A MITEL
PRODUCT
GUIDE

Mitel OpenScape Contact Center Enterprise V12

Web Interaction REST API Framework

Web Interaction REST API Framework

Programming Guide
10/2024

A31003 -S22C0 -R105 -01-7620

Mitel

Notices

The information contained in this document is believed to be accurate in all respects but is not warranted by Mitel Europe Limited. The information is subject to change without notice and should not be construed in any way as a commitment by Mitel or any of its affiliates or subsidiaries. Mitel and its affiliates and subsidiaries assume no responsibility for any errors or omissions in this document. Revisions of this document or new editions of it may be issued to incorporate such changes. No part of this document can be reproduced or transmitted in any form or by any means - electronic or mechanical - for any purpose without written permission from Mitel Networks Corporation.

Trademarks

The trademarks, service marks, logos, and graphics (collectively “Trademarks”) appearing on Mitel’s Internet sites or in its publications are registered and unregistered trademarks of Mitel Networks Corporation (MNC) or its subsidiaries (collectively “Mitel), Unify Software and Solutions GmbH & Co. KG or its affiliates (collectively “Unify”) or others. Use of the Trademarks is prohibited without the express consent from Mitel and/or Unify. Please contact our legal department at iplegal@mitel.com for additional information. For a list of the worldwide Mitel and Unify registered trademarks, please refer to the website: http://www.mitel.com/trademarks.

Contents

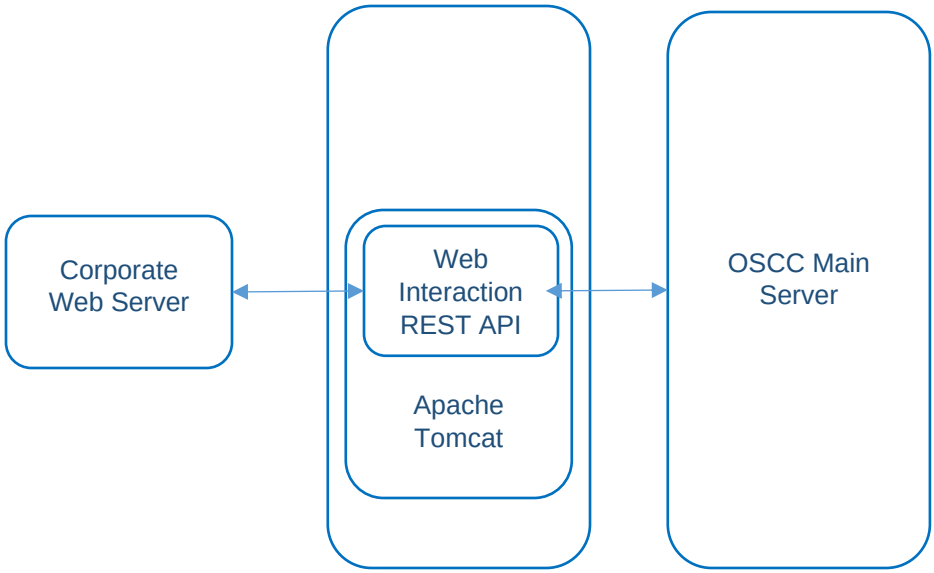
1 About the Web Interaction REST API

The Web Interaction REST API allows the development of Web Chat and Web Callback applications that integrate with the OpenScape Contact Center system.

It consists of a REST interface, which allows sending messages from the Corporate Web Page to the OpenScape Contact Center and sending messages from the OpenScape Contact Center to the Corporate Web Page. It also allows creating Callbacks on the OpenScape Contact Center from the Corporate Web Page.

1.1 Solution Overview

The figure below shows a high level overview of the Web Interaction REST API architecture.



2 Pre-requisites

For Web Interaction REST API interface there are some steps that must be performed before using the API.

- Knowledge on REST (Representational state transfer) web services.
- Install a Tomcat server or an **OpenScape Contact Center Application Server** (it can be collocated into OSCC server or into another machine).
- Configure the webinteractionsdk.xml file.

Note: For more details about installation of the Application Server, see *OpenScape Contact Center Installation Guide*.

3 Principles

3.1 Functional Description

To get access to the Web Interaction REST functions, the external application must register on the Web Interaction REST server.

The Web Chat / Web Callback application will be authenticated via OAuth 2.0 by means of a Client Id and a Client Secret. A token is generated to be used in all HTTP requests from the Web Chat / Web Callback application to the Web Interaction REST service.

In a multitenant system, the application must send the business unit name in the request for a new Web Interaction session. The OpenScape Contact Center will verify if the business unit name is configured.

3.2 Configuration

The Web Interaction REST API must be configured in the .xml file webinteractionsdk.xml which can be found in the conf directory of Apache Tomcat file structure. The following parameters must be configured:

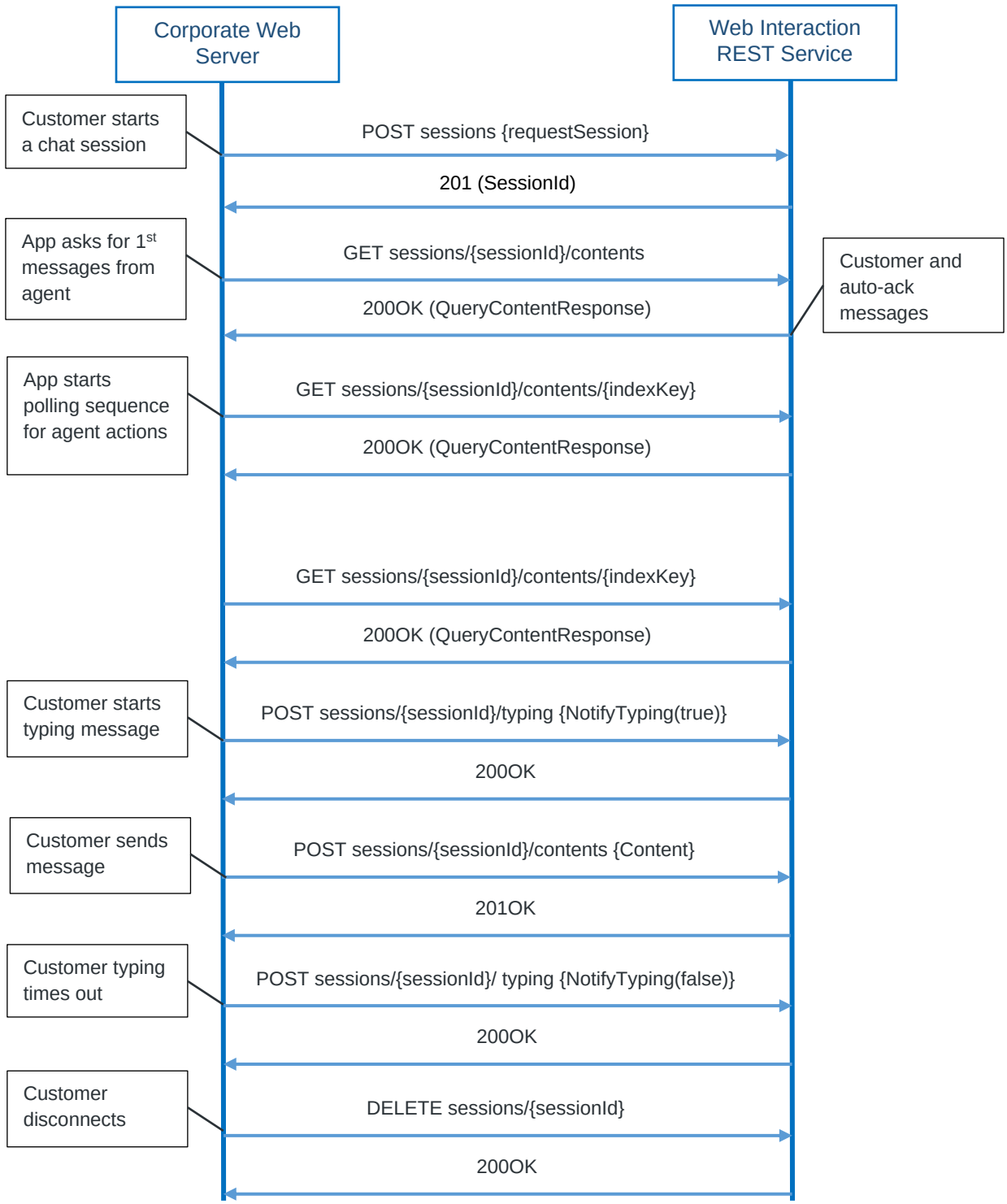
- server-name → contains the server name or IP address of the OpenScape Contact Center main server
- server-port → contains the port on which the OpenScape Contact Center is listening for Web Interaction messages.
- server-port-ssl → contains the port on which the OpenScape Contact Center is listening for Web Interaction messages if TLS is enabled.
- server-ssl-enabled → indicates if TLS is enabled between the Apache Tomcat server and the OpenScape Contact Center.
- socket-timeout → indicates for how long the TCP socket will remain open with no traffic.
- max-sockets → indicates the maximum number of sockets which can be created at a time.
- servlet-name → indicates the name of the servlet.
- oauth-client-id → application Client Id.
- oauth-client-secret → application Client Secret.
- oauth-token-validity-time → indicates for how long the token will be valid. If it is set to ‘0’, the token will not expire.

