



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

## Service Client API Reference

# Table of Contents

<b>Get started</b>	
Service Client API	4
<b>Namespaces</b>	
Agent namespace	21
Configuration namespace	25
Email namespace	34
Interaction namespace	36
Media namespace	63
System namespace	68
Voice namespace	87
Outbound namespace	97
Auth Namespace	116
Messenger namespace	119

---

Search the table of all articles in this guide, listed in alphabetical order, to find the article you need.

# Service Client API

## Contents

- [1 Getting started](#)
- [2 Security configuration](#)
  - [2.1 Origin](#)
  - [2.2 Rate Limit](#)
  - [2.3 Attached Data Access](#)
- [3 Working with the API](#)
  - [3.1 Notifications](#)
- [4 Event Type references](#)
  - [4.1 Outbound events](#)
- [5 Common actions with Service Client API](#)
  - [5.1 Controlling call recording from a third-party application](#)
  - [5.2 Embedding multiple third-party applications in Agent Workspace](#)
  - [5.3 Updating attached data from a third-party application](#)
  - [5.4 Enabling click-to-dial from a third-party application](#)
  - [5.5 Enabling Service Client API to invoke toast in Agent Workspace](#)
  - [5.6 Controlling case selection from a third-party application](#)
  - [5.7 Supporting multiple browser tabs](#)

Learn how to use the Service Client API to customize the way your web application integrates with Agent Workspace.

### Important

Depending on your environment, you might need to contact your Genesys representative to complete the configuration described on this page.

Use the Service Client API to customize how your web application or website integrates with Agent Workspace. This JavaScript API is based on `window.postMessage` and provides methods your application can use to communicate cross domain with Agent Workspace while maintaining secured isolation.

## Getting started

Here's an overview of the steps to access the API:

1. You have a web application that you've integrated in Agent Workspace. See [Enabling integration of web applications in the agent interface](#).
2. Download the sample application from [GitHub](#).
3. Copy the **www-service-client-api.js** file in the sample application to a location your web application can access.
4. Set configuration options related to security. See [Security configuration](#).
5. Review [Working with the API](#) for more information about how to use the API.
6. Review the methods and types available in each namespace:
  - Agent Namespace
  - Configuration Namespace
  - Email Namespace
  - Interaction Namespace
  - Media Namespace
  - System Namespace
  - Voice Namespace
  - Outbound Namespace
  - Auth Namespace
  - Messenger Namespace

7. See Common actions with Service Client API for ideas about how to use the API.

## Security configuration

The Service Client API involves two parties inside the agent's web browser: the service (the main web page) and the client (in an iframe on the same web page as the service). In order for the client web page to access the API, you need to set a few configuration options to work around web browser security restrictions for cross-origin requests and to enable request limits. You set these options on the **WWEWS Cluster** application only at the Application level; you can't set these options at the Agent or Agent Group level. Check out the Enabling the Service Client API topic in the *Workspace Web Edition Configuration Guide* for a full list of the options available to configure the API.

### Origin

First, to work around web browser security restrictions set the `service-client-api.accepted-web-content-origins` option to the domain you want to be able to access to the API. For example, if you want to give access to a web page located at `http://my-web-server/path/page.html`, then you would set **`service-client-api.accepted-web-content-origins`** to `http://my-web-server`.

If you have several pages that need access to the API and they're located at different domains, you can also provide **`service-client-api.accepted-web-content-origins`** with a list. For example: `http://my-web-server, http://my-second-web-server, http://my-third-web-server`.

Finally, if you want to allow *any* page to access the API, just set **`service-client-api.accepted-web-content-origins`** to `*`.

You can also set the **`service-client-api.accepted-web-content-origins`** option to values that filter by API request, using any of the following keywords:

- `agent.get`
- `agent.getState`
- `agent.getStateList`
- `agent.setState`
- `email.create`
- `interaction.deleteUserData`
- `interaction.getByInteractionId`
- `interaction.getInteractions`
- `interaction.selectCaseByCaseId`
- `interaction.setUserData`
- `interaction.singleStepTransfer(interactionId, targetQuery, userData, extensions, succeeded, failed)`
- `interaction.singleStepConference(interactionId, targetQuery, userData, extensions, succeeded, failed)`
- `interaction.consult(interactionId, targetQuery, userData, extensions, succeeded, failed)`
- `interaction.completeTransfer(consultInteractionId, succeeded, failed)`

- `interaction.completeConference(consultInteractionId, succeeded, failed)`
- `media.getMediaList`
- `media.setState`
- `voice.dial`
- `voice.dialEx(destination, userData, extensions, succeeded, failed)`
- `voice.pauseCallRecording`
- `voice.resumeCallRecording`
- `voice.startCallRecording`
- `voice.stopCallRecording`

For example, you could set **service-client-api.accepted-web-content-origins** to `http://my-web-server0`, `http://my-web-server1 (*)`, `http://my-web-server2 (agent.*, voice.dial)`, `http://my-web-server3 (agent.*, interaction.*)`. In this example, everything is allowed for the `http://my-web-server0` and `http://my-web-server1`. For the `http://my-web-server2` domain, only the `agent.get`, `agent.getStateList`, `agent.setState`, `agent.getState` and `voice.dial` requests are allowed.

As seen in the example above, you can also filter by wildcards, using the asterisk in parenthesis. For example, `http://my-web-server1 (*)` or `http://my-web-server3 (agent.*, interaction.*)`.

## Rate Limit

You can limit the maximum number of requests per minute on any Service Client API request by setting the `service-client-api.rate-limit` option. For example, setting the value to 50 would restrict the number of requests to 50 per minute. Set the value to 0 for unlimited requests.

If you want to limit the maximum number of requests per minute on a particular Service Client API request, use `service-client-api.rate-limit.`.

Consider the following sample configuration:

```
service-client-api.rate-limit=0
service-client-api.rate-limit.voice.dial=4
service-client-api.rate-limit.email.create=2
```

In this example, there are no limits globally, but `voice.dial` requests are limited to 4 requests per minute and `email.create` requests are limited to 2 requests per minute.

Workspace calculates the limitation as a fixed interval of time, each minute (this is not calculated on a costly sliding window).

When the number of requests reaches the limit, Workspace ignores all further requests of the same type for a configurable period of time, known as the quarantine delay. In response, Workspace Web Edition sends a result with an explicit error message to the first request it receives after the limit is reached:

```
{
  "errorMessage": "The rate limit for the request 'voice.dial' has been reached.\nFurther requests of the same type will be ignored for 30 seconds.",
  "request": "agent.getState"
```

```
}
```

To specify the global quarantine delay, set the `service-client-api.rate-limit-quarantine-delay` option. For example, setting the option to 60 means that Workspace Web Edition ignores requests for 60 seconds after the limit is reached. A value of 0 means that Workspace Web Edition ignores further requests forever, so use this value carefully.

## Attached Data Access

Workspace offers two configuration options to limit the read or write access to the key/value pairs in user data:

- `service-client-api.user-data.write-allowed` specifies the list of keys in user data that can be written with the `interaction.setUserData()` or `interaction.deleteUserData()` functions.
- `service-client-api.user-data.read-allowed` specifies the list of keys in user data that can be read. This applies in the `userData` property of the `interaction.deleteUserData()` object returned by a function or an event.

For example, consider the following configuration:

```
service-client-api.user-data.write-allowed=Key1,Key3  
service-client-api.user-data.read-allowed=Key1,Key2,Key3
```

This configuration lets you read the attached data with they keys Key1, Key2, and Key3, but only allows writes on keys Key1, and Key3.

## Working with the API

After you've completed the setup and security steps, you're ready to start working with the Service Client API. The first thing you need to do is add a tag to your web application that points to the **ww-service-client-api.js** file (remember, you stored it somewhere accessible in Step 3 above).

Now you can access the API through the **genesys.ww.service** namespace. For example:

```
Hello world
```

Here's an example of how you could modify attached data:

```
genesys.ww.service.interaction.setUserData(  
  "1",  
  {  
    MyKEY1: "MyValue1",  
    MyKEY2: "MyValue2"  
  }  
)
```

In the above example, the request is `interaction.setUserData` and the parameters are the `interactionId` of 1 and the `keyValues` of `MyKEY1` and `MyKEY2`. All methods provided in the Service Client API are asynchronous, so to get the successful or failed result, just add the matching callback:

```
genesys.wwe.service.interaction.setUserData(
  "1",
  {
    MyKEY1: "MyValue1",
    MyKEY2: "MyValue2"
  },
  function(result){
    console.debug("SUCCEEDED, result: " + JSON.stringify(result, null, '\t'));
  },
  function(result){
    console.debug("FAILED, result: " + JSON.stringify(result, null, '\t'));
  }
)
```

The global template for a service call is:

```
genesys.wwe.service.<(<... function parameters ...>, [, []]);
```

The `done()` callback is called when a request is successfully sent without an error.

The `fail()` callback is called when a request generates an error or an exception.

The result of these functions is provided in a JSON object as a unique parameter.

## Notifications

### Warning

You must call `genesys.wwe.service.subscribe` only once.

You can use the following code to subscribe to **agent** and **interaction** notifications:

```
function eventHandler(message) {
  console.debug("Event: " + JSON.stringify(message, null, '\t'));
}

genesys.wwe.service.subscribe([ "agent", "interaction" ], eventHandler, context);
```

In the above example, `eventHandler` is the event handler function and `context` is an optional contextual object. Here's an example with an agent `STATE_CHANGED` to `Ready`:

```
{
  "event": "agent",
  "data": {
```

```

    "eventType": "STATE_CHANGED",
    "mediaState": "READY"
  }
}

```

Here's an example with an agent `STATE_CHANGED` to Not Ready with a reason:

```

{
  "event": "agent",
  "data": {
    "eventType": "STATE_CHANGED",
    "mediaState": "NOT_READY_ACTION_CODE",
    "reason": "Break",
    "reasonCode": "1511"
  }
}

```

Finally, here's an example with an `ATTACHED_DATA_CHANGED` event on a voice interaction:

```

{
  "event": "interaction",
  "data": {
    "eventType": "ATTACHED_DATA_CHANGED",
    "media": "voice",
    "interaction": {
      "interactionId": "1",
      "caseId": "4ddalab6-aeab-4a33-f5d0-0153c9fdb43b",
      "userData": {
        "IWAttachedDataInformation": {
          "DispositionCode.Label": "DispositionCode",
          "Option.interaction.case-data.header-foreground-
color": "#FFFFFF",
          "CaseDataBusinessAttribute": "CaseData",
          "DispositionCode.Key": "ChooseDisposition",
          "Option.interaction.case-data.frame-color": "#17849D"
        },
        "IW_CaseUid": "4ddalab6-aeab-4a33-f5d0-0153c9fdb43b",
        "IW_BundleUid": "dfaca66c-4149-42a1-7244-337e949a12b5"
      },
      "parties": [
        {
          "name": "5001"
        }
      ],
      "callUuid": "4L6JGNEE9H7DT671FRPTKE6CQ000000G",
      "state": "DIALING",
      "previousState": "UNKNOWN",
      "isConsultation": false,
      "direction": "OUT",
      "callType": "Internal",
      "dnis": "5001",
      "isMainCaseInteraction": true
    }
  }
}

```

## Event Type references

The system eventType field can be one of the following:

eventType	Description
CUSTOM_TOAST_BUTTON_CLICK	<p>Uses the following parameters:</p> <ul style="list-style-type: none"> <li>• <b>customToastId</b>: The identifier of the toast where the button has been clicked. The identifier is returned by the popupToast method.</li> <li>• <b>buttonIndex</b>: The index of the clicked button. The index starts by 0.</li> </ul>
REALTIME_CONNECTION	<p>Uses the following parameters:</p> <ul style="list-style-type: none"> <li>• <b>state</b>: The attribute can take any of the following values: <ul style="list-style-type: none"> <li>• DISCONNECTED - The real-time connection with the Genesys Web Services server is disconnected.</li> <li>• RECONNECTED - The real-time connection with the Genesys Web Services server is established after a disconnection.</li> <li>• DOWN - The real-time connection with the Genesys Web Services server is down for more than one minute due to server inactivity. In this situation, we can consider the session as <i>Down</i>.</li> </ul> </li> </ul>

The interaction eventType field can be one of the following:

eventType	Description
Common events to all interaction types	
UNKNOWN	An unknown event occurs.
ADDED	The interaction has been added in the list of interactions.
REMOVED	The interaction has been removed from the list of interactions.
ATTACHED_DATA_CHANGED	The attached data have changed in the interaction.
CASE_OR_BUNDLE_ID_CHANGED	The case or the bundle identifier of this interaction has changed.
CASE_ID_CHANGED	The case identifier of this interaction has changed.
NEW_MESSAGE	This event represents a new message.
ERROR	An error occurs in the interaction.
CONTACT_CHANGED	A contact associated with the interaction is fully or partially modified.
Voice events	
CALL_RECORDING_STATE_CHANGED	The call recording state changed.

