it can result in substantial additional charges for your telecommunications services.

Avaya Toll Fraud intervention

If you suspect that you are being victimized by Toll Fraud and you need technical assistance or support, call Technical Service Center Toll Fraud Intervention Hotline at +1-800-643-2353 for the United States and Canada. For additional support telephone numbers, see the Avaya Support website: http://support.avaya.com or such successor site as designated by Avaya. Suspected security vulnerabilities with Avaya products should be reported to Avaya by sending mail to: securityalerts@avaya.com.

Trademarks

The trademarks, logos and service marks (“Marks”) displayed in this site, the Documentation, Hosted Service(s), and Product(s) provided by Avaya are the registered or unregistered Marks of Avaya, its affiliates, or other third parties. Users are not permitted to use such Marks without prior written consent from Avaya or such third party which may own the Mark. Nothing contained in this site, the Documentation, Hosted Service(s) and Product(s) should be construed as granting, by implication, estoppel, or otherwise, any license or right in and to the Marks without the express written permission of Avaya or the applicable third party.

Avaya is a registered trademark of Avaya Inc.

All non-Avaya trademarks are the property of their respective owners. Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

All non-Avaya trademarks are the property of their respective owners, and “Linux” is a registered trademark of Linus Torvalds.

Downloading Documentation

For the most current versions of Documentation, see the Avaya Support website: http://support.avaya.com or such successor site as designated by Avaya.

Contact Avaya Support

See the Avaya Support website: http://support.avaya.com for Product or Hosted Service notices and articles, or to report a problem with your Avaya Product or Hosted Service. For a list of support telephone numbers and contact addresses, go to the Avaya Support website: http://support.avaya.com (or such successor site as designated by Avaya), scroll to the bottom of the page, and select Contact Avaya Support.

Contents

Revision History

Date	Revision #	Summary of Changes
December 2015	1.0	Original HTML version
18 April 2016	2.0	Initial PDF version
August 2019	5.0	7.1 updates
September 2021	6.0	7.1.2 updates

Chapter 1: Changes in this release

The following sections describe the new features and changes in this document.

Features

Release 7.1.2 based on a previous release 7.1.1 with some fixes. New Agent API is provided. RoomStatus request is provided to check agent's state in a room. It’s necessary to send it when swithover is happened

Chapter 2: Overview

Avaya Contact Center Select (ACCS) 7.1 and Avaya Aura® Contact Center (AACC) 7.1 supports implementations of Web Chat; the traditional Web Communications option and a new Enterprise Web Chat (EWC) option.

This document and SDK provides information on the new Enterprise Web Chat (EWC) option only but to provide context, both options are discussed in this overview section:

Option 1: AACC Web Communications (as provided in AACC 6.4.2): This Web Communications offer has been available with Multimedia enabled AACC for a number of releases and continues to be supported in AACC 7.1. This version allows licensed AACC agents to handle Web Communication contacts via the out of the box Agent Desktop application. In addition to functionality provided as part of the AACC solution, two types of SDK download are available for this option. Both SDK's can be used by developers and solution integrators to develop a customer facing front-end / web application that can be used by customers to initiate Web Communication conversations. The reference implementation provided by the SDK's is designed to provide a starting point to build a Web Communications solution.

The SDK's are available to download from Dev Connect: http://www.devconnectprogram.com.

These are listed under ***Downloads*** for ***Avaya Aura Contact Center*** – *Web Communications SDK and Reference Implementation*. One version provides a reference implementation using JSP technology while the other provides similar functionality using PHP technology.

Option 2: **Enterprise Web Chat (EWC)**

Enterprise Web Chat (EWC) delivers the next generation of Web Chat in Avaya Aura Contact Center (AACC) 7.1. It is offered as a licensed alternative to the traditional AACC web chat providing a secure and high capacity chat solution. Enterprise Web Chat is rich in agent, supervisor and customer features, provided via new customer facing and agent facing set of APIs and reference implementations. Enterprise Web Chat provides:

- Scalable and high capacity multimedia chat solution (*Refer to the Avaya Aura Contact Center Specification and Overview document for supported capacity rates*)
- New Agent facing API and Agent SDK to customize agent& supervisor experience
- New customer facing API and reference Customer UI to customize customer experience
- Builds on the principles laid down by earlier releases of AACC – Uses existing administration, reporting and routing infrastructure of AACC
- Feature rich API providing in-chat behaviours, wait treatments, transcript management, and supervisor observe & barge-in etc.
- Mobile Ready – provides re-connect capabilities in the event of temporary network loss
- Supported on AACC 7.x Avaya Aura platforms and ACCS 7.1
- Requires Custom Desktop development (agent facing API is not implemented by 'out of the box' Avaya Agent Desktop)

- Supports multiplicity of chat sessions.

The Web Chat API allows customers to build and deploy custom agent and client solutions for their business. It offers a highly scalable application that will grow and shrink as necessary while maintaining all chat conversations. Maximum availability is guaranteed through the use of the Avaya High Availability (HA) solutions for AACC.

The solution uses an embedded XMPP server to host the conversations within AACC. In EWC, customer and agent controllers are used to request a chat and to pass messages into the desired chat room. Transcripts are kept for each chat and persisted in the AACC database allowing for each chat to be logged for further analysis. A notification for which user is typing is also included, as well as both comfort and on hold messages for the customer when the agent is slow to react to new chats. In addition, supervisors are granted the ability to observe other agents and even barge in when they are not performing adequately, these actions are guided by intrinsic data such as number of messages sent, time since last message, number of unanswered messages etc. EWC contacts can be monitored via AACC Real time and historical reporting.

Note: Both of the above options are controlled by license. AACC solutions may be licensed to use option 1 OR option 2. It is not possible to run both Web Chat options in parallel.

Chapter 3: Reference Architecture for Enterprise Web Chat

This section provides a reference architecture for a typical deployment.

Terminology

- Contact Center is either Avaya Aura© Contact Center or Avaya Contact Center Select.
- A co-resident server is one in which AACC uses a single server for both voice and multimedia applications. EWC is not supported on this deployment type.
- A standalone pair is where AACC uses separate servers for voice and for multimedia, e.g. one Voice Contact Server and one Multimedia Contact Server.
- A High Availability (HA) pair is a set of standalone servers which are configured to use high availability in a campus environment.

Overview

The Enterprise Web Chat reference architecture, as illustrated in Figure 1, is a suggested deployment and network topology for this system. It is based on a pair of reverse proxies set in a DMZ in front of Contact Center to provide an extra layer of protection for the contact center servers.

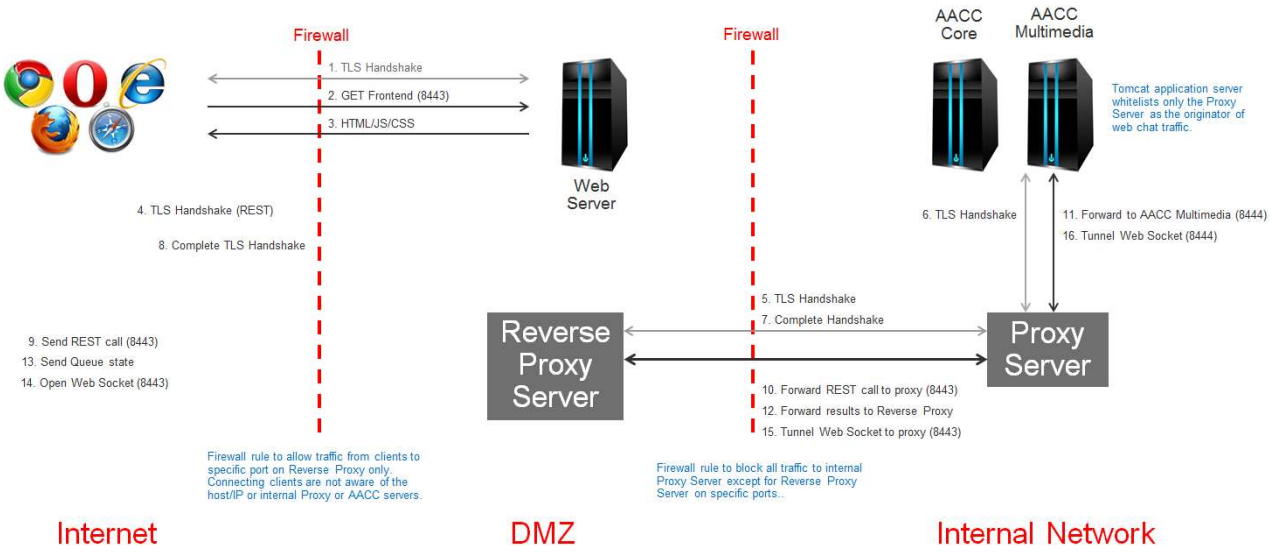


Figure 1 - Reference Solution Architecture

The customer-facing web page is deployed on a separate web server of your choice, and contains all the JavaScript used to open the chat, the styling of your website, and other details. If a secure connection is requested for the REST calls (e.g. check queue) or the WebSocket, the connection is preceded by a TLS handshake. TLS handshakes occur between the customer's browser and the outer proxy, one between each of the proxies, and one between the inner proxy and AACC. When the customer opens a chat, the WebSocket is opened from their browser to the outer proxy, whose address is visible to the customer. The outer proxy forwards the WebSocket handshake to the inner proxy, which then forwards it on to the Multimedia-only server in AACC.

Chapter 4: High Availability

This section describes Enterprise Web Chat High Availability for AACC and Business Continuity for ACCS.

Overview

The Enterprise Web Chat feature can be deployed in a High Availability / Business Continuity (HA/BC) environment and provides the following resiliency in the event of AACC Multimedia failure:

- Preservation of active chat conversations between customers and agents during an outage and subsequent switch-over of a Multimedia AACC server.
- Preservation of chat conversations already queueing in the system.

✱ **Note**: It should be noted that this section is intended as supplementary information to the existing Contact Center published user guides. It highlights any additional HA/BC configuration, features and limitations in the context of Enterprise Web Chat but should not be viewed as a replacement to the existing product documentation.

Campus High Availability / Business Continuity

Enterprise Web Chat supports the following level of campus HA/BC:

- Mission Critical High Availability (MCHA) for SIP-enabled Avaya Aura® Contact Center
- Business Continuity (BC) for Avaya Contact Center Select

When using the Enterprise Web chat feature, the following is required to deploy HA:

Option 1 (illustrated in figure 1)

- Two core AACC/ACCS 7.1 servers (voice only installation) are combined to provide a mission critical Active and Standby pair for AACC core components
- Two Multimedia servers (multimedia only install) are combined to provide a HA Active and Standby pair for Multimedia and Web Chat components

Option 2

- Two AACC/ACCS servers (voice and multimedia installation) are combined to provide a mission critical Active and Standby pair for core, multimedia and Web Chat components

EWC conversations are replicated between active and standby nodes using ejabberd clustering technology.

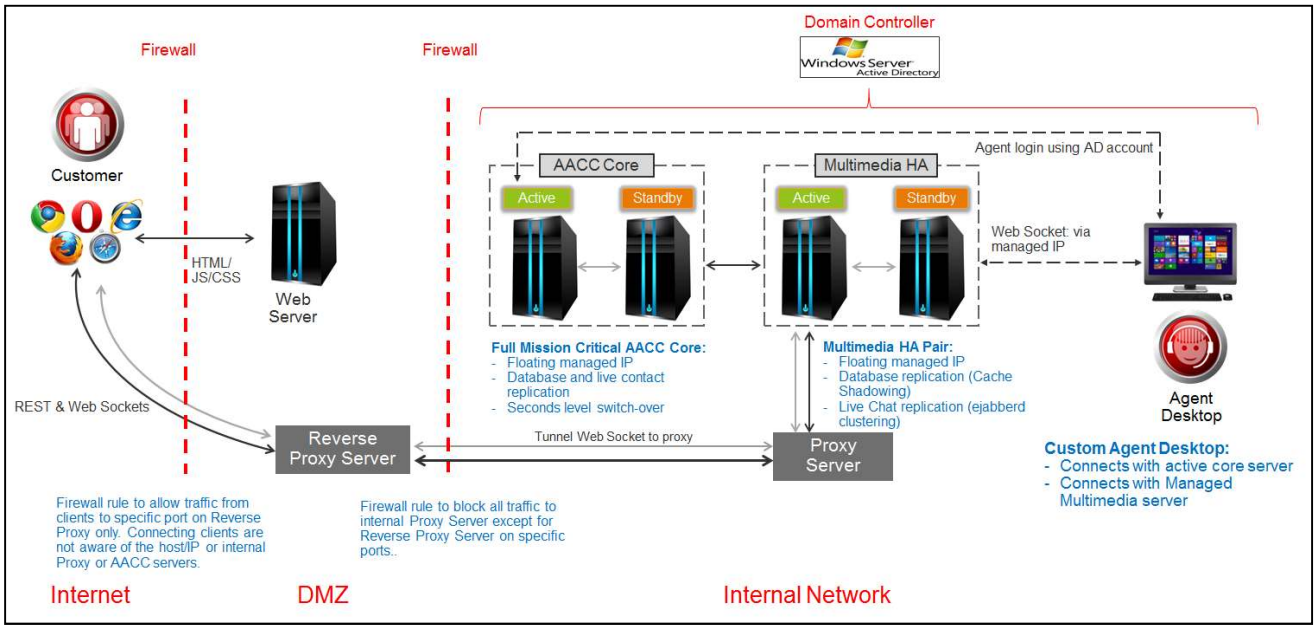


Figure 1 – Enterprise Web Chat High Availability

Contact Center Provisioning and Administration

In terms of Contact Center server provisioning and administration, the Enterprise Web Chat feature is fully integrated into existing high availability tools and applications. During a standard deployment of Contact Center Multimedia HA (as per existing commissioning documentation), a licensing check is performed by Contact Center to determine if Enterprise Web Chat is enabled. If Enterprise Web Chat is licensed, the Contact Center software will automatically enable web chat clustering between active and standby Multimedia nodes, thus preserving any active chat conversation in the event of a switch-over.

The following applies when Enterprise Web Chat is licensed:

- Active and standby CCMM servers must be synchronized before a manual or Enterprise Web Chat initiated switch-over can be performed. The term “Synchronized” means that both servers are running correctly and can replicate live chat conversations. This ensures that the currently running standby server is capable of taking over as the active server in the event of a switch-over.
- To check the Enterprise Web Chat health of a system (i.e. if the system is “Synchronized”), run the High Availability utility and get the current status as illustrated below:

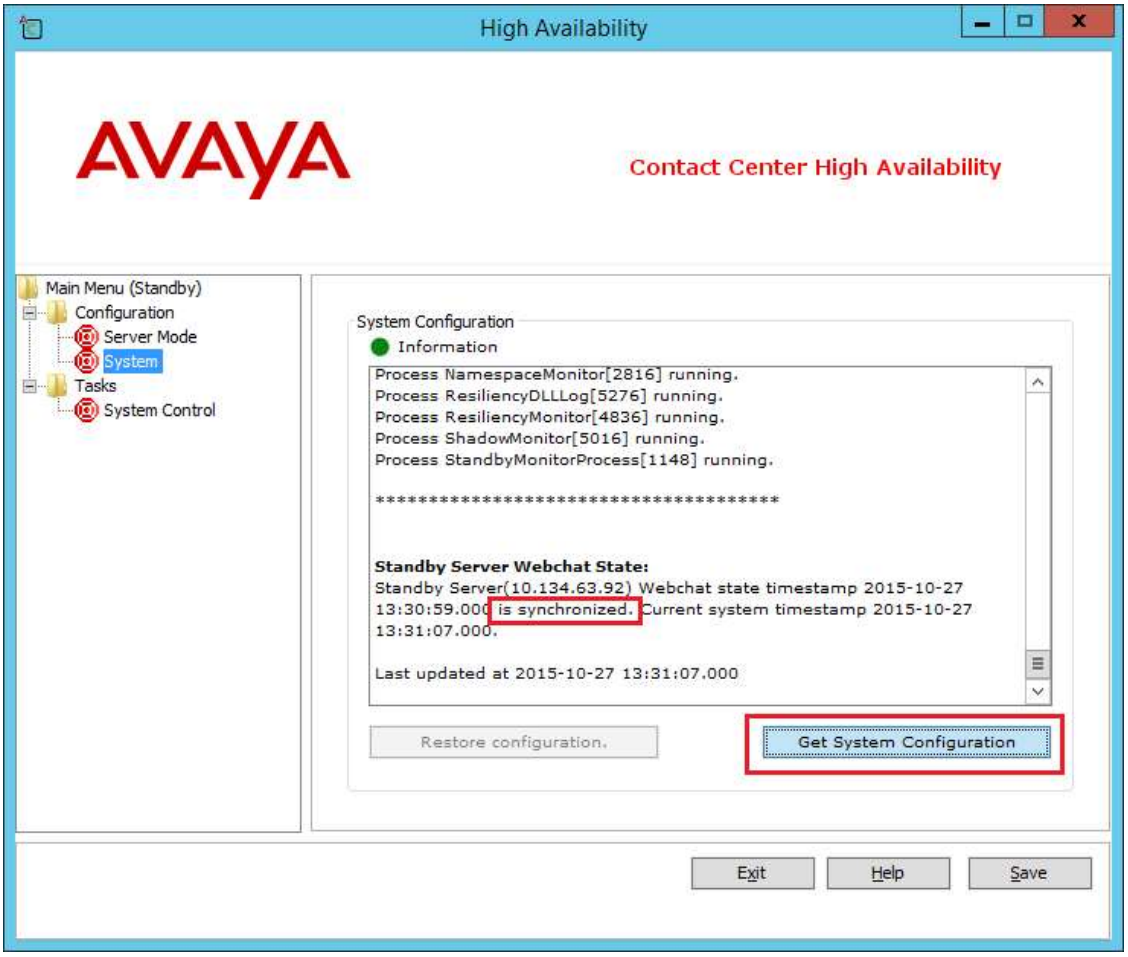


Figure 2 – Enterprise Web Chat High Availability Healthcheck

- This can be run from the active CCMM server or from the standby CCMM server (as illustrated above)

Campus Switchover scenarios and triggers

In addition to standard switchover scenarios EWC adds the following::

Manual:

- Before manual switchover there is a pre-check. Standby server EWC must report healthy state

Automatic:

- Critical web chat service/process outage on active server (Standby server EWC must report healthy state)

Customer Experience during CCMM server switch-over:

- The customer is automatically re-connected to newly active server (via managed address)
- Chat conversations are preserved during switch-over and can be continued post switch-over
- During the switch-over, the customer chat input box will be disabled for the duration of the switchover

Agent Experience during CCMM switchover

- The agent is automatically re-connected to newly active server (via managed address)
- Chat conversations are preserved during switch-over and can be continued post switch-over
- During the switch-over, the agent chat input box will be disabled for the duration of the switchover
- The agent will be transitioned into a 'Not Ready' state, this protects the agent from receiving any new chats while the system is transitioning from active to standby. The agent will remain logged in to the contact center.

Geographic Redundancy

The Avaya Aura® Contact Center High Availability feature supports geographic redundancy for data resiliency and disaster recovery. In geographic Wide Area Network (WAN) solutions, the standby server on the remote geographic site is called a Remote Geographic Node (RGN) server. Contact Center geographic High Availability supports data resiliency and disaster recovery

- Remote Geographic Nodes do not automatically take over if the campus fails
    - Manual intervention is required to bring the node into service (Down time required)
    - No live contacts are preserved
    - No chats in queue are preserved
- Provides valuable redundancy at another remote location

A typical campus and RGN solution:

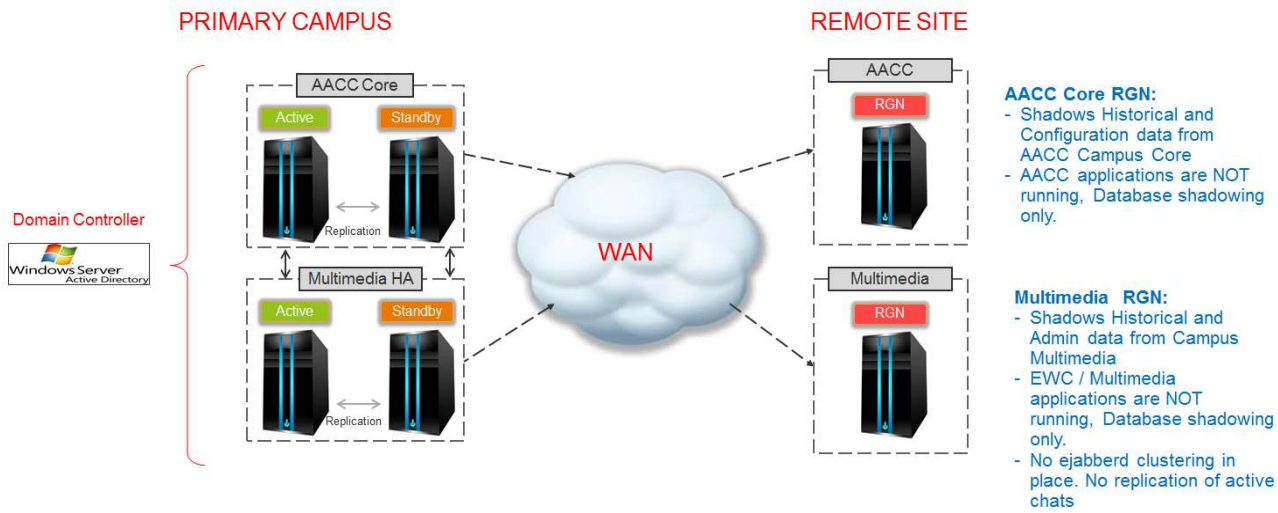


Figure 3 – Typical campus and RGN deployment

Geographic Business Continuity

Avaya Contact Center Select (ACCS) supports Geographic Business Continuity for data resiliency and disaster recovery. If the active ACCS fails, you must manually start up the Remote Geographic Node (RGN) server and redirect ACCS agents, supervisors, and administrators to use the RGN.

- Remote Geographic Nodes do not automatically take over if the campus fails
  - Manual intervention is required to bring the node into service (Down time required)
  - No live contacts are preserved
  - No chats in queue are preserved

Chapter 5: Security Considerations

This section details important security settings for Enterprise Web Chat and highlights ways to mitigate against common threats.

Certificates

Secure transmission requires a valid certificate from a certificate authority, which must be installed in the Contact Center Certificate Manager, refer to the Contact Center Server Administration documentation. The test certificates that come with Avaya Aura Contact Center are self-signed and are therefore invalid for production use; their intended purpose is for testing only. In general, browsers will refuse to connect to a page or a resource if they cannot guarantee a secure connection, such as from a self-signed certificate. In the case of a HTML page, it is possible for the user to make an exception for the certificate, but this is not possible for a sub-resource on the page (such as WebSockets or an XMLHttpRequest). Browsers will also refuse to connect to an unsecured resource from a secure page, e.g. an unsecured WebSocket will not be opened from a page that is being served over HTTPS. Secure connections are possible from an unsecured page. However, the JavaScript embedded in the page is not secure, and therefore not protected from a man-in-the-middle attack. It is recommended that the site for the Reference Web UI be served over HTTPS as well to prevent this.