3.3 Flow Examples

Below we can see message sequence flows that demonstrate how the Web Interaction application will integrate with the OpenScape Contact Center.

3.3.1 Web Chat Session

This is the example of a Web Chat session in which the new chat session is initiated, the customer sends a message and disconnects.

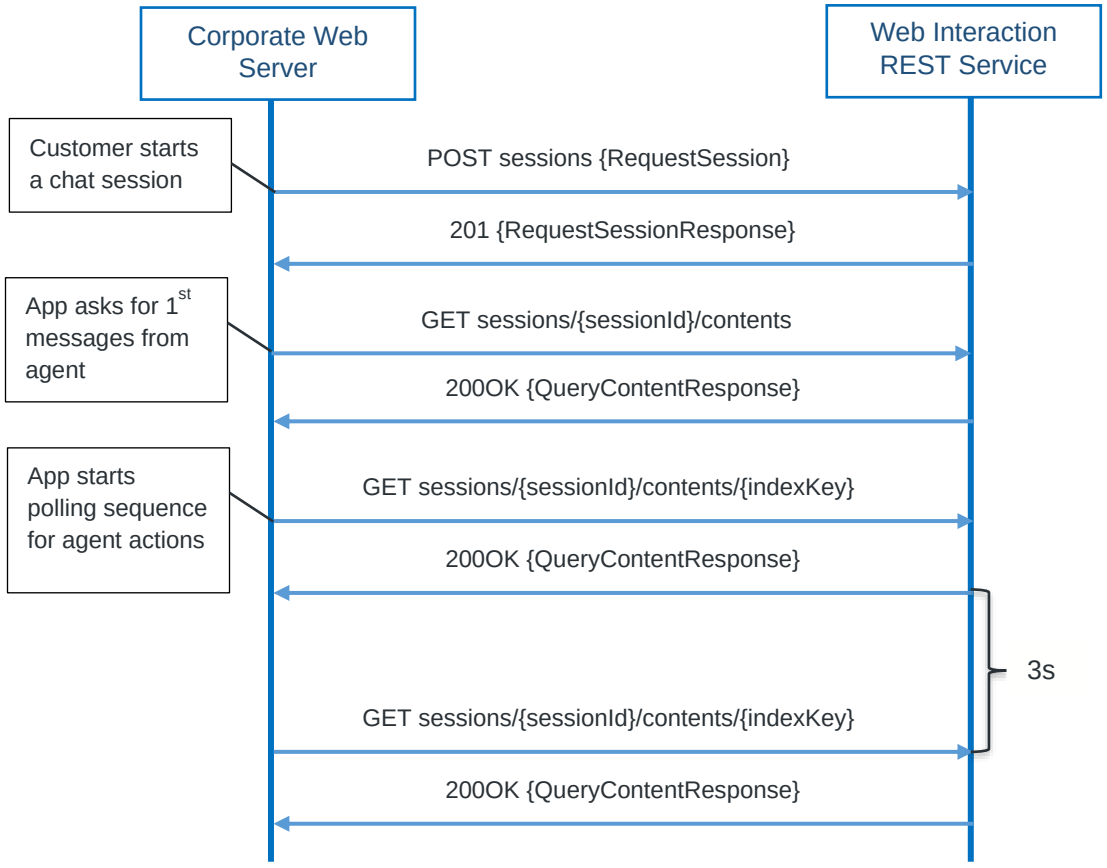

Corporate Web Server
Web Interaction REST Service
Customer starts a chat session
POST sessions {requestSession}
201 (SessionId)
App asks for 1st messages from agent
GET sessions/{sessionId}/contents
200OK (QueryContentResponse)
Customer and auto-ack messages
App starts polling sequence for agent actions
GET sessions/{sessionId}/contents/{indexKey}
200OK (QueryContentResponse)
GET sessions/{sessionId}/contents/{indexKey}
200OK (QueryContentResponse)
Customer starts typing message
POST sessions/{sessionId}/typing {NotifyTyping(true)}
200OK
Customer sends message
POST sessions/{sessionId}/contents {Content}
201OK
Customer typing times out
POST sessions/{sessionId}/ typing {NotifyTyping(false)}
200OK
Customer disconnects
DELETE sessions/{sessionId}
200OK

3.3.2 Initiate Web Chat Session

A Web Chat Session is initiated by a customer. The following data can be provided in the Session Request message:

- **Business Unit Name** – in a multi-tenant environment, it identifies the tenant for which the Web Chat session shall be established. In a single tenant environment, the value “DEFAULT” must be sent.
- **Caption** – this is the first message from the customer.
- **Contact Data** – this is a list of key-value pairs which can be used to pass further data about the contact or about the customer.
- **Customer Name** – name of the customer.
- **Destination** – this can be used to identify the subject of the contact as for example Sales or Service.
- **Language** – the language can be used to format the agent messages.
- **Priority** – the priority the contact shall be handled by the contact center.
- **Source** – this is the identification of the customer. This field will be used by the 360º Customer View to identify the customer.

Just after receiving the confirmation of the session initiation with the Session Id the Web Chat application shall send an initial Contents request and start polling for the further Contents.



-

3.3.3 Disconnect Web Chat Session

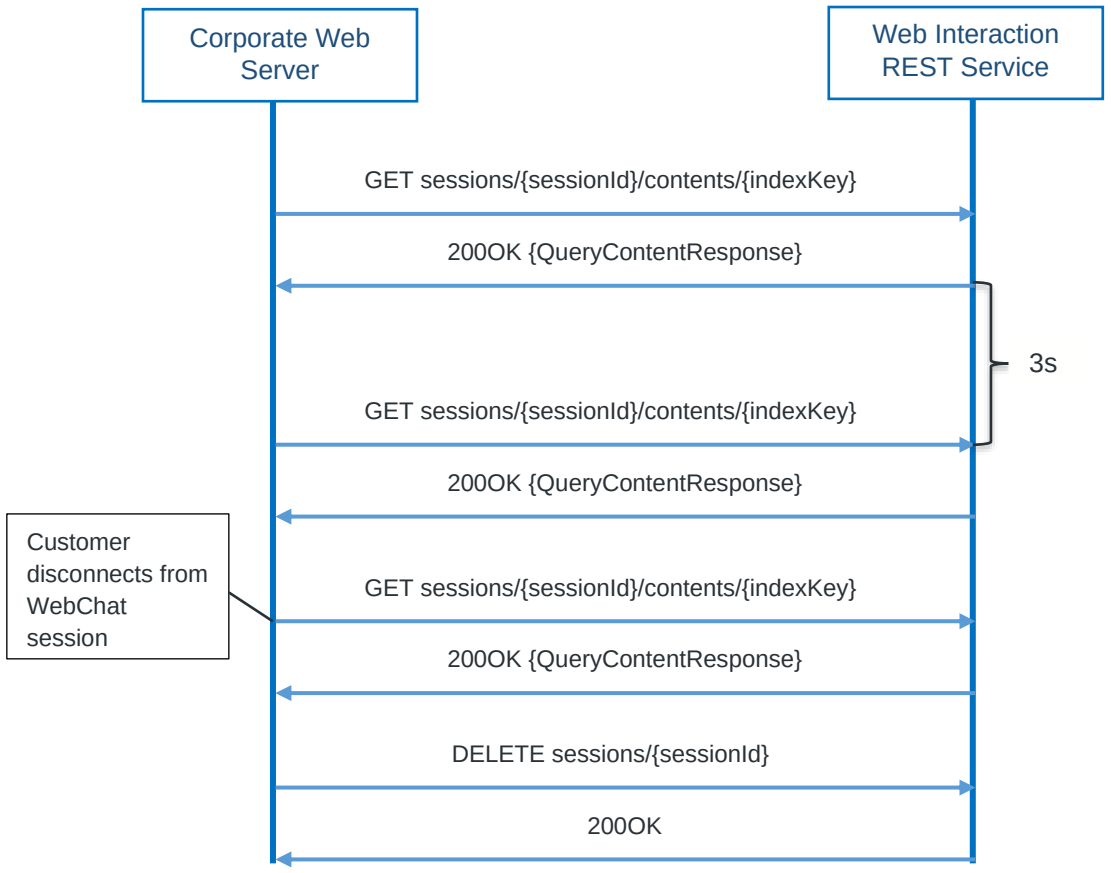
A Web Chat Session can be disconnected by the agent or by the customer.

If the Web Chat Session is disconnected by the customer, it is recommended to send a Content Request before sending the Delete Session request. The Content Request would get any pending message from the agent that had not been presented to the customer yet.

If the Web Chat Session is disconnected by the agent, the disconnection is informed to the Web Chat application in the response to the Content Request, which shall also contain any eventual pending messages written by the agent.

