Avaya Aura® Contact Center & Avaya Contact Center Select

Contact Control Service SDK (Java Script) Release 7.1.2.0.1

Release Notes & User Guide

This document contains information on SDK versioning, compatibility & user guide that is specific to this Avaya Contact Center SDK.

Table of Contents

Purpose 2
Publication History 2
Change History 3
Versions and Compatibility 4
What is Compatible 4
General Compatibility Information 4
Getting Started: Contact Control Service SDK 5
Compatibility Matrix 5
Supported Features 6
Getting Started: Contact Control Service SDK API 10
Getting Started: Contact Control Service SDK Reference Client 11
Reference Client Set Up 12
Launching the JavaScript Reference Client 12
Reference Client User Guide 13
Signing in to the Reference Client 14
Handling the first Interaction 15
Supervisor Controls Panel 15
Demo mode 15
Reference Client Design 16

Purpose

This document contains information on SDK compatibility and change history specific to this SDK. Please refer to the product documentation for more details.

Publication History

Issue	Change Summary	Date
01	Initial release of the CCS JavaScript Script SDK	19/12/2016
02	Release of CCS JavaScript Script SDK to align with CC 7.1.0.1	23/08/2019
03	Release of CCS SDK to align with CC 7.1.0.2	11/10/2019
04	Release of CCS SDK to align with CC 7.1.1.0	02/12/2020
05	Release of CCS SDK to align with CC 7.1.2.0	25/09/2021
06	Release of CCS SDK to align with CC 7.1.2.0.1	17/03/2023

Change History

SDK Version 7.0.1.0-1

This is the first release of the CCS JavaScript Script SDK. There are no previous versions.

SDK Version 7.1.0.1

The CCS SDK has been released to support Avaya Aura Contact Center and Avaya Contact Center Select release 7.1.0.1

Support has been added for the email and web chat contact types with new interfaces added.

No changes have been made to existing voice contact type interfaces.

SDK Version 7.1.0.2

The CCS SDK has been released to support Avaya Aura Contact Center and Avaya Contact Center Select release 7.1.0.2

This version of the API offers feature parity with Workspaces on AACC v. 7.1.0.2; with the following exceptions:

- Agent signature (get, set, update)

New features include Suggested responses(email), suggested phrases(web chat), page push urls, customer history, customer history media transcript, ad-hoc email, screenpops (basic implementation only, does not filter by contact type).

Interface changes: none

SDK Version 7.1.1.0

The CCS SDK has been released to support Avaya Aura Contact Center and Avaya Contact Center Select release 7.1.1.0

SDK Version 7.1.2.0

The CCS SDK has been released to support Avaya Aura Contact Center and Avaya Contact Center Select release 7.1.2.0

SDK Version 7.1.2.0.1

New feature Retrieve for User object has been added to this version.

Versions and Compatibility

What is Compatible

CCS SDK Version	AACC or ACCS 7.1.2.0	AACC or ACCS 7.1.1.0	AACC or ACCS 7.1.0.1	AACC or ACCS 7.0.1.0	AACC or ACCS 7.0.0.0	AACC or ACCS 6.x
7.0.1.0-1	No	No	No	Yes	No	No
7.1.0.1	Yes	Yes	Yes	No	No	No
7.1.0.2	Yes	Yes	Yes	No	No	No
7.1.1.0	Yes	Yes	Yes	No	No	No
7.1.2.0	Yes	Yes	Yes	No	No	No

General Compatibility Information

None.

Getting Started: Contact Control Service SDK

The JavaScript (JS) version of the CCS SDK is packaged in the *ccs-sdk-javascript-package.zip* file and is the starting point for accessing the Contact Control Service SDK. This packaged SDK zip file contains all the contents and resources required to use the CCS SDK.

Supported Versions

The CCS JS Reference Client has been tested on the following browsers and their versions:

Browser	Version
Internet Explorer	11
Chrome	53
Mozilla	48

The SDK will however work with a minimum of Internet Explorer 9, and the post 2012 versions of Chrome and Mozilla.

Compatibility Matrix

The below table is a compatibility matrix detailing the solutions, configurations and features that are supported or not supported through the CCS API.

	AACC AML	AACC SIP	ACCS
Solutions Supported?	N	Y	Y
Features, Configurations & Use Cases that are Supported?			
POM Contact Type	N/A	Y	N
Voice Contact Type	N/A	Y	Y
All Other Contact Types (including Email, EWC, IM, WebComms etc)	N/A	Y	Y

Multiple applications simultaneously controlling the same Agent *	N/A	N	N
Offsite Agent	N/A	N	N

The above table is not an exhaustive list of features supported through the CCS API.