<b>eventType</b>	<b>Description</b>
DIALING	The outbound call starts ringing.
ESTABLISHED	The call has been established.
HELD	The call has been held.
PARTY_CHANGED	The list of party has been changed in the interaction.
RELEASED	The call has been released.
RINGING	The inbound call starts ringing.
OpenMedia events	
ACCEPTED	The open media interaction is accepted.
COMPLETED	The open media interaction has been completed (Mark as done).
COMPOSING	The open media interaction is in composing mode.
CREATED	The open media interaction has been created.
INSERT_STANDARD_RESPONSE	A standard response has been inserted in the interaction.
INVITED	The open media interaction is an invitation.
INVITED_CONFERENCE	The open media interaction receive a conference invitation.
IN_QUEUE_FAILED	The place in queue has failed.
IN_WORKBIN	The interaction has been placed in the work-bin.
IN_WORKBIN_FAILED	The place in work-bin has failed.
LEFT_CONFERENCE	The open media interaction has left the conference.
PULLED	The open media interaction has been pulled from a work-bin.
PULL_FAILED	The pull from the queue has failed.
PULL_WORKBIN_FAILED	The pull from the work-bin has failed.
REVOKED	The open media interaction has been revoked.
TRANSFER_COMPLETED	The open media interaction has been transferred and the transfer has been completed.
Chat events (inherit from OpenMedia events)	
CANCELED	The interaction is already accepted in another chat session.
ENDED	The chat has been ended.
JOIN_FAILED	The connection with the chat server failed.
JOIN_PENDING	The interaction is trying to join the chat session.
Outbound email events (inherit from OpenMedia events)	
CANCELLED	The outbound email has been cancelled.
SENT	The outbound email has been sent.

## Outbound events

The **Outbound preview events** table lists the SCAPI event details for Pull Preview, Push Preview and Direct Push Preview records.

Outbound preview events

Mode	UI Event	Event Type	State	Call Type	Capabilities
Pull Preview	Preview record received	ADDED	PREVIEWING	OUTBOUND_PREVIEW	CALL, REJECT_RECORD, CANCEL_RECORD
		PREVIEWING	PREVIEWING	OUTBOUND_PREVIEW	CALL, REJECT_RECORD, CANCEL_RECORD
	Make call from preview	ADDED	DIALING	OUTBOUND	HANGUP
		DIALING	DIALING	OUTBOUND	HANGUP
		REMOVED	IDLE	OUTBOUND_PREVIEW	
	Release and mark done	RELEASED	IDLE	OUTBOUND	MARK_DONE
		MARKDONE_APPLY	IDLE	OUTBOUND	MARK_DONE
		REMOVED	IDLE	OUTBOUND	-
	Reject record	STATE_CHANGE	REJECTED	OUTBOUND_PREVIEW	MARK_DONE
	Cancel record	STATE_CHANGE	CANCELED	OUTBOUND_PREVIEW	MARK_DONE
Regular Push Preview	Record received	ADDED	INVITED	OUTBOUND_PUSH_PREVIEW	ACCEPT, REJECT
		INVITED	INVITED	OUTBOUND_PUSH_PREVIEW	ACCEPT, REJECT
	Accepted	PREVIEWING	PREVIEWING	OUTBOUND_PUSH_PREVIEW	CALL, REJECT_RECORD, CANCEL_RECORD
	Rejected	REMOVED	REJECTED	OUTBOUND_PUSH_PREVIEW	
	Make call	ADDED	DIALING	OUTBOUND	HANGUP
		DIALING	DIALING	OUTBOUND	HANGUP
		ESTABLISHED	TALKING	OUTBOUND	HANGUP, HOLD
	Release and mark done	RELEASED	IDLE	OUTBOUND	MARK_DONE
		MARKDONE_APPLY	IDLE	OUTBOUND	MARK_DONE
		REMOVED	IDLE	OUTBOUND_PUSH_PREVIEW	
REMOVED		IDLE	OUTBOUND	-	
Reject record	STATE_CHANGE	REJECTED	OUTBOUND_PUSH_PREVIEW	MARK_DONE	
Cancel record	STATE_CHANGE	CANCELED	OUTBOUND_PUSH_PREVIEW	MARK_DONE	
Direct Push Preview	Record received	ADDED	INVITED	OUTBOUND_PREVIEW	ACCEPT, REJECT
		INVITED	INVITED	OUTBOUND_PREVIEW	ACCEPT, REJECT
	Accepted	PREVIEWING	PREVIEWING	OUTBOUND_PREVIEW	CALL,

Mode	UI Event	Event Type	State	Call Type	Capabilities
					REJECT_RECORD, CANCEL_RECORD
	Rejected	REMOVED	REJECTED	OUTBOUND_PREVIEW	
Make call		ADDED	DIALING	OUTBOUND	HANGUP
		DIALING	DIALING	OUTBOUND	HANGUP
		ESTABLISHED	TALKING	OUTBOUND	HANGUP
		REMOVED	IDLE	OUTBOUND_PREVIEW	
Release and mark done		RELEASED	IDLE	OUTBOUND	MARK_DONE
		MARKDONE_APPLYIDLE		OUTBOUND	MARK_DONE
		REMOVED	IDLE	OUTBOUND	-
Reject record		STATE_CHANGE	REJECTED	OUTBOUND_PREVIEW	MARK_DONE
Cancel record		STATE_CHANGE	CANCELED	OUTBOUND_PREVIEW	MARK_DONE

The **Outbound campaign events** table lists the possible events for outbound campaigns.

Outbound campaign events

EventType	Trigger	Example
CampaignLoaded	When an outbound campaign is loaded.	<pre>{   "event": "outbound",   "data": {     "eventType": "CampaignLoaded",     "campaign": "Offer of the Month"   },   "userAgent": "WWE Server",   "protocolVersion": 2 }</pre>
CampaignUnloaded	When an outbound campaign is unloaded.	<pre>{   "event": "outbound",   "data": {     "eventType": "CampaignUnloaded",     "campaign": "Offer of the Month"   },   "userAgent": "WWE Server",   "protocolVersion": 2 }</pre>
CampaignStarted	When an outbound campaign starts.	<p>This event also has a "mode" property that describes the mode in which the campaign started.</p> <pre>{</pre>

EventType	Trigger	Example
		<pre> "event": "outbound", "data": {   "eventType": "CampaignStarted",   "campaign": "Offer of the Month",   "mode": "Predictive GVP" }, "userAgent": "WWE Server", "protocolVersion": 2 } </pre>
CampaignStopped	When an outbound campaign stops.	<pre> {   "event": "outbound",   "data": {     "eventType": "CampaignStopped",     "campaign": "Offer of the Month"   },   "userAgent": "WWE Server",   "protocolVersion": 2 } </pre>

### Chain of records events

The RECORDS\_RETRIEVED event is triggered on an outbound interaction when all of the records in the interaction's chain of records have been retrieved.

### Sample response

```

{
  "event": "interaction",
  "data": {
    "eventType": "RECORDS_RETRIEVED",
    "interaction": {
      "interactionId": "1",
      "caseId": "a26f59d2-2979-43c5-5c1d-b0757f9ab077",
      "parentInteractionId": null,
      "chainedRecords": [
        {
          Custom_Character: "c"
          Custom_Datetime: "2021-03-17 14:42:39"
          Custom_Float: "16.64"
          Custom_Integer: 0
          Custom_String_with_default: "Hi there!"
          Custom_VarChar: ""
          GSW_AGENT_ID: "+33298025000"
          GSW_APPLICATION_ID: 139
          GSW_ATTEMPTS: 0
          GSW_CALLING_LIST: "Calling List Custom"
          GSW_CALLING_LIST_DBID: 101
          GSW_CALL_ATTEMPT_GUID: "003DC7H6HG84DBRT1KMIF1TAES000031"
        }
      ]
    }
  }
}

```

```

    GSW_CALL_RESULT: 28
    GSW_CAMPAIGN_GROUP_DBID: 101
    GSW_CAMPAIGN_GROUP_DESCRIPTION: ""
    GSW_CAMPAIGN_GROUP_NAME: "Outbound Campaign Custom@Agent Group Outbound"
    GSW_CAMPAIGN_NAME: "Outbound Campaign Custom"
    GSW_CHAIN_ID: 3
    GSW_CONTACT_MEDIA_TYPE: "voice"
    GSW_FROM: 0
    GSW_PHONE: "+33647005"
    GSW_PHONE_TYPE: 1
    GSW_RECORD_HANDLE: 283
    GSW_REFERENCE_ID: 3
    GSW_SWITCH_DBID: 101
    GSW_TZ_NAME: "ACT"
    GSW_TZ_OFFSET: 34200
    GSW_UNTIL: 86399
    GSW_USER_EVENT: "PreviewRecord"
    IW_BundleUid: "27458420-0348-4345-c693-45bd95b5c81f"
    IW_CaseUid: "a26f59d2-2979-43c5-5c1d-b0757f9ab077"
    InteractionSubtype: "OutboundNew"
    InteractionType: "Outbound"
    WWE_OUTBOUND_CAMP_TYPE: "PreviewRecord"
  },
  {
    Custom_Character: "c"
    Custom_Datetime: "2021-03-17 14:42:32"
    Custom_Float: "51.69"
    Custom_Integer: 0
    Custom_String_with_default: "Hello General Kenobi"
    Custom_VarChar: ""
    GSW_AGENT_ID: "+33298025000"
    GSW_APPLICATION_ID: 139
    GSW_ATTEMPTS: 0
    GSW_CALLING_LIST: "Calling List Custom"
    GSW_CALLING_LIST_DBID: 101
    GSW_CALL_ATTEMPT_GUID: "003DC7H6HG84DBRT1KMIF1TAES000031"
    GSW_CALL_RESULT: 28
    GSW_CAMPAIGN_GROUP_DBID: 101
    GSW_CAMPAIGN_GROUP_DESCRIPTION: ""
    GSW_CAMPAIGN_GROUP_NAME: "Outbound Campaign Custom@Agent Group Outbound"
    GSW_CAMPAIGN_NAME: "Outbound Campaign Custom"
    GSW_CHAIN_ID: 3
    GSW_CONTACT_MEDIA_TYPE: "voice"
    GSW_FROM: 0
    GSW_PHONE: "+33647004"
    GSW_PHONE_TYPE: 1
    GSW_RECORD_HANDLE: 284
    GSW_REFERENCE_ID: 4
    GSW_SWITCH_DBID: 101
    GSW_TZ_NAME: "ACT"
    GSW_TZ_OFFSET: 34200
    GSW_UNTIL: 86399
    GSW_USER_EVENT: "ChainedRecord"
    InteractionSubtype: "OutboundNew"
    InteractionType: "Outbound"
  }
],
"userData": {
  "GSW_PHONE": "+33647005",
  "GSW_PHONE_TYPE": "1",
  "Custom_Character": "c",
  "Custom_Datetime": "2021-03-17 14:42:39",
  "Custom_Float": "16.64",

```

```

    "Custom_Integer": "0",
    "Custom_String_with_default": "Hi there! ",
    "Custom_VarChar": "",
    "GSW_FROM": "0",
    "GSW_UNTIL": "86399",
    "GSW_TZ_OFFSET": "34200",
    "GSW_CALLING_LIST": "Calling List Custom",
    "GSW_CAMPAIGN_NAME": "Outbound Campaign Custom",
    "InteractionType": "Outbound",
    "InteractionSubtype": "OutboundNew",
    "GSW_RECORD_HANDLE": "283",
    "GSW_APPLICATION_ID": "139",
    "GSW_CAMPAIGN_GROUP_DBID": "101",
    "GSW_CALLING_LIST_DBID": "101",
    "GSW_CAMPAIGN_GROUP_NAME": "Outbound Campaign Custom@Agent Group Outbound",
    "GSW_CAMPAIGN_GROUP_DESCRIPTION": "",
    "GSW_CHAIN_ID": "3",
    "GSW_ATTEMPTS": "0",
    "GSW_CALL_RESULT": "28",
    "GSW_TZ_NAME": "ACT",
    "GSW_CALL_ATTEMPT_GUID": "003DC7H6HG84DBRT1KMIF1TAES000031",
    "GSW_CONTACT_MEDIA_TYPE": "voice",
    "GSW_REFERENCE_ID": "3",
    "GSW_SWITCH_DBID": "101",
    "GSW_USER_EVENT": "PreviewRecord",
    "GSW_AGENT_ID": "+33298025000",
    "WWE_OUTBOUND_CAMP_TYPE": "PreviewRecord",
    "IW_BundleUid": "27458420-0348-4345-c693-45bd95b5c81f",
    "IW_CaseUid": "a26f59d2-2979-43c5-5c1d-b0757f9ab077"
  },
  "state": "PREVIEWING",
  "previousState": "UNKNOWN",
  "capabilities": [
    "CALL",
    "REJECT_RECORD",
    "CANCEL_RECORD"
  ],
  "parties": [
    {
      "name": "+33647005"
    }
  ],
  "startDate": null,
  "endDate": null,
  "callType": "OUTBOUND_PREVIEW",
  "isMainCaseInteraction": true,
  "isCaseSelected": true,
  "isCaseExpanded": false
}
},
"userAgent": "WWE Server",
"protocolVersion": 2
}

```

## Common actions with Service Client API

The following sections show some common actions you can perform with Service Client API:

## Controlling call recording from a third-party application

Review the following methods for details about call recording control:

- `pauseCallRecording`
- `resumeCallRecording`
- `startCallRecording`
- `stopCallRecording`

The call recording state is stored in the `recordingState` attribute on the `interaction.Interaction` object.

## Embedding multiple third-party applications in Agent Workspace

You can configure Agent Workspace to include more than one third-party web application, displayed as either a tab, a popup window, in the background at the interaction level, or hidden. Configure the following options:

- Set the `interaction.web-content` option to a list of option section names that correspond to web extension views.
- Make sure that the `service-client-api.accepted-web-content-origins` option references all the websites that should use the Service Client API.

## Updating attached data from a third-party application

Review the following methods for details about updating attached data:

- `deleteUserData`
- `getByInteractionId`
- `getInteractions`
- `setUserData`

The user data is stored in the `userData` attribute on the `interaction.Interaction` object.