Secure WebSockets and WebSocket proxies

Opening a secure WebSocket connection requires a secure port on Avaya Aura Contact Center (which is already included by default), and the use of the secure WebSocket protocol. The URLs for the WebSockets are defined in the *AppSettings.xml* file for the Custom Desktop, and the *links.js* file in the Customer Reference Application.

The following examples show the difference between a WebSocket running over a normal HTTP port versus a secure WebSocket over a secure port:

- ws://localhost:8081/CustomerControllerWeb/chat
- wss://localhost:{SECURE_PORT}/CustomerControllerWeb/chat

The Enterprise Web Chat reference architecture specifies that the Reference Web UI be kept on a separate server from Avaya Aura Contact Center and that at least one proxy be used to route chat traffic to AACC. The recommended topology is a pair of proxies based on Apache HTTP Server 2.4.16 between the Frontend and Avaya Aura Contact Center, with the following notes on configuration.

- The outer proxy should be configured to accept connections from all locations, as the WebSockets from the customers' browsers will connect directly to that.
- The inner proxy should be configured to only accept connections from the outer proxy.

- Avaya Aura Contact Center should be configured to only accept connections from the inner proxy.
- Any proxy servers used in the deployment of Enterprise Web Chat require individual certificates to operate in a secure environment. Apache does not accept certificates that have been encoded in DER format, PEM format is required. It is possible to convert certificates from DER to PEM formats for use with Apache.
- A common cause for errors with proxies is that the proxy's firewall has not been configured to allow traffic through a specific port, e.g. if Apache is set to listen on ports 8081 and {SECURE_PORT}, the operating system's firewall must allow traffic through those ports.
- {SECURE_PORT} is 8445 by default. This can be changed via Security Manager

Secure REST calls

Securing the REST calls for the queue status and callbacks involves the use of HTTPS and a secure port; below is an example of the difference between an unsecured call for the current queue and a secure version.

- http://localhost:8081/CustomerControllerWeb/currentqueue
- https://localhost:{SECURE_PORT}/CustomerControllerWeb/currentqueue

The REST calls for the queue status and to request a callback may also be passed over a proxy.

Further advice for securing Apache proxy servers

- Keep everything up-to-date wherever and whenever possible. This applies to any other proxy as well.
- The following Apache directives can hinder Denial of Service attacks:
  - Configure the ***RequestReadTimeout directive*** to limit the time a client may take to send a request.
  - Lower the value of the ***Timeout directive***. This may have performance issues, as Timeout is used for several other things as well.
  - Lower the value of the ***KeepAliveTimeout directive***. An alternative is to turn off KeepAlive altogether, but this has performance issues.
  - Configure the ***LimitRequestBody, LimitRequestFields, LimitRequestFieldSize, LimitRequestLine*** and ***LimitXMLRequestBody*** directives to limit resource consumption by users.
  - If the underlying OS of the server is supported, the ***AcceptFilter directive*** could be used to offload some of the request processing to the OS. The configurations for this are system-specific.
- By default, Apache is started by the root user, and then switches to the user defined in the ***User directive***. It is considered good practice to have a specific user with specific permissions.
- Prevent users from adding .htaccess files by adding a Directory directive which forbids access to system settings. This is done by default since Apache 2.3.9.
- Protect server files by refusing access to all directories.

- The ***mod_proxy*** module automatically adds the following headers when proxying a request: X-Forwarded-For, X-Forwarded-IP, X-Forwarded-Server, X-Powered-By. These contain information about the proxy and/or the backend, so it is advisable to unset them.
- Create customized error pages that do not leak any internal information such as the backend/proxy address, the operating system, or the version of the proxy used.

Authentication

Enterprise Web Chat does not include a mechanism for authenticating customers i.e. web chat originators, beyond the optional simplistic use of the customer's email address; a customized mechanism must be developed by adopters of the API.

Session Key Management

A session key has been implemented to hold the state of the customer's session, and to reconnect them if the chat is temporarily disconnected. If a null or invalid session key is received, the chat is refused. The Reference Web UI uses the browser's SessionStorage to hold details when logging into the chat. This is automatically emptied upon closing the chat, or when the browser session is closed.

Authorization

There are no authorization methods for websocket requests – they have access to the full range of messages. However, the range of messages that the customer can send are limited to normal chat messages and page push messages. Page push messages allow URLs to be sent to the agent, but agents must manually click these links. The authorization mechanism for agents consists of checking if the agent is a supervisor. Supervisors may observe and barge into a chat at their discretion, and remove the current agents from the chat. Both agents and supervisors may view a customer's history, but only when in a chat with that customer.

Data Validation

Data validation is achieved through a JSON validation schema on Contact Center.

Denial of Service

AACC currently includes an option to limit the maximum number of concurrent chats per customer. The default limit is 5 chats per customer, with a maximum of 50 chats in the queue for a particular web chat skillset at any moment. These can be configured in the Multimedia Admin for AACC. Callbacks requests are limited to three per customer. This is judged entirely on the presence of an email address. Other options to mitigate Denial of Service attacks would be to configure Tomcat or any proxies in front of AACC to limit the size of requests.

Data Protection

The Multimedia Admin includes an optional field for a link to a transcript filtering service. A sample service is available for testing, but this should be replaced with a customized one which will filter out sensitive data before the transcript is persisted to the database. This service MUST be accessed over a secure connection.

Chapter 6: Deployment and Configuration

This section provides deployment and configuration steps required when using the Enterprise Web Chat feature with Avaya Aura Contact Center. The information contained in this guide is specific to the Enterprise Web Chat feature and should be used in conjunction with the published Contact Center customer documentation. On completing the steps outlined in this guide, it will be possible to run the Reference Web User Interface (Customer facing) and reference Custom Desktop (agent/supervisor facing) applications with your Contact Center.

Deployment Specification

The following provides an overview of supported deployments for Enterprise Web Chat:

Enterprise Web Chat (EWC): Supported only in the following deployments		
Minimum AACC Release:	AACC 7.0	ACCS 7.1
Deployment Type:	Avaya Aura only *Enterprise Web Chat is not supported on CS1000 or IP Office deployments*	
Installation Type:	Voice Contact Server + Multimedia Contact Server on separate servers. *Enterprise Web Chat is not supported on co-resident Voice and Multimedia deployments.*	
Campus High Availability:	Campus Multimedia Server HA combined with Mission Critical Voice Contact Center deployment.	
Remote Geographic Node:	Multimedia Contact Server RGN combined with Campus solution.	

Licensing:	EWC agents must be licensed for Web Comms contact type. Enterprise Web Chat Feature must be licensed
EWC Agent & Supervisor support:	
Agent / Supervisor Desktop:	A Custom Desktop implementation required Enterprise Web Chat API (Web Socket based) and SDK are provided EWC is supported with Avaya Agent Desktop
Reference Custom Desktop:	EWC Reference Custom Desktop provided as part of this SDK. This combines limited CCT API integration and EWC API integration to allow chat handling. Provided as a guide for developers and solution implementers
Reference Custom Desktop – OS support:	.NET application deployed and verified on: ▪ Windows Server 2012 R2 ▪ Windows 7 Professional SP1 ▪ Windows 8.1 Windows 10
EWC Customer Interface:	
Customer Interface	New Enterprise Web Chat SDK (Web Sockets based) is provided as part of this SDK ✱ **Note**: AACC 6.x web facing chat applications are not compatible. Redevelopment is required.
Reference Web UI	EWC Reference Web UI (Web application) is provided as part of this SDK. This is provided as a guide for developers and solution implementers
Reference Web UI – Web Server Support:	Reference Web UI has been deployed and verified on: ▪ Tomcat 8 ▪ Ubuntu 15.04 ✱ Note: the above have been utilized as host web servers for development & test implementation only.
Reference Web UI – Browser Support:	Reference Web UI has been deployed and verified on: ▪ Internet Explorer: versions 10 & 11 ▪ Firefox: versions 39 - 42 Google Chrome 45 - 47

Configuring Custom Desktop Application Settings

Before you begin

The reference Custom Desktop application is provided in zip file format and is available as part of the Enterprise Web Chat SDK software. Ensure that you have extracted the contents of the zip file to your system, e.g. C:\Avaya EWC Custom Desktop. Once extracted, navigate to the 'bin' directory. This folder contains all the files necessary to run the Custom Desktop application.

About this task

Before running the Custom Desktop application for the first time, you must edit the AppSettings.xml file, this file is located in the previously extracted 'bin' directory. The AppSettings.xml file contains a number of server, user and security settings that must be defined prior to logging into Contact Center.

Once this task is complete, it will be possible to launch the Custom Desktop by running the CustomDesktop.exe application.

Procedure

1. In the extracted 'bin' folder, edit the *AppSettings.xml* and set the following values:
2. Set the hostname or IP address of the default CCT server in the `DefaultCCTServer` entry.

   ✱ **Note:** In High Availability (HA) / Business Continuity (BC) system this should be set to the active Contact Center voice server.
3. Set the CCT username of the agent or supervisor in the `DefaultCCTUserID` entry. This is the account that will be used to login to AACC when running Custom Desktop.
4. Set the domain of the configured CCT user account in the `DefaultCCTUserDomain` entry.

   ✱ Note: In HA/BC system user accounts must be part of a shared Active Directory domain. This ensures that the account is available in the event of a server outage.
5. Set the hostname or IP address of the default Multimedia server in the `DefaultMMServer` entry.

   ✱ **Note:** In HA/BC system this should be set to the managed hostname or managed IP address of the Multimedia pair.
6. Set the URL of the WebSocket in the `WebSocketURL` entry.

   Security on:
   wss:///(Multimedia Server IP):{SECURE_PORT}/AgentControllerWeb/chat
   Security off:
   ws:///(Multimedia Server IP):8081/AgentControllerWeb/chat

   ✱ **Note:** In HA/BC system this should be set to the managed hostname or managed IP address of the Multimedia pair. When the original standby server takes over as he active server, it takes ownership of the managed name / IP address. Setting the WebSocketURL as the managed name /IP address ensures that the Custom Desktop is always connected to the currently running server.
7. Set a value of false in the `HAEnabled` entry for solutions that are not using High Availability.

   ✱ **Note:** In HA/BC system this should be set to **true**
8. Set whether the Custom Desktop should automatically reconnect if disconnected unexpectedly. This is set in the `AutoReconnect` entry.

   This setting can either be true or false.
9. Set whether security is switched on for an email server (if applicable). This is set in the `IsSecure` entry.

This should be set to true if the security settings are activated on the CCMM server (i.e. set to 'on' in the Contact Center SecurityManager application). Otherwise, it can be set to false.

If this is not set correctly, agents will not be alerted about incoming emails.

10. Set the location where email attachments (if applicable) will be downloaded to in the `DownloadAttachmentsLocation` entry.
11. All other configuration settings can be left unchanged.
12. Save changes to the *AppSettings.xml* file.

✱ **Note:** In HA/BC system additional steps are required:

- Set the hostname or IP address of the secondary CCT server as the `CampusAlternateServer` entry. This should be set to the standby Contact Center voice server. In the event of the default CCT server being unavailable, this one will be used. In the event of the Campus Alternative Server being unavailable, the Default CCT server will be used.
- Set the hostname or IP address of the RGN Voice (CCT) server in the `GeographicAlternateServer` entry.

  ✱ **Note:** This is only required if a Remote Geographic Node is deployed in addition to a Campus HA/BC solution.
- Set `RGNWebSocket` as the Web Socket URL of the RGN CCMM server.

  ✱ **Note:** This is only required if a Remote Geographic Node is deployed in addition to a Campus HA/BC solution.
- Set `RGNMMServer` as the hostname or IP address of the RGN CCMM server.

  ✱ **Note:** This is only required if a Remote Geographic Node is deployed in addition to a Campus HA/BC solution.

Enabling Cross-Origin Resource Sharing Filtering

Cross-Origin Resource Sharing (CORS) is a recent W3C effort to introduce a standard mechanism for enabling cross-domain requests from web browsers to servers that wish to handle them.

About this task

Cross-domain requests in JavaScript (such as REST calls) are blocked by default, as they violate the Same-Origin Policy. To summarise this, the origin of a request consists of the protocol, the host and the port number, e.g. *http://localhost:8080* and *http://localhost:8082* are completely separate origins, even though they are on the same server and use the same protocol. The purpose of this policy is to limit cross-site forgery attacks and prevent malicious code from reading data from a domain.

The result of this is that if the Reference Web User Interface is deployed on *http://exampledomain1.com*, but Contact Center is deployed on *http://exampledomain2.com*, the REST call from the customer's browser will be rejected, and therefore the customer will not be able to enter the chat.

CORS filtering is enabled to allow CORS support to Java web applications. However, you need to specify the allowed domains.

Procedure

1. Open the Multimedia Administration utility. See Starting the CCMM Administration utility.
2. In the left pane, select Web Comms.
3. Click Config
4. Enter the domain name of the Reference Web UI e.g. exampledomain1

   ✱ **Note:**
   All that is required is the IP address or the domain name. Do not put in the full URL such as *http://exampledomain1.com*. Alternatively, enter an asterisk '*' to allow from all origins (not recommended for security reasons).
5. Press save.
6. Changes are applied immediately. It is not necessary to restart Contact Center or Tomcat services to apply changes.

Enabling Transcript Filtering Web Service

Before you begin

A sample Custom Filter Messages application is provided in zip file format and is available as part of the Enterprise Web Chat SDK software. Ensure that you have extracted the contents of the zip file to your system, e.g. C:\CustomerFilterMessages. Once extracted, the following folders are available:

'dist' folder: This contains a sample WAR file 'CustomFilterMessages.war' containing a test REST interface. This can be used to take a list of messages that will be stored in the database and applies a very basic filter to each one. In the case of this simple example, the text 'been filtered' will be added to the end of each message.

✱ **Note:** This application is intended as a basic test application to illustrate the transcript filtering integration. In a production environment, a more suitable filtering application would need to be developed. To use the test application, deploy the 'CustomFilterMessages.war' file on a web server e.g. Tomcat. This must not be installed on the Contact Center server; instead an internal network application server should be used. The server used to host this application must also have a valid security certificate.

'src' folder: This contains the source code for the Custom Filter Messages sample application.

About this task

Some customers may want to mask chat transcripts in order to hide sensitive data, such as account details etc. This functionality can be achieved in Enterprise Web Chat by making a REST call before the chat transcript is stored in the database. The sample filtering application described above can be used as a guide to integrating this feature into your solution. The procedure below explains how this can be configured and assumes you have already deployed a transcript filtering web application on a filtering web server.

Procedure

1. Open the Multimedia Administration utility. See Starting the CCMM Administration utility.
2. In the left pane, select Web Comms.
3. Click Config
4. In 'Transcript Filtering Web Service' put the path of the CustomerFilterMessages REST interface:

   If security is switched off (on the CCMM server) use the server IP of the server hosting the filtering application in the path:
   e.g. *http:///(IP address):8081/CustomerFilterMessages/rest/filter/*

   OR

   If security is switched on (on the CCMM server) use the hostname or FQDN specified in the certificate (see additional security information below) and SECURE PORT specified in Security Manager:
   e.g. *https:///(hostname):8445/CustomerFilterMessages/rest/filter/*

   ✱ **Note:** Ensure 'CustomerFilterMessages/rest/filter/' is included after the port.

   Additional security configuration if Contact Center Security is enabled (via the Certificate Manager on the CCMM server):

   ✱ **Note:** These procedures assume that the AACC default certificate is being used.

   a. The certificate name can be obtained from the server on which the CustomerFilterMessages WAR is located.
   b. To get the cert from the CCMM server:
      i. Export it using Security Manager on the CCMM server.
      ii. Open the keystore (on Contact Center installs this is D:\Avaya\Contact Center\Common Components\CCKeyStore) and export the certificate from there using the command line:

```
keytool -export -alias mydomain -file mydomain.crt -keystore keystore.jks
```

      ✱ **Note:**
      You can check which certificates are in the keystore with this command on the command line:

```
keytool -list -v -keystore keystore.jks
```

      You can check the cert name here.
   c. Add the CustomerFilterMessages server's cert to the Multimedia server's keystore by copying the CustomerFilterMessages server's cert over to the keystore directory on the Multimedia server and then using this command:
      `keytool -import -trustcacerts -alias mydomain -file mydomain.crt -keystore keystore.jks`
   d. Add the Multimedia server's cert to the CustomerFilterMessages server's keystore by copying the Multimedia server's cert over to the keystore directory on the CustomerFilterMessages server and then using this command:

```
keytool -import -trustcacerts -alias mydomain -file mydomain.crt -
keystore keystore.jks
```

e. Insert the IP and cert name of the CustomerFilterMessages server into the hosts file (C:\Windows\System32\drivers\etc\) of the Multimedia server

f. Insert the IP and cert name of the Multimedia server into the hosts file (C:\Windows\System32\drivers\etc\) of the CustomerFilterMessages server

Deploying the Reference Web User Interface

Before you begin

A Reference Web User Interface is provided in zip file format and is available as part of the Enterprise Web Chat SDK software. This sample customer web site represents the customer facing interface to request and engage in EWC chat conversations with EWC enabled Contact Center agents. Ensure that you have extracted the contents of the zip file to your system, e.g. C:\CustomerFrontendZip. Once extracted, note the following folder structure:

'src' folder: This folder contains the complete source of the Reference Web User Interface

'src\dist\dist' folder: This folder contains a distribution version of the application that can be deployed on your web server. Note: This should not be deployed on the Contact Center server.

About this task

This task explains how to deploy the reference customer-facing web user interface. A sample deployment on Apache Tomcat is as follows

Procedure

1. Navigate to the webapps directory of your Tomcat installation (e.g C:/servers/apache-tomcat/webapps).
2. Create a sub-directory for the CustomerFrontend (e.g. mySite)
3. Copy the contents of the distribution folder (mentioned above): e.g. all files and folders contained in 'src\dist\dist\' folder into the newly created sub-folder.
4. Restart Tomcat to ensure that it picks up the new site.

Configuring Reference Web User Interface

About this task

Configure the reference web user interface to communicate with the REST and Websocket listeners on the CCMM server.

Procedure

1. Edit the links.js file included in your deployed Reference Web User Interface application. Assuming you have deployed the application as described in the previous section, this file will located in: .\js\links.js.

   ✱ **Note:** If you need to locate this file in the output of the ZIP file, it is located in '*src/main/webapp/js'* and '*src/dist/dist/js'*

2. Edit the URL variables contained in the links.js file to point to your CCMM server IP address or hostname (assuming you are not using a proxy server). If security is turned on for the CCMM server, the URLs ***must*** use secure protocols (wss or https) and the secure port (8445). These fields are shown below:

   a. var webChatUrl = 'wss://<CCMM Server IP Address>:8445/CustomerControllerWeb/chat';
   b. var restUrl = 'https:// <CCMM Server IP Address>:8445/CustomerControllerWeb/currentqueue';
   c. var callbackUrl = 'https:// <CCMM Server IP Address>:8445/CustomerControllerWeb/callback';

   - ✱ **Note:**If security is not enabled on your CCMM server you should update the protocols to http and ws (non-secure) and change the port to 8081 (non-secure).
   - If you are using High Availability, use the managed name or IP Address for the CCMM server.

3. Ensure that you have set the External Web Server Domain field in the Multimedia Admin component. This has already been detailed in this guide – refer to the section 'Enabling Cross-Origin Resource Sharing Filtering'
4. Ensure that EWC enabled agents are logged into a Web Communications skillset and that the skillset has been assigned a valid route-point in the Multimedia Admin application.
5. Use your Internet browser to browse to your deployed web site and confirm that a chat request option is available.

Enabling Estimated Wait Time and Position in Queue

Before you begin

The Enterprise Web Chat feature can be configured to display Expected Wait Time and Position In Queue information to customers, this is done via the Reference Web User Interface. There is additional configuration required in order to use this feature. It is necessary to modify your Multimedia Flow using Orchestration Designer (assuming you are using the default flow) and also requires use of the Database Integration Wizard. The process is as follows: Once the EWC contact is queued to a skillset, it is possible to check the Expected Wait Time & Position In Queue intrinsics using Orchestration Designer. Using a Host block, the intrinsic values and the contact ID are passed to a soap web service which sets the value in the Contact Center database. The SOAP web service is part of the ***AccQueueService***, deployed as part of the EWC & CCMM server installation.

The Expected Wait Time & Position In Queue information is then available to the customer. The full configuration procedure is detailed below:

Note: Expected Wait Time & Position in Queue information will only be displayed if the wait time is greater than 60 seconds and the position in queue is greater than 0.

Refer to the following guides for more information on using Orchestration Designer and the Database Integration Wizard:

- Contact Center Server Administration guide: Database Integration Wizard configuration -> Configuring Web Services
- Using Contact Center Orchestration Designer guide

About this task

How to configure orchestration designer and update the multimedia flow so that the correct values are calculated for a customer's position in the queue and the estimated wait time based on average time it takes to answer a chat in that queue.

Procedure

1. Add the web service using the Database Integration Wizard on the core Voice Contact Center server:
   a. Open Database Integration wizard.
   b. Test connection to the default provider ID, 121.
   c. Add the package name and the address of the WSDL, e.g. *http://<multimediaServerIPAddress>:8081/AccQueueService/services/QueueServiceWs?WSDL,* then click import.
   e. If Contact Center security is on, the fully qualified domain name (FQDN) of the server must be used with SECURE PORT specified in Security Manager, e.g. *https://<multimediaServerFQDN>:8445/AccQueueService/services/QueueServiceWs?WSDL,* then import the X.509 certificate for the multimedia server and click import.
   d. Test the soap services and record the number for the ***updateStartingEwtandPIQ*** statement.
2. Open Orchestration Designer
3. Open the Multimedia Flow
4. Once in Queue view, add a new 'Host' block between *Queue to Skillset* and *Check Total Active Contacts*.

   ✱ **Note:** If you are using a flow other than the standard Multimedia flow to queue web chat contacts, ensure the Host block is added after the chat contact has been queued.
5. Open the Host block and changes its name e.g. *CallWebService*
6. In the Host block, click on the 'Setup' tab and add the Assignment expressions below: (*ProviderContactID* is the only variable to be typed the rest can be found in the variables menu)

   HDX_QueryNum_cv = 2
   externalCallid = CONTACT DATA ProviderContactID
   c_position_in_queue_cv = POSITION IN QUEUE contact_skillset_cv
   c_estimated_wait_time_cv = EXPECTED WAIT TIME contact_skillset_cv

   ✱ **Note:** *'HDX_QueryNum_cv' refers to the number assigned to the query when being added via the Database Integration Wizard.*

7. Select the host tab and ensure that the *Send Info* radio button is selected. Set the ProviderID to *HDX_Appid.*
8. In the Variables input box, add the the variables in the same order as shown below. This will call the SOAP web service (identified by the query number) and pass the contact ID, estimated wait time and position in queue values as parameters.