*If a CCS API developed application is controlling an Agent or Supervisor (known as User in the Contact Control Service), it must be the sole application controlling this Agent or Supervisor. No other application can control this Agent or Supervisor in parallel with the CCS API developed application instance. This restriction includes multiple instances of a CCS API developed application.

The applications referred to above include, Avaya Agent Desktop, any Contact Center SDK Reference Client or any application developed against the Contact Center SDK's listed below;

- CCT Open Interfaces SOAP SDK
- CCT Open Interfaces REST SDK
- CCT .NET SDK
- CCMM Web Communications SDK (all)
- CCMM Agent Web Services SDK
- Enterprise Web Chat SDK
- Contact Control Service SDK's

Supported Features

Listed below are the supported features that can be invoked through the CCS API based on Contact Type.

Features Supported	Voice	POM	Email	Web Chat
User Features				
Login	Y	Y	Y	Y
Logout	Y	Y	Y	Y
Ready	Y	Y	Y	Y
Not Ready	Y	Y	Y	Y

Not Ready with Code	Y	Y	Y	Y
After Call Work	Y	N	Y	N
Retrieve	N	N	Y	Y
Resource Features				
Call Supervisor	Y	N	N	N
Originate/Initiate *	Y	N	N	N
Interaction Features				
Answer **	Y	N	Y	Y
Hold	Y	Y	Y	Y
Un-Hold	Y	Y	Y	Y
End	Y	Y	Y	Y
End With Disposition Code	N/A	N/A	Y	Y
Initiate Single-Step transfer	Y	Y	Y	Y
Initiate Supervised Transfer	Y	Y	N	N
Initiate Supervised Conference	Y	Y	N	N
Complete Transfer	Y	Y	Y	Y
Complete Conference	Y	Y	N	N
Join Conference	Y	N	N	N
Emergency Call	Y	N	N	N
Play DTMF	Y	Y	N	N

Activity Codes	Y	N	Y	Y
Intrinsics	Y	N	Y	Y
UUI	Y	N	Y	Y
Attached Data	Y	N	N	N
Agent Notes	N/A	Y	Y	Y
Get Zones	N/A	Y	N/A	N/A
Wrap-Up	N/A	Y	N/A	N/A
Wrap-Up with Code	N/A	Y	N/A	N/A
Extend Wrap-Up	N/A	Y	N/A	N/A
Create Callback	N/A	Y	N/A	N/A
Preview Dial	N/A	Y	N/A	N/A
Preview Cancel	N/A	Y	N/A	N/A
Redial	N/A	Y	N/A	N/A
Change Conference Owner	N/A	Y	N/A	N
End Conference ***	N/A	Y	N/A	N/A
Update Customer Details	N/A	Y	Y	Y
Add To DNC	N/A	Y	N/A	N/A

View Customer Details	N/A	Y	Y	Y
View Interaction details	Y	Y	Y	Y
Screenpops	N	N	N	N
Supervisor Features				
Force Logout	Y	Y	N	N

Force Not Ready	Y	N	N	N
Force Ready	Y	N	N	N
Observe	Y	N	N	N
Barge-In	Y	N	N	N
Whisper	Y	N	N	N
Web Chat Features				
IsTyping	N/A	N/A	N/A	Y
Suggested Phrases	N/A	N/A	N/A	Y
Push Url	N/A	N/A	N/A	Y
Receive Url	N/A	N/A	N/A	Y
Agent Label	N/A	N/A	N/A	Y
Customer Label	N/A	N/A	N/A	Y
Comfort Messages	N/A	N/A	N/A	Y
Send/Receive Message	N/A	N/A	N/A	Y
Transcript of transferred chat	N/A	N/A	N/A	Y
Email Features				
Read Incoming Email (HTML & Plain text)	N/A	N/A	Y	N/A
View Email details	N/A	N/A	Y	N/A
Suggested responses	N/A	N/A	Y	N/A
Reply	N/A	N/A	Y	N/A
Reply all	N/A	N/A	Y	N/A
CC	N/A	N/A	Y	N/A
BCC			N	
Attach Files*	N/A	N/A	N	N/A

Insert / edit Signature*	N/A	N/A	N	N/A

*The Originate feature is not applicable for POM. The Preview Dial or Redial features would be the closest equivalent POM feature. Initiate
The Answer feature is not applicable for POM. POM interactions are automatically answered. * End Conference is a POM specific feature. In Voice, a User can drop themselves out of a conference by invoking the End feature.