You should also set the options related to user data in the Service Client section of Agent Setup or configure the `service-client-api.user-data.read-allowed` and `service-client-api.user-data.write-allowed` options.

## Enabling click-to-dial from a third-party application

If you configure Agent Workspace to display your web application in a new tab in the Agent Workspace user interface, then the service API only gives access to the dial operation.

## Enabling Service Client API to invoke toast in Agent Workspace

Review the following methods for details about enabling and updating toast:

- system.popupToast
- system.updateToast
- system.closeToast

## Controlling case selection from a third-party application

Review the following method for details about case selecting control:

- selectCaseByCaseId

The case selection state is stored in the *isCaseSelected* attribute and the *isCaseExpanded* attribute on the **interaction.Interaction** object.

## Supporting multiple browser tabs

Service Client API supports multiple browser tabs in a session. The API uses the concept of a leader tab and following tab or tabs. When multiple tabs are open, certain actions (typically automatic) are performed only by the leader tab, such as auto-answer for chat, email, and voice interactions, and contact management in Universal Contact Server. The API also tracks which tab was the last active because some actions are performed only by this tab, such as sounds, toasts, and supervisor-forced log out.

The state of a given browser tab is determined by an internal election process, which can be triggered when an agent closes a leader tab. The state is exposed through the **data.frameState** property on system events. The **frameState** property has three possible values:

- LEADING: The election happened and this tab is the leader.
- FOLLOWING: The election happened and this tab is a follower.
- NEGOTIATING: The election is in progress and no tab is a leader or follower until the election is finished.

You can subscribe to system events as follows:

```
function eventHandler(message) {
  switch (message.event) {
    case 'system':
      log('Received system event: ', JSON.stringify(message, null, '\t'));
      break;
    default:
      break;
  }
}
```

```
genesys.wwe.service.subscribe(['system'], eventHandler, this);
```

When an election is triggered, you should see these types of system events:

```
Received system event:
{
  "event": "system",
  "data": {
    "frameState": "LEADING"
  },
  "userAgent": "WWE Server",
```

```
    "protocolVersion": 2
  }
```

Received system event:

```
{
  "event": "system",
  "data": {
    "frameState": "NEGOTIATING"
  },
  "userAgent": "WWE Server",
  "protocolVersion": 2
}
```

Service Client API provides some helper functions through the System namespace to determine the state of a tab:

- `isFrameLeading`
- `isFrameFollowing`
- `isFrameNegotiating`
- `isFrameLeadingOrNegotiating`
- `isLastActiveFrame`

Service Client API updates the attached data for an interaction in the leader tab with a new **caseId** on eventType `CASE_ID_CHANGED`.

```
{
  "event": "interaction",
  "data": {
    "eventType": "CASE_ID_CHANGED",
    "caseId": "e6470563-af78-4942-657d-976a25dd9de3",
    "previousCaseId": "5f7e5f3a-fb6e-43f3-c404-eaee21d64ef1"
  },
  "userAgent": "WWE Server",
  "protocolVersion": 2
}
```

# Agent namespace

## Contents

- **1 Methods**
  - **1.1 get**
  - **1.2 getState**
  - **1.3 getStateList**
  - **1.4 setState**
- **2 Type definitions**
  - **2.1 Agent**

Learn about the Agent namespace methods and type definitions in the Service Client API.

## Methods

The Agent namespace includes the following methods:

- get
- getState
- getStateList
- setState

### get

<b>Signature</b>	get( <i>succeeded, failed</i> ) → {agent.Agent}		
<b>Description</b>	Gets the agent's attributes.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.
<b>Returns</b>	agent.Agent		

### getState

<b>Signature</b>	getState( <i>succeeded, failed</i> ) → {media.State}		
<b>Description</b>	Gets the agent's state.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when

<b>Signature</b>	getState( <i>succeeded, failed</i> ) → {media.State}								
	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td>the operation fails.</td> </tr> </tbody> </table>			Name	Type	Description			the operation fails.
Name	Type	Description							
		the operation fails.							
<b>Returns</b>	media.State								

### getStateList

<b>Signature</b>	getStateList( <i>succeeded, failed</i> ) → {Array.}											
<b>Description</b>	Gets the list of possible agent states.											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>			Name	Type	Description	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.
Name	Type	Description										
succeeded	function	A function called when the operation succeeds.										
failed	function	A function called when the operation fails.										
<b>Returns</b>	Array.											

### setState

<b>Signature</b>	setState( <i>stateOperationName, succeeded, failed</i> )														
<b>Description</b>	Sets the agent's state.														
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>stateOperationName</td> <td>string</td> <td>An operationName from the agent states list. See State.</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>			Name	Type	Description	stateOperationName	string	An operationName from the agent states list. See State.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.
Name	Type	Description													
stateOperationName	string	An operationName from the agent states list. See State.													
succeeded	function	A function called when the operation succeeds.													
failed	function	A function called when the operation fails.													

## Type definitions

The agent namespace includes the following object types:

- Agent

### Agent

<b>Description</b>	Represents the JSON structure of the agent.																	
<b>Type</b>	Object																	
<b>Properties</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>employeeid</td><td>string</td><td>The agent's unique identifier used for routing purposes.</td></tr><tr><td>firstname</td><td>string</td><td>The agent's first name.</td></tr><tr><td>lastname</td><td>string</td><td>The agent's last name.</td></tr><tr><td>username</td><td>string</td><td>The agent's username. This is a global unique ID.</td></tr></tbody></table>	Name	Type	Description	employeeid	string	The agent's unique identifier used for routing purposes.	firstname	string	The agent's first name.	lastname	string	The agent's last name.	username	string	The agent's username. This is a global unique ID.		
Name	Type	Description																
employeeid	string	The agent's unique identifier used for routing purposes.																
firstname	string	The agent's first name.																
lastname	string	The agent's last name.																
username	string	The agent's username. This is a global unique ID.																

# Configuration namespace

## Contents

- **1 Methods**
  - **1.1 `getOption`**
  - **1.2 `getContextualOption`**
- **2 Type definitions**
  - **2.1 Section**
- **3 Examples**
  - **3.1 `getContextualOption()`**
  - **3.2 `getOption()`**

Learn about the Configuration namespace methods and type definitions in the Service Client API.

### Important

Depending on your environment, you might need to contact your Genesys representative to complete the configuration described on this page.

## Methods

The Configuration namespace includes the following methods:

- `getOption`
- `getContextualOption`

### getOption

<b>Signature</b>	<code>getOption(<i>options</i>, succeeded, failed) → {Array. }</code>								
<b>Description</b>	Get configuration options and values for a specific option name or a subset of options from the <b>[interaction-workspace]</b> section or a custom section.								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code>options</code></td><td>string</td><td>An array of configuration options or sections to return. Unless otherwise specified, the API returns options from the <b>[interaction-workspace]</b> section by default. You can specify</td></tr></tbody></table>	Name	Type	Description	<code>options</code>	string	An array of configuration options or sections to return. Unless otherwise specified, the API returns options from the <b>[interaction-workspace]</b> section by default. You can specify		
Name	Type	Description							
<code>options</code>	string	An array of configuration options or sections to return. Unless otherwise specified, the API returns options from the <b>[interaction-workspace]</b> section by default. You can specify							

Signature	<code>getOption(<i>options</i>, succeeded, failed) → {Array. }</code>		
	Name	Type	Description
			<p>any of the following:</p> <ul style="list-style-type: none"> <li>• A single option: genesys.wwe.service.configuration.getOption('voice.auto-answer', succeeded, failed)</li> <li>• A single option in a specific section: genesys.wwe.service.configuration.getOption('CustomSection/option.customer.code', succeeded, failed)</li> <li>• Multiple options: genesys.wwe.service.configuration.getOption(['voice.auto-answer', 'privilege.email.can-mark-done'], succeeded, failed)</li> <li>• Multiple</li> </ul>

Signature	<code>getOption(options, succeeded, failed) → {Array.}</code>		
	Name	Type	Description
			<p>options in different sections:  <code>genesys.w</code>  <code>we.servic</code>  <code>e.configu</code>  <code>ration.ge</code>  <code>tOption([</code>  <code>'privileg</code>  <code>e.*',</code>  <code>'CustomSe</code>  <code>ction/</code>  <code>option.cu</code>  <code>stom.cust</code>  <code>omer.code</code>  <code>'],</code>  <code>succeeded</code>  <code>, failed)</code></p> <p>You can use an asterisk '*' as a wildcard, but only at the end of each word. For example:</p> <ul style="list-style-type: none"> <li>• <code>voice.*</code></li> <li>• <code>voice.auto*</code></li> <li>• <code>sipendpoint.*</code></li> <li>• <code>CustomAPI/test.*</code></li> </ul> <p>You cannot use an asterisk at the start of an option or section. For example, the following values are not allowed:</p> <ul style="list-style-type: none"> <li>• <code>*.mark-done</code></li> <li>• <code>*.auto</code></li> </ul>
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when

<b>Signature</b>	getOption( <i>options</i> , succeeded, failed) → {Array. }		
	<b>Name</b>	<b>Type</b>	<b>Description</b>
			the operation fails.
<b>Returns</b>	Array.		

### getContextualOption

<b>Signature</b>	getContextualOption( <i>options</i> , <i>interactionId</i> , succeeded, failed) → {Array. }		
<b>Description</b>	<p>Get configuration options and values in the context of this interaction when they are overridden by a routing strategy. If the interaction is not found or this parameter is missing, the API throws an exception. Note: The getContextualOption method can be applied to any option, even if the option doesn't support overriding options with a routing strategy in Workspace Web Edition. Only Workspace Web Edition options with the following text in their descriptions can be overridden by a routing strategy: "This option can be overridden by a routing strategy as described in this Configuration Guide." Using the getContextualOption method doesn't interfere with how Workspace Web Edition handles options.</p>		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	options	string	<p>An option or array of options and their values. Unless otherwise specified, the API returns options from the <b>[interaction-workspace]</b> section by default. You can specify any of the following:</p> <ul style="list-style-type: none"> <li>A single option by name:</li> </ul>

Signature	<code>getContextualOption(options, interactionId, succeeded, failed) → {Array.}</code>		
	Name	Type	Description
			<p>genesys.we.service.configuration.getOption('voice.auto-answer', succeeded, failed)</p> <ul style="list-style-type: none"> <li>• A subset of options defined by '*':</li> <li>• A subset of options in a specific section: genesys.we.service.configuration.getOption('interaction-workspace / interaction.case-data.*', succeeded, failed)</li> </ul> <p>You can use an asterisk '*' as a wildcard, but only at the end of each word. For example:</p> <ul style="list-style-type: none"> <li>• voice.*</li> <li>• voice.auto*</li> <li>• sipendpoint.*</li> <li>• CustomAPI/test.*</li> </ul> <p>You cannot use an asterisk at the</p>

<b>Signature</b>	getContextualOption( <i>options</i> , <i>interactionId</i> , succeeded, failed) → {Array. }		
	<b>Name</b>	<b>Type</b>	<b>Description</b>
			start of an option or section. For example, the following values are not allowed: <ul style="list-style-type: none"> <li>• *.mark-done</li> <li>• *.auto</li> </ul>
	interactionId	string	The unique identifier for the interaction.
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.
<b>Returns</b>	Array.		

## Type definitions

The Configuration namespace includes the following object types:

- Section

### Section

<b>Description</b>	Represents the JSON structure of a configuration section. Each section includes a list of key/value pairs for the matching option(s).		
<b>Type</b>	Object		
<b>Properties</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	name	string	The name of the configuration option.

Description	Represents the JSON structure of a configuration section. Each section includes a list of key/value pairs for the matching option(s).								
	<table border="1"> <thead> <tr> <th data-bbox="821 354 1032 401">Name</th> <th data-bbox="1039 354 1237 401">Type</th> <th data-bbox="1243 354 1448 401">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="821 409 1032 518">value</td> <td data-bbox="1039 409 1237 518">string or array of strings</td> <td data-bbox="1243 409 1448 518">The value of the configuration option.</td> </tr> </tbody> </table>			Name	Type	Description	value	string or array of strings	The value of the configuration option.
Name	Type	Description							
value	string or array of strings	The value of the configuration option.							

## Examples

Consider the following scenario:

1. You set `interaction-workspace/interaction.case-data.frame-color="#FFBA00"`.
2. You also set the override option key: `interaction-workspace/interaction.override-option-key="IW_OverrideOptions"`.
3. An interaction arrives with the attached data `IW_OverrideOptions="CaseDataColor"`.
4. The transaction object `"CaseDataColor"` annex has the option `interaction-workspace/interaction.case-data.frame-color="#FF00088"`.

Here's how this scenario would look for each of the Configuration namespace methods:

### getContextualOption()

Use `getContextualOption()` to get the option:

```
genesys.wwe.service.configuration.getContextualOption("interaction.case-data.frame-color",
"1", succeeded, failed)
```

You receive this response:

```
{
  "request": "configuration.getContextualOption",
  "data": {
    "interaction-workspace": {
      "interaction.case-data.frame-color": "#FF008000"
    }
  },
  "userAgent": "WWE Server",
  "protocolVersion": 2
}
```

If the interaction isn't found, the failed callback receives this response:

```
{
  "request": "configuration.getContextualOption",
  "errorMessage": "Error: Interaction not found.",
  "userAgent": "WWE Server",
  "protocolVersion": 2
}
```

```
}
```

## getOption()

Use `getOption()` to get the option::

```
genesys.wwe.service.configuration.getOption("interaction.case-data.frame-color", succeeded,  
failed)
```

The response includes the original default value of the option instead of the overridden value:

```
{  
  "request": "configuration.getOption",  
  "data": {  
    "interaction-workspace": {  
      "interaction.case-data.frame-color": "#FFBA00"  
    }  
  },  
  "userAgent": "WWE Server",  
  "protocolVersion": 2  
}
```

# Email namespace

## Contents

- [1 Methods](#)
  - [1.1 create](#)

Learn about the Email namespace methods in the Service Client API.