   HDX_QueryNum_cv
   externalcallid
   c_estimated_wait_time_cv
   c_position_in_queue

9. Save flow and exit to save changes.

   ✱ **Note:** *The PIQ and EST won't be displayed on the reference web UI unless PIQ is greater than 0 and EWT is greater than 60 seconds.*

Chapter 7: Using the Reference Custom Desktop Client

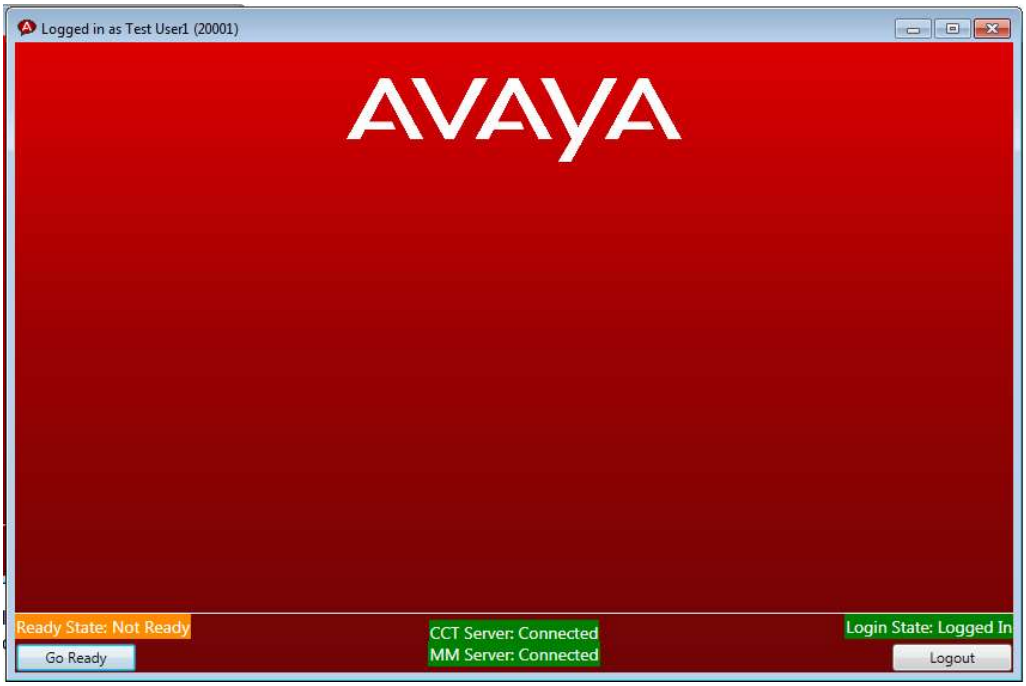
This section describes the functionality provided by Enterprise Web Chat Custom Desktop application for AACC.

Logging on to Custom Desktop

Steps in logging in an agent

1. Launch the Custom Desktop application by running the CustomDesktop.exe, provided in the Custom Desktop folder.
2. The agent will be presented with a Login screen containing the following fields;
    i. FQDN or Address of the CCT server
    ii. Port Number of the CCT server
    iii. User ID
    iv. The Domain address for the CCT User Account
    v. Password
    vi. CCMM password (optional)

3. Four of the fields will be pre-populated from the appSettings.xml file in the custom desktop. (See Configuring Custom Desktop Application Settings for more information)
4. The agent must enter their user ID and corresponding Password to log in. The agent will also need to enter their CCMM password if the “Use default CCMM password” checkbox is unchecked. Otherwise, the agent logon id is automatically used as the CCMM password.
5. Once the agent has been successfully logged in, the Ready State of the agent is set to “Not Ready” and the Login State is set to “Logged in”.



6. The agent needs to press the Go Ready button in order to set their Ready State to “Ready”.

   Once the agent is logged in and in a Ready State, customers who require a particular skillset can be routed to a matching agent.

   ✱ ***Note regarding Call Presentation classes when using the Reference Custom Desktop with EWC.***
   The ‘Call Force Delay’ call presentation option (agent automatically answers call after a designated amount of time) in CCMA is not implemented on Custom Desktop and the contact will continue to alert indefinitely until manually answered. Any agent using the desktop and assigned to this call presentation class will have to decline the new contact to return it to the skillset queue.

Accepting a Web communications contact

An agent accepts a Web communications contact to communicate privately with a customer in real time over the Internet.

An agent and a customer can use the text chat component of the Custom Desktop to conduct a two-way conversation by exchanging messages.

Before you begin

- Ensure that the Ready Status is set to “Ready”.
- Ensure that the agent who is logged in is assigned to a skillset for handling Web communications.

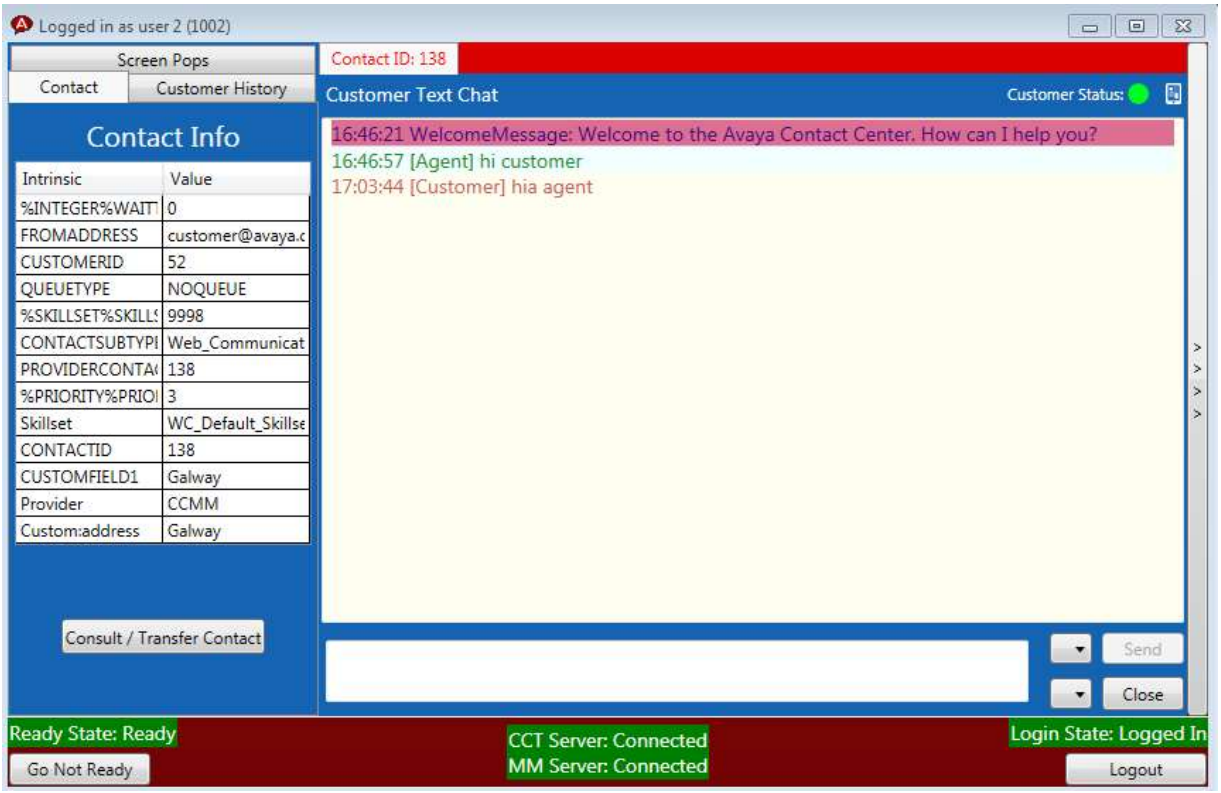
About this task

The agent will be presented with a notification that a customer chat has been routed to them by a flashing icon in the center of the bottom bar on the Custom desktop.

This icon informs the agent which skillset the customer has requested and allows the agent accept the contact using the green speech bubble.

Note: An agent can receive multiple contacts. The site administrator can configure the maximum number of additional contacts that will be routed to an agent.

If the agent accepts the Web Chat they will be presented with a screen containing 2 main panels;



1. **Customer Text Chat.**
   - The Customer Text Chat panel, on the right, is the main interface for discourse between the agent and the customer.
   - It will record and timestamp all messages sent from the customer to the agent and agent to customer until the agent or customer closes the Chat.
   - All messages are colour-coded to differentiate between customer and agent messages.
   - The agent will be informed in this panel when the current customer is typing a message.
   - The agent can type and send messages to the customer using the text field and send button in the bottom panel of the frame.
   - The agent can close a conversation using the Close button.
   - The agent can push a URL or an Auto-phrase to the customer using the drop-down menus in the bottom panel of the frame.
   - The agent is informed of the device type that the customer is logged in on.
   - The agent is informed of the customer connection status.

2. **Contact Info.**
   - This customer Info panel is responsible for presenting information about the connecting customer in three tabs.
   - The Contact tab shows Information about the currently connected chat, such as the customer's email address, requested skillset and other relevant information.
   - The customer History tab allows the agent to view previous conversations with the customer if they have used an email address to identify themselves.

- The Screen pops tab allows the agent to access web sites and applications within the Custom Desktop.

Declining a Web communications contact

About this task

An agent can decline a Web communications contact when they are not available to chat with a customer and want to return the contact to the queue.

Procedure

When the Chat notification appears, click the red speech-bubble to reject the incoming call. The contact returns to the queue and into Web-on-hold. The agent status is set to Not Ready.

> ✱ **Note:**If two or more Web communications contacts are simultaneously presenting to the agent on Custom Desktop and one Web communications contact is rejected, all contacts are rejected.

Sending a chat message

About this task

Send a chat message to communicate privately with a customer in real time over the Internet.

Procedure

1. Type the message.
2. Click the Send button.

The agent name or agent label appears in the Conversation frame, followed by the message.

Adding an auto-phrase to a chat message

About this task

Use the automatic phrase drop-down to select a commonly-used phrase from a configured list and send it to the customer.

Procedure

1. Select the auto-phrase drop box.

2. Select the auto-phrase to use from the list.

The phrase is automatically added to the Conversation frame and the text chat log.

Pushing a Web page to a customer

About this task

It is possible to send and open a Web page on a customer's browser.

An agent uses the Page Push feature to push a Web page to the customer.

The customer can also push a Web page to the agent's browser.

Page Push has the following limitations:

- Some Web pages do not appear the same when pushed to another person's browser. For example, in the case of Web pages that can be personalized to display weather or news, each person sees their own version of the page.
- Dynamic pages that are customized through the use of cookies may be presented differently because different cookies will be present on the customer and agent's browsers.
- Framed pages cannot be pushed completely. The URL defining the frameset is pushed, but the individual URLs in each frame are not pushed.
- After a page is pushed, if either the agent or the customer follows a link to another page, the other person does not see that change.
- If the agent and the customer both push the selected web page at approximately the same time, one sees the form that was pushed last. However, both URLs appear in the text chat log.
- The Page Push URLs are not saved to the text chat log.
- The Page Push URLs are predefined from the Multimedia admin. An agent can send any URL from the predefined list.

Procedure

1. Select the Page Push drop box.

2. Select the URL to send from the list.

The URL is automatically added to the Conversation frame.

Performing a Web Communications transfer to a skillset

About this task

An agent can transfer a Web Communications (WC) contact to a skillset, if they have the correct permissions.

The agent who accepts the contact is presented with a transcript of the preceding conversation between the customer and the previous agent.

Important:

A customer cannot be transferred to a skillset that does not have active agents.

Procedure

1. Click Consult/Transfer Contact in the Contact tab.

2. In the left pane, under Transfer, select the skillset from the drop-down list.

3. Optionally, enter the reason for transfer under Transfer Reason.

The agent that answers the transferred contact can view the transfer details, including the reason for the transfer.

4. Click Transfer.

Custom Desktop refreshes the list of agents active on the selected skillset and queues the contact to the selected skillset.

Transferring a chat session

About this task

Transfer a chat session to another available agent. A chat session can be transferred only once.

Procedure

1. Click Consult.

2. In the left pane, under Transfer, select the skillset from the drop-down list.

3. Click the appropriate agent name.

4. Click Consult.

The Conversation window splits into two. In the Consult Text Chat window, the consulting agent can inform the consulted agent about the reason for the transfer.

> ✱ **Note:** The conversation between the agent and the customer is visible to the consulted agent.

5. Click Complete Transfer.

The chat session is now transferred to the new agent and it is no longer displayed on the previous agent's Custom Desktop. The status of the agent who has left the conversation is set to Ready.

Conferencing a chat session

About this task

An agent can choose to Conference with another agent and a customer, allowing all parties to communicate.

Only one extra agent can be added to a conference.

Procedure

1. Click Consult.

2. In the left pane, under Transfer, select the skillset from the drop-down list.

3. Click the appropriate agent name.

4. Click Consult.

The Conversation window splits into two. In the Consult Text Chat window, the consulting agent can inform the consulted agent about the reason for the conference.

> ✱ **Note:** The conversation between the agent and the customer is visible to the consulted agent.

5. Click Complete Conference.

The Consult Text Chat window closes and the original Conversation window reappears.

When the agent who was added to the conference begins typing, it is visible to all three participants.

> ✱ **Note:** On Custom Desktop, for the agent who initiated the conference, the Consult button is not visible for the entire duration of the Web communications contact. The agent cannot initiate another consult during this period.

6. An agent can leave a conference by clicking the Close button. If both agents leave the Chat, it is closed.

A message appears in the Conversation window that indicates when a participant leaves the conference.

Observing a Web Communications contact

Before you begin

- Log in a supervisor/agent to the Custom Desktop who has the Observe privilege set.

About this task

A supervisor/agent can observe an agent-customer Web Chat. When a supervisor/agent starts observing a Web Communications contact, the agent receives a notification that a supervisor/agent is observing the contact. The agent can continue the conversation without interference from the

supervisor/agent. The customer does not receive a notification and is not aware that a supervisor/agent is observing the contact.

Note:

The supervisor/agent must log on to the Custom Desktop to observe a Web Communications contact.

When a supervisor/agent is observing a contact, only the agent already interacting with the contact can set an activity code for that contact.

When a supervisor/agent logs in, they are presented with the Supervisor Observe tab window.

Procedure

1. Select an agent under that supervisor from the drop box.

In the Contacts list, the red flag indicates the contacts whose intrinsic values are exceeding the threshold defined by the administrator in the CCMM Administration tool.

2. Move the mouse cursor over contact to view the intrinsic data for that contact.

The intrinsics are assigned with priority 1 to 5. Contacts exceeding threshold limits for high priority intrinsics, appear higher in the Contacts list.

> ✱ **Note:** Administrators must configure intrinsic priorities in the CCMM Administration tool.

3. Select a Web Communications contact and click Observe.

The Custom Desktop displays the customer Text Chat window.

> ✱ **Note:** When using the Observe function, a supervisor/agent cannot participate in the Web Communications contact. Use the barge-in function to participate in the Web Communications contact.

4. A supervisor/agent can click the Close button to finish observing the Web Communications contact.

Barging-in on a Web Communications contact

Before you begin

Log in a supervisor\agent to the Custom Desktop that has the Observe and Barge privilege set.

About this task

A supervisor/agent can participate in an agent-customer Web Communications contact.

> ✱ Note: The supervisor/agent must log on to the Custom Desktop to barge-in on a Web Communications contact.

The supervisor/agent can barge-in on a Web Communications contact only after they have observed that Web Communications contact.

After barging, the supervisor becomes the owner of the chat room and is allowed to send messages to the contact. The original agent becomes passive and can no longer send messages to the customer, but can still whisper to the supervisor.

Procedure

1. In the Supervisor Observe window, click Barge-in, to barge-in on a Web Communications contact that the supervisor/agent is observing.

> ✱ **Note:** The Barge-in button for a supervisor/agent is enabled only after they observe the agent's Web Communications contact.

The Custom Desktop displays the customer Text Chat window.

2. In the Text box, type a message to the customer.

3. Click Send.

The agent name or agent label appears before the message in the Conversation frame.

4. Click Close when the conversation is complete.

The agent-customer Web Communications contact continues until all the agents or the supervisor/agents have left the Web Communications contact or the customer leaves the Web Communications contact.

Ending the Text Chat session

About this task

End the text chat session when the contact is completed. The text chat history is saved automatically and can be emailed to the customer.

Procedure

1. Click Finish.

2. In the Notes box, type additional information about the contact.

3. If Closed Reasons are required, select one of the configured Closed Reason codes that best describes the reason for closing the text chat session.

4. Click Close.

5. Repeat step 1 to step 4 to close each additional Web communications contact.

Sending the chat log in an email message

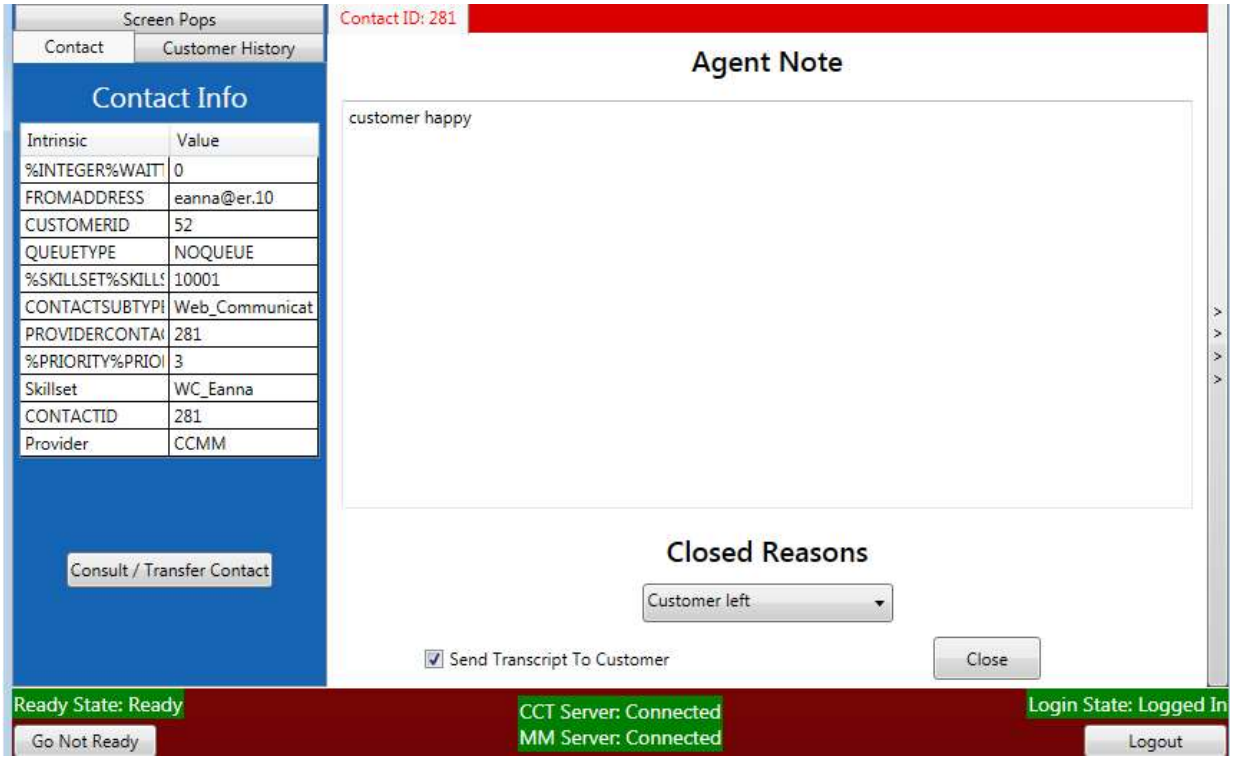
About this task

If the Contact Center is licensed for email, an agent can send a copy of the completed text chat log by email to the customer.

Procedure

When an agent closes a chat the agent will be presented with a screen where they set the reason that the Chat was closed. On this screen, next to the Closed Reasons, there is a check box marked “Send transcript to customer”.

Tick this box to email the transcript to the customer.



Using screen pops on Custom Desktop

The screen pop feature allows administrators to configure a website or application that will be presented (screen popped) to an agent while they are handling a chat contact. This can be configured via the CCMM Administration application (Complete details of how to use this application are contained in the Server Administration document.)

Custom Desktop displays screen pops when a contact alerts or is answered. Screen pops can display relevant information about the alerting or answered web chat based on the contact ID. Screen pops can be used to open a browser in the Custom Desktop or to launch an application such as Notepad.

IMPORTANT: Enterprise Web Chat and the Reference Custom Desktop will pass the contact ID as a parameter to the screen pop application. It does not use any intrinsics that may be configured within CCMM Administration.

Screen pops open in an additional tab within Custom Desktop (see Figure 1 & Figure 2).