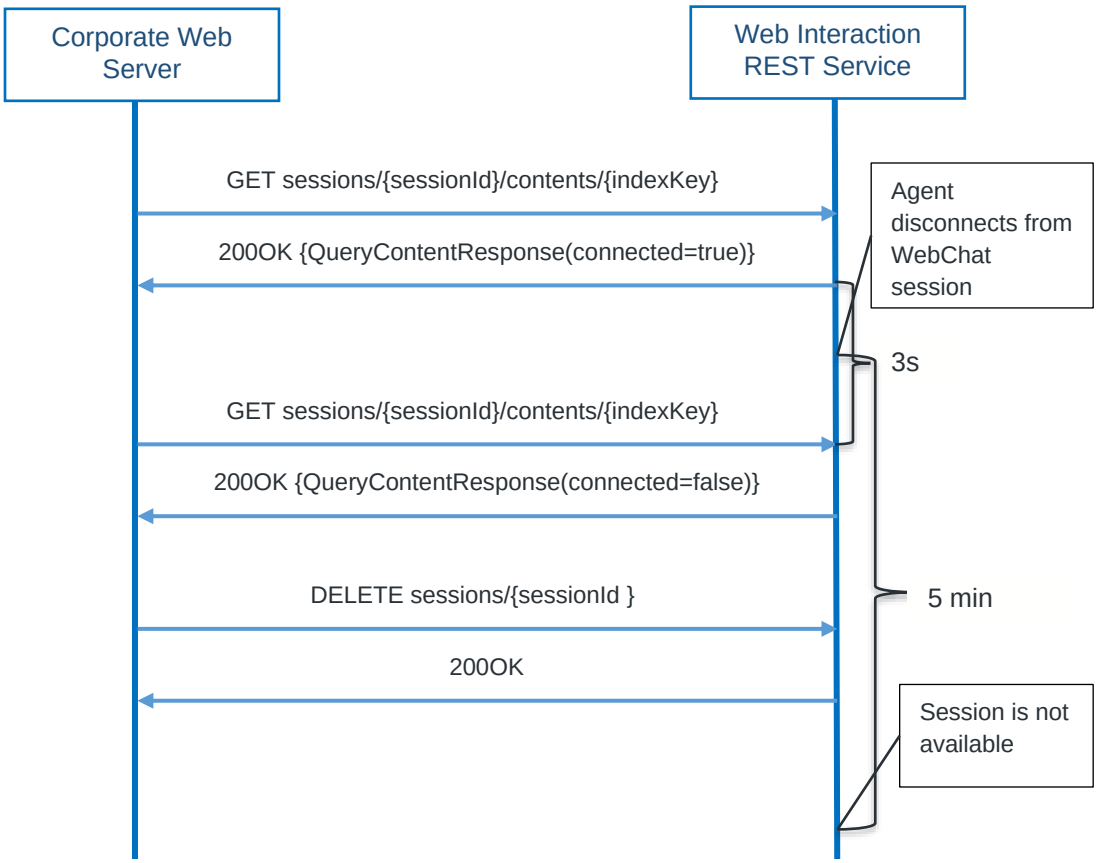
Note: after the disconnection, the Web Chat Session will remain available for 5 minutes. Before sending a DELETE Session request, the Web Chat application can still ask for content in this period of time.

Disconnection from Customer:



-

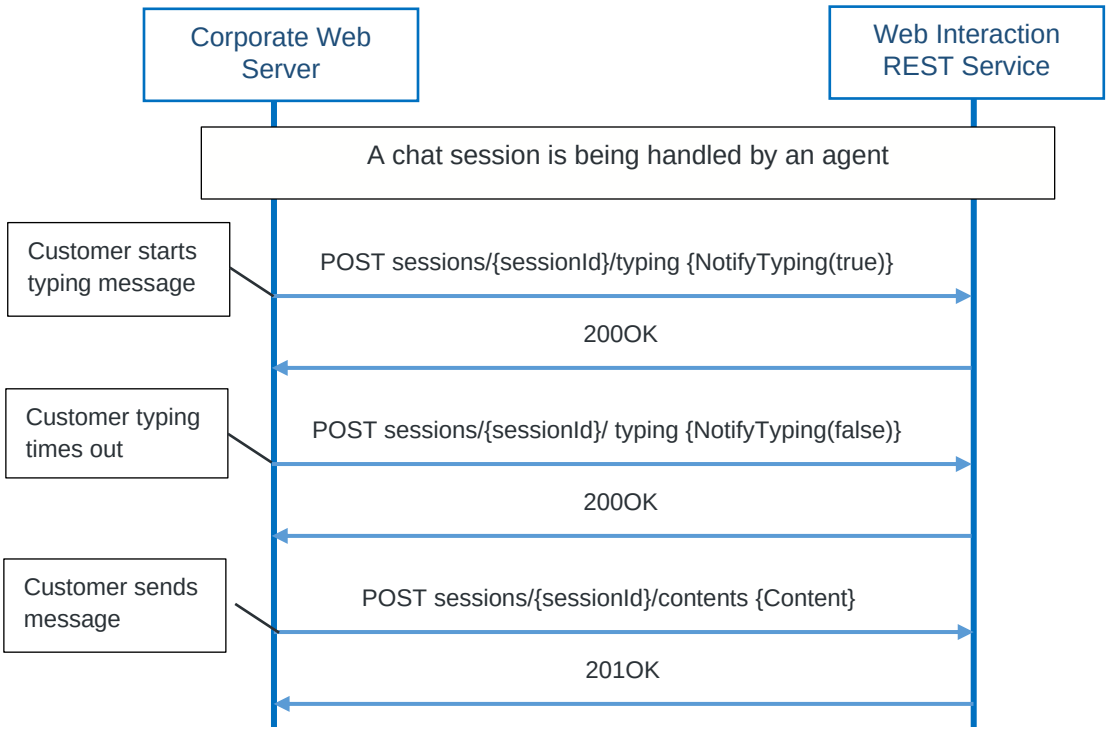
Disconnection from Agent:



-

3.3.4 Customer Messages

The messages from the customer are sent from the Web Chat application to the Web Interaction REST service by posting contents. When the customer starts typing, a “typing” message with content “true” can be posted to the server. When the customer stops typing, a “typing” message with content “false” can be posted to the server.

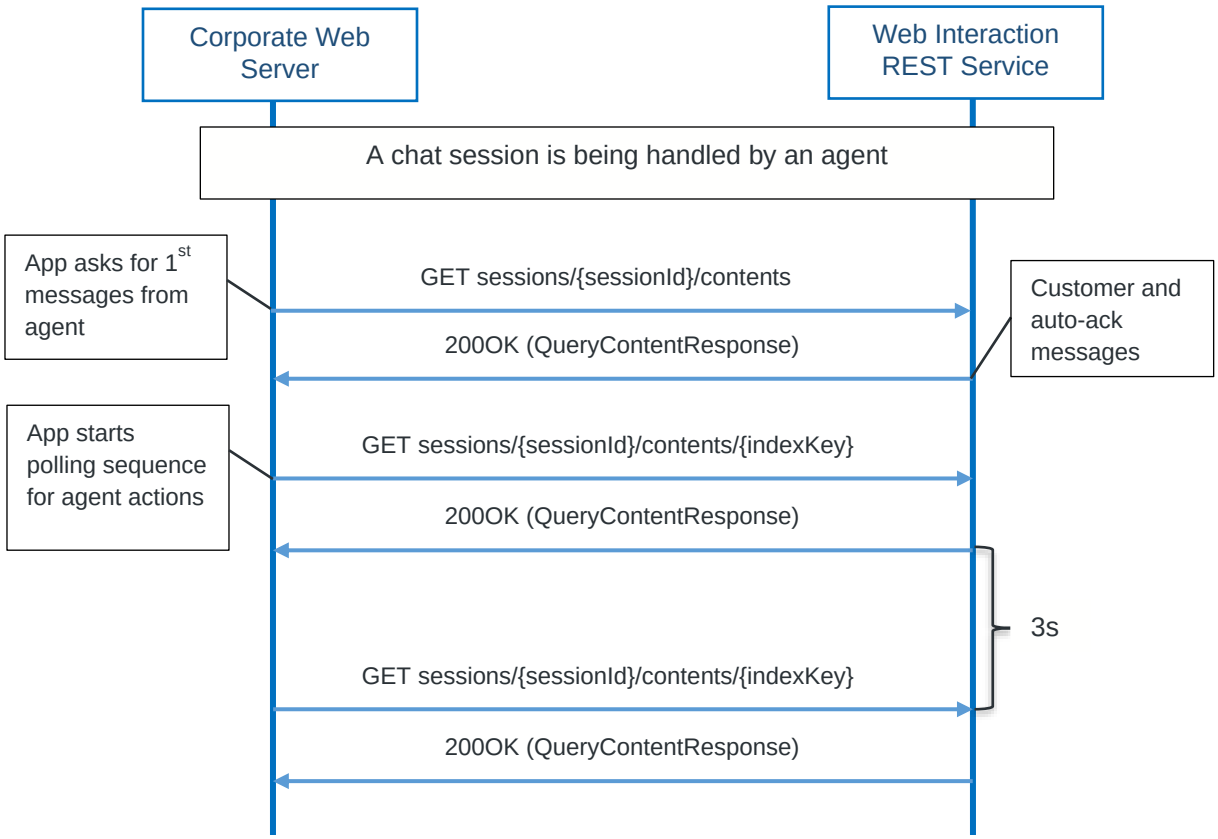


3.3.5 Agent Messages

The Agent Messages are collected by the Web Chat application by polling the server for new messages.