## Methods

The Email namespace includes the following methods:

- create

### create

<b>Signature</b>	<code>create(destination, userData, succeeded, failed)</code>																				
<b>Description</b>	Creates a new empty email.																				
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Argument</th><th>Description</th></tr></thead><tbody><tr><td>destination</td><td>string</td><td></td><td>The destination address for the email.</td></tr><tr><td>userData</td><td>object</td><td></td><td>The attached user data key/value object that is updated with each interaction event.</td></tr><tr><td>succeeded</td><td>function</td><td></td><td>A function called when the operation succeeds.</td></tr><tr><td>failed</td><td>function</td><td></td><td>A function called when the operation fails.</td></tr></tbody></table>	Name	Type	Argument	Description	destination	string		The destination address for the email.	userData	object		The attached user data key/value object that is updated with each interaction event.	succeeded	function		A function called when the operation succeeds.	failed	function		A function called when the operation fails.
	Name	Type	Argument	Description																	
	destination	string		The destination address for the email.																	
	userData	object		The attached user data key/value object that is updated with each interaction event.																	
	succeeded	function		A function called when the operation succeeds.																	
failed	function		A function called when the operation fails.																		

# Interaction namespace

## Contents

- **1 Methods**
  - 1.1 completeConference
  - 1.2 completeTransfer
  - 1.3 consult
  - 1.4 deleteUserData
  - 1.5 getByInteractionId
  - 1.6 getInteractions
  - 1.7 selectCaseByCaseId
  - 1.8 setUserData
  - 1.9 markdone
  - 1.10 blockMarkdone
  - 1.11 singleStepConference
  - 1.12 singleStepTransfer
  - 1.13 unblockMarkdone
  - 1.14 accept
  - 1.15 reject
- **2 Type definitions**
  - 2.1 Interaction
  - 2.2 Party
  - 2.3 Contact

Learn about the Interaction namespace methods and type definitions in the Service Client API.

### Important

Depending on your environment, you might need to contact your Genesys representative to complete the configuration described on this page.

## Methods

The Interaction namespace includes the following methods:

- completeConference
- completeTransfer
- consult
- deleteUserData
- getByInteractionId
- getInteractions
- selectCaseByCaseId
- setUserData
- markdone
- blockMarkdone
- singleStepConference
- singleStepTransfer
- unblockMarkdone
- accept
- reject

### completeConference

<b>Signature</b>	<code>completeConference(<i>consultInteractionId</i>, <i>succeeded</i>, <i>failed</i>)</code>
<b>Description</b>	Completes a conference.

<b>Signature</b>	completeConference( <i>consultInteractionId</i> , <i>succeeded</i> , <i>failed</i> )														
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>consultInteractionId</td> <td>string</td> <td>The unique identifier for the consultation interaction.</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	consultInteractionId	string	The unique identifier for the consultation interaction.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.		
Name	Type	Description													
consultInteractionId	string	The unique identifier for the consultation interaction.													
succeeded	function	A function called when the operation succeeds.													
failed	function	A function called when the operation fails.													

### completeTransfer

<b>Signature</b>	completeTransfer( <i>consultInteractionId</i> , <i>succeeded</i> , <i>failed</i> )														
<b>Description</b>	Completes a transfer.														
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>consultInteractionId</td> <td>string</td> <td>The unique identifier for the consultation interaction.</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	consultInteractionId	string	The unique identifier for the consultation interaction.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.		
Name	Type	Description													
consultInteractionId	string	The unique identifier for the consultation interaction.													
succeeded	function	A function called when the operation succeeds.													
failed	function	A function called when the operation fails.													

### consult

<b>Signature</b>	consult( <i>interactionId</i> , <i>targetQuery</i> , <i>userData</i> , <i>extensions</i> , <i>succeeded</i> , <i>failed</i> )								
<b>Description</b>	Make a consultation interaction.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td>The unique</td> </tr> </tbody> </table>	Name	Type	Description	interactionId	string	The unique		
Name	Type	Description							
interactionId	string	The unique							

Signature	consult( <i>interactionId</i> , <i>targetQuery</i> , <i>userData</i> , <i>extensions</i> , <i>succeeded</i> , <i>failed</i> )		
	Name	Type	Description
			identifier for the interaction.
	targetQuery	object or string	The destination target object, or a character string (for example, phone number). <ul style="list-style-type: none"> <li>• If targetQuery is a character string, the Service Client API creates the operation that uses a target of type <b>CustomContact</b> with a destination set to this value.</li> <li>• If targetQuery is a JSON object, specify the following sub-parameters:                             <ul style="list-style-type: none"> <li>• <b>target (string):</b> The target type. The possible values are: "AGENT", "AGENT_GROUP",</li> </ul> </li> </ul>

Signature	consult( <i>interactionId</i> , <i>targetQuery</i> , <i>userData</i> , <i>extensions</i> , <i>succeeded</i> , <i>failed</i> )		
	Name	Type	Description
			<p>"SKILL", "INTERACTION_QUEUE", "ROUTING_POINT", and "CUSTOM_CONTACT".</p> <ul style="list-style-type: none"> <li> <b>destination (string):</b>                      The destination. The supported values are: the employeeld of an agent, the name of an AgentGroup, the name of a Skill, the name of an InteractionQueue, the name of a RoutingPoint, and a phone number for CustomContact.                 </li> <li> <b>[media] (string):</b>                      An optional media used to make the consultation. If not specified,                 </li> </ul>

Signature	<code>consult(interactionId, targetQuery, userData, extensions, succeeded, failed)</code>		
	Name	Type	Description
			<p>uses the same media as the specified interaction. For example, if the interaction has a "chat" media, and you want to make a voice consultation, you must specify "voice" here.</p>
	userData	object	<p>The attached user data key/value object. Set an undefined or empty JSON object if you don't want to set any user data.</p>
	extensions	object	<p>The extensions key/value object. Set an undefined or empty JSON object if you don't want to set any extensions. This is not applicable for the chat media.</p>

<b>Signature</b>	<code>consult(interactionId, targetQuery, userData, extensions, succeeded, failed)</code>		
	<b>Name</b>	<b>Type</b>	<b>Description</b>
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.

## deleteUserData

<b>Signature</b>	<code>deleteUserData(interactionId, key, succeeded, failed)</code>		
<b>Description</b>	Deletes the user data attached to the interaction. The List of User Data Write Allowed setting in Agent Setup or the <code>service-client-api.user-data.write-allowed</code> configuration option might restrict the allowed key/value pairs.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	interactionId	string	The unique identifier for the interaction.
	key	string	The key to delete from the attached data.
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.

## getByInteractionId

<b>Signature</b>	<code>getByInteractionId(interactionId, succeeded, failed) → {interaction: Interaction}</code>
<b>Description</b>	Gets an interaction by its unique identifier.

<b>Signature</b>	getByInteractionId( <i>interactionId</i> , succeeded, failed) → {interaction.Interaction}														
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td>The unique identifier for the interaction.</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	interactionId	string	The unique identifier for the interaction.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.		
Name	Type	Description													
interactionId	string	The unique identifier for the interaction.													
succeeded	function	A function called when the operation succeeds.													
failed	function	A function called when the operation fails.													
<b>Returns</b>	interaction.Interaction or null if the interaction doesn't exist.														

## getInteractions

<b>Signature</b>	getInteractions(succeeded, failed) → {Array.}											
<b>Description</b>	Gets all the interactions.											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.		
Name	Type	Description										
succeeded	function	A function called when the operation succeeds.										
failed	function	A function called when the operation fails.										
<b>Returns</b>	Array.											

## selectCaseByCaseId

<b>Signature</b>	genesys.wwe.service.interaction.selectCaseByCaseId(caseId, succeeded, failed)
<b>Description</b>	<p>Select the case in the UI by case identifier. If you subscribe to the "interaction" events (genesys.wwe.service.subscribe([ "interaction" ], eventHandler, this);), you will receive the following event:</p> <pre>Received interaction event: {   "event": "interaction",</pre>

<b>Signature</b>	genesys.wwe.service.interaction.selectCaseByCaselId(caselId, succeeded, failed)														
	<pre>       "data": {         "eventType": "CASE_COLLAPSED",         "selectedCaseId": "4401820b- c4e6-4994-69c2-6ae7fdbc4905"       },       "userAgent": "WWE Server",       "protocolVersion": 2     } Received interaction event: {   "event": "interaction",   "data": {     "eventType": "CASE_EXPANDED",     "selectedCaseId": "4401820b- c4e6-4994-69c2-6ae7fdbc4905"   },   "userAgent": "WWE Server",   "protocolVersion": 2 } Received interaction event: {   "event": "interaction",   "data": {     "eventType": "CASE_SELECTED",     "selectedCaseId": "d4187b87-9fe1-4db8-0515-6a91e666e22d"   },   "userAgent": "WWE Server",   "protocolVersion": 2 } </pre>														
<b>Parameters</b>	<table border="1"> <thead> <tr> <th data-bbox="824 1178 1032 1220">Name</th> <th data-bbox="1032 1178 1239 1220">Type</th> <th data-bbox="1239 1178 1446 1220">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="824 1220 1032 1318">caselId</td> <td data-bbox="1032 1220 1239 1318">string</td> <td data-bbox="1239 1220 1446 1318">The unique identifier for the case.</td> </tr> <tr> <td data-bbox="824 1318 1032 1444">succeeded</td> <td data-bbox="1032 1318 1239 1444">function</td> <td data-bbox="1239 1318 1446 1444">A function called when the operation succeeds.</td> </tr> <tr> <td data-bbox="824 1444 1032 1570">failed</td> <td data-bbox="1032 1444 1239 1570">function</td> <td data-bbox="1239 1444 1446 1570">A function called when the operation fails.</td> </tr> </tbody> </table>			Name	Type	Description	caselId	string	The unique identifier for the case.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.
Name	Type	Description													
caselId	string	The unique identifier for the case.													
succeeded	function	A function called when the operation succeeds.													
failed	function	A function called when the operation fails.													

## setUserData

<b>Signature</b>	setUserData( <i>interactionId</i> , <i>keyValues</i> , succeeded, failed)
<b>Description</b>	Sets the user data on the live interaction (for voice, this means the interaction is not in the IDLE state).

<b>Signature</b>	setUserData( <i>interactionId</i> , <i>keyValues</i> , succeeded, failed)																	
	This request overwrites any existing keys on the user data. The List of User Data Write Allowed setting in Agent Setup or the service-client-api.user-data.write-allowed configuration option might restrict the allowed key/value pairs.																	
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td>The unique identifier for the interaction.</td> </tr> <tr> <td>keyValues</td> <td>object</td> <td>The key value pairs to set on the user data.</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	interactionId	string	The unique identifier for the interaction.	keyValues	object	The key value pairs to set on the user data.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.		
Name	Type	Description																
interactionId	string	The unique identifier for the interaction.																
keyValues	object	The key value pairs to set on the user data.																
succeeded	function	A function called when the operation succeeds.																
failed	function	A function called when the operation fails.																

## markdone

<b>Signature</b>	markdone( <i>interactionId</i> , succeeded, failed)														
<b>Description</b>	Mark done the selected interaction.														
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td>The unique identifier for the interaction.</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	interactionId	string	The unique identifier for the interaction.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.		
Name	Type	Description													
interactionId	string	The unique identifier for the interaction.													
succeeded	function	A function called when the operation succeeds.													
failed	function	A function called when the operation fails.													

## Outbound interactions

The markdone operation can be used for outbound interactions such as pull preview, push preview, and direct push preview, but there are some details you need to know:

- Pull preview - Mark done is similar to doing a 'Done and Stop' action, where the next preview record is not fetched.
- Push preview - No special behaviour.
- Direct push preview - Mark done is similar to doing a 'Done and Stop' action, where it triggers a notification to Outbound Contact Server to stop sending direct push preview records.

## blockMarkdone

<b>Signature</b>	blockMarkdone( <i>interactionId</i> , <i>warningMessage</i> , <i>succeeded</i> , <i>failed</i> )																	
<b>Description</b>	Block the mark done operation on the selected interaction. The "markdone" event must be subscribed to receive the event which informs that there is a delay in blocking the markdone operation with this method.																	
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td>The unique interaction identifier of the interaction to prevent the mark done operation.</td> </tr> <tr> <td>warningMessage</td> <td>string</td> <td>The warning message.</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	interactionId	string	The unique interaction identifier of the interaction to prevent the mark done operation.	warningMessage	string	The warning message.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.		
Name	Type	Description																
interactionId	string	The unique interaction identifier of the interaction to prevent the mark done operation.																
warningMessage	string	The warning message.																
succeeded	function	A function called when the operation succeeds.																
failed	function	A function called when the operation fails.																