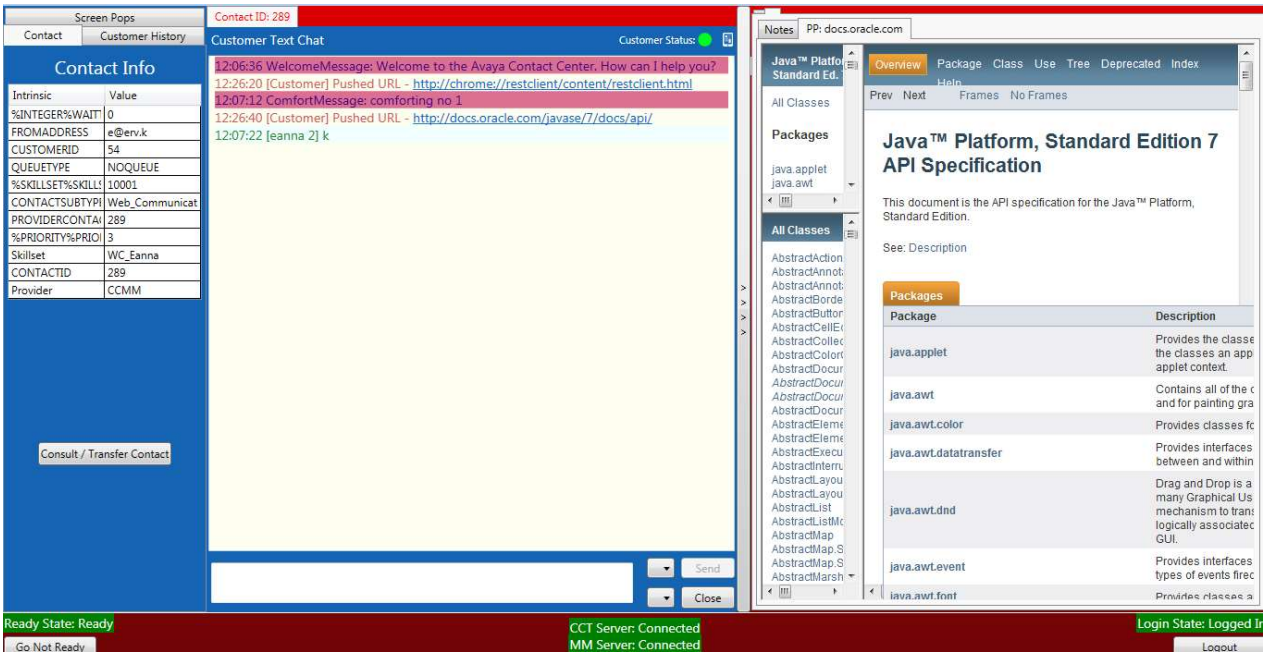


Figure 1

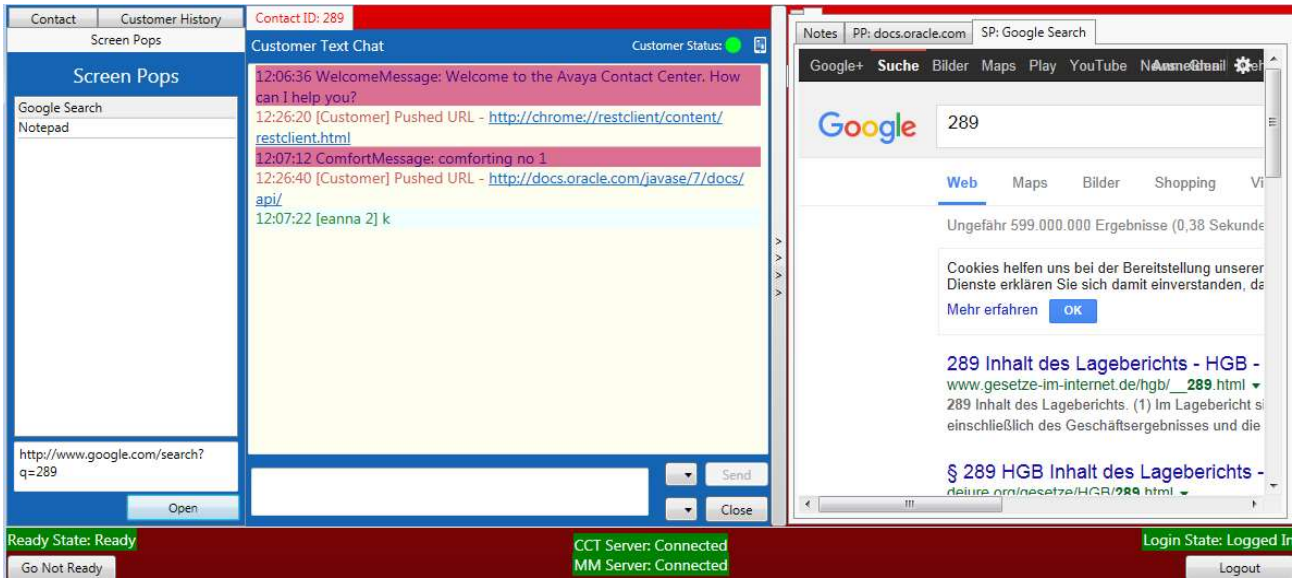


Figure 2

Customize Custom Desktop Window Settings

You can customize the following window settings on Custom Desktop:

- Window size and position.
- Window state (maximized or minimized).
- Multimedia panel state (expanded or collapsed).

Once you close and reopen Custom Desktop, the system restores the window settings of Custom Desktop.

In a multiple screen setup, once you close and reopen Custom Desktop, Custom Desktop opens on the screen that Custom Desktop was last open on.

> ✱ **Note:** On Windows 7 and Windows 8.1, you can configure the Control Panel Display settings to change the size of text and other items on your screen. You must not change the default display settings. Custom Desktop supports the Smaller - 100% (default) option only.

Changing agent status to Ready

About this task

Change the agent status to Ready when they are available to create and receive contacts.

Procedure

1. On the Custom Desktop Top bar menu, click the Status icon.
2. Select Go Ready to create or receive both Voice and Multimedia contacts.

   The status icon changes to green.

Changing agent status to Not Ready

About this task

Change the agent status to Not Ready when they are unavailable to receive contacts.

Procedure

1. On the Custom Desktop Top bar menu, click the Status icon.
2. Click Go Not Ready.

   The Status icon changes to orange.

   The status changes to Not Ready on the Top bar.

Changing agent status to Not Ready when on a contact

About this task

Change the agent status to Not Ready while active on a contact to indicate that they are not ready to accept contacts when the current contact is complete.

Procedure

1. While active on a contact, on the Custom Desktop Top bar menu click the Status icon.

2. Click Go Not Ready.

The Status icon changes to orange.

Logging off from Custom Desktop

Before you begin

• Ensure that there are no open contacts. If a contact is open the agent will not be permitted to log out.

About this task

A warning message appears asking the agent to confirm that they want to close the Custom Desktop and log off.

Procedure

1. On the Custom Desktop Top bar menu, click the Status icon.

2. Click Log Out.

Custom Desktop logs off. The status icon changes to red and the Top bar displays the Logged Out status.

Checking Customer History

About this task

The customer history information is located in the “Customer History” tab in the left pane. An agent can use the Customer History pane to view the details of previous interactions with the customer, including automatic responses sent and the content of previous email messages.

Procedure

1. On the Custom Desktop Customer History pane, use the “Amount of customer history” textbox to enter the number of transcripts to be viewed.

2. Click Get History.

The specified amount of transcripts of that customer will be returned in a list ordered by most recent.

3. Click on a transcript and click View History.

Accepting an Incoming Email Message

About this task

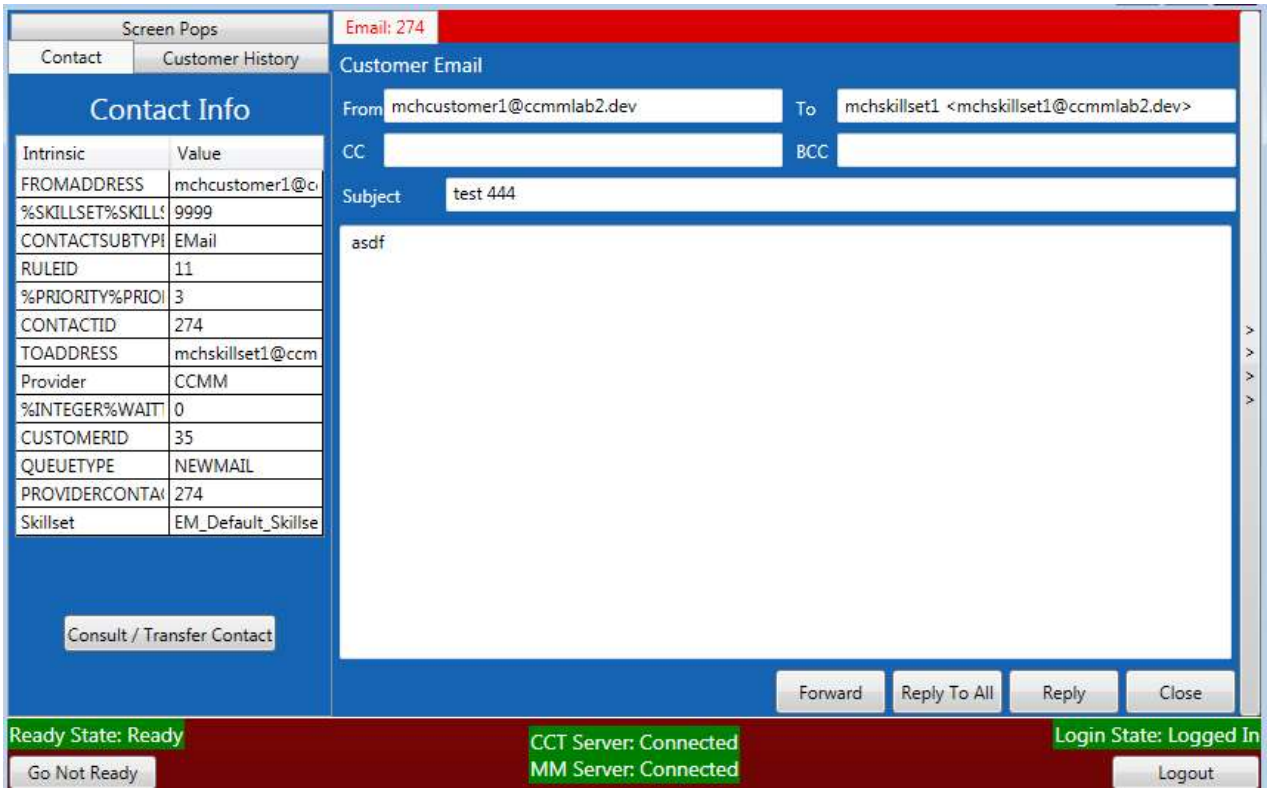
An agent can accept an incoming email message when they are ready to receive the customer's email, display customer details and begin contact with a customer. The Custom Desktop displays the customer details and the call timer appears on the work item. The new incoming email message is presented as a new work item in the Work List window.

Procedure

On the Custom Desktop, click Accept.

The email message opens in the E-mail Display panel.

The customer details associated with the email message appears in the left pane in the Customer Info panel.



Declining An Incoming Email Message

About this task

Decline an incoming email message if the agent cannot handle the email message. The new incoming email message disappears from the agent's work list.

Procedure

On the Custom Desktop, click Reject.

The contact returns to the queue and the agent's status is set to Not Ready.

Chapter 8: Using the Avaya Aura Contact Center Enterprise Reference Web User Interface

This section describes the use the of the customer-facing web user interface reference implementation of Avaya Aura Contact Center Enterprise Web Chat.

Overview

Use the web user interface to provide your customers with an interface to request online chat or agent call back. There are two main components, the *online chat panel* and the *scheduled callback panel* illustrated below

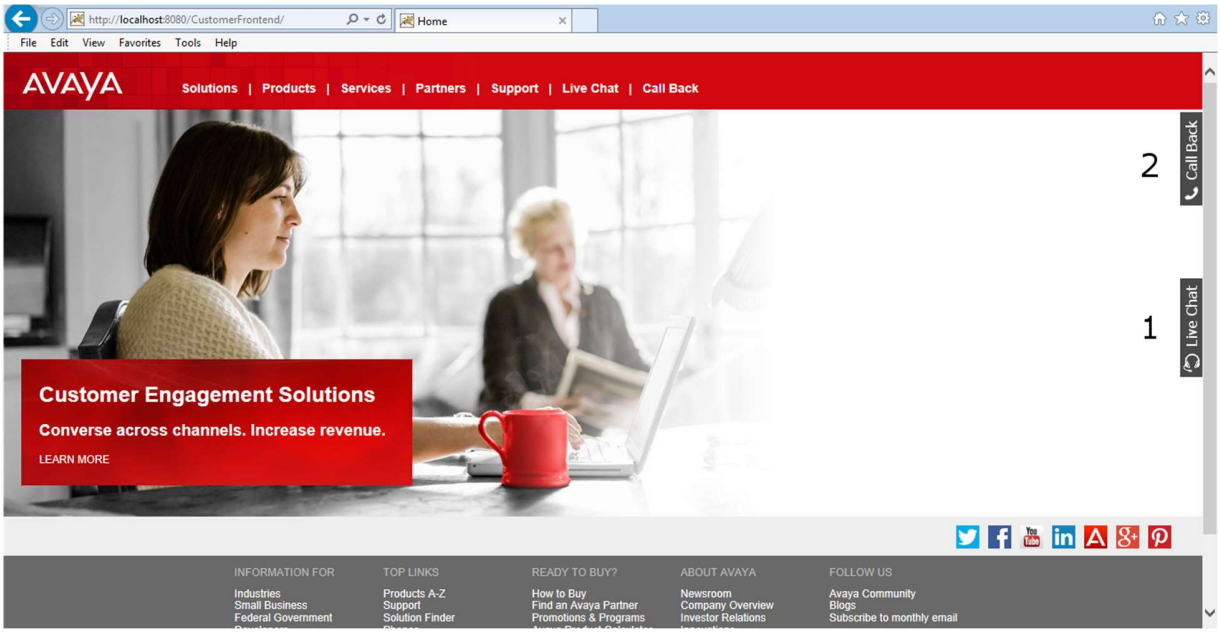


Figure 1 Reference Implementation of a customer-facing web user interface.

Online chat panel

The online chat tab is used to open a chat. When clicked, it slides into the page and shows the panel to log into a chat as illustrated in Figure 2. When the customer has entered their details and clicked the button to enter a chat, this switches to the chat interface as illustrated in Figure 3. The chat login details are gathered from a form enclosed inside a table.

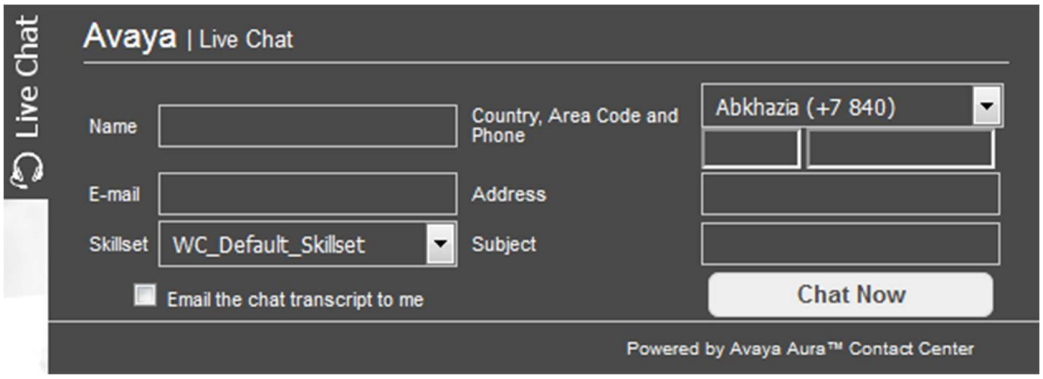


Figure 2 Online chat panel when first opened

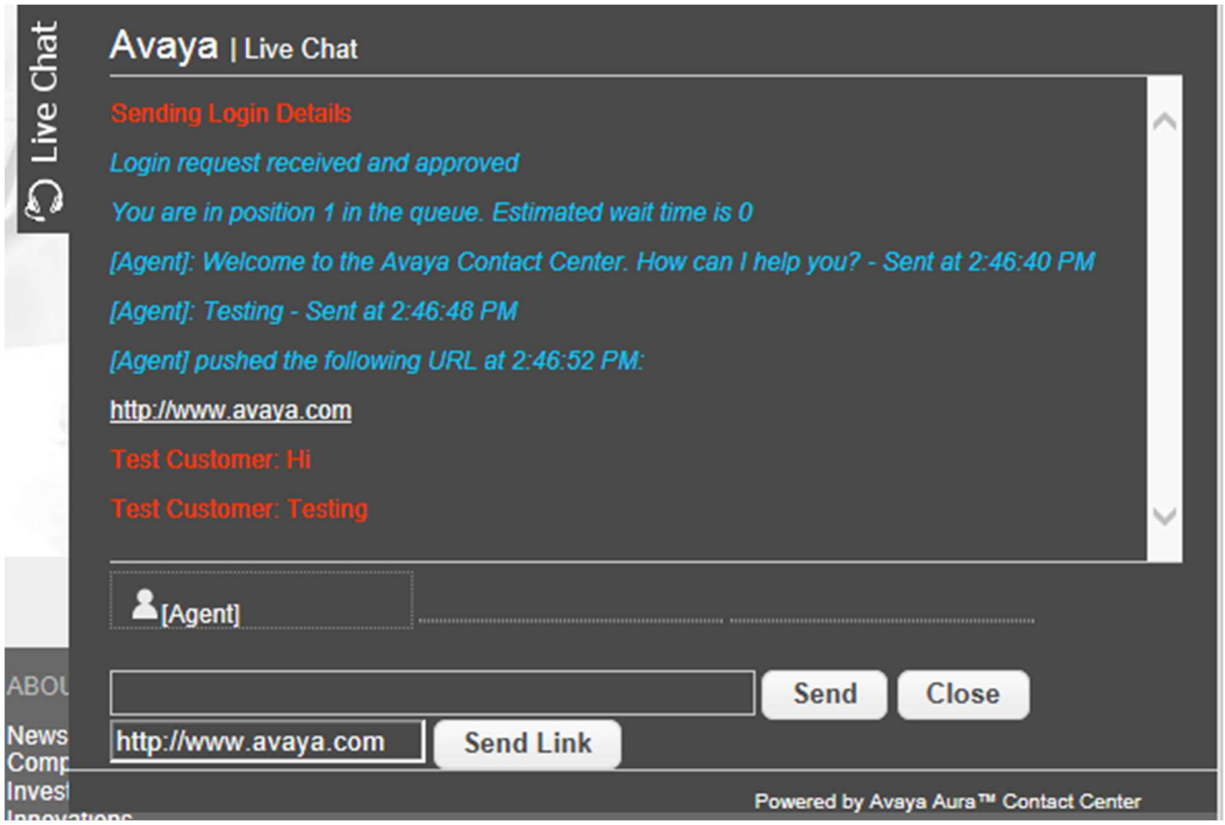


Figure 3 – The chat interface during chat. The [Agent] label here is the default for CCMM.

The chat transcript appears in the center of the chat interface and contains a copy of the principal messages in the chat. Below that is the current users section, which displays a maximum of three agents that are currently in the chat, along with their typing status. At the bottom lie the controls for the chat, comprising a text box for the customer's next message and buttons to send a message or leave the chat, along with a specialized input for sending URLs to an agent.

Callback panel

The callback panel is used by customers to request a scheduled callback without entering a chat, and is illustrated in figure 4.

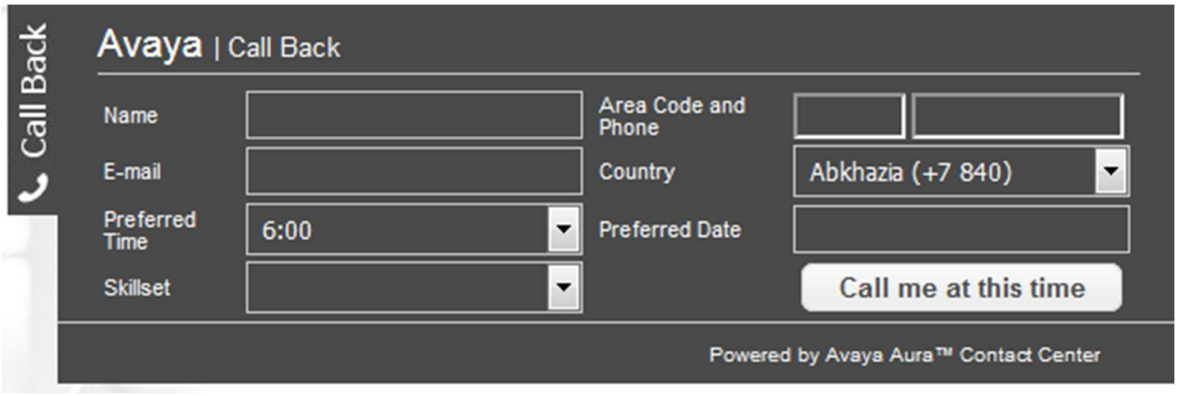


Figure 4 – The request call back panel

The first column contains entries for the customer's name, email address, a time that suits them and a skillset. The second column contains entries for the customer's phone number at the top, country and area code below that, followed by an input for the date and the button to send the callback request. A jQuery datepicker widget is attached to the preferred date entry to aid the customer. For a request to be sent, the customer must enter their phone number and select a date, time, skillset and country.

Requesting a chat

To request a chat using the Reference Web UI, visit the home.html page and click on the chat tab. Enter the user's details, select an available skillset, and click the "Chat Now" button. The customer's position in the queue is calculated after a delay and then sent back to the customer as a specific chat message. When an agent answers, the chat may proceed as normal. While the customer is in the queue and waiting for an agent, any Web-On-Hold URLs and comfort messages that have been configured for the skillset will play automatically. These are configured in the Avaya Aura Contact Center Multimedia Administration.

Sending and Receiving Position in Queue and Estimated Wait Time

The customer's position in queue and estimated wait time are automatically sent after the customer has entered the queue. This is sent at a delay of 3 ½ seconds to allow the service time to calculate the estimated wait time based on the current queue metrics. If an agent answers within this period, the request will not be sent. The results will not be shown if the estimated wait time is less than 1 minute.

Sending a chat message

When an agent has joined the chat, the customer may send chat messages. To do so, click on the text box next to the “Send” button and enter a message. When you are finished, press Enter or click the “Send” button to send the message.

Receiving chat messages

Chat messages from the agent or server are processed automatically according to their type. Normal chat messages, comfort messages or Web-On-Hold messages are automatically written into the chat transcript box. IsTyping messages cause the image of the relevant agent to change color and explain that the agent is typing. NewParticipant and ParticipantLeave notifications add and remove agents to/from the current users section. Page push messages are written into the chat transcript box.

Pushing a URL to the agent

If required, the customer may send a page push message to the agent(s). To do so, enter a URL into the box at the bottom of the tab and click the “Send Link” button. It is not necessary to prepend the URL with http://, e.g. www.avaya.com is acceptable in lieu of http://www.avaya.com.

Reconnecting to the chat

A reconnection mechanism is included for when the customer loses their connection. This will attempt to reopen the WebSocket every three seconds, for a maximum of 150 seconds (2 ½ minutes). If the chat cannot be re-established within this time frame, the connection will be closed.

Retrieving callback skillsets

As with the web chat, the callback skillsets are retrieved from Avaya Aura Contact Center automatically upon page load. A request is sent to Contact Center, which returns an array of JSON objects containing the queues for the outbound skillsets that have been configured. Unlike the web chat, there is no accounting for the number of agents logged into the queue for that skillset – each object simply contains the name of a skillset that has been configured on Avaya Aura Contact Center.

Requesting a scheduled callback

To request a scheduled callback, visit the home.html page and click the callback tab. Enter a valid phone number into the phone box, and enter the area code into the box next to the country menu.

Select a country and a skillset from the drop-down menus. Select a date from the date input, and a time from the relevant drop down menu. Entering a name and email address is optional, but helpful. When these are complete, click the “Call me at this time” button. An alert will appear in the browser window if the callback has succeeded. The time and date entered for the callback are in the customer’s current time zone, which is calculated using the browser’s system time. Country codes for the customer’s phone number are automatically loaded into the country menu on page load. Each code contains a dialing code for a particular country, and the display name of that country in English.

Chapter 9: Setting Up a Development Environment

This section describes the development environment for adopters of Avaya Aura Contact Center Enterprise Web Chat.