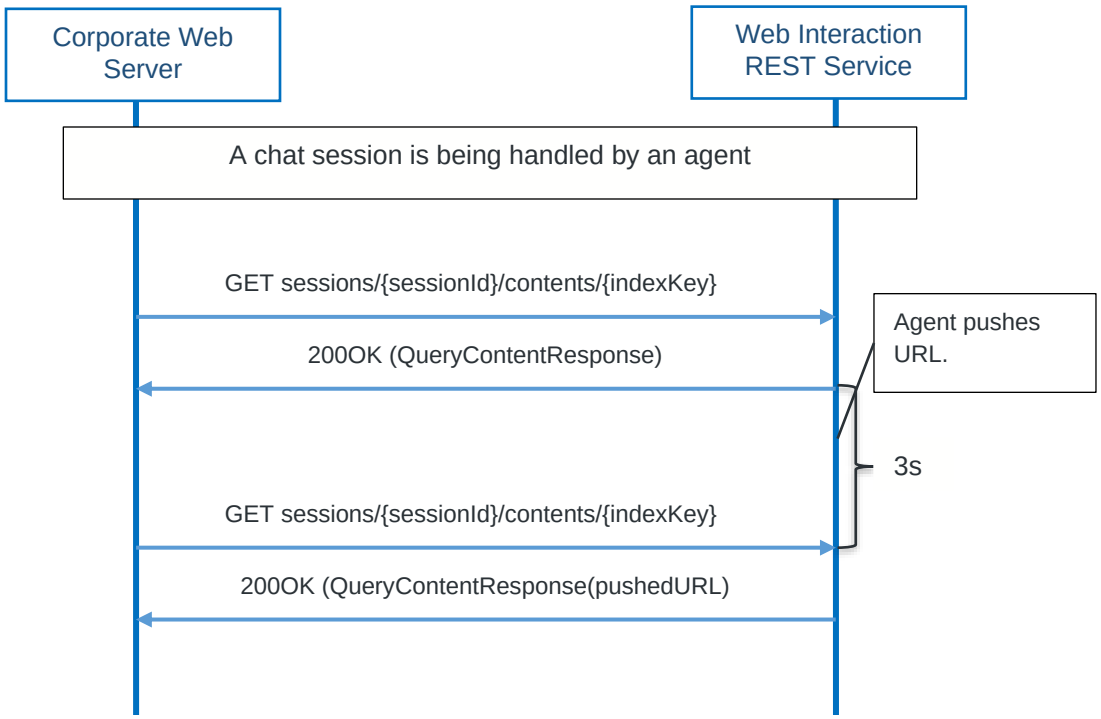
When the Web Chat session is initiated, the Web Chat application can send a contents polling request to get all messages which have been exchanged so far.

From the second polling request, the Web Chat application can send a contents polling request to get messages from a specific entry index. It is recommended that the Web Chat application sends a polling request every 3 seconds.



3.3.6 Pushed URL

When the agent pushes a URL to the customer, this is informed to the Web Chat application in the Content response.



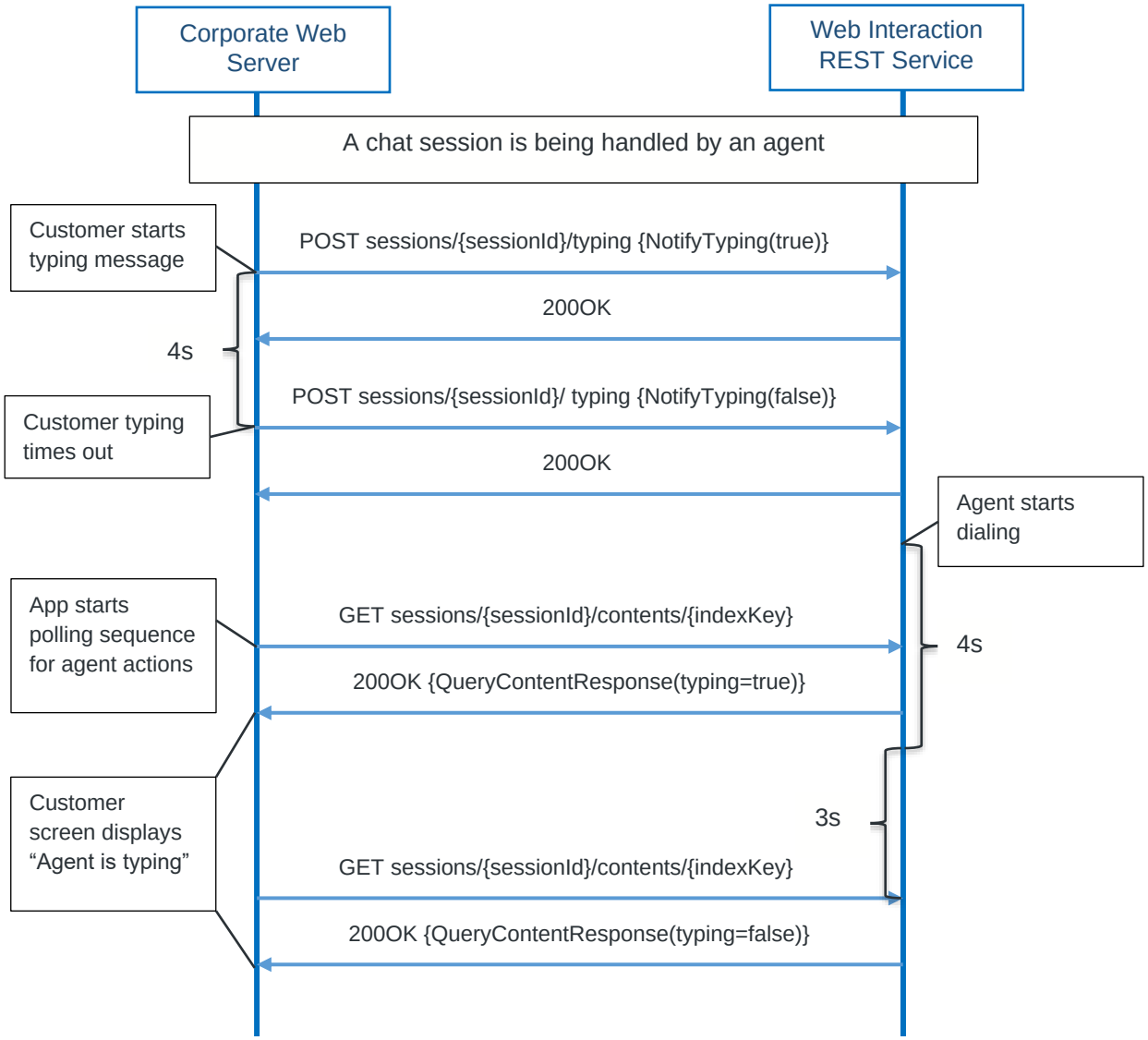
Example of a Content response message:

```
{
  "sessionId": "W16612020001010011101756 53c3d287b449c5b7e010462f3c6638",
  "content": [
    {
      "messageType": "AGENT_PUSH_URL",
      "from": "14",
      "agentFirstName": "Claudio",
      "agentLastName": "Bastos",
      "content": "<FONT size=2 face=Tahoma>Agent pushed <FONT tabIndex=-1
contentEditable=false style=\"BACKGROUND-COLOR: #ffffff\" size=2
face=Tahoma>http://www.atos.net</FONT> to you.</FONT>",
      "formattedContent": "<TABLE width='95%'  border='0' cellspacing='0'
cellpadding='2'><TR><TD width='2%' valign='top'><IMG SRC='images/agenticon.gif'
WIDTH='16' HEIGHT='16'></TD><TD width='98%'><FONT size=2 face=Tahoma>Agent pushed
<FONT tabIndex=-1 contentEditable=false style=\"BACKGROUND-COLOR: #ffffff\" size=2
face=Tahoma>http://www.atos.net</FONT> to you.</FONT></TD></TR></TD></TR></TABLE>",
      "pushedUrl": http://www.atos.net
    }
  ],
  "pushUrl": "http://www.atos.net",
  "callMeAgentKey": 0,
  "newIndexKey": 9,
  "whoIsTyping": "",
  "redirectUrl": "",
  "connected": true
}
```

3.3.7 Who is Typing

When the customer starts typing, a “typing” request with content “true” must be sent by the Web Chat application to the Web Interaction REST Service. When the customer stops typing, a “typing” request with content “false” must be sent. In order to avoid spam of “typing” requests, it is recommended to have a timer of 4 seconds after the customer typed the last character before reporting that typing stopped.

The typing status of the agent is sent in the content polling response. The agent has status typing set to “”true” for 4 seconds when a customer types something.