## singleStepConference

<b>Signature</b>	singleStepConference( <i>interactionId</i> , <i>targetQuery</i> , <i>userData</i> , <i>extensions</i> , <i>succeeded</i> , <i>failed</i> )											
<b>Description</b>	Make a single step conference.											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td>The unique identifier for the interaction.</td> </tr> <tr> <td>targetQuery</td> <td>object or string</td> <td>The destination</td> </tr> </tbody> </table>	Name	Type	Description	interactionId	string	The unique identifier for the interaction.	targetQuery	object or string	The destination		
Name	Type	Description										
interactionId	string	The unique identifier for the interaction.										
targetQuery	object or string	The destination										

Signature	singleStepConference( <i>interactionId</i> , <i>targetQuery</i> , <i>userData</i> , <i>extensions</i> , <i>succeeded</i> , <i>failed</i> )		
	Name	Type	Description
			<p>target object, or a character string (for example, phone number).</p> <ul style="list-style-type: none"> <li>If <i>targetQuery</i> is a character string, the Service Client API creates the operation that uses a target of type <b>CustomContact</b> with a destination set to this value.</li> <li>If <i>targetQuery</i> is a JSON object, specify the following sub-parameters: <ul style="list-style-type: none"> <li><b>type (string):</b> The target type. The possible values are: "AGENT", "AGENT_GROUP", "SKILL", "INTERACTION_QUEUE", "ROUTING_POINT", and "CUSTOM_CONTACT".</li> </ul> </li> </ul>

Signature	singleStepConference( <i>interactionId</i> , <i>targetQuery</i> , <i>userData</i> , <i>extensions</i> , <i>succeeded</i> , <i>failed</i> )		
	Name	Type	Description
			<ul style="list-style-type: none"> <li> <b>destination (string):</b>                      The destination. The supported values are: the employeeld of an agent, the name of an AgentGroup, the name of a Skill, the name of an InteractionQueue, the name of a RoutingPoint, and a phone number for CustomContact.                 </li> </ul>
	userData	object	The attached user data key/value object. Set an undefined or empty JSON object if you don't want to set any user data.
	extensions	object	The extensions key/value object. Set an undefined or empty JSON

<b>Signature</b>	<code>singleStepConference(interactionId, targetQuery, userData, extensions, succeeded, failed)</code>		
	<b>Name</b>	<b>Type</b>	<b>Description</b>
			object if you don't want to set any extensions. This is not applicable for the chat media.
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.

### singleStepTransfer

<b>Signature</b>	<code>singleStepTransfer(interactionId, targetQuery, userData, extensions, succeeded, failed)</code>		
<b>Description</b>	Make a single step transfer.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	interactionId	string	The unique identifier for the interaction.
	targetQuery	object or string	<p>The destination target object, or a character string (for example, phone number).</p> <ul style="list-style-type: none"> <li>If targetQuery is a character string, the Service Client API creates the</li> </ul>

Signature	singleStepTransfer( <i>interactionId</i> , <i>targetQuery</i> , <i>userData</i> , <i>extensions</i> , <i>succeeded</i> , <i>failed</i> )		
	Name	Type	Description
			<p>operation that uses a target of type <b>CustomContact</b> with a destination set to this value.</p> <ul style="list-style-type: none"> <li>If targetQuery is a JSON object, specify the following sub parameters: <ul style="list-style-type: none"> <li><b>type (string):</b> The target type. The possible values are: "AGENT", "AGENT_GROUP", "SKILL", "INTERACTION_QUEUE", "ROUTING_POINT", and "CUSTOM_CONTACT".</li> <li><b>destination (string):</b> The destination. The supported values are: the employeeId of an Agent, the name of an</li> </ul> </li> </ul>

Signature	singleStepTransfer( <i>interactionId</i> , <i>targetQuery</i> , <i>userData</i> , <i>extensions</i> , <i>succeeded</i> , <i>failed</i> )		
	Name	Type	Description
			AgentGroup, the name of a Skill, the name of an InteractionQueue, the name of a RoutingPoint, and a phone number for CustomContact.
	userData	object	The attached user data key/value object. Set an undefined or empty JSON object if you don't want to set any user data.
	extensions	object	The extensions key/value object. Set an undefined or empty JSON object if you don't want to set any extensions.
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.

## unblockMarkdone

<b>Signature</b>	unblockMarkdone( <i>interactionId</i> , succeeded, failed)														
<b>Description</b>	Unblock the mark done operation on the selected interaction that was previously blocked.														
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td>The unique interaction identifier of the interaction to prevent the mark done operation.</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	interactionId	string	The unique interaction identifier of the interaction to prevent the mark done operation.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.		
Name	Type	Description													
interactionId	string	The unique interaction identifier of the interaction to prevent the mark done operation.													
succeeded	function	A function called when the operation succeeds.													
failed	function	A function called when the operation fails.													

## accept

<b>Signature</b>	accept( <i>interactionId</i> , succeeded, failed)														
<b>Description</b>	Accept an interaction when it is ringing in Agent Workspace.														
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td>The unique interaction identifier of the interaction to be accepted.</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	interactionId	string	The unique interaction identifier of the interaction to be accepted.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.		
Name	Type	Description													
interactionId	string	The unique interaction identifier of the interaction to be accepted.													
succeeded	function	A function called when the operation succeeds.													
failed	function	A function called when the operation fails.													

## reject

<b>Signature</b>	<code>reject(interactionId, succeeded, failed)</code>		
<b>Description</b>	Reject an interaction when it is ringing in Agent Workspace.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	interactionId	string	The unique interaction identifier of the interaction to be rejected.
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.

## Type definitions

The Interaction namespace includes the following object types:

- Interaction
- Party
- Contact

## Interaction

<b>Description</b>	Represents the JSON structure of an interaction. Attributes specific to voice interactions are: <code>callUuid</code> , <code>direction</code> , <code>callType</code> , <code>ani</code> , <code>dnis</code> and <code>recordingState</code> .		
<b>Type</b>	Object		
<b>Properties</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	interactionId	string	The unique identifier for the interaction. <b>Note:</b> This is a client-side ID that is lost

Description	Represents the JSON structure of an interaction. Attributes specific to voice interactions are: callUuid, direction, callType, ani, dnis and recordingState.																				
	<table border="1"> <thead> <tr> <th data-bbox="833 384 1032 426">Name</th> <th data-bbox="1032 384 1239 426">Type</th> <th data-bbox="1239 384 1437 426">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="833 426 1032 520"></td> <td data-bbox="1032 426 1239 520"></td> <td data-bbox="1239 426 1437 520">                             on the next session or refresh.                         </td> </tr> <tr> <td data-bbox="833 520 1032 814">parentInteractionId</td> <td data-bbox="1032 520 1239 814">string</td> <td data-bbox="1239 520 1437 814">                             The unique identifier for the parent interaction. <b>Note:</b> This is a client-side ID that is lost on the next session or refresh.                         </td> </tr> <tr> <td data-bbox="833 814 1032 968">caseId</td> <td data-bbox="1032 814 1239 968">string</td> <td data-bbox="1239 814 1437 968">                             This identifier targets the case that this interaction is part of.                         </td> </tr> <tr> <td data-bbox="833 968 1032 1199">userData</td> <td data-bbox="1032 968 1239 1199">object</td> <td data-bbox="1239 968 1437 1199">                             The attached user data key/value object that is updated with each interaction event.                         </td> </tr> <tr> <td data-bbox="833 1199 1032 1827">state</td> <td data-bbox="1032 1199 1239 1827">string</td> <td data-bbox="1239 1199 1437 1827">                             The current state of the interaction. Possible values are:                             <ul style="list-style-type: none"> <li>• UNKNOWN — An unknown state.</li> <li>• IDLE — Specifies a non-active interaction which could be closed.</li> <li>• RINGING — The inbound</li> </ul> </td> </tr> </tbody> </table>			Name	Type	Description			on the next session or refresh.	parentInteractionId	string	The unique identifier for the parent interaction. <b>Note:</b> This is a client-side ID that is lost on the next session or refresh.	caseId	string	This identifier targets the case that this interaction is part of.	userData	object	The attached user data key/value object that is updated with each interaction event.	state	string	The current state of the interaction. Possible values are: <ul style="list-style-type: none"> <li>• UNKNOWN — An unknown state.</li> <li>• IDLE — Specifies a non-active interaction which could be closed.</li> <li>• RINGING — The inbound</li> </ul>
Name	Type	Description																			
		on the next session or refresh.																			
parentInteractionId	string	The unique identifier for the parent interaction. <b>Note:</b> This is a client-side ID that is lost on the next session or refresh.																			
caseId	string	This identifier targets the case that this interaction is part of.																			
userData	object	The attached user data key/value object that is updated with each interaction event.																			
state	string	The current state of the interaction. Possible values are: <ul style="list-style-type: none"> <li>• UNKNOWN — An unknown state.</li> <li>• IDLE — Specifies a non-active interaction which could be closed.</li> <li>• RINGING — The inbound</li> </ul>																			

<p><b>Description</b></p>	<p>Represents the JSON structure of an interaction. Attributes specific to voice interactions are: callUuid, direction, callType, ani, dnis and recordingState.</p>								
	<table border="1"> <thead> <tr> <th data-bbox="824 384 1032 426">Name</th> <th data-bbox="1032 384 1240 426">Type</th> <th data-bbox="1240 384 1445 426">Description</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td data-bbox="1240 426 1445 1831"> <p>call is ringing.</p> <ul style="list-style-type: none"> <li>• DIALING — The outbound call is ringing.</li> <li>• TALKING — The call is established.</li> <li>• HELD — The call is on hold.</li> <li>• PREVIEW — The interaction is a call preview.</li> <li>• INVITED — The open media interaction is inviting.</li> <li>• ACCEPTED — The open media interaction is accepted.</li> <li>• CREATED — The open media interaction has been created.</li> <li>• PULLED — The open media interaction has been pulled</li> </ul> </td> </tr> </tbody> </table>			Name	Type	Description			<p>call is ringing.</p> <ul style="list-style-type: none"> <li>• DIALING — The outbound call is ringing.</li> <li>• TALKING — The call is established.</li> <li>• HELD — The call is on hold.</li> <li>• PREVIEW — The interaction is a call preview.</li> <li>• INVITED — The open media interaction is inviting.</li> <li>• ACCEPTED — The open media interaction is accepted.</li> <li>• CREATED — The open media interaction has been created.</li> <li>• PULLED — The open media interaction has been pulled</li> </ul>
Name	Type	Description							
		<p>call is ringing.</p> <ul style="list-style-type: none"> <li>• DIALING — The outbound call is ringing.</li> <li>• TALKING — The call is established.</li> <li>• HELD — The call is on hold.</li> <li>• PREVIEW — The interaction is a call preview.</li> <li>• INVITED — The open media interaction is inviting.</li> <li>• ACCEPTED — The open media interaction is accepted.</li> <li>• CREATED — The open media interaction has been created.</li> <li>• PULLED — The open media interaction has been pulled</li> </ul>							

<p><b>Description</b></p>	<p>Represents the JSON structure of an interaction. Attributes specific to voice interactions are: callUuid, direction, callType, ani, dnis and recordingState.</p>								
	<table border="1"> <thead> <tr> <th data-bbox="824 384 1032 426">Name</th> <th data-bbox="1032 384 1240 426">Type</th> <th data-bbox="1240 384 1453 426">Description</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td data-bbox="1240 426 1453 1841"> <p>from a workbin.</p> <ul style="list-style-type: none"> <li>• <b>REVOKED</b> — The open media interaction has been revoked.</li> <li>• <b>COMPLETED</b> — The open media interaction has been completed (Mark as done).</li> <li>• <b>ERROR</b> — The open media interaction has an error.</li> <li>• <b>SAVED</b> — The open media interaction has been saved.</li> <li>• <b>TRANSFERRING</b> — The open media interaction is being transferred.</li> <li>• <b>TRANSFER_COMPLETED</b> — The open media interaction has been transferred</li> </ul> </td> </tr> </tbody> </table>			Name	Type	Description			<p>from a workbin.</p> <ul style="list-style-type: none"> <li>• <b>REVOKED</b> — The open media interaction has been revoked.</li> <li>• <b>COMPLETED</b> — The open media interaction has been completed (Mark as done).</li> <li>• <b>ERROR</b> — The open media interaction has an error.</li> <li>• <b>SAVED</b> — The open media interaction has been saved.</li> <li>• <b>TRANSFERRING</b> — The open media interaction is being transferred.</li> <li>• <b>TRANSFER_COMPLETED</b> — The open media interaction has been transferred</li> </ul>
Name	Type	Description							
		<p>from a workbin.</p> <ul style="list-style-type: none"> <li>• <b>REVOKED</b> — The open media interaction has been revoked.</li> <li>• <b>COMPLETED</b> — The open media interaction has been completed (Mark as done).</li> <li>• <b>ERROR</b> — The open media interaction has an error.</li> <li>• <b>SAVED</b> — The open media interaction has been saved.</li> <li>• <b>TRANSFERRING</b> — The open media interaction is being transferred.</li> <li>• <b>TRANSFER_COMPLETED</b> — The open media interaction has been transferred</li> </ul>							