Browser Support

The following browsers are supported for testing: Internet Explorer 10 & 11, Microsoft Edge, Firefox 39 or later, Chrome 41 or later, and Apple Safari 6.

Reference Tools

The following tools and integrated design environments were used to develop the SDK. It is not necessary that these be used, but they are included as a reference.

- Eclipse Luna (4.4.2)
- Visual Studio 2013 (.NET 4.0)
- Apache Maven 3.3.1
- Grunt 0.4.5 and various Grunt packages
- Node.js 0.12.0
- Node.js package manager (npm) 2.5.1
- SonarQube/Findbugs for static analysis

Folders and File Breakdown

The following SDK files are provided as part of the Enterprise Web Chat download:

Avaya EWC Custom Desktop.ZIP	The EWC reference Custom Desktop – This contains the source for the Custom Desktop

	application and also a binary version that can run to test with your AACC(ACCS) / EWC solution. Extracted content includes the following: ▪ \bin ▪ \src
Avaya EWC Documentation.ZIP	The Enterprise Web Chat SDK documentation
Avaya EWC Reference Web User Interface.ZIP	The reference Customer User Interface and source code. Extracted content contains: \CustomerFrontendZip

Reference Web User Interface

The '\CustomerFrontendZip' folder contains the following:

src\main\webapp	Source code folder
src\dist\dist	Distribution folder for Grunt
src\test\test	Folder for the frontend unit tests
target	Distribution folder for Maven
.jshintrc	Configures JSHint for linting JavaScript files
bower.json	Configures Bower
Gruntfile.js	Configures Grunt
package.json	Configures Grunt dependencies
pom.xml	Configures Maven
sonar-project.properties	Configures Sonar for local static analysis
BUILD.txt	Contains details about the build in which the folder was generated

The source folder for the Reference Web UI contains the raw source files for the customer reference implementation. It contains the following:

images	Images for the customer reference website
jquery-1.11.3	jQuery library
jquery-plugins	jQuery plugin to slide a tab out from the web page
jquery-ui-1.11.4.custom	jQuery UI library
js	JavaScript files for the library
licenses	Licence files
META-INF	Java META directory, required by Maven
WEB-INF	Contains web.xml file for Tomcat deployments
style.css	stylesheet for the web pages
home.html	sample web page
WebChat.html	sample web page for a pop-up chat window
webChatLogon.html	sample web page for a pop-up chat window

The distribution folder for the Customer Frontend contains some modified files and is laid out as follows:

images	Images for the customer reference website
jquery-1.11.3	jQuery library
jquery-plugins	jQuery plugin to slide a tab out from the web page
jquery-ui-1.11.4.custom	jQuery UI library
js	Modified JavaScript files for the library
META-INF	Java META directory, required by Maven
WEB-INF	Contains web.xml file for Tomcat deployments

style.css	Unmodified version of style.css
style.min.css	Minified version of style.css
home.html	sample web page
WebChat.html	sample web page for a pop-up chat window
webChatLogon.html	sample web page for a pop-up chat login window
home.min.html	minified version of home.html
WebChat.min.html	minified version of WebChat.html
webChatLogon.min.html	minified version of webChatLogon.html

Note: *the un-minified sample pages in the distribution folder have been configured to use a specific JavaScript file, avayaChat.js, instead of the multiple JavaScript files that the source folder contains. The jQuery files and images are not changed.*

Custom Desktop

Once extracted the reference Custom Desktop has the following content:

bin	Contains pre-built binaries for the Custom Desktop
src	Contains the source of the Custom Desktop
BUILD.txt	Contains information about the build which produced the folder.

The *src* folder is further broken down into the following:

.nuget	Configures NuGet
CustomDesktop	Source files for the Custom Desktop window
EncoderDecoder	Contains the source code of and encoders for the JSON messages

EncoderDecoderUnitTest	Contains unit tests for the JSON encoder
Licenses	Licenses for libraries used in the Custom Desktop
CustomDesktop.sln	Visual Studio solution for the project
sonar-project.properties	Configures Sonarqube for static analysis

The *bin* folder contains a sample Custom Desktop. It is broken down into the following files:

AppSettings.xml	Configuration file for URLs and other details
CustomDesktop.exe	Desktop application
CustomDesktop.exe.config	Configuration file for the desktop application
CustomDesktop.exe.log4net	Configuration file for log4net
EncoderDecoder.dll	Library for JSON encoders and decoders
log4net.dll	Logging library
Newtonsoft.Json.dll	Json.NET parsing library
Nortel.CCT.dll	Library for CCT operations
Nortel.CCT.WCF.dll	Library for CCT operations
websocket-sharp.dll	C# WebSocket implementation library

Building the Reference Web User Interface

Building the Reference Web UI requires Grunt and Maven. If Maven is installed, this can be done on the command line. The procedure is as follows:

1. Edit the links.js file (*src/main/webapp* or *src/dist/dist*) so that the URLs point towards your Contact Center Multimedia server or proxy server.
2. Navigate to the CustomerFrontend directory on the command line.
3. Run the command mvn clean install to empty the target directory (where Maven puts its results) and then to build the project.

Note: *Maven has been configured to automatically install Node.js, Grunt and its required dependencies, and then run Grunt. The required versions of the Grunt packages used are stored in the package.json file. The result files from Grunt are under src/dist/dist.*

Building the Custom Desktop

Building the Custom Desktop requires importing it into Visual Studio. The procedure is as follows:

1. Open the CustomerDesktop.sln file in Visual Studio.
2. Change the AppSettings.xml file to match your Contact Center servers.
3. Under the “Build” menu, click “Build Solution”.

Chapter 10: Web User Interface API

This section describes the customer facing API available to adopters of Enterprise Web Chat feature. The operations described in this section are intended for use when building your own customer web site and represents the main entry point for customers to access an AACC agent using EWC. The customer facing API described here is implemented in the Reference Web User Interface which can be used to demonstrate the API and can be used as a guide when building your own web site. For more details on this implementation, please refer to the section: Using the Avaya Contact Center Enterprise Reference Web User Interface.

Notation

- Strings/texts are marked in "`double quotes`".
- Booleans are marked in **`bold and underline`**.
- Numbers are presented as numbers (`0, 1, 234`).
- All timestamps and dates are in UTC.

WebSocket API

General

The purpose of this API is to allow the customer to chat with an agent via a WebSocket. The WebSocket URI should use either of the following formats depending on whether security is enabled or disabled on the CCMM server, and are defined inside the links.js file:

- `Security off: "ws://{CCMM Server hostname/address}:8081/CustomerControllerWeb/chat"`
- `Security on: "wss://{CCMM Server hostname/address}:{SECURE_PORT}/CustomerControllerWeb/chat"`

* Note: For High Availability deployments, the managed CCMM server hostname should be used.

Request Chat

This is used to log a customer into the chat. It is sent as soon as the chat opens. The `guid` and `authenticationKey` variables are used to reconnect to an existing session, in which case they will not be null. **`requestTranscript`** is set by the customer if they wish to receive a transcript of the chat by email, and by default is false unless the customer has entered a valid email address and specified that they wish to receive a transcript. The `customFields` entry handles customized fields such as the customer's address, or other details that you wish to add to the database.

```
{
    "apiVersion" : "1.0",
    "type" : "request",
    "body" :
    {
        "method" : "requestChat",
        "guid" : 654321,
        "authenticationKey" : "bgc714f138g2236823hchc",
        "deviceType"   :   "Mozilla/5.0   (Windows   NT   6.1;   WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.130 Safari/537.36",
        "requestTranscript" : false,
        "intrinsics" : {
            "email" : "test@test.com",
            "name" : "John",
            "country" : "+353",
            "area" : "091",
            "phoneNumber" : "733277",
            "skillset" : "WC_Default_Skillset",
            "customFields" :
                    [
                      {"title": "address", "value" : "Springfield"},
                      {"title": "locale", "value" : "en-GB"}
                    ]
        }
    }
}
```

Request Chat Notification

This informs the customer that they have logged into the chat, and holds messages to play while the customer is waiting for an agent. Web-on-Hold URLs and comfort messages are defined inside CCMM, and are played by the customer's browser. The `authenticationKey` variable will be used to log back into the chat if the customer temporarily loses their connection.

```
{
    "apiVersion" : "1.0",
    "type" : "notification",
    "body" :
    {
        "method" : "requestChat",
        "guid" : 123456,
        "authenticationKey" : "0FaGRg8HQoi8rXAHjhoL7",
```

```
            "webOnHoldComfortGroups" : [
                {
                    "groupName" : "test group 1",
                    "delay" : 45,
                    "numberOfMessages" : 2,
                    "messages" : [
                        {
                            "message" : "first comfort message",
                            "sequence" : 1
                        },
                        {
                            "message" : "first comfort message",
                            "sequence" : 2
                        }
                    ]
                }
            ],
            "webOnHoldURLs" : [
                {
                    "tag" : "Some tag name",
                    "description" : "Test homepage",
                    "holdTime" : 30,
                    "sequence" : 1,
                    "urls" : [
                        {
                            "url" : "http://www.google.ie"
                        },
                        {
                            "url" : "http://www.avaya.com"
                        }
                    ]
                }
            ]

        }
    }
```

Queue Status Request

Used to request the customer's position in the queue, and how long they are likely to wait before an agent replies. It is sent once, after the customer has entered the queue. By default, this is sent at a delay of 3 seconds, and if an agent answers within this delay it is not sent.

```
{
    "apiVersion" : "1.0",
    "type" : "request",
    "body" :
    {
        "method" : "queueStatus"
    }
}
```

Queue Status Notification

Informs the customer of their position in the queue and approximately how long they will be waiting for an agent. The estimated wait time is in seconds, and is converted to minutes upon receipt. If the estimated wait time is less than 1 minute, or their position in the queue is zero, neither value will be shown to the customer.

```
{
    "apiVersion" : "1.0",
    "type" : "notification",
    "body" :
    {
        "method": "queueStatus",
        "positionInQueue": 0,
        "estimatedWaitTime": 20
    }
}
```

New Participant Notification

This is used to alert the customer that an agent has joined the room. The `participants` array is the updated list of agents in the room.

```
{
    "apiVersion" : "1.0",
    "type" : "notification",
    "body" :
    {
        "method" : "newParticipant",
        "agentId" : "534422",
        "displayName" : "John",
        "role"                                                                    :
"active_participant|passive_participant|supervisor_observe",
        "numberOfParticipants": 2,
        "participants" : [
                                {
                                    "id":"123",
                                    "name":"owen",
                                    "type":"passive_participant"
                                },
                                {
                                    "id":"4325",
                                    "name":"bob",
                                    "type":"active_participant"
                                }
                            ]
    }
}
```

Participant Leave Notification

This is sent to alert the customer that an agent has left the room. The `participants` array is the updated list of agents in the room. If the **`endchatFlag`** variable is set to true, it signifies that the active agent has closed the chat, and the conversation will end.

```
{
    "apiVersion" : "1.0",
    "type" : "notification",
    "body" :
    {
        "method" : "participantLeave",
        "agentId" : "54776",
        "endChatFlag" : false,
        "numberOfParticipants": 2,
        "participants" : [
                                {
                                     "id":"123",
                                     "name":"owen",
                                     "type":"passive_participant"
                                 },
                                 {
                                     "id":"4325",
                                     "name":"bob",
                                     "type":"active_participant"
                                 }
                            ]
    }
}
```

Is Typing Request

This is used to let the agent or agents know that the customer is typing. The **`isTyping`** variable must not be null.

```
{
    "apiVersion" : "1.0",
    "type" : "request",
    "body" :
    {
        "method" : "isTyping",
        "isTyping" : true
    }
}
```

Is Typing Notification

This is used to let the customer know that a particular agent is typing. If **`isTyping`** is true, then this agent is typing. By default, if another Is Typing Notification is not received within three seconds, the agent is assumed to no longer be typing.

```
{
    "apiVersion" : "1.0",
    "type" : "notification",
    "body" :
    {
        "method" : "isTyping",
        "agentId" : "43222",
        "displayName" : "John",
        "isTyping" : true
    }
}
```

New Message Request

This is used to send a normal chat message to an agent. The `message` variable must be between 1 and 10000 characters in length, which is verified on the server.

```
{
    "apiVersion" : "1.0",
    "type" : "request",
    "body" :
    {
        "method" : "newMessage",
        "message" : "Test message from customer"
    }
}
```

New Message Notification

The purpose of this is to send a normal chat message from an agent to the customer.

```
{
    "apiVersion" : "1.0",
    "type" : "notification",
    "body" :
    {
        "method" : "newMessage",
        "displayName" : "Frank",
        "message" : "Test message from agent",
        "timestamp" : 1427294421
    }
}
```

Close Conversation Request

This is sent to the server when the customer wishes to end the chat.

```
{
    "apiVersion" : "1.0",
    "type" : "request",
    "body" :
    {
        "method" : "closeConversation"
    }
}
```

Close Conversation Notification

The purpose of this message is to let the customer know the result of closing the chat. If the **`result`** variable is true, the WebSocket closes.

```
{
    "apiVersion" : "1.0",
    "type" : "notification",
    "body" :
    {
        "method" : "closeConversation",
        "result" : true
    }
}
```

New Page Push Request

The purpose of this is to allow the customer to send a URL to an agent. The `pagePushURL` variable must not be blank.

```
{
    "apiVersion" : "1.0",
    "type" : "request",
    "body" :
    {
        "method" : "newPushPageMessage",
        "pagePushURL" : "www.avaya.com"
    }
}
```

New Page Push Notification

This is used to process a URL that was sent by an agent to the customer. The `pagePushDestination` variable defines where this hyperlink opens, and by default is in a new browser tab.

```
{
    "apiVersion" : "1.0",
    "type" : "notification",
    "body" :
    {
        "method" : "newPushPageMessage",
        "displayName" : "Kevin",
        "pagePushURL" : "www.avaya.com",
        "pagePushDestination" : "newTab",
        "timestamp" : 1427294421
    }
}
```

Error Notification

Used to alert the customer that an error has occurred. The error codes that are currently in use, and their corresponding message, are

- 1 (Invalid message);
- 2 (Session key is invalid or expired);
- 500 (Internal Server Error).

```
{
    "apiVersion" : "1.0",
    "type" : "error",
    "body" :
    {
        "method" : "newMessage",
        "code" : 123,
        "errorMessage" : "Some error"
    }
}
```

Acknowledgement Notification

The purpose of this message is to confirm that a message was received, but it is not used in the current implementation.

```
{
    "apiVersion" : "1.0",
    "type" : "acknowledgement",
    "body" :
    {
```

```
            "method" : "newMessage",
            "accepted" : true
        }
    }
```

Ping

Although WebSockets implement their own 'ping/pong' mechanism at the protocol level, these messages are not (as of http://www.w3.org/TR/2011/CR-websockets-20111208/) controllable at the application layer. In order to more effectively manage timeouts at the application layer EWC implements it's own client-driven ping mechanism[1][2].

1. Implementation of this application 'ping/pong' mechanism is entirely orthogonal to a client's use of WebSocket 'ping/pong' packets.
2. An application should not have defined expectations regarding the server's use of WebSocket 'ping/pong' packets.

The EWC ping mechanism maintains the application session on the CCMM server and provides a means for the client application to (configurably) timeout and start a reconnection attempt (per the reconnection mechanism detailed above). Although the specific timeout will be application and environment dependent it is *mandatory* that applications institute this and, regardless of their tolerance for latency, send 'ping' messages at least every 60 seconds.

✱ **Note:** see reference implementation for general recommendation.

As all messages are delivered via TCP there is no possibility that they will be lost, overlapping or out of sequence and, as such, only one 'ping' message will generally be emitted at any time by an instance of the client application. The client application will thus await a 'pong' notification prior to sending further 'ping' messages. An application may emit multiple 'ping' messages and should expect a 'pong' notification in response to each, however, the implications of this are undefined (and thus discouraged).

For simplicity, 'ping' messages should be considered independent of other application messaging in regards a server's liveliness (that is, an application should not delay sending 'ping' messages because other messages are being exchanged and the maximum delay of 60 seconds should be observed).

It is at an application's discretion how (and if) it executes reconnection based on these messages; that is, they may ignore them completely (allowing WebSocket and TCP behaviour to specify) or they may, follow the reference implementation behaviour (e.g. close with application code '4001' to initiate reconnect).

```
{
    "apiVersion" : "1.0",
    "type" : "request",
    "body" : {
        "method" : "ping"
    }
}
```

Pong

The 'pong' notification will be sent by the server, to the client, when a 'ping' message is recieved[1]. Any instance of this notification should be used by the client to reset timers monitoring the liveliness of the chat session and allow it to continue. If an application does not recieve 'pong' notification after a period, it may choose to close the chat session with an application error code (e.g. 4001). An application may also choose to ignore notifications and allow normal WebSocket and TCP behaviour to initiate reconnection events.

Applications *SHOULD NOT* terminate the chat session in a manner that prevents reconnection, simply because of missing 'pong' notifications (e.g. they should not use pre-defined RFC closure codes).

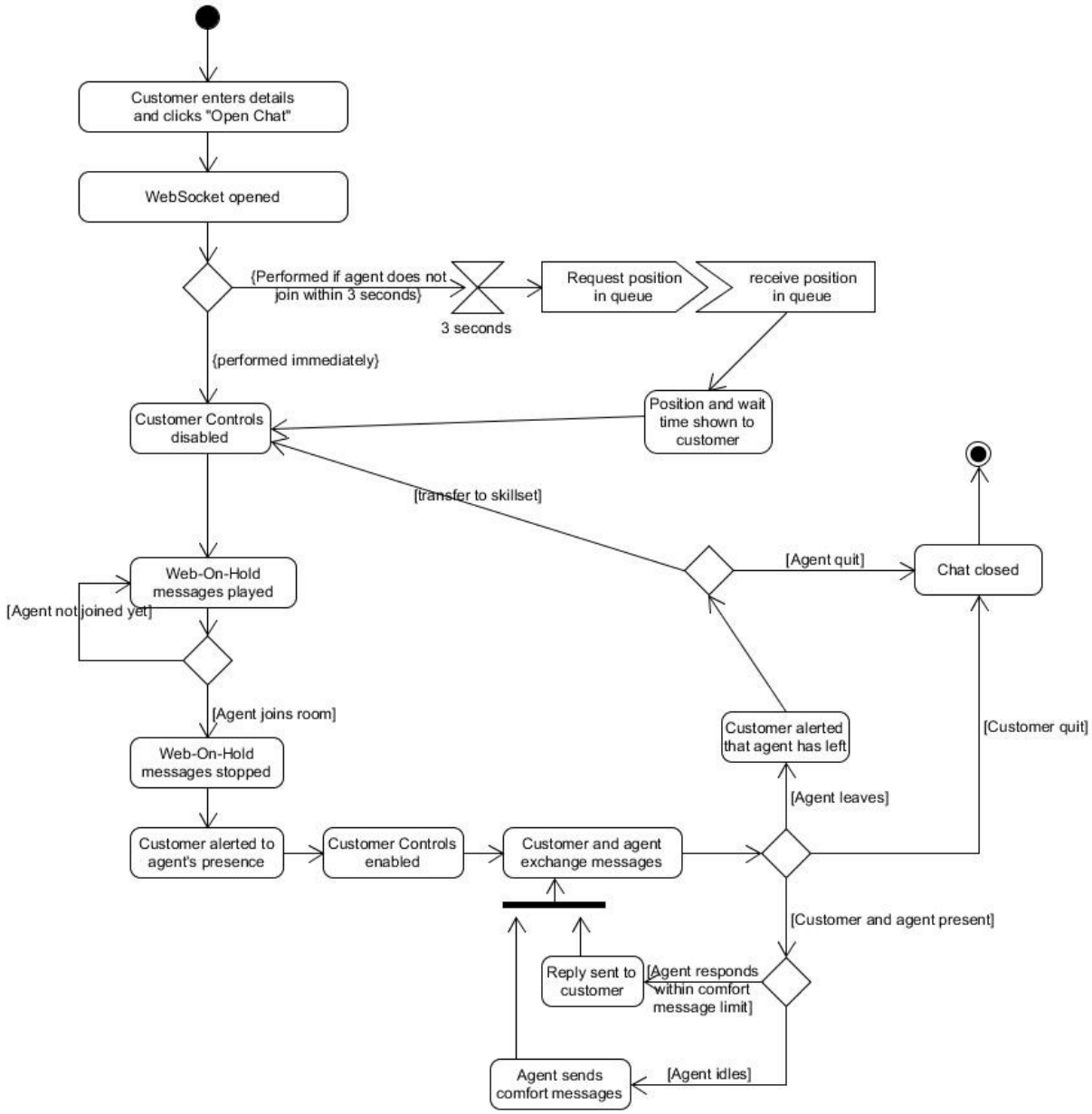
```
{
    "apiVersion" : "1.0",
    "type" : "request",
    "body" : {
        "method" : "pong"
    }
}
```

✱ **Note:** the 'method' is correctly documented above as 'ping' (i.e. 'pong' *IS NOT* a defined 'method').

1. Latency of the server processing should be negligable, however, it will be included in the general considerations for total latency in 'ping/pong' notifications.