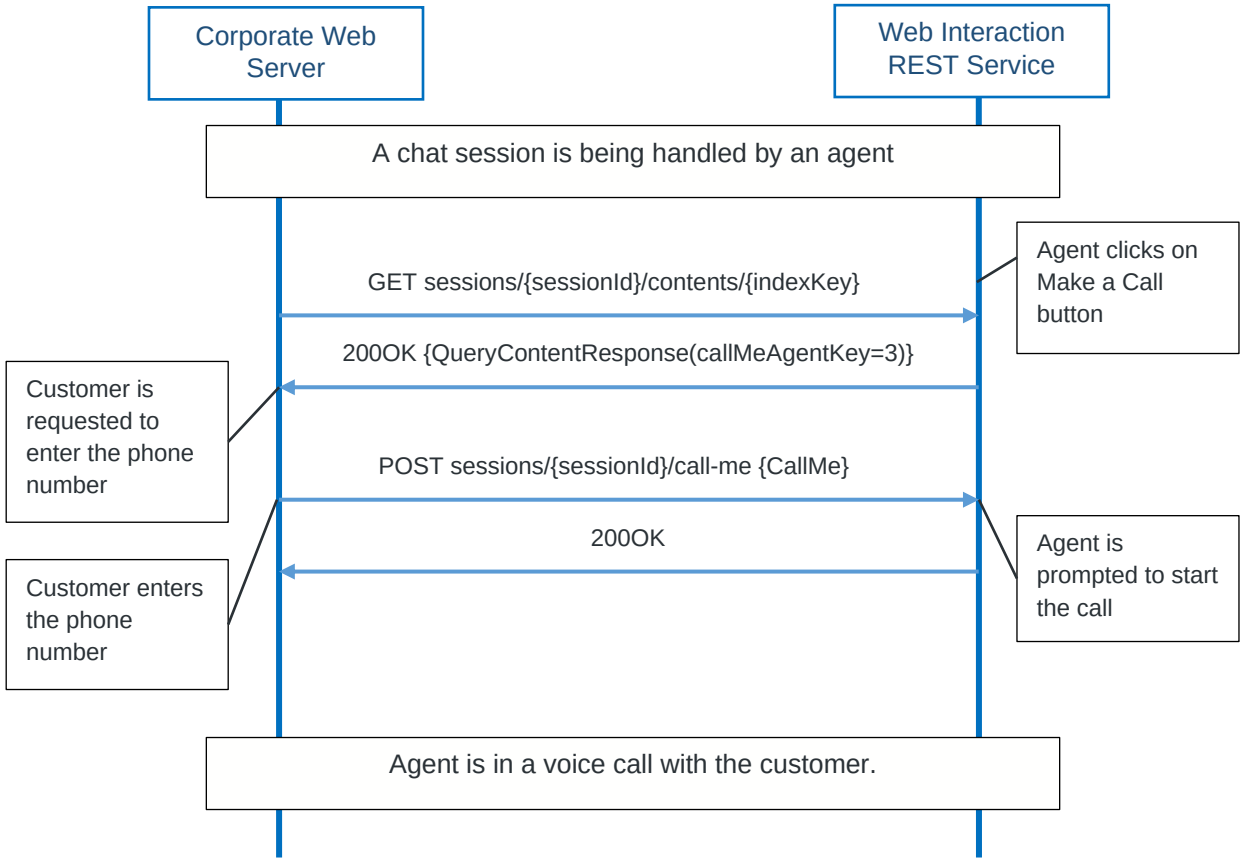


3.3.8 Call-me Function

The Call-me function allows that a voice call is established during the Web Chat session.

The procedure to start a voice call during a Web Chat session is the following:

- The Agent clicks on the “Make a Call” button.
- The Web Chat application will receive a Content response with the field callMeAgentKey set to the key of the agent.
- The Web Chat application shall prompt the customer to enter the number to be dialed, preferably in E.164 format as for example either +554180911597 or +55 (41) 80911597.
- The entered number shall be sent to the server in a Call-me request.
- A screen pop will be presented to the agent with the number provided by the customer.
- The customer clicks on OK to start the voice call.

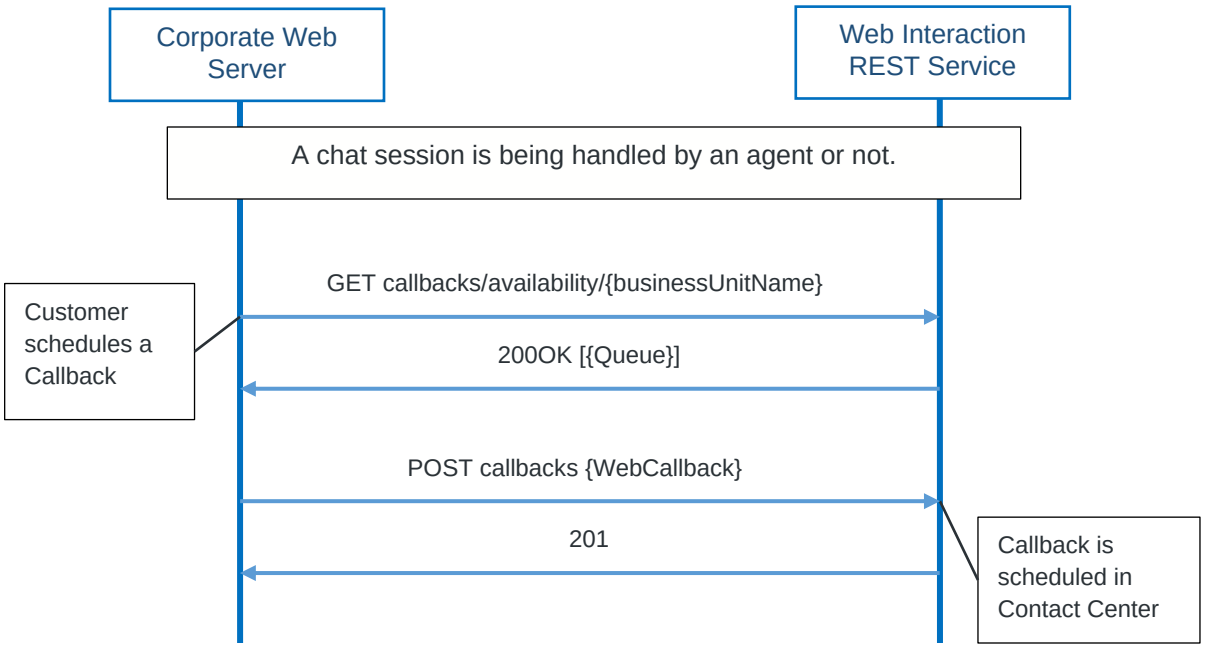


-

3.3.9 Web Callback

The customer shall be able to schedule a Web Callback from the Corporate Web Page. The customer can indicate up to two-time intervals for the execution of the callback.

The Web Callback application can send a request to the Web Interaction REST Service for the queues which are eligible to handle callbacks.



-

3.4 HTTP Messages

All Web Interaction REST HTTP requests need to pass the Token in the Authorization Header.

The Registration does not require this Authorization in the header because the session was not established yet.

Sample HTTP Token Request with body:

```
POST /restinteractionsdk/oauth/token
Host: 10.80.199.15
Accept: application/json, text/plain, */*
Authorization: Basic YXBwbGljWXRpb24xOmiY2QxMjM0
Content-Type: application/x-www-form-urlencoded
Set-Fetch-Mode: cors
Set-Fetch-Site: same-origin
grant_type=client_credentials
```

Note: The header "Authorization" is set with a Basic authentication which is filled as "Basic <base64Encode(client-id:client-secret)>"

A Token will be received in the Token Response as below:

Sample HTTP Token Response:

```
HTTP/1.1 200
Cache-Control: no-store
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Date: Thu, 18 Oct 2018 12:38:58 GMT
{
  "access_token" : "f95612a9-17a2—412a-8684-e7b9a14f9c04",
  "token_type" : "bearer",
  "scope" : "all"
}
```

The HTTP message for an HTTP Request is as follows:

Sample request:

```
POST /restinteractionsdk/v1/sessions
Host: 10.80.199.15
Content-Type: application/json; charset=utf-8
authorization: bearer f95612a9-17a2—412a-8684-e7b9a14f9c04
Set-Fetch-Mode: cors
Set-Fetch-Site: same-origin
User-Agent: …
{JSON Content}
```

4 Messages

4.1 OAuth Commands

4.1.1 Token

Description: request to register a Web Interaction application. The Web Interaction REST Service answers with a session token which must be used on subsequent requests (authorization header).
Note: In order to register, the client must provide a valid client id and client secret. The client id and client secret are configured in the webinteractionsdk.xml file.
URL: POST /oauth/token

Token Response Object		
Attribute name	**Attribute type**	**Description**
access_token	String	It contains the session token which shall be used in all further application requests (authorization header).
token_type	String	Type of the token. In this case always “bearer”.
scope	String	Scope value. In this case always “all”.

Message Example

```
//Token Response
{
  "access_token" : "f95612a9-17a2—412a-8684-e7b9a14f9c04",
  "token_type" : "bearer",
  "scope" : "all"
}
```

Note: The grant type must be added to the body of the message.