Description	Represents the JSON structure of an interaction. Attributes specific to voice interactions are: callUuid, direction, callType, ani, dnis and recordingState.																										
	<table border="1"> <thead> <tr> <th data-bbox="820 380 1031 426">Name</th> <th data-bbox="1034 380 1239 426">Type</th> <th data-bbox="1242 380 1445 426">Description</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td data-bbox="1242 430 1445 573">                             and the transfer has been completed.                         </td> </tr> <tr> <td></td> <td></td> <td data-bbox="1242 577 1445 814"> <ul style="list-style-type: none"> <li>• INVITED_CONFERENCE — The open media interaction receives a conference invitation.</li> </ul> </td> </tr> <tr> <td></td> <td></td> <td data-bbox="1242 819 1445 1056"> <ul style="list-style-type: none"> <li>• LEFT_CONFERENCE — The open media interaction has left the conference.</li> </ul> </td> </tr> <tr> <td></td> <td></td> <td data-bbox="1242 1060 1445 1241"> <ul style="list-style-type: none"> <li>• USER_DATA_ATTACHED — Data has been attached to the interaction.</li> </ul> </td> </tr> <tr> <td></td> <td></td> <td data-bbox="1242 1245 1445 1446"> <ul style="list-style-type: none"> <li>• USER_DATA_UPDATED — The attached data has changed in the interaction.</li> </ul> </td> </tr> <tr> <td></td> <td></td> <td data-bbox="1242 1451 1445 1614"> <ul style="list-style-type: none"> <li>• JOIN_PENDING — Trying to join the chat session.</li> </ul> </td> </tr> <tr> <td></td> <td></td> <td data-bbox="1242 1619 1445 1803"> <ul style="list-style-type: none"> <li>• JOIN_FAILED — The connection with the chat server</li> </ul> </td> </tr> </tbody> </table>			Name	Type	Description			and the transfer has been completed.			<ul style="list-style-type: none"> <li>• INVITED_CONFERENCE — The open media interaction receives a conference invitation.</li> </ul>			<ul style="list-style-type: none"> <li>• LEFT_CONFERENCE — The open media interaction has left the conference.</li> </ul>			<ul style="list-style-type: none"> <li>• USER_DATA_ATTACHED — Data has been attached to the interaction.</li> </ul>			<ul style="list-style-type: none"> <li>• USER_DATA_UPDATED — The attached data has changed in the interaction.</li> </ul>			<ul style="list-style-type: none"> <li>• JOIN_PENDING — Trying to join the chat session.</li> </ul>			<ul style="list-style-type: none"> <li>• JOIN_FAILED — The connection with the chat server</li> </ul>
Name	Type	Description																									
		and the transfer has been completed.																									
		<ul style="list-style-type: none"> <li>• INVITED_CONFERENCE — The open media interaction receives a conference invitation.</li> </ul>																									
		<ul style="list-style-type: none"> <li>• LEFT_CONFERENCE — The open media interaction has left the conference.</li> </ul>																									
		<ul style="list-style-type: none"> <li>• USER_DATA_ATTACHED — Data has been attached to the interaction.</li> </ul>																									
		<ul style="list-style-type: none"> <li>• USER_DATA_UPDATED — The attached data has changed in the interaction.</li> </ul>																									
		<ul style="list-style-type: none"> <li>• JOIN_PENDING — Trying to join the chat session.</li> </ul>																									
		<ul style="list-style-type: none"> <li>• JOIN_FAILED — The connection with the chat server</li> </ul>																									

<p><b>Description</b></p>	<p>Represents the JSON structure of an interaction. Attributes specific to voice interactions are: callUuid, direction, callType, ani, dnis and recordingState.</p>														
	<table border="1"> <thead> <tr> <th data-bbox="824 384 1032 426">Name</th> <th data-bbox="1032 384 1239 426">Type</th> <th data-bbox="1239 384 1453 426">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="824 426 1032 1623"></td> <td data-bbox="1032 426 1239 1623"></td> <td data-bbox="1239 426 1453 1623"> <p>failed.</p> <ul style="list-style-type: none"> <li>HISTORY_IN_PROGRESS — Loading the content of the chat interaction.</li> <li>HISTORY_DONE — The content of the chat interaction has been loaded.</li> <li>CANCELLED — The outbound email is cancelled.</li> <li>SENT — The outbound email is sent.</li> <li>READY — The call preview is ready.</li> <li>CANCELED — The call preview is cancelled.</li> <li>REJECTED — The call preview is rejected.</li> </ul> </td> </tr> <tr> <td data-bbox="824 1623 1032 1717">previousState</td> <td data-bbox="1032 1623 1239 1717">string</td> <td data-bbox="1239 1623 1453 1717">The previous state of the interaction.</td> </tr> <tr> <td data-bbox="824 1717 1032 1812">parties</td> <td data-bbox="1032 1717 1239 1812">Array.</td> <td data-bbox="1239 1717 1453 1812">A collection of all the parties involved in</td> </tr> </tbody> </table>			Name	Type	Description			<p>failed.</p> <ul style="list-style-type: none"> <li>HISTORY_IN_PROGRESS — Loading the content of the chat interaction.</li> <li>HISTORY_DONE — The content of the chat interaction has been loaded.</li> <li>CANCELLED — The outbound email is cancelled.</li> <li>SENT — The outbound email is sent.</li> <li>READY — The call preview is ready.</li> <li>CANCELED — The call preview is cancelled.</li> <li>REJECTED — The call preview is rejected.</li> </ul>	previousState	string	The previous state of the interaction.	parties	Array.	A collection of all the parties involved in
Name	Type	Description													
		<p>failed.</p> <ul style="list-style-type: none"> <li>HISTORY_IN_PROGRESS — Loading the content of the chat interaction.</li> <li>HISTORY_DONE — The content of the chat interaction has been loaded.</li> <li>CANCELLED — The outbound email is cancelled.</li> <li>SENT — The outbound email is sent.</li> <li>READY — The call preview is ready.</li> <li>CANCELED — The call preview is cancelled.</li> <li>REJECTED — The call preview is rejected.</li> </ul>													
previousState	string	The previous state of the interaction.													
parties	Array.	A collection of all the parties involved in													

Description	Represents the JSON structure of an interaction. Attributes specific to voice interactions are: callUuid, direction, callType, ani, dnis and recordingState.																							
	<table border="1"> <thead> <tr> <th data-bbox="833 384 1032 426">Name</th> <th data-bbox="1032 384 1239 426">Type</th> <th data-bbox="1239 384 1437 426">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="833 426 1032 499"></td> <td data-bbox="1032 426 1239 499"></td> <td data-bbox="1239 426 1437 499">the interaction.</td> </tr> <tr> <td data-bbox="833 499 1032 701">isConsultation</td> <td data-bbox="1032 499 1239 701">boolean</td> <td data-bbox="1239 499 1437 701">This property is true if the interaction is a consultation; otherwise, it's false.</td> </tr> <tr> <td data-bbox="833 701 1032 1272">isMainCaseInteraction</td> <td data-bbox="1032 701 1239 1272">boolean</td> <td data-bbox="1239 701 1437 1272">This property is true if the interaction is the main interaction in the customer case; otherwise, it's false. In Workspace Web Edition, the main interaction is related to Case Information, Disposition, Note, Contact Profile, and so on.</td> </tr> <tr> <td data-bbox="833 1272 1032 1423">callUuid</td> <td data-bbox="1032 1272 1239 1423">string</td> <td data-bbox="1239 1272 1437 1423">The UUID of the call. This attribute is only on voice interactions.</td> </tr> <tr> <td data-bbox="833 1423 1032 1717">direction</td> <td data-bbox="1032 1423 1239 1717">string</td> <td data-bbox="1239 1423 1437 1717">The call direction. Possible values are: IN, OUT or UNKNOWN. This attribute is only on voice interactions.</td> </tr> <tr> <td data-bbox="833 1717 1032 1808">callType</td> <td data-bbox="1032 1717 1239 1808">string</td> <td data-bbox="1239 1717 1437 1808">The call type. Possible values are:</td> </tr> </tbody> </table>			Name	Type	Description			the interaction.	isConsultation	boolean	This property is true if the interaction is a consultation; otherwise, it's false.	isMainCaseInteraction	boolean	This property is true if the interaction is the main interaction in the customer case; otherwise, it's false. In Workspace Web Edition, the main interaction is related to Case Information, Disposition, Note, Contact Profile, and so on.	callUuid	string	The UUID of the call. This attribute is only on voice interactions.	direction	string	The call direction. Possible values are: IN, OUT or UNKNOWN. This attribute is only on voice interactions.	callType	string	The call type. Possible values are:
Name	Type	Description																						
		the interaction.																						
isConsultation	boolean	This property is true if the interaction is a consultation; otherwise, it's false.																						
isMainCaseInteraction	boolean	This property is true if the interaction is the main interaction in the customer case; otherwise, it's false. In Workspace Web Edition, the main interaction is related to Case Information, Disposition, Note, Contact Profile, and so on.																						
callUuid	string	The UUID of the call. This attribute is only on voice interactions.																						
direction	string	The call direction. Possible values are: IN, OUT or UNKNOWN. This attribute is only on voice interactions.																						
callType	string	The call type. Possible values are:																						

Description	Represents the JSON structure of an interaction. Attributes specific to voice interactions are: callUuid, direction, callType, ani, dnis and recordingState.																							
	<table border="1"> <thead> <tr> <th data-bbox="829 384 1032 426">Name</th> <th data-bbox="1032 384 1235 426">Type</th> <th data-bbox="1235 384 1448 426">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="829 426 1032 688"></td> <td data-bbox="1032 426 1235 688"></td> <td data-bbox="1235 426 1448 688">                             INTERNAL, INBOUND, OUTBOUND, CONSULT or UNKNOWN. This attribute is only on voice interactions.                         </td> </tr> <tr> <td data-bbox="829 688 1032 898">ani</td> <td data-bbox="1032 688 1235 898">string</td> <td data-bbox="1235 688 1448 898">                             The Automatic Number Identification service. This attribute is only on voice interactions.                         </td> </tr> <tr> <td data-bbox="829 898 1032 1108">dnis</td> <td data-bbox="1032 898 1235 1108">string</td> <td data-bbox="1235 898 1448 1108">                             The Dialed Number Identification Service. This attribute is only on voice interactions.                         </td> </tr> <tr> <td data-bbox="829 1108 1032 1423">recordingState</td> <td data-bbox="1032 1108 1235 1423">string</td> <td data-bbox="1235 1108 1448 1423">                             The call recording state. Possible values are: STOPPED, RECORDING or PAUSED. This attribute is only on voice interactions.                         </td> </tr> <tr> <td data-bbox="829 1423 1032 1633">isCaseSelected</td> <td data-bbox="1032 1423 1235 1633">boolean</td> <td data-bbox="1235 1423 1448 1633">                             Is true if the case containing this interaction is selected, otherwise is false.                         </td> </tr> <tr> <td data-bbox="829 1633 1032 1816">ronaCallState</td> <td data-bbox="1032 1633 1235 1816">string</td> <td data-bbox="1235 1633 1448 1816">                             This value is populated on event RELEASED when an agent receives                         </td> </tr> </tbody> </table>			Name	Type	Description			INTERNAL, INBOUND, OUTBOUND, CONSULT or UNKNOWN. This attribute is only on voice interactions.	ani	string	The Automatic Number Identification service. This attribute is only on voice interactions.	dnis	string	The Dialed Number Identification Service. This attribute is only on voice interactions.	recordingState	string	The call recording state. Possible values are: STOPPED, RECORDING or PAUSED. This attribute is only on voice interactions.	isCaseSelected	boolean	Is true if the case containing this interaction is selected, otherwise is false.	ronaCallState	string	This value is populated on event RELEASED when an agent receives
Name	Type	Description																						
		INTERNAL, INBOUND, OUTBOUND, CONSULT or UNKNOWN. This attribute is only on voice interactions.																						
ani	string	The Automatic Number Identification service. This attribute is only on voice interactions.																						
dnis	string	The Dialed Number Identification Service. This attribute is only on voice interactions.																						
recordingState	string	The call recording state. Possible values are: STOPPED, RECORDING or PAUSED. This attribute is only on voice interactions.																						
isCaseSelected	boolean	Is true if the case containing this interaction is selected, otherwise is false.																						
ronaCallState	string	This value is populated on event RELEASED when an agent receives																						

<p><b>Description</b></p>	<p>Represents the JSON structure of an interaction. Attributes specific to voice interactions are: callUuid, direction, callType, ani, dnis and recordingState.</p>																				
	<table border="1"> <thead> <tr> <th data-bbox="824 384 1032 426">Name</th> <th data-bbox="1032 384 1243 426">Type</th> <th data-bbox="1243 384 1445 426">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="824 426 1032 663"></td> <td data-bbox="1032 426 1243 663"></td> <td data-bbox="1243 426 1445 663"> <p>an inbound call and does not answer. Possible values are: REDIRECTED or NO_ANSWER.</p> </td> </tr> <tr> <td data-bbox="824 663 1032 867">isCaseExpanded</td> <td data-bbox="1032 663 1243 867">boolean</td> <td data-bbox="1243 663 1445 867"> <p>Is true if the case containing this interaction is expanded, otherwise is false.</p> </td> </tr> <tr> <td data-bbox="824 867 1032 1104">interactionUUID</td> <td data-bbox="1032 867 1243 1104">string</td> <td data-bbox="1243 867 1445 1104"> <p>The attr_itx_id for a multimedia interaction or the callUuid for a voice interaction.</p> </td> </tr> <tr> <td data-bbox="824 1104 1032 1230">connId</td> <td data-bbox="1032 1104 1243 1230">string</td> <td data-bbox="1243 1104 1445 1230"> <p>The unique connection ID from the T-Server.</p> </td> </tr> <tr> <td data-bbox="824 1230 1032 1356">contact</td> <td data-bbox="1032 1230 1243 1356">interaction.Contact</td> <td data-bbox="1243 1230 1445 1356"> <p>An object representing the contact's information.</p> </td> </tr> </tbody> </table>			Name	Type	Description			<p>an inbound call and does not answer. Possible values are: REDIRECTED or NO_ANSWER.</p>	isCaseExpanded	boolean	<p>Is true if the case containing this interaction is expanded, otherwise is false.</p>	interactionUUID	string	<p>The attr_itx_id for a multimedia interaction or the callUuid for a voice interaction.</p>	connId	string	<p>The unique connection ID from the T-Server.</p>	contact	interaction.Contact	<p>An object representing the contact's information.</p>
Name	Type	Description																			
		<p>an inbound call and does not answer. Possible values are: REDIRECTED or NO_ANSWER.</p>																			
isCaseExpanded	boolean	<p>Is true if the case containing this interaction is expanded, otherwise is false.</p>																			
interactionUUID	string	<p>The attr_itx_id for a multimedia interaction or the callUuid for a voice interaction.</p>																			
connId	string	<p>The unique connection ID from the T-Server.</p>																			
contact	interaction.Contact	<p>An object representing the contact's information.</p>																			