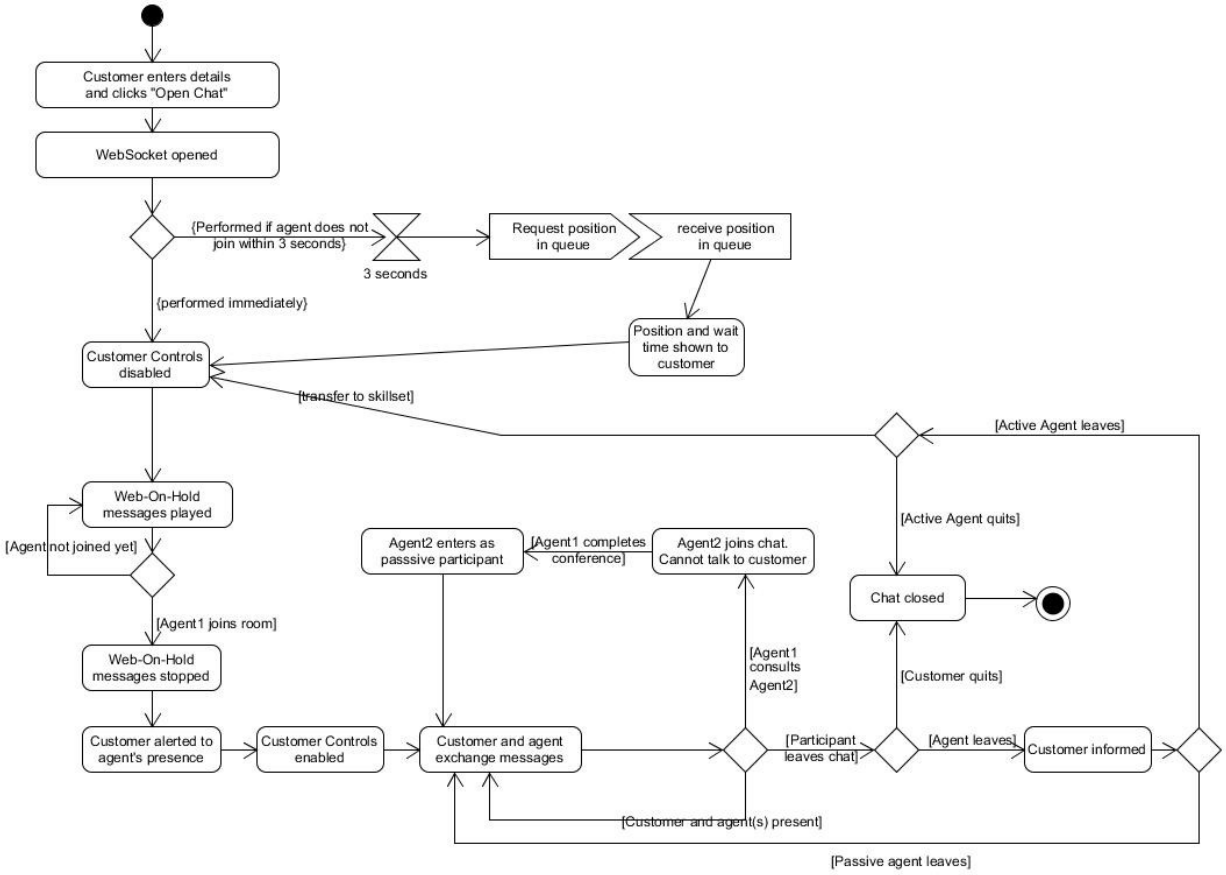
End-to-end Chat Flow

The following flow represents a typical chat flow as implemented using the Reference User Interface



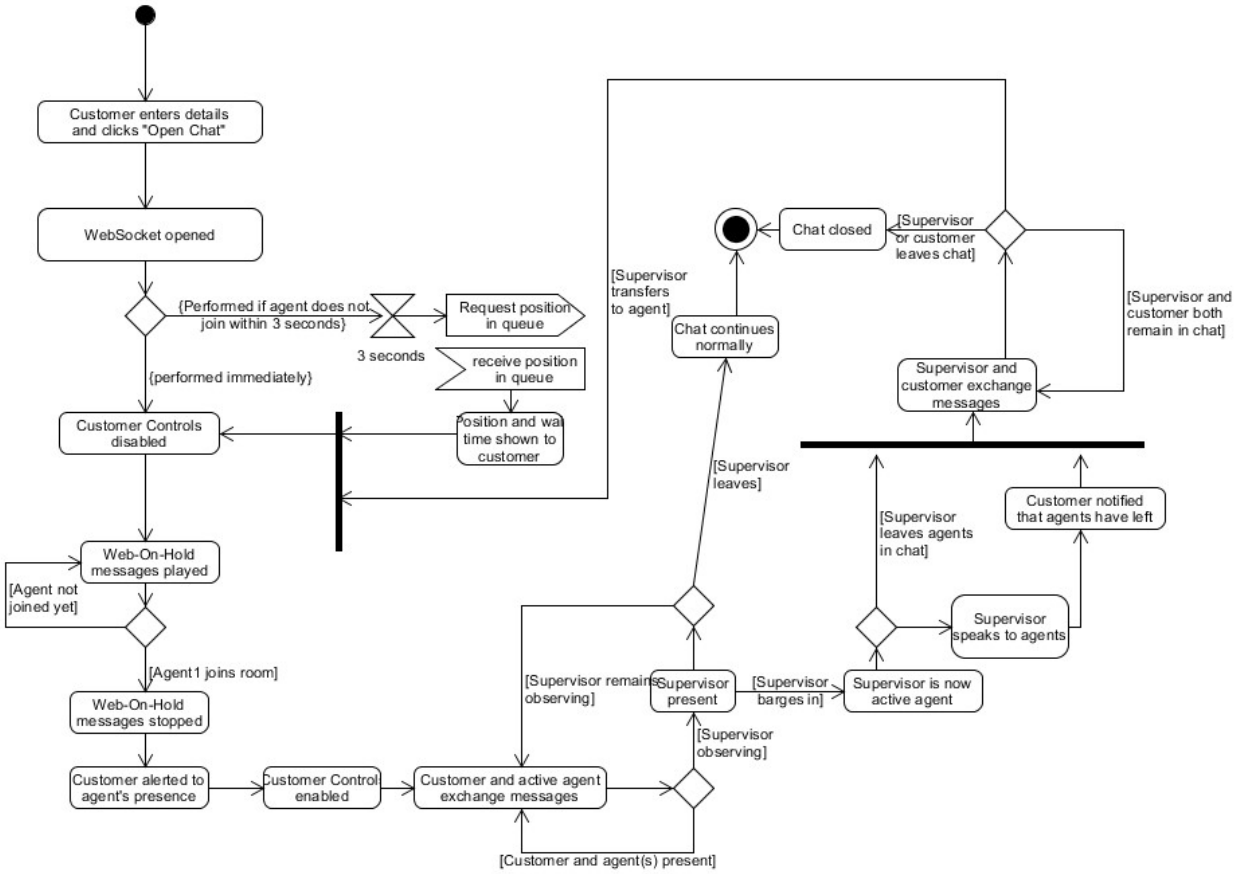
Agent Conferencing Chat Flow

The following flow represents a typical conferencing flow as implemented using the Reference User Interface



Supervisor Observe/Barge-in Flow

The following flow represents a typical Supervisor Observe and Barge-in flow as implemented using the Reference User Interface.



REST API

General

The purpose of this API is to perform various tasks outside of the WebSocket chat. The following URLs should be used for this purpose:

- "http://{multimedia hostname/address}:8081/CustomerControllerWeb/currentqueue" (request queue metrics, unsecure connection)
- "https://{multimediahostname/address}:{SECURE_PORT}/CustomerControllerWeb/currentqueue" (request queue metrics, secure connection)
- "http://{multimedia hostname/address}:8081/CustomerControllerWeb/callback" (scheduled callbacks, unsecure connection)
- "https://{multimedia hostname/address}:{SECURE_PORT}/CustomerControllerWeb/callback" (scheduled callbacks, secure connection)

Callback Skillsets Request

This is used to request the skillsets that have been configured on the server for handling outbound calls. It is sent automatically upon page load.

```
{
    "method" : "getSkillsets"
}
```

Callback Skillsets Response

The purpose of this is to send the configured outbound skillsets to the customer's browser. Upon receipt, they are added to the callback panel.

```
{
    "OB_Default_Skillset" : "Default Skillset",
    "OB_Test" : "Testing"
}
```

Callback Request

Used outside the chat to request a scheduled callback. The date is in UTC.

```
{
    "method" : "callBack",
    "name" : "Adam Smith",
    "phone" : "733277",
    "email" : "test@test.com",
    "countryCode" : "+353",
    "areaCode": "091",
    "skillset" : "OB_Default_Skillset",
    "date" : 1234567890
}
```

Callback Notification

This confirms that the request has been received. The frontend will currently spawn an alert window to inform the user of this.

```
{
    "status" : "Your request has been received"
}
```

Queue Metrics Request

Used outside the chat to check the availability of a queue. If the skillset is an empty string, it is treated as a request for all queues.

```
{
    "skillset" : "WC_Default_Skillset"
}
```

Queue Metrics Notification

The purpose of this message is to return queues for the chat, which will be processed on the frontend. If a queue has sufficient agents and customers, that queue will be available for the customer, and it will be added to the chat login panel. The default criterion for a sufficient number of agents is 1 agent for every 2 customers. The `maxConcurrentChats` value is the maximum number of customers allowed in the queue (i.e. the number of customers that are waiting for an agent).

```
{
    "apiVersion" : "1.0",
    "type" : "notification",
    "body" :
    {
        "method" : "queueMetrics",
        "metrics" : [
```

```
                {
                    "queueName" : "wc_sales",
                    "availableAgentsInQueue" : 20,
                    "customersInQueue" : 30,
                    "maxConcurrentChats" : 50
                },
                {
                    "queueName" : "wc_support",
                    "availableAgentsInQueue" : 3,
                    "customersInQueue" : 40,
                    "maxConcurrentChats" : 50
                },
                {
                    "queueName" : "wc_billing",
                    "availableAgentsInQueue" : 5,
                    "customersInQueue" : 0,
                    "maxConcurrentChats" : 50
                }
            ]
        }
    }
```

Chapter 11: Enterprise Web Chat Agent Desktop API

This section describes the Agent Desktop API available to adopters of Enterprise Web Chat (EWC). The operations described in this section are intended for use when building your own desktop for Contact Center agents & supervisors when handing chat contacts. This API is already implemented by the 'out of the box' Avaya Agent Desktop applications as well as it is used in ContactControlService that is responsible for the Workspaces clients operation. However, this API could be used for a custom desktop development. The agent facing API described here is implemented by the reference Agent Desktop supplied as part of the Enterprise Web Chat SDK, please refer to the Using the Reference Agent Desktop Client section of the documentation for more information. The reference application can be used as a guide to developing a desktop and using the agent facing API.

Notation

- Strings/text is marked in "`double quotes`".
- Booleans are marked in **`bold and underline`**.
- Numbers are presented as numbers (`0, 1, 234`).
- All timestamps and dates are in UTC.
- A messageType of `whisper` means that a customer cannot see this message.

Using the API

The Custom Desktop uses WebSockets to open a connection to the Contact Center Multimedia server, this ensures the connection is maintained throughout the agent session and all EWC API messages are sent using this WebSocket.

If CCMM security is enabled, the custom desktop should connect to:

wss://<multimediaServerAddress>:***<SECURE_PORT>***/AgentControllerWeb/chat

If CCMM security if disabled, the custom desktop should connect to:

ws://<multimediaServerAddress>:***8081***/AgentControllerWeb/chat

Note the different port number requirements and secure versus unsecure WebSocket protocols depending on your security configuration.

More information on how to deploy and configure the custom desktop is contained in the 'Deployment and Configuration' section.

Typical operations implemented in the reference Custom Desktop are:

- The ***Agent Login Request*** is used to authenticate and login an agent with CCMM, the agent is authenticated with CCT using the CCT API. If successful, the custom desktop will receive the ***Agent Login Notification*** which allows Custom Desktop to proceed to the waiting screen.
- When a chat is offered to the agent, accepting that chat uses the ***Agent Join Room Request*** with the contact ID for the customer passed as the GUID and the variable *joinType* to specify if they enter the room as an active agent or to consult or to observe (in the case of a supervisor).
- During a chat with a customer, the ***is Typing Request*** and ***is Typing Notification*** are triggered when the agent begins typing or when the customer does the same from the Customer UI, this shows a little flag to indicate which side is typing.
- When a message is sent by the Custom Desktop, the ***New Message Request*** is used to notify participants. The number of participants that receive the message is based on the value set for *messageType* (normal or whisper). This allows certain parties to be excluded. The ***New Message Notification*** is used to alert the agent that a new message has arrived.
- To begin a consult or transfer, the agent first views the available agents in AACC/ACCS, this is accomplished using the ***Get Destinations Request***, which takes the contact ID as the GUID argument and the agent will receive the ***get Destinations Notification***. The notification will contain a list of agents that are qualified to assist with that customer. The ***Agent Operations Request*** can then be used to initiate the consult, conference or barge using the other agents' id in the *destinationID* variable. When using it for a blind transfer the destination type is set to "queue".
- When an agent joins a room the ***Agent Active Notification*** is sent to everyone currently in the room, e.g. the customer or the agent who owns the room. The *usertype* can be set to allow the joining agent to join as an active, passive OR a supervisor observing the chat. From here the agent who owns the chat can promote a passive user to the conference owner using the ***Agent Change Conference Owner Request***. The passive agents can be removed from the chat using the A***gent Drop Passive Participant Request***. To finish the conference, the conference owner calls the ***Agent Conference Close Request*** which closes the chat room and sends passive agents the ***Agent Exit Room Notification***.
- A supervisor using Custom Desktop has access to the same operations as a standard agent and also the ability to observe a standard agent's performance throughout a web chat session. Once the supervisor has logged in the same way as a standard agent, the Custom Desktop will send the ***Agent Data Request*** which returns with a notification showing the list of agents assigned to the supervisor. This list contains the intrinsic data for each active chat as well as agent status (ready/not ready) and allows the supervisor to observe a chat using the ***Agent Operations Request***.
- Once observing a chat, a supervisor can barge in using another ***Agent Operations Request*** and become the new owner of the room, making the standard agent(s) passive. After this, the

supervisor is treated as a standard agent and has the same ability to drop the passive agents or to close the conference altogether. All comfort messages and contact messages will come from the supervisor after a barge as the agent is blocked from sending messages to the customer. The agent can only communicate with the supervisor using the whisper messages.

Screen-pop in Enterprise Web Chat:

In AACC 7.x with EWC, screen pops operate in a different manner to AACC 6.4. The intrinsic value that will be passed as a parameter to the screen pop application is the contact ID of the chat contact. The Custom Desktop does not make use of any screen-pop intrinsic settings that are set in the Multimedia Admin application (launched via CCMA).

When the agent enters a new chat room, a ***Screen Pops Request*** is sent to the server, which returns the screen pop applications or website URLs that have been configured on the server. The %VALUE% returned is the contact ID of the customer chat.

Flow diagrams providing a graphical view of agent conference scenarios and also supervisor observe and barge in are included at the end of the Agent API section.

Agent API

Agent Login Request

Sent when an agent logs into the Custom Desktop.

```
{
   "apiVersion" : "1.0",
   "type" : "request",
   "body" :
   {
      "method" : "agentLogin",
      "agentId" : "12345",
      "password" : "12345",
      "reconnect" : false,
      "clientType" : "custom"
   }
}
```

Agent Login Notification

Used to inform the agent that they have successfully logged into the desktop.

```
{
   "apiVersion" : "1.0",
   "type" : "notification",
   "body" :
   {
      "method" : "agentLogin",
      "loginSuccess" : true
   }
}
```

Agent Logout Request

Sent when the agent is logging out of the Custom Desktop.

```
{
   "apiVersion" : "1.0",
   "type" : "request",
   "body" :
   {
      "method" : "agentLogout"
   }
}
```

Agent Join Room Request

Sent when an agent is joining a room. The type is either active (when they are the primary agent), consult (used when another agent has requested their assistance) or observe (used by supervisors only).

```
{
   "apiVersion" : "1.0",
   "type" : "request",
   "body" :
   {
      "method" : "agentJoinRoom",
      "guid" : 123456,
      "joinType" : "active|consult|observe"
   }
}
```

Agent Join Room Notification

Inform agent of the details of the room.

```
{
   "apiVersion" : "1.0",
   "type" : "notification",
   "body" :
   {
      "method" : "agentJoinRoom",
      "guid" : 123456,
      "deviceType" : "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/43.0.2357.130 Safari/537.36",
      "previousTranscripts" : 12,
      "conversationTranscript" :
      {
         "messages": [
            {
               "type": "message",
               "sender": {
                  "type": "customer"
               },
               "displayName": "Customer",
               "timestamp": 1427294421,
               "message": "first message",
               "messageType": "normal"
            },
            {
               "type": "message",
               "sender": {
                  "type": "agent",
                  "agentId": "12345"
               },
               "displayName": "Frank",
               "timestamp": 1427294421,
               "message": "second message",
               "messageType": "whisper"
            },
            {
               "type": "pagePush",
               "sender": {
                  "type": "agent",
                  "agentId": "12345"
               },
               "url": "http://example.com",
               "displayName": "Jimmy",
               "timestamp": 1427294421,
               "destination": "newTab"
            }
         ]
      },
      "currentParticipantList" :
      {
```

```
    "participants" : [
      {
        "agentId" : "00123",
        "userType" : "active_participant"
      },
      {
        "agentId" : "123",
        "userType" : "passive_participant"
      },
      {
        "agentId" : "01234",
        "userType" : "supervisor_observe"
      }
    ]
},
"timerList" :
{
    "keepAliveTime_inSeconds" : 30,
    "desirableResponseToCust_inSeconds" : 30,
    "desirableResponseToAgent_inSeconds" : 60,
    "consultRequestTimeout_inSeconds" : 30,
    "forceIdleCustomerCheck" : false,
    "forceIdleCustomerCheckTimeout_inSeconds" : 180,
    "saveTimestampOnMessages" : true,
    "saveChatHistory" : true,
    "concurrentChatsLimitPerCustomer" : 3,
    "requestedCallBacksLimitPerCustomer" : 3,
    "emailChatLogToCustomer" : true
},
"resources" :
{
    "agentLabel" : "first name last name",
    "customerLabel" : "customer"
},
"frequentlyUsedPhraseList" : [
    {
      "skillset" : "all",
      "phrases" : [
        {
          "name" : "phrase1",
          "phraseText" : "Can I help you with anything else"
        },
        {
          "name" : "phrase2",
          "phraseText" : "how can I help"
        },
        {
          "name" : "phrase3",
          "phraseText" : "some phrase"
        },
```

```
            {
               "name" : "phrase4",
               "phraseText" : "welcome to chat"
            }
         ]
      },
      {
         "skillset" : "wc_sales",
         "phrases" : [
            {
               "name" : "phrase1",
               "phraseText" : "Can I help you with anything else"
            },
            {
               "name" : "phrase2",
               "phraseText" : "how can I help"
            }
         ]
      }
   ],
   "pagePushURLList" : [
      {
         "skillset" : "all",
         "urls" : [
            {
               "url" : "www.google.com",
               "description" : "Google search"
            },
            {
               "url" : "www.avaya.com",
               "description" : "Avaya Homepage"
            },
            {
               "url" : "www.test.com",
               "description" : "test page"
            },
            {
               "url" : "www.avaya.com",
               "description" : "Avaya Homepage"
            }
         ]
      },
      {
         "skillset" : "wc_sales",
         "urls" : [
            {
               "url" : "www.google.com",
               "description" : "Google search"
            },
            {
```

```
                "url" : "www.avaya.com",
                "description" : "Avaya Homepage"
            }
        ]
    }
  ]
 }
}
```

Agent Status Request

Sent when an agent is rejoining a room.

```
{
   "apiVersion" : "1.0",
   "type" : "request",
   "body" :
   {
      "method" : "agentReJoinRoom",
      "guid" : 123456
   }
}
```

Room Status Request

Sent when an agent is rejoining a room. It’s necessary to send it when swithover is happened.

```
{
   "apiVersion" : "1.0",
   "type" : "request",
   "body" :
   {
      "method" : "RoomStatus",
      "guid" : 123456
   }
}
```

Room Status Notification

Returns a boolean value about the agent’s presence in the room

```
{
```

```
    "apiVersion" : "1.0",
    "type" : "notification",
    "body" :
    {
        "method" : "roomStatus",
        "guid" : 123456,
        "isAgentInRoom" : false
    }
}
```

Screen Pops Request

Sent to retrieve the configured screen pop windows.

```
{
   "apiVersion" : "1.0",
   "type" : "request",
   "body" :
   {
      "method" : "screenPops"
   }
}
```

Screen Pops Notification

Returns the list of configured screen pop windows. The **forceLaunch** variable defines whether or not this will be automatically opened.

```
{
    "apiVersion" : "1.0",
    "type" : "notification",
    "body" :
    {
        "method" : "screenPops",
        "screenPops" : [
           {
              "name" : "Google Search",
              "path" : "http://www.google.com/search?q=%VALUE%",
              "forceLaunch" : false
           },
           {
              "name" : "Notepad",
              "path" : "notepad %VALUE%",
              "forceLaunch" : false
           }
        ]
    }
}
```

Is Typing Request

Used to let another agent or a customer know that this agent is typing.

```
{
   "apiVersion" : "1.0",
   "type" : "request",
   "body" :
   {
      "method" : "isTyping",
      "guid" : 123456,
      "messageType" : "normal|whisper",
      "isTyping" : true
   }
}
```

Is Typing Notification

Let the agent know that another user is typing.

```
{
   "apiVersion" : "1.0",
   "type" : "notification",
   "body" :
   {
      "method" : "isTyping",
      "sender": {
         "type": "agent",
         "agentId": "1234"
      },
      "guid" : 123456,
      "displayName": "Agent Smith",
      "timestamp": 1427294421,
      "isTyping": true,
      "messageType": "normal|whisper"
   }
}
```

New Message Request

Send a new chat message to all participants in a chat, or just to other agents.

```
{
   "apiVersion" : "1.0",
```

```
   "type" : "request",
   "body" :
   {
      "method" : "newMessage",
      "guid" : 123456,
      "message" : "Test message from customer",
      "messageType" : "normal|whisper"
   }
}
```

New Message Notification

Used to alert the agent that a new chat message has arrived. Comfort messages are messages sent from the server to the customer while the agent is occupied in another chat. The welcome message is played once, when the agent enters the chat.

```
{
   "apiVersion" : "1.0",
   "type" : "notification",
   "body" :
   {
      "method" : "newMessage",
      "sender": {
         "type": "agent",
         "agentId": "1234"
      },
      "guid" : 123456,
      "displayName": "Agent Smith",
      "timestamp": 1427294421,
      "message" : "Test message from agent",
      "messageType" : "normal|comfortMessage|welcomeMessage|whisper"
   }
}
```

Agent Quit Chat Request

Sent when an agent wishes to leave a chat.

```
{
   "apiVersion" : "1.0",
   "type" : "request",
   "body" :
   {
      "method" : "agentQuitChat",
      "guid" : 123456,
   }
```

```
}
```

Agent Quit Chat Notification

Received from the server when the agent is leaving the chat. It contains a list of reasons for why the chat is being closed.

```
{
   "apiVersion" : "1.0",
   "type" : "notification",
   "body" :
   {
      "method" : "agentQuitChat",
      "guid" : 123456,
      "closedReasonCodes" :
      {
         "defaultReasonCode" : 2,
         "codes" : [
            {
               "id" : 1,
               "closedReason" : "problem solved"
            },
            {
               "id" : 2,
               "closedReason" : "could not fix"
            }
         ]
      }
   }
}
```

Customer Disconnect Notification

Received from the server when the customer has left the chat.

```
{
   "apiVersion" : "1.0",
   "type" : "notification",
   "body" :
   {
      "method" : "customerDisconnect",
      "guid" : 332233,
      "timestamp" : 1427294421,
      "closedReasonCodes" :
      {
         "defaultReasonCode" : 2,
         "codes" : [
```

```
            {
               "id" : 1,
               "closedReason" : "problem solved"
            },
            {
               "id" : 2,
               "closedReason" : "could not fix"
            }
         ]
      }
   }
}
```