Getting Started: Contact Control Service SDK API

Information about the CCS API is available in the JavaScript Script documentation overview. The JS SDK documentation provides the following;

1) A description of the Contact Control Service (CCS) & CCS SDK
2) A description of the Contact Control Service API
3) A list of third party dependencies
4) The structure of the CCS API
5) How to get started developing a Contact Control Service API compliant application

The documentation for the CCS API are located in *documentation* directory of the SDK package. To launch the documentation, browse to the *documentation* directory and open the *index.html* file. This will launch a browser and load the overview page of the CCS SDK.

Getting Started: Contact Control Service SDK Reference Client

Within the SDK package, you will also find a reference client implementation.

The goal of the Reference Client (RefClient) is to provide a sample implementation of how the JavaScript CCS API could be used. The RefClient is an example of how to develop an application against the CCS API but it should not be taken as the only way to use the CCS API.

The GUI of the RefClient has been implemented with the primary goal of understanding how to use the CCS API. There are many buttons which are disabled most of the time. This was done to make it easier to see when a capability becomes enabled as the button, wherever possible, are driven solely from capabilities provided in CCS API events.

For a complete understanding of every operation possible through the RefClient please refer to the JavaScript CCS API documentation referred to in the section Getting Started: Contact Control Service SDK API.

Reference Client Set Up

An understanding of JavaScript and web development is assumed.

1. Host the API Reference Client on a web server. The API Reference Client has been developed and tested while hosted on Windows 2012 IIS. This is the preferred method of hosting but it will work on other options including a locally running python server. It is important to note that it is not permitted to host custom applications including those utilizing the CCS SDK on AACC/ACCS servers. This and any other custom web applications must be hosted external to the AACC/ACCS server.

2. Copy all files within the reference-client Folder to the application folder on the web server.

3. The websocket connection is always secure over the 443 port therefor the URL to the reference client must use the https protocol. An example of the URL will be: https://WebServerName/RefclientJS

4. The server root certification must be applied to the hosting server, the client and the AACC/ACCS server.

Note: In mission critical HA environment where the Reference Client is connecting to the contact center managed host name. If the contact center security certificates are created as per the contact center documentation the hostname of the hosting web server must be used in the browser URL when browsing to the Reference Client application.

If it is preferred to use the hostname FQDN of the webserver hosting the Reference Client the contact center managed hostname FQDN must be added to the contact center Subject Alternative Name (SAN) of the contact center server's certificates. This allows the user to browse to the Reference Client using the hosting web server hostname FQDN and connect to the contact center using the managed contact center hostname FQDN.

Launching the JavaScript Reference Client

To launch;

1. On the client machine browse to the Reference Client URL (example URL: https://<<WebServerhostname>>/<<ReferenceClientWebSiteName>>) and see the home screen render,

a. populate the AACC/ACCS server hostname (managed hostname for mission critical systems),

b. windows user name associated with the agent account

c. password of user account

d. Click the Sign In button.

Reference Client User Guide

Detailed below is a guide providing instructions on how to use the Reference Client.

Prerequisites:

1. A fully functional AACC/ACCS 7.0.1.0 or higher server (7.1.0.1 for Email/Web Chat).
2. Have successfully launched the RefClient as described in the section Launching the JavaScript Reference Client

Signing in to the Reference Client

Below is a step by step guide on how to sign in to the Contact Center from the Reference Client.

1. Sign into the AACC/ACCS server with the following parameters:

    a. AACC server hostname; the server hostname must resolve to the AACC/ACCS server and this same name must be used to identify the server by its signed certificate. The root certificate of the CA that signed the AACC/ACCS certificate must be loaded on the client machine. In a mission critical/business continuity environment the managed server hostname must be used and the certificate SAM should include the managed hostname and the physical server FQDN.

    b. Username – The username of the Windows User Account that is associated with the User (Agent/Supervisor) that you want to connect to the Contact Center. This is the same username which would be used for other Contact Center client applications, i.e. Avaya Agent

    Desktop. The format required is Domain\Username

    c. Password – The password of the Windows User Account that is associated with the User (Agent/Supervisor)

A full description on these items can be found in the JavaScript documentation for the CCS API in the Client object.

2. Click the SignIn button. The displayed page changes to the signedin route.

3. All browsers have a console usually displayed by clicking F12 on the keyboard and will provide more detailed information required for troubleshooting the application.