Party

<p><b>Description</b></p>	<p>Represents the JSON structure of a party.</p>								
<p><b>Type</b></p>	<p>Object</p>								
<p><b>Properties</b></p>	<table border="1"> <thead> <tr> <th data-bbox="824 1591 1032 1633">Name</th> <th data-bbox="1032 1591 1243 1633">Type</th> <th data-bbox="1243 1591 1445 1633">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="824 1633 1032 1705">name</td> <td data-bbox="1032 1633 1243 1705">string</td> <td data-bbox="1243 1633 1445 1705"> <p>The name of the party.</p> </td> </tr> </tbody> </table>			Name	Type	Description	name	string	<p>The name of the party.</p>
Name	Type	Description							
name	string	<p>The name of the party.</p>							

## Contact

<b>Description</b>	Represents the JSON structure of a contact.														
<b>Type</b>	Object														
<b>Properties</b>	<table border="1"><thead><tr><th><b>Name</b></th><th><b>Type</b></th><th><b>Description</b></th></tr></thead><tbody><tr><td>displayName</td><td>string</td><td>The contact's display name.</td></tr><tr><td>firstName</td><td>string</td><td>The contact's first name.</td></tr><tr><td>lastName</td><td>string</td><td>The contact's last name.</td></tr></tbody></table>	<b>Name</b>	<b>Type</b>	<b>Description</b>	displayName	string	The contact's display name.	firstName	string	The contact's first name.	lastName	string	The contact's last name.		
<b>Name</b>	<b>Type</b>	<b>Description</b>													
displayName	string	The contact's display name.													
firstName	string	The contact's first name.													
lastName	string	The contact's last name.													

# Media namespace

## Contents

- **1 Methods**
  - **1.1 getMediaList**
  - **1.2 getMediaByName**
  - **1.3 setState**
- **2 Type definitions**
  - **2.1 Media**
  - **2.2 State**
  - **2.3 Device**

Learn about the Media namespace methods and type definitions in the Service Client API.

## Methods

The Media namespace includes the following methods:

- `getMediaList`
- `getMediaByName`
- `setState`

### getMediaList

<b>Signature</b>	<code>getMediaList(succeeded, failed) → {Array.}</code>									
<b>Description</b>	Get the list of media with attributes.									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code>succeeded</code></td><td>function</td><td>A function called when the operation succeeds.</td></tr><tr><td><code>failed</code></td><td>function</td><td>A function called when the operation fails.</td></tr></tbody></table>	Name	Type	Description	<code>succeeded</code>	function	A function called when the operation succeeds.	<code>failed</code>	function	A function called when the operation fails.
Name	Type	Description								
<code>succeeded</code>	function	A function called when the operation succeeds.								
<code>failed</code>	function	A function called when the operation fails.								
<b>Returns</b>	Array.									

### getMediaByName

<b>Signature</b>	<code>getMediaByName(name, succeeded, failed)</code>												
<b>Description</b>	Get the media attributes.												
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code>name</code></td><td>string</td><td>The media name.</td></tr><tr><td><code>succeeded</code></td><td>function</td><td>A function called when the operation succeeds.</td></tr><tr><td><code>failed</code></td><td>function</td><td>A function</td></tr></tbody></table>	Name	Type	Description	<code>name</code>	string	The media name.	<code>succeeded</code>	function	A function called when the operation succeeds.	<code>failed</code>	function	A function
Name	Type	Description											
<code>name</code>	string	The media name.											
<code>succeeded</code>	function	A function called when the operation succeeds.											
<code>failed</code>	function	A function											

<b>Signature</b>	getMediaByName( <i>name</i> , succeeded, failed)		
	<b>Name</b>	<b>Type</b>	<b>Description</b>
			called when the operation fails.

## setState

<b>Signature</b>	setState( <i>name</i> , <i>stateOperationName</i> , succeeded, failed)		
<b>Description</b>	Sets the media state.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	name	string	The media name.
	stateOperationName	string	An operationName from the agent states list. See State.
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.

## Type definitions

The Media namespace includes the following object types:

- Media
- State
- Device

### Media

<b>Description</b>	Represents the JSON structure of a media.
<b>Type</b>	Object

Description	Represents the JSON structure of a media.		
Properties	Name	Type	Description
	name	string	The media name.
	state	media.State	The media state object.

## State

Description	Represents the JSON structure of a media state.		
Type	Object		
Properties	Name	Type	Description
	type	string	The type of operation. Possible values are: <ul style="list-style-type: none"> <li>• LOGOUT</li> <li>• READY</li> <li>• PARTIAL_READY *</li> <li>• NOT_READY</li> <li>• NOT_READY_ACTION_CODE</li> <li>• NOT_READY_AFTER_CALLW</li> <li>• NOT_READY_AFTER_CALLW</li> <li>• DND_ON</li> <li>• OUT_OF_SERVICE *</li> <li>• LOGOUT_DND_ON *</li> <li>• UNKNOWN *</li> </ul>
	displayName	string	The display name of the state.
	operationName	string	The operation name to use with agent.setState and media.setState.

\* States that are limited to an event and can't be applied by code

## Device

<b>Description</b>	Represents the JSON structure of a media.		
<b>Type</b>	Object		
<b>Properties</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	number	string	The phone number configured for an agent - the physical DN.  <b>Note:</b> This property is applicable only for voice data.
	dynamicPhoneNumber	string	The dynamic phone number configured for the agent for the session.  <b>Note:</b> This property is applicable only for voice data. This property is applicable only when there is an alternate phone number and applicable for the current session only.

# System namespace

## Contents

- **1 Methods**
  - 1.1 `amlVisible`
  - 1.2 `closeDialog`
  - 1.3 `closeToast`
  - 1.4 `closeViewInApplicationMenuBar`
  - 1.5 `getAllowedServices`
  - 1.6 `isFrameLeading`
  - 1.7 `isFrameFollowing`
  - 1.8 `isFrameNegotiating`
  - 1.9 `isFrameLeadingOrNegotiating`
  - 1.10 `isLastActiveFrame`
  - 1.11 `openDialog`
  - 1.12 `popupToast`
  - 1.13 `triggerActivity`
  - 1.14 `updateViewInApplicationMenuBar`
  - 1.15 `updateToast`

Learn about the System namespace methods in the Service Client API.

### Important

Depending on your environment, you might need to contact your Genesys representative to complete the configuration described on this page.

## Methods

The System namespace includes the following methods:

- amIVisible
- closeDialog
- closeToast
- closeViewInApplicationMenuBar
- getAllowedServices
- isFrameLeading
- isFrameFollowing
- isFrameNegotiating
- isFrameLeadingOrNegotiating
- isLastActiveFrame
- popupToast
- openDialog
- triggerActivity
- updateViewInApplicationMenuBar
- updateToast

### amIVisible

<b>Signature</b>	amIVisible(succeeded, failed) → {boolean}
<b>Description</b>	Get the current visibility state of the frame.

<b>Signature</b>	amIVisible(succeeded, failed) → {boolean}		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.
<b>Returns</b>	true if the frame is visible.		

## Sample request

```
setTimeout(function() {
  genesys.wwe.service.system.amIVisible(succeeded, failed);
}, 3000); // This gives 3 seconds to switch the panel to test.
```

## Sample response

The asynchronous answer is included in the `data` attribute:

```
{
  "request": "system.amIVisible",
  "data": true,
  "userAgent": "WWE Server",
  "protocolVersion": 2
}
```

## closeDialog

<b>Signature</b>	closeDialog(dialogId, succeeded, failed) → {boolean}		
<b>Description</b>	Close a previously opened dialog.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	dialogId	string	The dialog identifier (returned in the response of openDialog).
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.

<b>Signature</b>	<code>closeDialog(dialogId, succeeded, failed) → {boolean}</code>
<b>Returns</b>	true if the dialog is closed; false if the dialog is not found.

## closeToast

<b>Signature</b>	<code>closeToast(id, succeeded, failed) → {boolean}</code>														
<b>Description</b>	Closes the specified toast.														
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>id</td> <td>string</td> <td>The identifier of the toast to close. The identifier is returned by the <code>popupToast</code> method.</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	id	string	The identifier of the toast to close. The identifier is returned by the <code>popupToast</code> method.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.		
Name	Type	Description													
id	string	The identifier of the toast to close. The identifier is returned by the <code>popupToast</code> method.													
succeeded	function	A function called when the operation succeeds.													
failed	function	A function called when the operation fails.													
<b>Returns</b>	true if the toast has been updated; false if the toast identifier has not been found.														

## closeViewInApplicationMenuBar

<b>Signature</b>	<code>closeViewInApplicationMenuBar(parameters, succeeded, failed) → {boolean}</code>														
<b>Description</b>	Removes the given view from the <b>Application Menu</b> bar region.														
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>name</td> <td>string</td> <td>The name of the custom view to be removed.</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function</td> </tr> </tbody> </table>	Name	Type	Description	name	string	The name of the custom view to be removed.	succeeded	function	A function called when the operation succeeds.	failed	function	A function		
Name	Type	Description													
name	string	The name of the custom view to be removed.													
succeeded	function	A function called when the operation succeeds.													
failed	function	A function													

<b>Signature</b>	closeViewInApplicationMenuBar(parameters, succeeded, failed) → {boolean}		
	<b>Name</b>	<b>Type</b>	<b>Description</b>
			called when the operation fails.
<b>Returns</b>	true if the view is removed; false if the view name is not found.		

### Sample request

```
genesys.wwe.service.system.closeViewInApplicationMenuBar("view1", succeeded, failed)
```

### Sample response

```
{
  "request": "system.closeViewInApplicationMenuBar",
  "data": true,
  "userAgent": "WWE Server",
  "protocolVersion": 2
}
```

### Sample request

```
genesys.wwe.service.system.closeDialog("wweCustomDialog1", succeeded, failed)
```

### Sample response

The asynchronous answer is included in the data attribute:

```
{
  "request": "system.closeDialog",
  "data": true,
  "userAgent": "WWE Server",
  "protocolVersion": 2
}
```

## getAllowedServices

<b>Signature</b>	getAllowedServices(succeeded, failed) → {Array.}		
<b>Description</b>	Gets the list of allowed services, as determined by the Security configuration. If the domain of the web application that calls this method isn't listed in the service-client-api.accepted-web-content-origins option, then this method fails.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	succeeded	function	A function called when the operation succeeds.

<b>Signature</b>	getAllowedServices(succeeded, failed) → {Array.}								
	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>			Name	Type	Description	failed	function	A function called when the operation fails.
Name	Type	Description							
failed	function	A function called when the operation fails.							
<b>Returns</b>	Array.								

### isFrameLeading

<b>Signature</b>	isFrameLeading( <i>succeeded, failed</i> ) → {boolean}											
<b>Description</b>	Find out if the browser tab is leading.											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>			Name	Type	Description	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.
Name	Type	Description										
succeeded	function	A function called when the operation succeeds.										
failed	function	A function called when the operation fails.										
<b>Returns</b>	true if the browser tab is the leader.											

### isFrameFollowing

<b>Signature</b>	isFrameFollowing( <i>succeeded, failed</i> ) → {boolean}											
<b>Description</b>	Find out if the browser tab is following.											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>			Name	Type	Description	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.
Name	Type	Description										
succeeded	function	A function called when the operation succeeds.										
failed	function	A function called when the operation fails.										
<b>Returns</b>	true if this browser tab is following.											

## isFrameNegotiating

<b>Signature</b>	isFrameNegotiating( <i>succeeded, failed</i> ) → {boolean}											
<b>Description</b>	Find out if there is an election in progress and the browser tab state is not yet set to leading or following (the tab is "negotiating.")											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.		
Name	Type	Description										
succeeded	function	A function called when the operation succeeds.										
failed	function	A function called when the operation fails.										
<b>Returns</b>	true if the tab is negotiating.											

## isFrameLeadingOrNegotiating

<b>Signature</b>	isFrameLeadingOrNegotiating( <i>succeeded, failed</i> ) → {boolean}											
<b>Description</b>	Find out if the browser tab is leading or there is an election in progress and the tab state is not yet set to leading or following (the tab is "negotiating.").											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.		
Name	Type	Description										
succeeded	function	A function called when the operation succeeds.										
failed	function	A function called when the operation fails.										
<b>Returns</b>	true if the browser tab is leading or negotiating.											

## isLastActiveFrame

<b>Signature</b>	isLastActiveFrame( <i>succeeded, failed</i> ) → {boolean}								
<b>Description</b>	Find out if this is the last active browser tab.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>succeeded</td> <td>function</td> <td>A function called when</td> </tr> </tbody> </table>	Name	Type	Description	succeeded	function	A function called when		
Name	Type	Description							
succeeded	function	A function called when							

<b>Signature</b>	isLastActiveFrame( <i>succeeded, failed</i> ) → {boolean}		
	<b>Name</b>	<b>Type</b>	<b>Description</b>
			the operation succeeds.
	failed	function	A function called when the operation fails.
<b>Returns</b>	true if this is the last active browser tab.		