Agent Set Closed Reason Request

Sent when the agent is setting the closed reason code for a chat that has closed. If **sendTranscript** is true, the transcript of the chat will be emailed to the customer.

```
{
   "apiVersion" : "1.0",
   "type" : "request",
   "body" :
   {
      "method" : "agentSetClosedReason",
      "guid" : 112233,
      "closedReasonCode" : 44,
      "agentNote" : "Customer wasn't happy",
      "sendTranscript" : false
   }
}
```

Agent Set Closed Reason Notification

Used to inform the agent that their closed reason has been accepted or rejected.

```
{
   "apiVersion" : "1.0",
   "type" : "notification",
   "body" :
   {
      "method" : "agentSetClosedReason",
      "guid" : 112233,
      "accepted" : true
   }
}
```

Read Customer History Request

Sent when an agent requests a customer's previous history. The amount variable defines how many previous interactions should be returned.

```
{
   "apiVersion" : "1.0",
   "type" : "request",
   "body" :
   {
      "method" : "customerHistory",
      "guid" : 443344,
      "amount" : 5
   }
}
```

Read Customer History Notification

Holds the results of the customer history request.

```
{
   "apiVersion" : "1.0",
   "type" : "notification",
   "body" :
   {
      "method" : "customerHistory",
      "guid" : 443344,
      "history" : [
         {
            "startTime" : 45638485689,
            "transcript" : "hello there",
            "source": "Text Chat History",
            "contactType": "Web_Communications"
         },
         {
            "startTime" : 54657688968,
            "transcript" : "hello there, hi customer",
            "source": "Text Chat History With Whispers",
            "contactType": "Web_Communications"
         },
         {
            "startTime" : 23456456678,
            "transcript" : "help me",
            "source": null,
            "contactType": "EMail"
         }
      ]
   }
}
```

New Page Push Request

Used when the agent selects a link to send to the customer.

```
{
   "apiVersion" : "1.0",
   "type" : "request",
   "body" :
   {
      "method" : "pagePush",
      "guid" : 123456,
      "url" : "www.avaya.com",
      "destination" : "newBrowserTab"
   }
}
```

New Page Push Notification

Used to alert the agent that another user has sent them a link. If it comes from a customer, the destination field is not present.

```
{
   "apiVersion" : "1.0",
   "type" : "notification",
   "body" :
   {
      "method" : "pagePush",
      "guid" : 123456,
      "sender": {
         "type": "agent",
         "agentId": "1234"
      },
      "displayName" : "Frank",
      "url" : "www.avaya.com",
      "destination" : "newTab",
      "timestamp": 1434019311123
   }
}
```

Get Destinations Request

Sent when the agent wishes to transfer the chat to another skillset.

```
{
```

```
   "apiVersion" : "1.0",
   "type" : "request",
   "body" :
   {
      "method" : "getDestinations",
      "guid" : 62445
   }
}
```

Get Destinations Notification

Sent back to the agent, and contains the list of queues into which the chat can be transferred. If the **allowTransferToSkillset** and **allowTransferToAgent** variables are set to false, then no transfers are allowed.

```
{
   "apiVersion" : "1.0",
   "type" : "notification",
   "body" :
   {
      "method" : "getDestinations",
      "guid" : 62445,
      "allowTransferToSkillset" : true,
      "allowTransferToAgent" : true,
      "queues" : [
         {
            "queueName" : "WC_Default_Skillset",
            "queueId" : 13,
            "queueCCMSId" : 9998,
            "routepoint" : "WebChatRP",
            "agents" : [
               {
                  "agentId" : "111",
                  "displayName" : "John"
               },
               {
                  "agentId" : "222",
                  "displayName" : "Mike"
               }
            ]
         },
         {
            "queueName" : "WC_New_Skillset",
            "queueId" : 14,
            "queueCCMSId" : 9999,
            "routepoint" : "WebChatRP",
            "agents" : [
               {
                  "agentId" : "333",
```

```
                "displayName" : "Bob"
            },
            {
                "agentId" : "444",
                "displayName" : "Kevin"
            }
        ]
    }
  ]
 }
}
```

Agent Operation Request

Sent when the agent wishes to perform a particular action.

```
{
  "apiVersion" : "1.0",
  "type" : "request",
  "body" :
  {
    "method" : "operationRequest",
    "guid" : 12345,
    "destinationType" : "queue|agent|consult|conference|barge",
    "destinationId" : "654321"
  }
}
```

Agent Operation Notification

Alerts the agent about the result of the previous request.

```
{
  "apiVersion" : "1.0",
  "type" : "notification",
  "body" :
  {
    "method" : "operationRequest",
    "guid" : 12345,
    "success" : true
  }
}
```

Agent Active Notification

Notifies the agent that another agent has joined the room.

```
{
   "apiVersion" : "1.0",
   "type" : "notification",
   "body" :
   {
      "method" : "activeAgentNotification",
      "guid" : 12345,
      "agentId" : "54321",
      "userType" : "active_participant|passive_participant|supervisor_observe"
   }
}
```

Agent Change Conference Owner Request

Sent when the active agent in a chat wishes to promote another agent to the active agent.

```
{
   "apiVersion" : "1.0",
   "type" : "request",
   "body" :
   {
      "method" : "changeConferenceOwner",
      "guid" : 12345
   }
}
```

Agent Change Conference Owner Notification

Informs agent of the new conference owner, i.e. who is the active agent.

```
{
   "apiVersion" : "1.0",
   "type" : "notification",
   "body" :
   {
      "method" : "changeConferenceOwner",
      "guid" : 12345,
      "newOwner" : "123456"
   }
}
```

Agent Leave Room Request

Used when the passive agent in a conference or a consultation wishes to leave the conference.

```
{
   "apiVersion" : "1.0",
   "type" : "request",
   "body" :
   {
      "method" : "leaveRoom",
      "guid" : 112233
   }
}
```

Agent Drop Passive Participant Request

Sent when the active agent in a conference or a consultation wishes to drop the passive agent.

```
{
   "apiVersion" : "1.0",
   "type" : "request",
   "body" :
   {
      "method" : "dropPassiveParticipant",
      "guid" : 112233
   }
}
```

Agent Conference Close Request

Allow the active agent in a conference to close the conference.

```
{
   "apiVersion" : "1.0",
   "type" : "request",
   "body" :
   {
      "method" : "conferenceClose",
      "guid" : 112233
   }
}
```

Agent Exit Room Notification

Sent to the passive agent when they are removed from the conference or consultation.

```
{
   "apiVersion" : "1.0",
```

```
   "type" : "notification",
   "body" :
   {
      "method" : "exitRoom",
      "guid" : 112233,
      "agentId" : "12345",
      "scenario" : "dropped by conference owner"
   }
}
```

Agent Data Request

Sent by supervisor agents to check the status of the agents they are supervising.

```
{
   "apiVersion" : "1.0",
   "type" : "request",
   "body" :
   {
      "method" : "agentData"
   }
}
```

Agent Data Notification

Sent to supervisor agents with the status of the agents that they are supervising.

```
{
   "apiVersion" : "1.0",
   "type" : "notification",
   "body" :
   {
      "method" : "agentData",
      "thresholds" :
      {
         "conversationLength_inSeconds" : {"threshold": 20, "priority": 1},
         "lastMessageOut_inSeconds" :  {"threshold": 20, "priority": 3},
         "lastMessageIn_inSeconds" :  {"threshold": 20, "priority": 2},
         "numberOfAgentMessages" :  {"threshold": 5, "priority": 4},
         "numberOfUnansweredCustomerMessages" : {"threshold": 1, "priority": 5}
      },
      "agents" : [
         {
            "agentId" : "112233",
            "displayName" : "John Smith",
            "conversations" : [
               {
```

```
            "guid" : 111222,
            "conversationLength_inSeconds" : 30,
            "lastMessageOut_inSeconds" : 30,
            "lastMessageIn_inSeconds" : 30,
            "numberOfAgentMessages" : 10,
            "numberOfUnansweredCustomerMessages" : 3
         },
         {
            "guid" : 555666,
            "conversationLength_inSeconds" : 20,
            "lastMessageOut_inSeconds" : 20,
            "lastMessageIn_inSeconds" : 20,
            "numberOfAgentMessages" : 5,
            "numberOfUnansweredCustomerMessages" : 1
         }
      ]
   },
   {
      "agentId" : "553311",
      "displayName" : "David Jones",
      "conversations" : [
         {
            "guid" : 9990,
            "conversationLength_inSeconds" : 30,
            "lastMessageOut_inSeconds" : 30,
            "lastMessageIn_inSeconds" : 30,
            "numberOfAgentMessages" : 10,
            "numberOfUnansweredCustomerMessages" : 3
         },
         {
            "guid" : 8880,
            "conversationLength_inSeconds" : 20,
            "lastMessageOut_inSeconds" : 20,
            "lastMessageIn_inSeconds" : 20,
            "numberOfAgentMessages" : 5,
            "numberOfUnansweredCustomerMessages" : 1
         }
      ]
   },
   {
      "agentId" : "553311",
      "displayName" : "David Jones",
      "conversations" : [
         {
            "guid" : 543,
            "conversationLength_inSeconds" : 30,
            "lastMessageOut_inSeconds" : 30,
            "lastMessageIn_inSeconds" : 30,
            "numberOfAgentMessages" : 10,
            "numberOfUnansweredCustomerMessages" : 3
```

```
            }
         ]
      }
   ]
  }
}
```

Supervisor Barge Notification

Sent to all users when a supervisor barges into the chat, except for the supervisor. The supervisor becomes the active participant, and all other agents are put into a consult state.

```
{
   "apiVersion" : "1.0",
   "type" : "notification",
   "body" :
   {
      "method" : "supervisorBarge",
      "guid" : 12345,
      "supervisorId" : "20000"
   }
}
```

Customer Connection Status Notification

Sent when the customer has lost their connection.

```
{
   "apiVersion" : "1.0",
   "type" : "notification",
   "body" :
   {
      "method" : "customerConnectionStatus",
      "guid" : 123456,
      "isConnected" : false,
      "timestamp": 1427294421
   }
}
```

Acknowledgement Notification

Sent when the server is acknowledging that a message was received.

```
{
   "apiVersion" : "1.0",
```

```
   "type" : "acknowledgement",
   "body" :
   {
      "method" : "newMessage",
      "guid": 123,
      "accepted" : true
   }
}
```

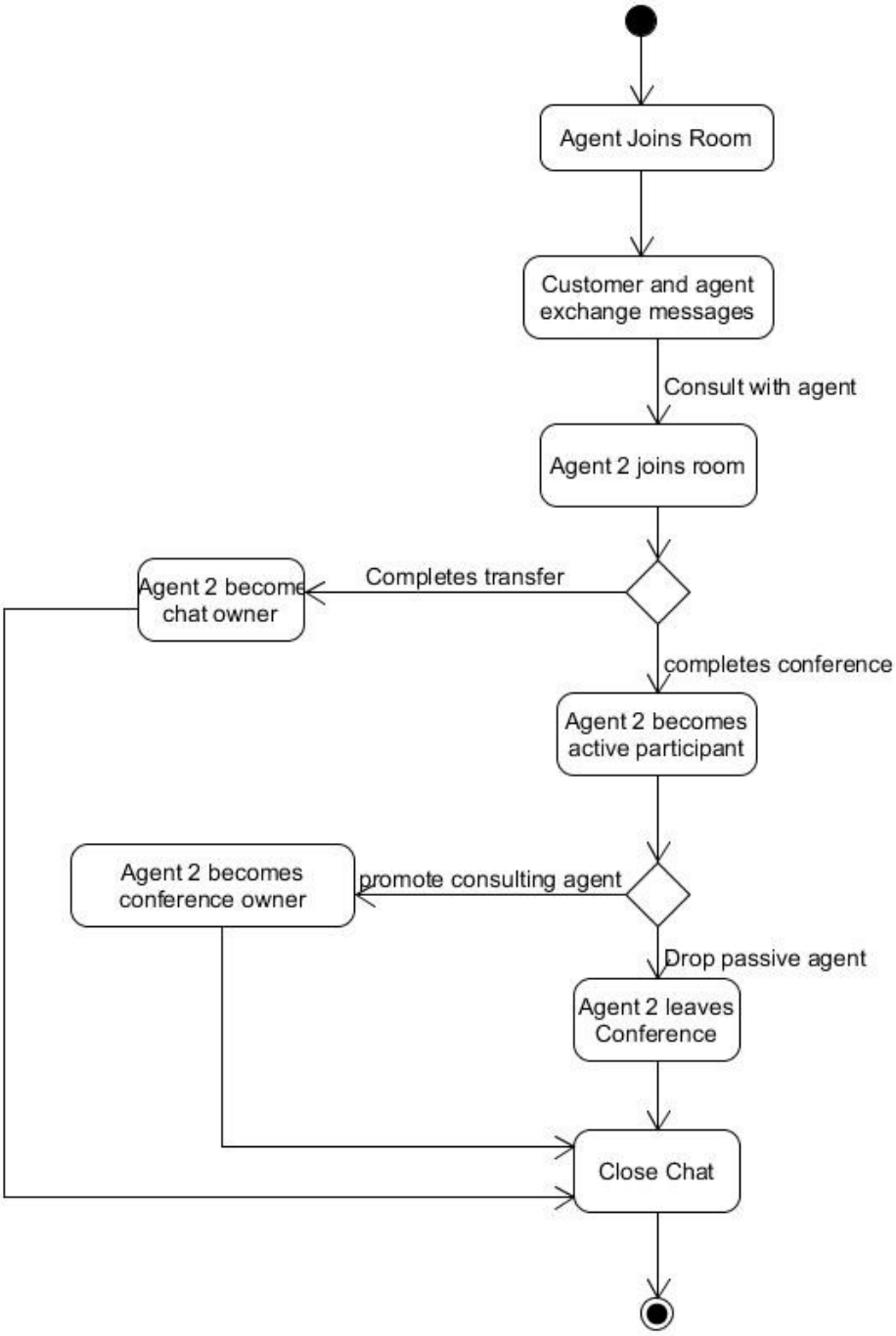
Error Notification

Sent to the agent when an error occurs.

```
{
   "apiVersion" : "1.0",
   "type" : "error",
   "body" :
   {
      "method" : "newMessage",
      "code" : 123,
      "errorMessage" : "Some error"
   }
}
```

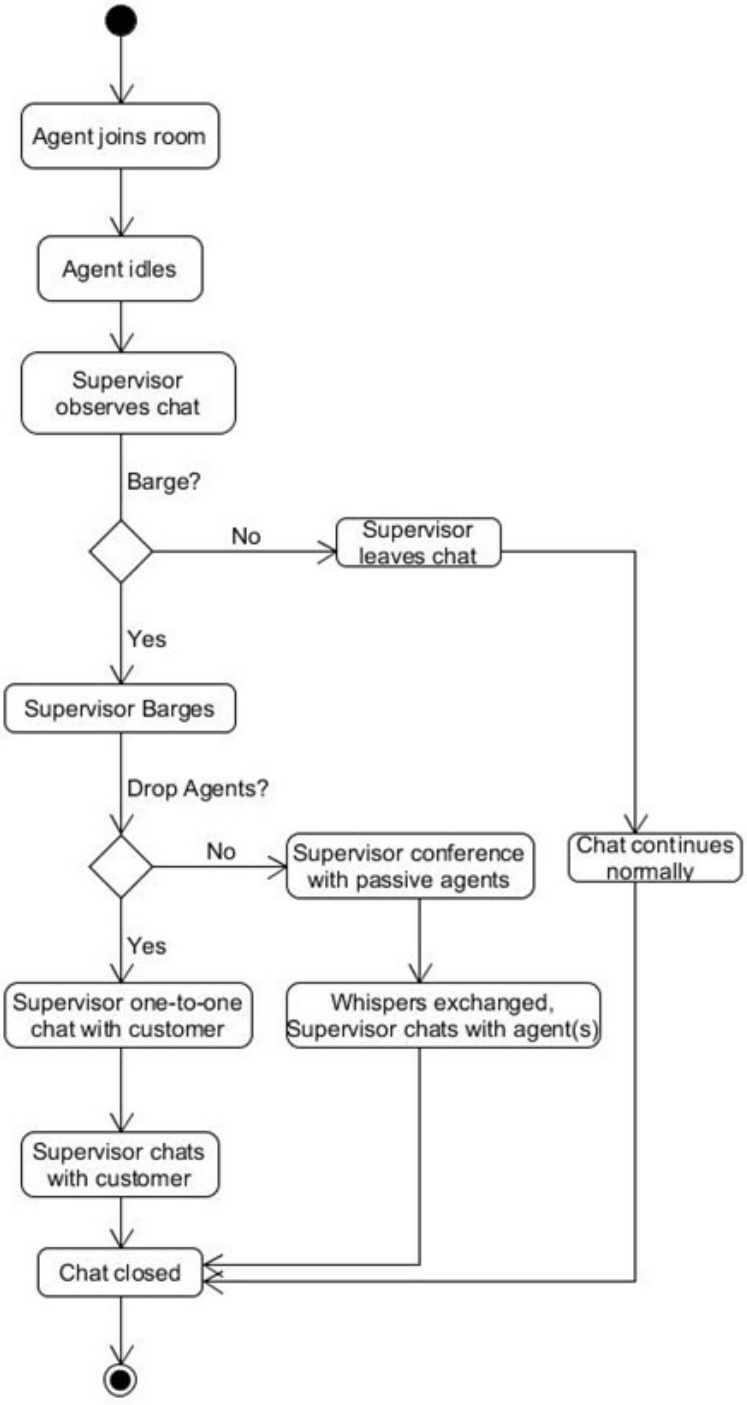
Agent Conference

The following provides a typical conference flow when using the Agent API as implemented using the Custom Desktop.



Supervisor Barge

The following provides a typical supervisor observe and barge-in flow when using the Agent API as implemented using the Custom Desktop.



Chapter 12: Troubleshooting

This section describes troubleshooting and deployment tips for Enterprise Web Chat 7.1.

Troubleshooting tips

This section describes how to troubleshoot Enterprise Web Chat. You can troubleshoot problems better by planning your solution deployment in advance and having up-to-date information if troubleshooting is required. For example, ensure that you know the correct user accounts, passwords, solution resources, and network details. Plan ahead and ensure that you are using the correct information to install and commission the Contact Center solution. Review all applicable customer documentation in advance of any deployment.

The main stages of the troubleshooting process are:

1. *Identify the problem*. Is the problem intermittent or reproducible? Has the problem always existed, or was it introduced after a recent configuration change?
2. *Determine the cause of the problem*. Determining the most likely cause is a process of elimination - eliminating potential causes of a problem. Begin your investigation by double-checking the solution configuration data and reviewing against the published customer documentation.
3. *Solve the problem*. Identify and test the solution to the problem. Intermittent problems require additional and prolonged soak testing.

If you are not able to solve the problem, collect all the relevant information and have it available before contacting Avaya Technical Support.

Important Log files

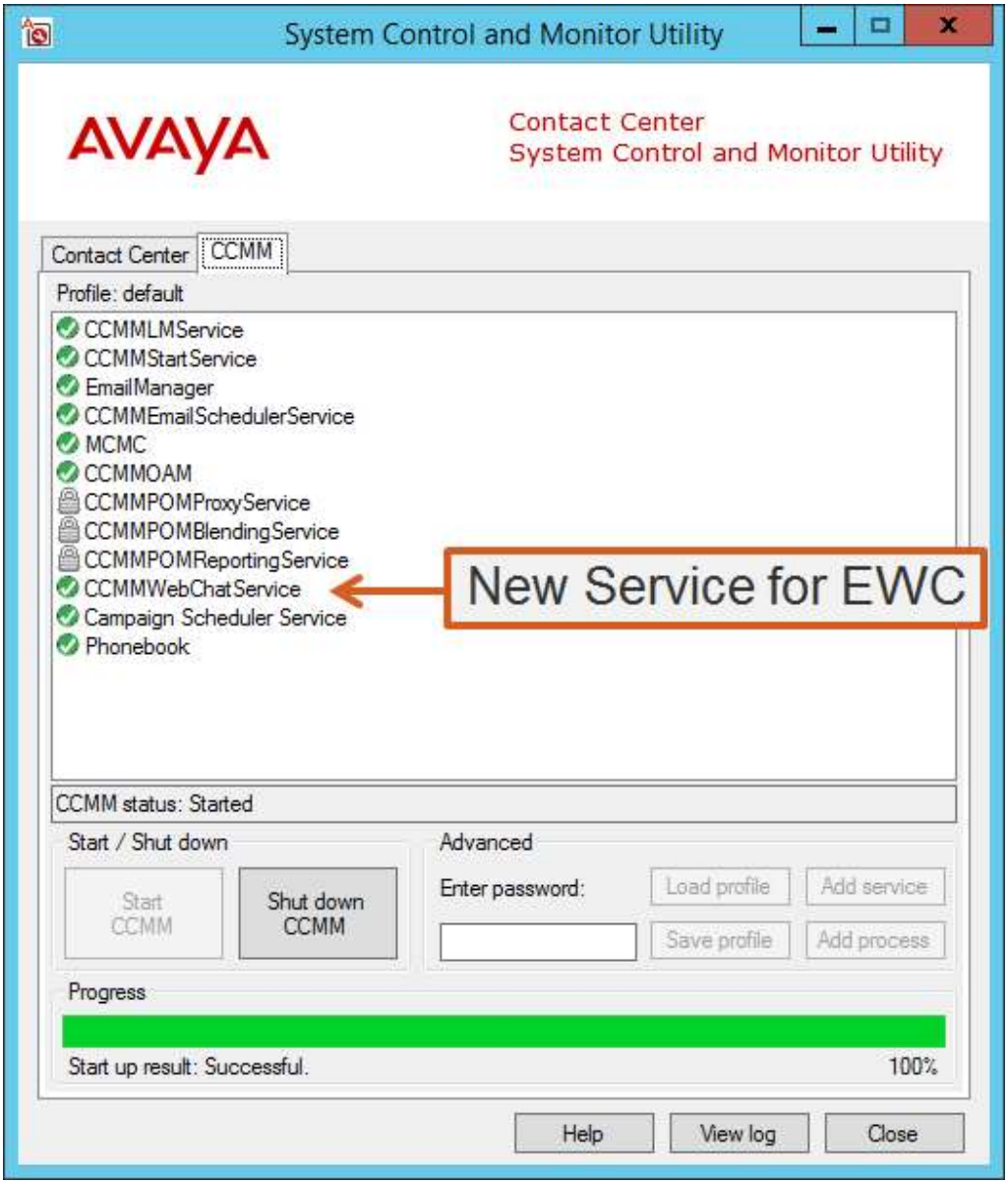
The following logs are important resources when monitoring or troubleshooting Enterprise Web Chat.