4.2 Session Management Commands

4.2.1 Initiate Session

Description: request to start a new Web Interaction session. In a multi-tenant system, the session request will indicate the Business Unit for which the session is being started. The response will contain the session id to identify the session in any further request which is sent to the server.
URL: POST /v1/sessions

RequestSession Object			
Attribute name	**Attribute type**	**Required/ optional**	**Description**
businessUnitName	String	Required	Target Business Unit Name.
caption	String	Required	Initial message that the customer is sending.
contactData	Array of ContactData	Optional	Array of additional Key/Value contact data.
customerName	String	Required	Customer name.
destination	String	Optional	The Destination can be used to make Routing decisions and appears on the Agent Portal contact details. It must be an ASCII character excluding special characters like exclamation mark, quotation mark, percent sign, asterisk, comma, underscore, grave accent and vertical bar (pipe).

language	String	Required	The Name of the language that shall be used for this session. You can add new languages on the OpenScape Contact Center Manager.
priority	Integer	Required	The priority determines how this contact is scored compared to other contacts. The higher the priority of the contact relative to other contacts, the higher the importance of the contact. Priority can range from 1 to 100.
source	String	Required	The Source can be used to make Routing decisions and appears on the Agent Portal contact details. The Source field is also used on the 360 feature, to identify further contacts for the same customer. It must be an ASCII character excluding special characters like exclamation mark, quotation mark, percent sign, asterisk, comma, underscore, grave accent and vertical bar (pipe).

Message Example

```
//POST Sessions Request
{
  "businessUnitName" : "Company1",
  "caption" : "Can you help me with my problem?",
  "contactData" : [
    {
      "key" : "Address",
      "value" : "Street"
    },
    {
      "key" : "Phone Number",
      "value" : "+55 (41) 99999-9999"
    }],
  "customerName" : "Jose Silva",
  "destination" : "Sales",
  "language" : "English",
  "priority" : 5,
  "source" : "jose.silva@email.com"
}
```

RequestSessionResponse Object		
Attribute name	**Attribute type**	**Description**
sessionId	String	The Web Interaction Session ID.

Message Example

```
//Sessions Response
{
  "sessionId" : "W16612020000 7092418001dac3e753c0c141268d153bfac7c5f69e"
}
```

Possible Response Codes:

POST Session Response Codes		
HTTP Code	**Description**	**Schema**
201	Session successfully started	RequestSessionResponse
400	One or more required fields are invalid or empty.	No content
401	The authentication was not successful.	No content
403	The Web Interaction feature is not licensed.	No content

500	The server could not process the request.	No content
503	The Web Interaction server has reached the maximum allowed sessions	No content

4.2.2 Disconnect Session

Description: Request to disconnect a Web Interaction Session.
URL: DELETE /v1/sessions

Sessions Request Object			
Attribute name	**Attribute type**	**Required/ optional**	**Description**
sessionId	String	Required.	The Session ID of the Web Interaction to be disconnected.

Message Example

```
//DELETE Sessions Request
{
  "sessionId" : "W16612020000 7092418001dac3e753c0c141268d153bfac7c5f69e"
}
```

Possible Response Codes:

DELETE Session Responses		
HTTP Code	**Description**	**Schema**
200	Session successfully started	No content
400	One or more required fields are invalid or empty.	No content
401	The authentication was not successful.	No content
403	The Web Interaction feature is not licensed.	No content
404	The Web Interaction session could not be found.	No content
500	The server could not process the request.	No content

4.3 Web Chat Messaging Commands

4.3.1 Query Contents

Description: Queries a session for contents. This command is used to get the following content changes from the agent:

- Messages from the agent.
- The agent started/stopped typing.
- The agent requested a voice call.
- The agent pushed a URL.
- The agent redirected a URL.

There are two possible ways to query a session for new messages:
1.Query all contents - This query is equivalent to querying for 'indexKey' 0. Each content has an indexKey, starting from 1. So, given a session with contents A, B and C:

- 1 -> A
- 2 -> B
- 3 -> C

This query will return A, B and C with 'newIndexKey' 4. A chat application will first make a query for all contents, then start polling with the received 'newIndexKey'. When a new content is received, the query will return all contents and a new 'newIndexKey'. If the session is disconnected, it will remain available for queries for 5 minutes. After this period, the session will be completely removed from the system and further request will return a 404 HTTP Status Code.
URL: GET /v1/sessions/{sessionId}/contents
2. Queries a session for the contents starting at the given indexKey. Each content has an indexKey, starting from 1.
So, given a session with contents A, B and C:

- 1 -> A
- 2 -> B
- 3 -> C

A query for indexKey 2, will return the content items including the indexKey 2 up to the last one: B and C. The 'newIndexKey' will be 4. A chat application will first make a query for all contents, then start polling with the received 'newIndexKey'. When a new content is received, the query will return the new contents and a new 'newIndexKey'. If there were no new contents, the 'newIndexKey' will be the same and content array will be empty. If the session is disconnected, the content items of the session will remain available for queries for 5 minutes. After this period, the session is completely removed from the system and further request will return a 404 HTTP Status Code.
URL: GET /v1/sessions/{sessionId}/contents/{indexKey}

QueryContentResponse Object			
Attribute name	**Attribute type**	**Required/ optional**	**Description**
callMeAgentKey	Integer	Optional	If an agent sends a call me notification, this will contain the agent’s key, otherwise it will be 0.
connected	Boolean	Optional	True if the session is currently connected. After the session is disconnected it will stay available for queries for 5 minutes.
content	Array of Content	Optional	An array of Content, which describes each message. If the index given does not exist or the server is sending a redirectUrl, this content will be null.
newIndexKey	Integer	Optional	The indexKey the next received content will have. This field will remain the same until a new content is received. If the 'newIndexKey' remains the same between two queries, it means no new content was received. For example, given a session with two contents, the 'newIndexKey' will be 3.
pushUrl	String	Optional	URL pushed by the server. It will be empty if no url was pushed since the last content query.
redirectUrl	String	Optional	If the session creation workflow pointed to a redirection, this field will contain the redirect url.
sessionId	String	Required	The Web Interaction Session ID.
whoIsTyping	String	Optional	A string indicating that an agent is typing at the request time. The field will contain the identification of the agent who is typing. If more than one agent are typing at the same time, each agent will be separated by a comma.

Content Object			
Attribute name	**Attribute type**	**Required/ optional**	**Description**
messageType	String	Required	The type of the message.
from	String	Required	It identifies the element which sent this message. If the message is from a customer, it contains the customer name. If the message is from an agent, it contains the agent key If the

			message is from the system, it contains the configured System name.
agentFirstName	String	Required	If the message comes from an agent, it contains the agent first name
agentLastName	String	Required	If the message comes from an agent, it contains the agent last name
content	String	Required	The content of the message. Note: if the message was sent by the agent, the font of the content is formatted according to the configuration in Manager.
formattedContent	String	Required	This is the legacy content field which contains an HTML formatted message.
pushedUrl	String	Required	It contains the link of the pushed URL.

The following Message Types are possible:

CUSTOMER_MESSAGE,
CUSTOMER_DISCONNECT,
AGENT_MESSAGE,
AGENT_DISCONNECT,
AGENT_PUSH_URL,
AGENT_REQUEUE,
AGENT_CONFERENCE_JOIN,
SYSTEM_MESSAGE,
SYSTEM_PUSH_URL