4. After successful authentication the Reference Client is now connected to Contact Center but the User is not logged into the Contact Center. Observe the current state of the User is “LoggedOut” and the agent name drop down button is enabled on the top left of the page.

    a. If the state is different in your client this may be because the User was already logged in, possibly using a different client.

    b. If required it should be possible to Logout the user by clicking on the user name and selecting “Log out” to return to a “LoggedOut” state

5. It is not necessary for the User to login to Contact Center to receive a direct call (personal call).

6. Selecting the drop down of agent name and toggling the “demo” option reveals extra panels and all buttons on the Reference Client. The status of capabilities is reflected in disabled/enabled buttons. The extra panels list all capabilities and properties for each object.

Handling the first Interaction

Below is a step by step guide to handle an Interaction in the Reference Client.

Prerequisites:

1. Have successfully signed in to the Reference Client as described in the section Signing in to the Reference Client

Steps:

1. From a phone set or another client application place a call directly to this User's number.
2. A new Interaction should appear in a panel depending on the size of the screen in the middle of the Reference Client
    a. Observe that the Interaction is in the "Ringing" state
    b. Observe that the *Answer* button is enabled
    c. To view intrinsics for the Interaction click the enabled "Intrinsics" button to reveal the pop up intrinsics form.
3. Click the *Answer* button
    a. Observe that the Interaction transitions to the "Active" state
    b. Observe that several Interaction buttons now become enabled
4. Click the *Drop* button to release the Interaction

Supervisor Controls Panel

Click on the agent name in the top right of the page and select "Supervisor Controls" from the list of options.

If the User is a supervisor the "Supervisor Controls" option is enabled in the drop down and after clicking the "Supervisor Controls" panel is revealed. The agents reporting to this Supervisor are listed in the top table and when ticked their monitored calls appear in the second table. Agent state can be forced and voice calls can be observed, whispered to and barged-into.

Demo mode

Demo mode is toggled on and off by click on the agent name on top left of page and selecting "Demo".

Reference Client Design

The Reference Client is a single page application written in Angular JS. All code required is retrieved with a single page load. Every angular application starts from creating an angularJS module. An angularJS module is a container for the different parts which in this reference client includes: templates of html, angularJS controllers, services and factories.

The 'templates-app.js' file at the top level of the application contains the entire application html.

The folder structure of the JavaScript reference client is as follows:

Folder	Significant Files	Comment
assets	rfcljs-0.0.1.css	Files available to the application. Includes the application css style file.
assets\ccs	ccs-min.js	Contact Control Service (CCS) API library required for connecting to the server and all agent and call control features.
fonts	48	Required by third party library FontAwesome.
src		All application JavaScript code.
src\app	app.js	Main angular application file.
vendor		Code of third party libraries used by the application including AngularJS

The assets folder contains files available to the application – all of the application pngs in use are stored here.
Within the ccs folder the ccs-min.js SDK library file is located which is what the application uses to create the websocket connection and communicate with the server. The SDK comes with a full suite of documentation which describes each of the available objects and their properties and methods. The src folder contains all code specific to this application. Apart from the "app.js" file this folder is filled with subfolders corresponding to high-level sections of the application, often corresponding to top-level CCS objects.

The "app.js" is the main application configuration file. It starts the application process by defining the "refClientJS" angularJS module and requiring the high-level angularJS modules from "src/app" that are required.

Each app folder contains the related angular factory or service implementations where the applications functionality is defined. So all we need to do in `app.js` is specify a default route to follow. In this case, our `signin` angularJS module is where we want to start, which has a defined route for `/signin` in `src/app/signin/main.js`.

In `src/app/signin' the 'main.js' file contains the two main sub routes of this application, sub route "signin" and sub route "signedin" and their associated template files.

The html rendered on the browser is changed appropriately as the user changes from not signed in (sub route "signin") to signed in (sub route "signedin") after successful authentication with the AACC/ACCS server.

"main.js" also contains the main controller of the application "MainCtrl". This main angularJS controller contains the signin method and the isSignedIn Boolean used by the view when in the "signin" state.

<SNIP from controller MainCtrl in main.js>

```
//sessionService will manage the session of the user   sessionService.signIn($scope.server,
$scope.username, $scope.password);  </SNIP>
```

The "refClientJS.session" angularJS module is required by "refClientJS.main"making the angularJS factory "sessionService" available to the "MainCtrl" angularJS controller allowing it to make the call to the "sessionService" "signin" method.

Within "src/app/session" is the file sessionService.js which contains the angularJS factory "sessionService". This angularJS factory contains all the methods required to interact with the Contact Control Service (CCS) *Session* object.

Sign in requires an instance of the CCS *Client* object and the method SignIn is called with the username, password and callback functions for the CCS *Session* and the CCS *Client*. These call back functions also exist in the "sessionService" factory.