Component	Log file location:
CCMM Web Chat Service	D:\Avaya\Logs\CCMM\ CCMM_WebChat_Service.log
CCMM Starter Service	D:\Avaya\Logs\CCMM\ CCMM_StarterService_1.log
CCMM LM Service	D:\Avaya\Logs\CCMM\ LMService_1.log
Agent Controller (Tomcat)	D:\Avaya\Logs\CCMM\ CCMM_EnterpriseWebChat_AgentController.log
Customer Controller (Tomcat)	D:\Avaya\Logs\CCMM\ CCMM_EnterpriseWebChat_CustomerController.log
ACC Queue Service (Tomcat)	D:\Avaya\Logs\CCMM\ CCMM_EnterpriseWebChat_AccQueueService.log
General Tomcat logs:	D:\Avaya\Contact Center\apache-tomcat\logs

CCMM Services & Enterprise Web Chat

A new service named *CCMM Web Chat Service* is delivered as part of Enterprise Web Chat. Understanding the role of this service and the logging it generates is an important part in troubleshooting EWC issues. The service is responsible for the following:

- Management and monitoring of EWC applications, including Tomcat and ejabberd chat engine
- Management and monitoring of High Availability and web chat system health



The *CCMM Web Chat Service* is controlled and started by the *CCMM Start Service*. Refer to the CCMM Start service logging if you encounter issues starting the *CCMM Web Chat Service*. The *CCMM Web Chat Service* starts only under the following conditions:

- Enterprise Web Chat feature must be licensed. This will be checked and logged by the *CCMM LM Service*
- CCMM must be deployed on Multimedia Only server

Service start-up / shut-down events are logged to the Windows event viewer

The *CCMM Web Chat Service* monitors the state of the system at regular intervals and logs this information to the standard logs. An example of a successfully running (non-HA) system is shown below:

```
55635  INFO  2015-11-27 04:39:49,878 102072986ms WebChatService         OnTimedEvent             - ================================================
55636  INFO  2015-11-27 04:39:49,878 102072986ms WebChatService         CheckModeChange          - Check Mode Change
55637  INFO  2015-11-27 04:39:49,878 102072986ms WebChatService         CheckHealth              - Check Health
55638  INFO  2015-11-27 04:39:49,878 102072986ms EJabberdController     GetStatus                - ejabberdctl status
55639  DEBUG 2015-11-27 04:39:49,890 102072998ms EJabberdController     WriteProcessOutputToLog  - Process standard output: Using path D:/Avaya/Contact Center/EnterpriseWebChat/eJabberd
55640  DEBUG 2015-11-27 04:39:50,139 102073247ms EJabberdController     WriteProcessOutputToLog  - Process standard output: The node ejabberd@CHATORANMORE7 is started with status: started
55641  DEBUG 2015-11-27 04:39:50,139 102073247ms EJabberdController     WriteProcessOutputToLog  - Process standard output: ejabberd 15.07.40 is running in that node
55642  DEBUG 2015-11-27 04:39:50,148 102073256ms EJabberdController     GetStatus                - ExitCode=[0]
55643  INFO  2015-11-27 04:39:50,152 102073260ms TomcatController       GetStatus                - D:\Avaya\Contact Center\EnterpriseWebChat\eJabberd\tomcat_status.bat 8445
55644  DEBUG 2015-11-27 04:39:50,182 102073290ms TomcatController       WriteProcessOutputToLog  - Process standard output: tomcat is running
55645  DEBUG 2015-11-27 04:39:50,183 102073291ms TomcatController       GetStatus                - ExitCode=[0]
55646  INFO  2015-11-27 04:39:50,183 102073291ms WebChatService         EJabberdBusinessLogic    - EJabberd Status
55647  INFO  2015-11-27 04:39:50,183 102073291ms WebChatService         EJabberdBusinessLogic    - eJabberd is UP.
55648  INFO  2015-11-27 04:39:50,183 102073291ms WebChatService         TomcatBusinessLogic      - Tomcat Status
55649  INFO  2015-11-27 04:39:50,183 102073291ms WebChatService         OnTimedEvent             - Tomcat is UP.
55650  INFO  2015-11-27 04:40:00,191 102083300ms WebChatService         OnTimedEvent             - ================================================
```

- The ejabberd state and Tomcat state is checked and reported on at regular intervals in the logs.

Enterprise Web Chat Tomcat applications

To support Enterprise Web Chat, Tomcat is automatically installed on all CCMM (Multimedia) servers. The version installed is standard across all Contact Center components. Tomcat is installed under the location: *D:\Avaya\Contact Center\apache-tomcat* and a new service is added to the system named: *CCTOMCATSRV*.

> ✱ **Note:** the *CCTOMCATSRV* service is not displayed or controlled by the SCMU application but is instead deployed as an automatic service and will always be running on the system.

The following EWC applications are deployed:

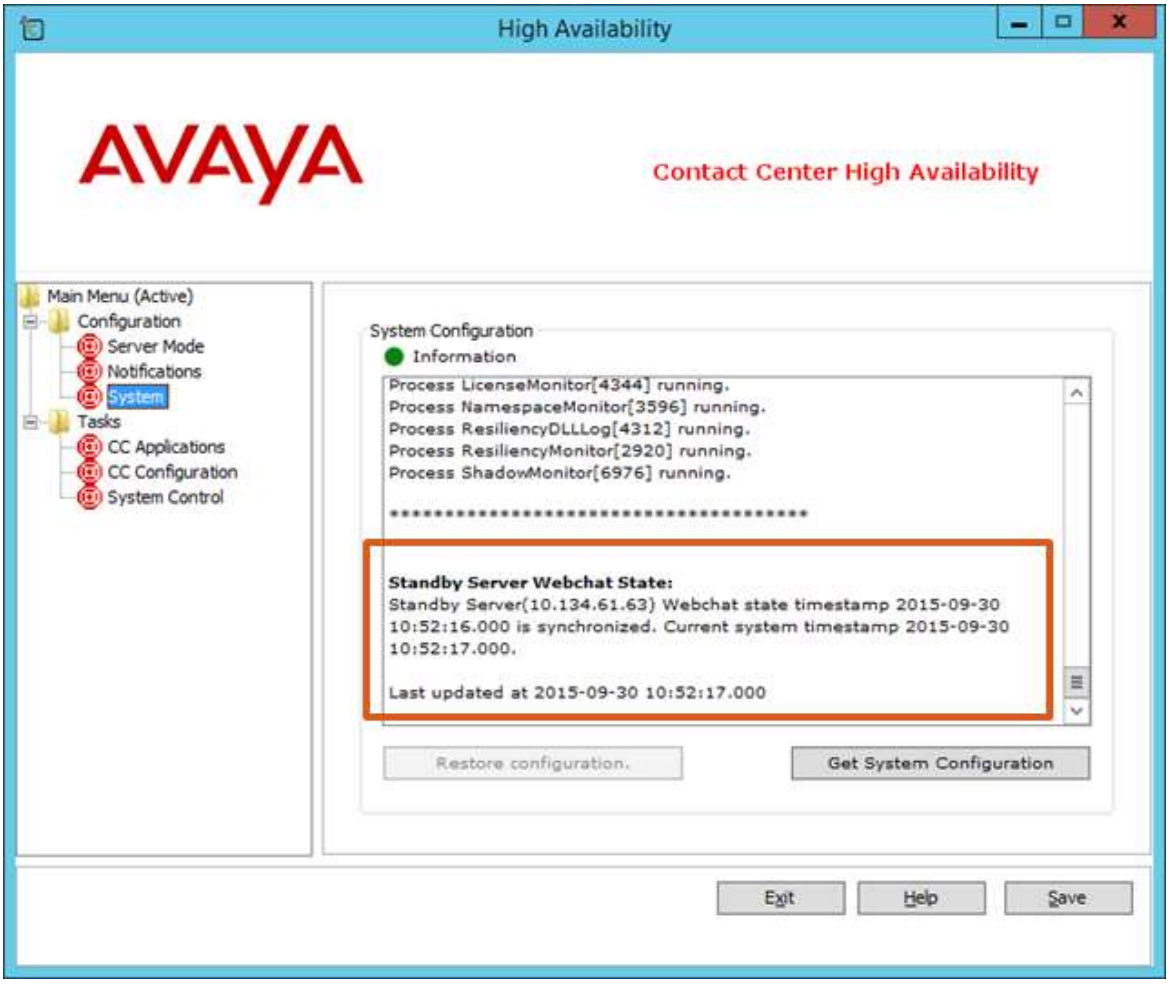
- ***Agent Controller***: Provides agent chat features, used by sample agent reference client.
- ***Customer Controller***: Provides customer chat features, used by sample Reference Web UI.
- ***ACC Queue Service***: Provides an interface to set a queued chat's position in queue & expected wait time. This can later be retrieved by the Customer UI.

High Availability

The *CCMM Web Chat Service* ensures:

- If EWC feature is on, and servers are configured for HA, the CCMM Web Chat service will automatically configure chat room clustering (ejabberd) and monitor critical processes.
- EWC will trigger a switch-over in the event of ejabbed OR Tomcat application failures.
- The *CCMM Web Chat Service* monitors the state of ejabberd clustering and reports system health in the log files.

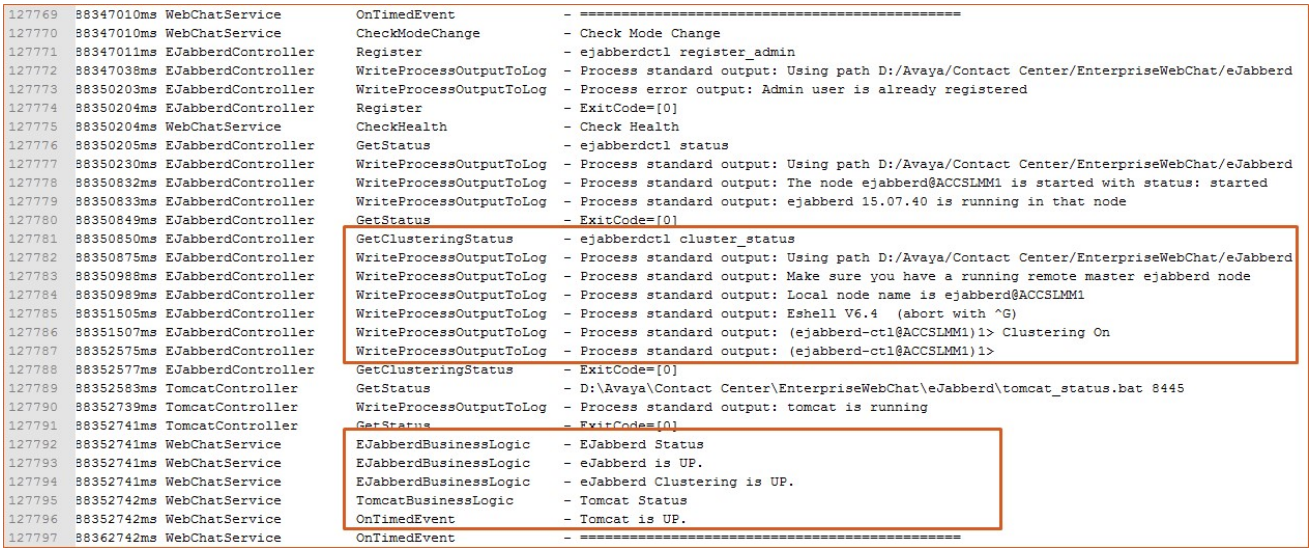
High Availability – UI:



For EWC HA systems, the Web Chat state is displayed.

Important: The system must be synchronized to allow manual OR EWC switch-overs. If the system is not synchronised, consult the CCMM Web Chat logs to investigate where the issue lies.

Sample logging for a system when both active and standby are synchronized:



```
127769  88347010ms WebChatService         OnTimedEvent             - ============================================
127770  88347010ms WebChatService         CheckModeChange          - Check Mode Change
127771  88347011ms EJabberdController     Register                 - ejabberdctl register_admin
127772  88347038ms EJabberdController     WriteProcessOutputToLog  - Process standard output: Using path D:/Avaya/Contact Center/EnterpriseWebChat/eJabberd
127773  88350203ms EJabberdController     WriteProcessOutputToLog  - Process error output: Admin user is already registered
127774  88350204ms EJabberdController     Register                 - ExitCode=[0]
127775  88350204ms WebChatService         CheckHealth              - Check Health
127776  88350205ms EJabberdController     GetStatus                - ejabberdctl status
127777  88350230ms EJabberdController     WriteProcessOutputToLog  - Process standard output: Using path D:/Avaya/Contact Center/EnterpriseWebChat/eJabberd
127778  88350832ms EJabberdController     WriteProcessOutputToLog  - Process standard output: The node ejabberd@ACCSLMM1 is started with status: started
127779  88350833ms EJabberdController     WriteProcessOutputToLog  - Process standard output: ejabberd 15.07.40 is running in that node
127780  88350849ms EJabberdController     GetStatus                - ExitCode=[0]
127781  88350850ms EJabberdController     GetClusteringStatus      - ejabberdctl cluster_status
127782  88350875ms EJabberdController     WriteProcessOutputToLog  - Process standard output: Using path D:/Avaya/Contact Center/EnterpriseWebChat/eJabberd
127783  88350988ms EJabberdController     WriteProcessOutputToLog  - Process standard output: Make sure you have a running remote master ejabberd node
127784  88350989ms EJabberdController     WriteProcessOutputToLog  - Process standard output: Local node name is ejabberd@ACCSLMM1
127785  88351505ms EJabberdController     WriteProcessOutputToLog  - Process standard output: Eshell V6.4  (abort with ^G)
127786  88351507ms EJabberdController     WriteProcessOutputToLog  - Process standard output: (ejabberd-ctl@ACCSLMM1)1> Clustering On
127787  88352575ms EJabberdController     WriteProcessOutputToLog  - Process standard output: (ejabberd-ctl@ACCSLMM1)1>
127788  88352577ms EJabberdController     GetClusteringStatus      - ExitCode=[0]
127789  88352583ms TomcatController       GetStatus                - D:\Avaya\Contact Center\EnterpriseWebChat\eJabberd\tomcat_status.bat 8445
127790  88352739ms TomcatController       WriteProcessOutputToLog  - Process standard output: tomcat is running
127791  88352741ms TomcatController       GetStatus                - ExitCode=[0]
127792  88352741ms WebChatService         EJabberdBusinessLogic    - EJabberd Status
127793  88352741ms WebChatService         EJabberdBusinessLogic    - eJabberd is UP.
127794  88352741ms WebChatService         EJabberdBusinessLogic    - eJabberd Clustering is UP.
127795  88352742ms WebChatService         TomcatBusinessLogic      - Tomcat Status
127796  88352742ms WebChatService         OnTimedEvent             - Tomcat is UP.
127797  88362742ms WebChatService         OnTimedEvent             - ============================================
```

Verifying that Multimedia Services are running

About this task

Verify that the Multimedia Contact Center services are started. Use the System Control and Monitor Utility to verify that all necessary Avaya Contact Center services are running.

Procedure

1. Log on to the Multimedia Contact Server.
2. On the Apps screen, in the Avaya section, select System Control and Monitor Utility.
4. Click the CCMM tab (see figure 1).
5. Verify that the Contact Center License Manager service (CCMMLMService) is running.
6. Verify that the Contact Center Multimedia services are running.
7. Verify that the CCMM Web Chat Service is running.
    a. Ensure CCMMWebChatService is running.
    b. Ensure the EmailManager is running if you are having problems with email.
    c. Verify that CCMM is started and running.
8. Click Start Contact Center or Start CCMM to start any stopped services.

Verifying that the Contact Center Tomcat instance is running

About this task

Verify that the Contact Center Tomcat is started. Use the Services administrative tool to verify that tomcat is running.

Procedure

1. Open the Services administrative tool (see Figure 2 below).
2. Check if the status of the *Contact Center Tomcat Instance* is running
3. If the service is not running, select it and click Restart or Start.

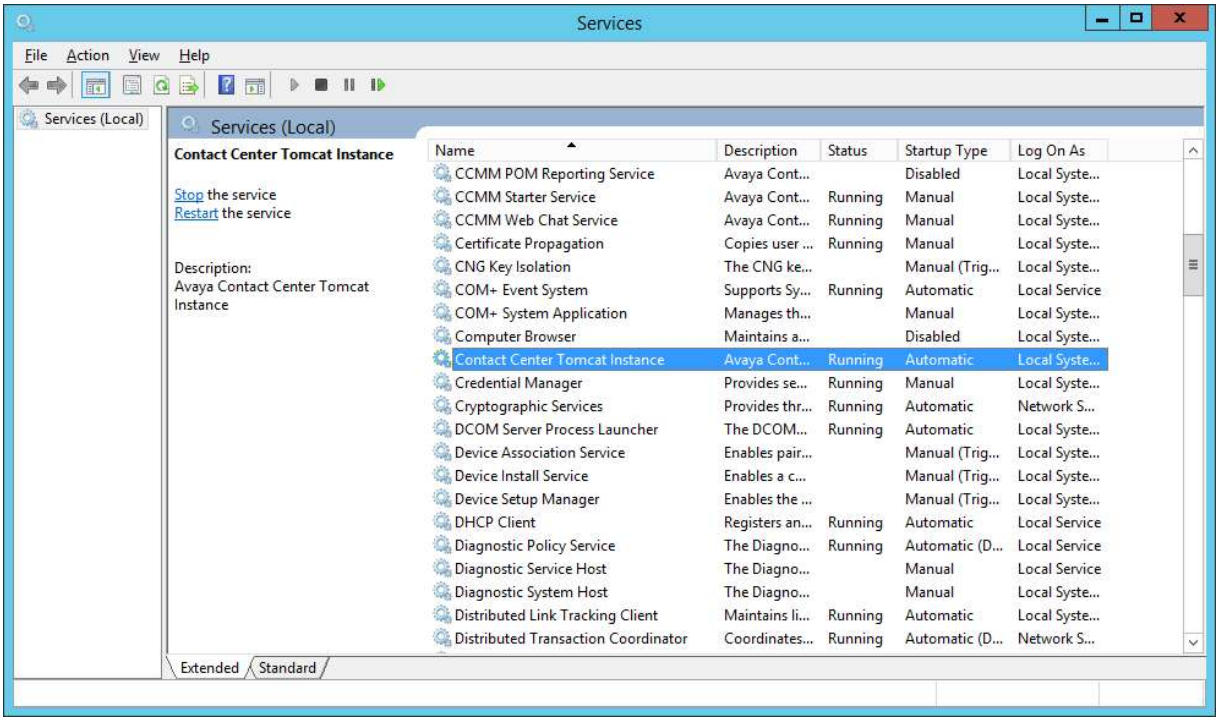


Figure 2 Windows Services

Using the CCMM dashboard

About this task

Use the CCMM dashboard (Figure 3) to verify that Contact Center is communicating with the email server and is monitoring the recipient mailbox.

Procedure

1. Log on to the Contact Center server.
2. On the Apps screen, in the Avaya section, select Multimedia Dashboard.
3. Ensure Contact Center is communicating with the inbound email server.
4. Ensure Contact Center is monitoring the recipient mailbox.

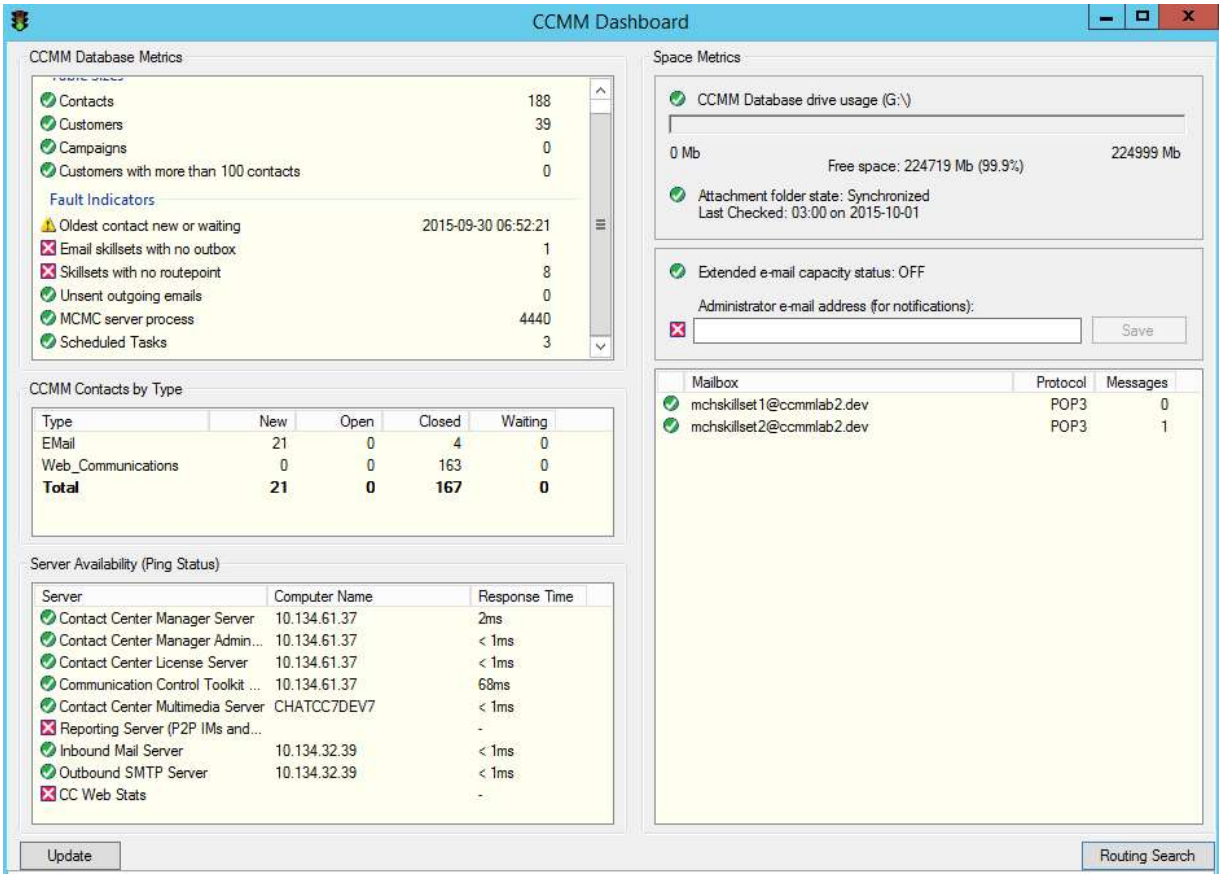


Figure 3 Avaya Aura Contact Center CCMM dashboard.

Reference Custom Desktop

Before running the reference Custom Desktop (compiled version)

- AppSettings.xml file must be updated to set parameters for AACC, HA and CCT user information.
- Default log file location: *C:\Logs\SMF\CustomDesktop.log*

Full deployment details are available with this SDK documentation.

Reference Web User Interface

If you experience issues when running the reference web user interface, check the following:

- Configure Cross-Origin Resource Sharing (CORS) filtering: available via the CCMM Administration Utility, Web Comms, Config
- EWC WebSocket URLs must be configured to connect to the CCMM server – security on/off options are available

Full deployment details are available with this SDK documentation.

LAST PAGE