Message Examples:

```
//GET All Contents Request
{
  "sessionId": "W16612020000711194600417f5ec2b2da343449daf1698a9f5f21e",
  "content": [
    {
      "messageType": "CUSTOMER_MESSAGE",
      "from": "Firstname Lastname",
      "agentFirstName": "",
      "agentLastName": "",
      "content": "Can you help me with...",
      "formattedContent": "<TABLE width='95%'  border='0' cellspacing='0'
cellpadding='2'><TR><TD width='2%' valign='top'><IMG SRC='images/customericon.gif'
WIDTH='16' HEIGHT='16'></TD><TD width='98%'><FONT size=2 face=Tahoma><FONT tabIndex=-1
contentEditable=false style=\"BACKGROUND-COLOR: #ffffff\" size=2
face=Tahoma>07/01/2020</FONT>, <FONT tabIndex=-1 contentEditable=false
style=\"BACKGROUND-COLOR: #ffffff\" size=2 face=Tahoma>11:19 AM</FONT> <FONT
tabIndex=-1 contentEditable=false style=\"BACKGROUND-COLOR: #ffffff\" size=2
face=Tahoma><STRONG>Firstname Lastname</STRONG></FONT><STRONG>
says</STRONG></FONT></TD></TR><TR><TD> </TD><TD valign='top'><FONT face=Tahoma
size=2>Can you help me with...</FONT></TD></TR></TABLE>",
      "pushedUrl": ""
    },
    {
      "messageType": "SYSTEM_MESSAGE",
      "from": "<FONT size=2 face=Tahoma>System</FONT>",
      "agentFirstName": "",
      "agentLastName": "",
      "content": "<FONT face=Tahoma size=2>Welcome to the contact center, powered by
OpenScape Contact Center.  Please hold while we connect you to an agent.</FONT>",
      "formattedContent": "<TABLE width='95%'  border='0' cellspacing='0'
cellpadding='2'><TR><TD width='2%' valign='top'><IMG SRC='images/systemicon.gif'
WIDTH='16' HEIGHT='16'></TD><TD width='98%'><FONT size=2 face=Tahoma><FONT tabIndex=-1
contentEditable=false style=\"BACKGROUND-COLOR: #ffffff\" size=2
```

```
face=Tahoma>07/01/2020</FONT>, <FONT tabIndex=-1 contentEditable=false
style=\"BACKGROUND-COLOR: #ffffff\" size=2 face=Tahoma>11:19
AM</FONT> Agent<STRONG> says</STRONG></FONT></TD></TR><TR><TD> </TD><TD
valign='top'><FONT face=Tahoma size=2>Welcome to the contact center, powered by
OpenScape Contact Center.  Please hold while we connect you to an
agent.</FONT></TD></TR></TABLE>",
      "pushedUrl": ""
    },
    {
      "messageType": "AGENT_MESSAGE",
      "from": "14",
      "agentFirstName": "Claudio",
      "agentLastName": "Bastos",
      "content": "<FONT size=2 face=Tahoma><FONT tabIndex=-1 contentEditable=false
style=\"BACKGROUND-COLOR: #ffffff\" size=2 face=Tahoma>07/01/2020</FONT>, <FONT
tabIndex=-1 contentEditable=false style=\"BACKGROUND-COLOR: #ffffff\" size=2
face=Tahoma>11:19 AM</FONT> How can I help you?</FONT>",
      "formattedContent": "<TABLE width='95%'  border='0' cellspacing='0'
cellpadding='2'><TR><TD width='2%' valign='top'><IMG SRC='images/agenticon.gif'
WIDTH='16' HEIGHT='16'></TD><TD width='98%'><FONT size=2 face=Tahoma><FONT tabIndex=-1
contentEditable=false style=\"BACKGROUND-COLOR: #ffffff\" size=2
face=Tahoma>07/01/2020</FONT>, <FONT tabIndex=-1 contentEditable=false
style=\"BACKGROUND-COLOR: #ffffff\" size=2 face=Tahoma>11:19 AM</FONT> How can I help
you?</FONT></TD></TR></TABLE>",
      "pushedUrl": ""
    }
  ],
  "pushUrl": "",
  "callMeAgentKey": 0,
  "newIndexKey": 4,
  "whoIsTyping": "",
  "redirectUrl": "",
  "connected": true
}

//GET Contents Request from an Index Key
{
  "sessionId": "W16612020000714233200051f96bd33ac56477882219babd0f7839b",
  "content": [
    {
      "messageType": "AGENT_MESSAGE",
      "from": "14",
      "agentFirstName": "Claudio",
      "agentLastName": "Bastos",
      "content": "<p style=\"margin: 0\">How are you doing?</p>",
      "formattedContent": "<TABLE width='95%'  border='0' cellspacing='0'
cellpadding='2'><TR><TD width='2%' valign='top'><IMG SRC='images/agenticon.gif'
WIDTH='16' HEIGHT='16'></TD><TD width='98%'><FONT size=2 face=Tahoma><FONT tabIndex=-1
contentEditable=false style=\"BACKGROUND-COLOR: #ffffff\" size=2
face=Tahoma>07/01/2020</FONT>, <FONT tabIndex=-1 contentEditable=false
style=\"BACKGROUND-COLOR: #ffffff\" size=2 face=Tahoma>2:24
PM</FONT> Agent<STRONG> says</STRONG></FONT></TD></TR><TR><TD> </TD><TD
valign='top'><p style=\"margin: 0\">How are you doing?</p></TD></TR></TABLE>",
      "pushedUrl": ""
    }
  ],
  "pushUrl": "",
  "callMeAgentKey": 0,
  "newIndexKey": 5,
  "whoIsTyping": "",
  "redirectUrl": "",
```

```
  "connected": true
}
```

Possible Response Codes:

GET Contents Response Codes		
HTTP Code	**Description**	**Schema**
200	Successfully queried the content.	QueryContentResponse
400	One or more required fields are invalid or empty.	No content
401	The authentication was not successful.	No content
403	The Web Interaction feature is not licensed.	No content
404	The Web Interaction session could not be found.	No content
500	The server could not process the request.	No content
503	The Web Interaction server has reached the maximum allowed sessions	No content

4.3.2 Add Content

Description: Request to add new content to the Web Interaction session.
URL: POST /v1/sessions/{sessionId}/contents

AddContent Object			
Attribute name	**Attribute type**	**Required/ optional**	**Description**
content	String	Required	Content to be added to the session.

Message Example

```
//POST Contents Request
{
  "content": "Hey, can you help me?"
}
```

Possible Response Codes:

POST Contents Response Codes		
HTTP Code	**Description**	**Schema**
201	Message successfully added to the session.	No content
400	One or more required fields are invalid or empty.	No content
401	The authentication was not successful.	No content
403	The Web Interaction feature is not licensed.	No content
404	The Web Interaction session could not be found.	No content
500	The server could not process the request.	No content
503	The session has already been disconnected.	No content

4.3.3 Typing

Description: Request to inform the agent(s) that the customer is typing.
URL: POST /v1/sessions/{sessionId}/typing

NotifyTyping **Object**			
Attribute name	**Attribute type**	**Required/ optional**	**Description**
isTyping	Boolean	Required	If 'true' is sent, the server will consider that customer is typing until it receives either a new request with isTyping set to 'false' or a new message.

Message Example

```
//POST Typing Request
{
  "isTyping": "true"
}
```

Possible Response Codes:

Typing Response Codes		
HTTP Code	**Description**	**Schema**
200	Successfully sent Typing notify.	No content
400	One or more required fields are invalid or empty.	No content
401	The authentication was not successful.	No content
403	The Web Interaction feature is not licensed.	No content
404	The Web Interaction session could not be found.	No content
500	The server could not process the request.	No content

4.3.4 Call-Me

Description: Sends a phone number and the agent key to the specified agent. This data allows the agent to call the customer number. The agent key of the agent which started the Call-Me request can be got by performing a content query. Example:
1) Agent starts a "Make a call" request.
2) For the next Contents query request which is sent by the application, the agent key will appear on the callMeAgentKey field.
3) Subsequent Query Content will return to have callMeAgentKey as 0.
4) The customer sends its phone number through the Call Me request.
URL: POST /v1/sessions/{sessionId}/call-me

CallMe Object			
Attribute name	**Attribute type**	**Required/ optional**	**Description**
agentKey	Integer	Required	The agent that will receive the phone number.
phoneNumber	String	Required	The phone which shall be used to dial to the customer.

Message Example

```
//POST Call-Me Request
{
  "agentKey": 14,
  "phoneNumber": "554180911597"
}
```

Possible Response Codes:

Call-Me Response Codes		
HTTP Code	**Description**	**Schema**
200	Successfully sent phone number.	No content
400	One or more required fields are invalid or empty.	No content
401	The authentication was not successful.	No content
403	The Web Interaction feature is not licensed.	No content
404	The Web Interaction session could not be found.	No content
500	The server could not process the request.	No content

4.4 Web Callback Commands

4.4.1 Create Web Callback

Description: Request the creation of a new Callback to the server.

.

URL: POST /v1/callbacks

WebCallback Request Object			
Attribute name	**Attribute type**	**Required/ optional**	**Description**
callbackQueueKey	Integer	Required	The queue's key that the callback will be inserted. The callback queue key is given by the Check Callback Availability request.
comment	String	Optional	Additional information to be included with the Web Callback.
contactData	Array of ContactData	Optional	Additional contact data.
contactName	String	Required	The contact name
firstAttemptEndTime	String	Required	The end time for the first callback attempt. The date must be in the UTC timezone and in ISO-8601 Date and Time format. The date timezone offset must be omitted.
firstAttemptStartTime	String	Required	The start time for the first callback attempt. The date must be in the UTC timezone and in ISO-8601 Date and Time format. The date timezone offset must be omitted.
phoneNumber	String	Required	The Web Callback phone number which shall be dialed.
priority	Integer	Required	The priority determines how this contact is scored compared to other contacts. The higher the priority of the contact relative to other contacts, the higher the importance of the contact. Priority can range from 1 to 100.
secondAttemptEndTime	String	Optional	The end time for the second callback attempt. The date must be in the UTC timezone and in ISO-8601 Date and Time format. The date timezone offset must be omitted.
secondAttemptStartTime	String	Optional	The start time for the second callback attempt. The date must be in the UTC timezone and in ISO-8601 Date and Time format. The date timezone offset must be omitted.