Once signed in the application state is "signedin" and the "SignedinCtrl" angularJS controller takes over control of the view.

All other sub folders contain .js files for each angularJS service or angularJS factory of that section of the application. These angularJS services and angularJS factories contain all the necessary methods and data objects the application needs to communicate with the AACC/ACCS server over the connected websocket connection.

The notifications received and processed by the client handler call back functions are as follows:

Notification Type	Description
NEW	Notifies the client of a presence of a new object in the object hierarchy.
UPDATE	Notifies the client that an existing object in the hierarchy has been updated.
DELETE	Notifies the client that an existing object has been deleted from the object hierarchy.
RESPONSE	Notifies the client of a response to a request it sent that had been tagged with a RequestID.
ERROR	Notifies the client that an error has occurred in the handling of a request.

The application view reflects the up to date information sent to the application service call back methods for each section of the application.

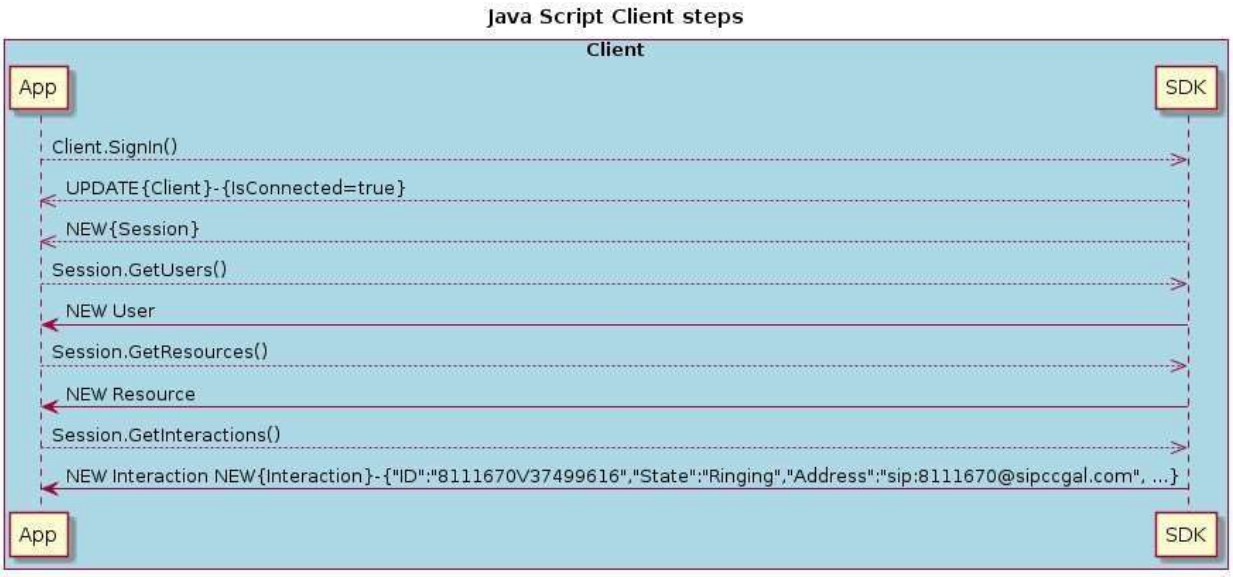
On successful authentication the application state changes to “signedin” and the “signedin/signedin.tpl.html” is displayed in the application view.

The *User* Object name property populates a button which is a drop down on the top left of the application. The options beneath this dropdown include “Log in” which when selected by the user pops the “LoginModal.tpl.html” modal form.

The *User* login information (user id, password and Pom Zone if relevant) is collected on the “LoginModal.tpl.html” modal form. The login executes the logInToContactCenter method in the “app/signedin/signed.js” file which uses the Login method of the CCS *User* object. This object is persisted in the “userData” angularJS factory user object.

<SNIP from ctrl.logInToContactCenter in src\app\signedin\signedin.js> userData.user.Login($scope.loginPassword, $scope.pzones.repeatSelect); </SNIP>

The following diagram displays the high level messages between the SDK and the JavaScript client during the sign in procedure and receiving a new interaction:



For most of the requests in the Reference Client the methods relating to these requests do not return a value. This is due to the fact that most requests into the CCS API do not return a success or failure. Instead a separate incoming event is received.

The state of the Reference Client is driven by incoming events and not from outgoing requests; it is never assumed an outgoing request will succeed. The Reference Client instead relies solely on the incoming event to indicate a state change and update the display.