Message Examples:

```
//POST Web Callback Request
{
  "callbackQueueKey": 3,
  "comment": "Call to negotiate contract.",
  "contactData": [
    {
      "key" : "Address",
      "value" : "Street"
    },
    {
      "key" : "Phone Number",
      "value" : "+55 (41) 99999-9999"
    }
  ],
  "contactName": "Jose Silva",
  "firstAttemptEndTime": "2020-01-20T12:00",
  "firstAttemptStartTime ": "2020-01-20T08:00",
  "phoneNumber": "554180911597"
  "priority": 10,
  "secondAttemptEndTime": "2020-01-21T12:00",
  "secondAttemptStartTime ": "2020-01-21T08:00"
}
```

Possible Response Codes:

Web Callback Response Codes		
HTTP Code	**Description**	**Schema**
201	Successfully created the Web Callback.	No content
400	One or more required fields are invalid or empty.	No content
401	The authentication was not successful.	No content
403	The Web Interaction feature is not licensed.	No content
500	The server could not process the request.	No content
503	It failed to create a Web Callback.	No content

4.4.2 Check Web Callback

Description: Checks if the Web Callback feature is currently enabled. If the feature is enabled, an array with the possible callback queues is returned.
URL: GET /v1/callbacks/availability/{businessUnitName}

CheckWebCallback Response Object			
Attribute name	**Attribute type**	**Required/ optional**	**Description**
	Array of Queue	Required	The list of the queues which are eligible for callbacks. Note: the list will be empty if there is no eligible queue for callback.

Queue Object

Attribute name	Attribute type	Required/ optional	Description
key	Integer	Required	The key of the queue.
name	String	Required	The name of the queue.
description	String	Required	The description of the queue.

Message Examples:

```
//POST Web Callback Request
[
  {
  "key": 9,
  "name": "Default Callback Queue",
  "description": "Default callback queue"
  },
  {
  "key": 10,
  "name": "Sales Callback Queue",
  "description": "Sales callback queue"
  }
]
```

Possible Response Codes:

Check Web Callback Response Codes		
HTTP Code	**Description**	**Schema**
200	The Web Callback feature is enabled.	No content
400	One or more required fields are invalid or empty.	No content
401	The authentication was not successful.	No content
403	The Web Interaction feature is not licensed.	No content
500	The server could not process the request.	No content
503	The Web Callback feature is disabled.	No content

5 Click to Contact Component

5.1 About

The Click to Contact Component is a Web Component made with lit-element from the polymer-project (https://lit-element.polymer-project.org/guide/start).

The component allows a web page to have the WebRTC functionality, letting users dial to a configured pilot number directly from the browser. The component also allows the user to share their video or screen, as well as receiving the agent's video or screen (in case the agent is on an Integrated Phone). For more information about implementation and configuration, please see the document WebRTC, Description Guide.

5.2 Customization

The Click-To-Dial component is highly customizable, allowing a complete change of the component's texts, styles and other relevant properties.

5.2.1 Modify Texts

The component has been developed by default with English texts. Those texts can be modified or translated into another language. The customization is done by adding the corresponding attributes to the tag “<oscc-click-to-dial>”. In the example below the header text of the component is changed from default “Call Us” to “Click here to call us”:

```
<oscc-click-to-dial
       ...
       headerLabel=”Click here to call us”
></oscc-click-to-dial>
```

The tables below show the default texts, which can be modified.

Header	
Attribute name	**Description**
headerLabel	Call Us
notSupportedError	Open this website on Google Chrome to call us

Instructions State	
Attribute name	**Description**
instructionsLabel	Instructions
line1Label	Ensure that you have a good internet connection
line2Label	Ensure that you have both audio and video devices plugged into your computer
line3Label	Allow application permission in browser to use audio and video devices when requested.
continueLabel	Continue

Input State	
Attribute name	**Description**
inputStateTitle	Please input your number to continue
howCanWeHelpLabel	How can we help you?

Device Selection State	
Attribute name	**Description**
deviceSelectionTitleLabel	Device Selection
audioInputLabel	Audio input device
audioOutputLabel	Audio output device
videoInputLabel	Video input device
dialUsLabel	Dial Us
backLabel	Back

Dial State	
Attribute name	**Description**
ringingLabel	Ringing...
disconnectLabel	Disconnect
disconnectedLabel	Disconnected...
callAgainUsLabel	Make another call

Tooltips	
Attribute name	**Description**
cameraTooltip	Camera
screenshareTooltip	Screen share
muteTooltip	Mute
unmuteTooltip	Unmute
clickToExpandTooltip	Click to Expand
fitToScreenTooltip	Fit To Screen
zoomOutTooltip	Zoom Out
zoomInTooltip	Zoom In
fullScreenTooltip	Fullscreen
keypadTooltip	Keypad

Errors	
Attribute name	**Description**
videoFailedError	Failed to start video
screenShareFailedError	Failed to start screenshare
callFailedError	Failed to start call
noAudioPermissionError	Microphone permissions are required. Please allow the page to access your device
noVideoPermissionError	Failed to get video permissions

5.2.2 Pre-configure user selections

Some user selections in the component can be hidden and the associated functionality can be given a value by configuration. The customization is done by adding the corresponding attributes to the tag “<oscc-click-to-dial>”. E.g. if the user should not select a destination, because all calls should go to a single destination, then the “How can we help” label can be hidden and that single destination can be configured:

```
<oscc-click-to-dial

       ...

       destinationKey=”...name...”

></oscc-click-to-dial>
```

The name of the destination must be the same as configured in the Application Server Configuration Center**:** Tab **“Webinteraction SDK” > Integrated Phone > Destinations > Key**, which is associated there with the Pilot Number.

The table below shows the user selections, which can be hidden and pre-configured.

Selection	
Attribute name	**Description**
customerNumber	The telephone number associated with label “Please input your number to continue”
destinationKey	The destination associated with label “How can we help you?”

5.2.3 Modify Styles

The component has been developed with default CSS styles (fonts, colors, etc.). Those styles can be modified by configuring CSS variables with the desired CSS values. The customization is done by adding those CSS variables in the section “<head>” by the tag “<style>” of the web page. The scope of the variables should be global, i.e. “body” or “:root”. E.g. the background color of the component’s header can be changed from default green to blue:

```
<style>

	body {

		--oscc-ctd-header-background-color: blue;

	}

</style>
```

The tables below show the styles, which can be modified. The last part of each CSS variable name shows the standard CSS property, e.g. “font-family” which defines the possible CSS value syntax, e.g. “Verdana, Arial”.

General	
Attribute name	**Description**
--oscc-ctd-font-family	Component font
--oscc-ctd-max-width	Maximum width of the component
--oscc-ctd-max-height	Maximum height of the component

Header variables	
Attribute name	**Description**
--oscc-ctd-header-background-color	The color of the header
--oscc-ctd-header-opacity	The opacity of the header
--oscc-ctd-header-font-color	The color of the text in the header
--oscc-ctd-header-font-size	The size of the text in the header

Component's body variables	
Attribute name	**Description**
--oscc-ctd-body-background-color	The color of the component's body background
--oscc-ctd-body-opacity	The opacity of the component's body
--oscc-ctd-font-color	The color of the text in the component's body

Buttons - Extra indentation: Secondary button/Cancel button	
Attribute name	**Description**
--oscc-ctd-btn-background	Button's background
--oscc-ctd-btn-sec-background	
--oscc-ctd-btn-background-active	Button's background when the pseudo class :active is set
--oscc-ctd-btn-sec-background-active	
--oscc-ctd-btn-hover-background-color	Button's background when the pseudo class :hover is set
--oscc-ctd-btn-sec-background-hover	

--oscc-ctd-btn-font-color	The color of the text in buttons
--oscc-ctd-btn-sec-font-color	
--oscc-ctd-btn-border	The border of the buttons
--oscc-ctd-btn-sec-border	
--oscc-ctd-btn-box-shadow	The box shadow of the buttons
--oscc-ctd-btn-sec-box-shadow	
--oscc-ctd-btn-box-shadow-focus	The box shadow of the buttons when the button is focused
--oscc-ctd-btn-transition	The transition property of the button

Tooltips	
Attribute name	**Description**
--oscc-ctd-tooltip-background-color	The background color of tooltips
--oscc-ctd-tooltip-font-color	The color of the text in tooltips

Video Element	
Attribute name	**Description**
--oscc-ctd-video-background	The background color for the incoming video

Video Control Bar	
Attribute name	**Description**
--oscc-ctd-video-controls-container-background	The background color for the video controller bar on call screen

Error Banner	
Attribute name	**Description**
--oscc-ctd-error-background-color	The background color of the error banner
--oscc-ctd-error-box-shadow	The box shadow of the error banner
--oscc-ctd-error-font-color	The color of the text in the error banner

Component not supported	
Attribute name	**Description**
--oscc-ctd-not-supported-font-size	The size of the text in the header when the component is not supported

Keypad	
Attribute name	**Description**
--oscc-ctd-keypad-background-color	Background color of the keypad
--oscc-ctd-keypad-border	Border element of the keypad
--oscc-ctd-keypad-font-color	The color of the keypad's digits
--oscc-ctd-keypad-font-size	The font size of the keypad's digits
--oscc-ctd-keypad-height	The height of the keypad dialog
--oscc-ctd-keypad-width	The width of the keypad dialog
--oscc-ctd-keypad-font-color-hover	The color when keypad's digits are hovered (pseudo class :hover is set)