### openDialog

<b>Signature</b>	openDialog( <i>url, options, succeeded, failed</i> ) → {string}		
<b>Description</b>	Open an iframe in a dialog, based on the configured parameters.		
	<b>Name</b>	<b>Type</b>	<b>Description</b>
	url	string	The URL of the iframe to load in the dialog.
<b>Parameters</b>	options	object	<p>Optional parameters to configure the dialog. This value can't be null, so you must pass {} if there are no specific options. You can include any of the following options:</p> <ul style="list-style-type: none"> <li>label - Set a custom value for the aria-label attribute on the dialog. When the dialog pops up, this value identifies it</li> </ul>

Signature	openDialog( <i>url, options, succeeded, failed</i> ) → {string}														
	<table border="1"> <thead> <tr> <th data-bbox="833 327 1031 369">Name</th> <th data-bbox="1031 327 1239 369">Type</th> <th data-bbox="1239 327 1445 369">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="833 369 1031 1056"></td> <td data-bbox="1031 369 1239 1056"></td> <td data-bbox="1239 369 1445 1056">                     to accessibility tools like screen readers.                     <ul style="list-style-type: none"> <li>width - The initial width of the dialog. Valid formats are px or %.</li> <li>height - The initial height of the dialog. Valid formats are px or %.</li> </ul> </td> </tr> <tr> <td data-bbox="833 1056 1031 1182">succeeded</td> <td data-bbox="1031 1056 1239 1182">function</td> <td data-bbox="1239 1056 1445 1182">A function called when the operation succeeds.</td> </tr> <tr> <td data-bbox="833 1182 1031 1308">failed</td> <td data-bbox="1031 1182 1239 1308">function</td> <td data-bbox="1239 1182 1445 1308">A function called when the operation fails.</td> </tr> </tbody> </table>			Name	Type	Description			to accessibility tools like screen readers. <ul style="list-style-type: none"> <li>width - The initial width of the dialog. Valid formats are px or %.</li> <li>height - The initial height of the dialog. Valid formats are px or %.</li> </ul>	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.
Name	Type	Description													
		to accessibility tools like screen readers. <ul style="list-style-type: none"> <li>width - The initial width of the dialog. Valid formats are px or %.</li> <li>height - The initial height of the dialog. Valid formats are px or %.</li> </ul>													
succeeded	function	A function called when the operation succeeds.													
failed	function	A function called when the operation fails.													
Returns	The dialog identifier or null if the url parameter is not defined.														

### Sample request

```
genesys.wwe.service.system.openDialog("", {
  label: "Dialog $Agent.FullName$",
  width: "430px",
  height: "325px"
}, succeeded, failed)
```

### Sample response

The asynchronous answer is included in the data attribute:

```
{
  "request": "system.openDialog",
  "data": "wweCustomDialog1",
```

```
"userAgent": "WWE Server",
"protocolVersion": 2
}
```

## popupToast

Signature	popupToast( <i>parameters</i> , succeeded, failed) → {string}																													
Description	Pops up a new custom toast.																													
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td> <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>title</td> <td>string</td> <td>The title</td> </tr> <tr> <td>iconUrl</td> <td>string</td> <td>The URL of the icon you want to display in the title bar of the custom toast popup.</td> </tr> <tr> <td>subject</td> <td>string</td> <td>Optional. The subject</td> </tr> <tr> <td>message</td> <td>string</td> <td>Optional. The message</td> </tr> <tr> <td>keyValues</td> <td>array</td> <td>Optional. JSON object used to fill the key value pair list. For example:</td> </tr> </tbody> </table> </td> </tr> <tr> <td>parameters</td> <td>object</td> <td></td> </tr> </tbody> </table>			Name	Type	Description			<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>title</td> <td>string</td> <td>The title</td> </tr> <tr> <td>iconUrl</td> <td>string</td> <td>The URL of the icon you want to display in the title bar of the custom toast popup.</td> </tr> <tr> <td>subject</td> <td>string</td> <td>Optional. The subject</td> </tr> <tr> <td>message</td> <td>string</td> <td>Optional. The message</td> </tr> <tr> <td>keyValues</td> <td>array</td> <td>Optional. JSON object used to fill the key value pair list. For example:</td> </tr> </tbody> </table>	Name	Type	Description	title	string	The title	iconUrl	string	The URL of the icon you want to display in the title bar of the custom toast popup.	subject	string	Optional. The subject	message	string	Optional. The message	keyValues	array	Optional. JSON object used to fill the key value pair list. For example:	parameters	object	
	Name	Type	Description																											
		<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>title</td> <td>string</td> <td>The title</td> </tr> <tr> <td>iconUrl</td> <td>string</td> <td>The URL of the icon you want to display in the title bar of the custom toast popup.</td> </tr> <tr> <td>subject</td> <td>string</td> <td>Optional. The subject</td> </tr> <tr> <td>message</td> <td>string</td> <td>Optional. The message</td> </tr> <tr> <td>keyValues</td> <td>array</td> <td>Optional. JSON object used to fill the key value pair list. For example:</td> </tr> </tbody> </table>	Name	Type	Description	title	string	The title	iconUrl	string	The URL of the icon you want to display in the title bar of the custom toast popup.	subject	string	Optional. The subject	message	string	Optional. The message	keyValues	array	Optional. JSON object used to fill the key value pair list. For example:										
Name	Type	Description																												
title	string	The title																												
iconUrl	string	The URL of the icon you want to display in the title bar of the custom toast popup.																												
subject	string	Optional. The subject																												
message	string	Optional. The message																												
keyValues	array	Optional. JSON object used to fill the key value pair list. For example:																												
parameters	object																													

Signature	popupToast( <i>parameters</i> , succeeded, failed) → {string}														
	Name	Type	Description												
			<table border="1"> <thead> <tr> <th data-bbox="1253 394 1318 436">Name</th> <th data-bbox="1318 394 1383 436">Type</th> <th data-bbox="1383 394 1461 436">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="1253 436 1318 638"></td> <td data-bbox="1318 436 1383 638"></td> <td data-bbox="1383 436 1461 638">{"key1" ; "value one", "key2" ; "value two", "key3" ; "value three"}.</td> </tr> <tr> <td data-bbox="1253 638 1318 1520"></td> <td data-bbox="1318 638 1383 1520">Array</td> <td data-bbox="1383 638 1461 1520">Optional. Each character string in this array becomes a button. All buttons are displayed as buttons, not hyperlinks, in the following order: [Button 2] [Button 3] ... [Button N] [Button 1].</td> </tr> <tr> <td data-bbox="1253 1520 1318 1841"></td> <td data-bbox="1318 1520 1383 1841">boolean</td> <td data-bbox="1383 1520 1461 1841">Optional. If set to true, displays the <b>Show and Dismiss</b></td> </tr> </tbody> </table>	Name	Type	Description			{"key1" ; "value one", "key2" ; "value two", "key3" ; "value three"}.		Array	Optional. Each character string in this array becomes a button. All buttons are displayed as buttons, not hyperlinks, in the following order: [Button 2] [Button 3] ... [Button N] [Button 1].		boolean	Optional. If set to true, displays the <b>Show and Dismiss</b>
Name	Type	Description													
		{"key1" ; "value one", "key2" ; "value two", "key3" ; "value three"}.													
	Array	Optional. Each character string in this array becomes a button. All buttons are displayed as buttons, not hyperlinks, in the following order: [Button 2] [Button 3] ... [Button N] [Button 1].													
	boolean	Optional. If set to true, displays the <b>Show and Dismiss</b>													

Signature	popupToast( <i>parameters</i> , succeeded, failed) → {string}														
	Name	Type	Description												
			<table border="1"> <thead> <tr> <th data-bbox="1252 390 1312 426">Name</th> <th data-bbox="1312 390 1372 426">Type</th> <th data-bbox="1372 390 1461 426">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="1252 426 1312 1220"></td> <td data-bbox="1312 426 1372 1220"></td> <td data-bbox="1372 426 1461 1220">                     buttons and pops up the current iframe if the <b>Show</b> button is pushed. If set to false, displays "OK" or custom buttons based on the parameter's buttons.                 </td> </tr> <tr> <td data-bbox="1252 1220 1312 1682">autoClose</td> <td data-bbox="1312 1220 1372 1682">boolean</td> <td data-bbox="1372 1220 1461 1682">Optional. If set to greater than 0, the popup is automatically closed after the specified milliseconds.</td> </tr> <tr> <td data-bbox="1252 1682 1312 1841">sendTo</td> <td data-bbox="1312 1682 1372 1841">string</td> <td data-bbox="1372 1682 1461 1841">Optional. If set to</td> </tr> </tbody> </table>	Name	Type	Description			buttons and pops up the current iframe if the <b>Show</b> button is pushed. If set to false, displays "OK" or custom buttons based on the parameter's buttons.	autoClose	boolean	Optional. If set to greater than 0, the popup is automatically closed after the specified milliseconds.	sendTo	string	Optional. If set to
Name	Type	Description													
		buttons and pops up the current iframe if the <b>Show</b> button is pushed. If set to false, displays "OK" or custom buttons based on the parameter's buttons.													
autoClose	boolean	Optional. If set to greater than 0, the popup is automatically closed after the specified milliseconds.													
sendTo	string	Optional. If set to													

Signature	popupToast( <i>parameters</i> , succeeded, failed) → {string}																							
	<table border="1"> <thead> <tr> <th data-bbox="820 321 1031 373">Name</th> <th data-bbox="1031 321 1242 373">Type</th> <th data-bbox="1242 321 1453 373">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="820 373 1031 1455"></td> <td data-bbox="1031 373 1242 1455"></td> <td data-bbox="1242 373 1453 1455"> <table border="1"> <thead> <tr> <th data-bbox="1250 384 1307 436">Name</th> <th data-bbox="1307 384 1364 436">Type</th> <th data-bbox="1364 384 1445 436">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="1250 436 1307 835"></td> <td data-bbox="1307 436 1364 835"></td> <td data-bbox="1364 436 1445 835">true, sends the <b>subject</b>, <b>iconUrl</b>, <b>title</b>, <b>keyValues</b>, and <b>message</b> parameters to the <b>MyMessage</b> panel.</td> </tr> <tr> <td data-bbox="1250 835 1307 1444">width</td> <td data-bbox="1307 835 1364 1444">number</td> <td data-bbox="1364 835 1445 1444">Optional. The width of the custom toast popup, in pixels. This value takes precedence over the service-client-api.toast.width configuration option.</td> </tr> </tbody> </table> </td> </tr> <tr> <td data-bbox="820 1455 1031 1575">succeeded</td> <td data-bbox="1031 1455 1242 1575">function</td> <td data-bbox="1242 1455 1453 1575">A function called when the operation succeeds.</td> </tr> <tr> <td data-bbox="820 1575 1031 1707">failed</td> <td data-bbox="1031 1575 1242 1707">function</td> <td data-bbox="1242 1575 1453 1707">A function called when the operation fails.</td> </tr> </tbody> </table>			Name	Type	Description			<table border="1"> <thead> <tr> <th data-bbox="1250 384 1307 436">Name</th> <th data-bbox="1307 384 1364 436">Type</th> <th data-bbox="1364 384 1445 436">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="1250 436 1307 835"></td> <td data-bbox="1307 436 1364 835"></td> <td data-bbox="1364 436 1445 835">true, sends the <b>subject</b>, <b>iconUrl</b>, <b>title</b>, <b>keyValues</b>, and <b>message</b> parameters to the <b>MyMessage</b> panel.</td> </tr> <tr> <td data-bbox="1250 835 1307 1444">width</td> <td data-bbox="1307 835 1364 1444">number</td> <td data-bbox="1364 835 1445 1444">Optional. The width of the custom toast popup, in pixels. This value takes precedence over the service-client-api.toast.width configuration option.</td> </tr> </tbody> </table>	Name	Type	Description			true, sends the <b>subject</b> , <b>iconUrl</b> , <b>title</b> , <b>keyValues</b> , and <b>message</b> parameters to the <b>MyMessage</b> panel.	width	number	Optional. The width of the custom toast popup, in pixels. This value takes precedence over the service-client-api.toast.width configuration option.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.
Name	Type	Description																						
		<table border="1"> <thead> <tr> <th data-bbox="1250 384 1307 436">Name</th> <th data-bbox="1307 384 1364 436">Type</th> <th data-bbox="1364 384 1445 436">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="1250 436 1307 835"></td> <td data-bbox="1307 436 1364 835"></td> <td data-bbox="1364 436 1445 835">true, sends the <b>subject</b>, <b>iconUrl</b>, <b>title</b>, <b>keyValues</b>, and <b>message</b> parameters to the <b>MyMessage</b> panel.</td> </tr> <tr> <td data-bbox="1250 835 1307 1444">width</td> <td data-bbox="1307 835 1364 1444">number</td> <td data-bbox="1364 835 1445 1444">Optional. The width of the custom toast popup, in pixels. This value takes precedence over the service-client-api.toast.width configuration option.</td> </tr> </tbody> </table>	Name	Type	Description			true, sends the <b>subject</b> , <b>iconUrl</b> , <b>title</b> , <b>keyValues</b> , and <b>message</b> parameters to the <b>MyMessage</b> panel.	width	number	Optional. The width of the custom toast popup, in pixels. This value takes precedence over the service-client-api.toast.width configuration option.													
Name	Type	Description																						
		true, sends the <b>subject</b> , <b>iconUrl</b> , <b>title</b> , <b>keyValues</b> , and <b>message</b> parameters to the <b>MyMessage</b> panel.																						
width	number	Optional. The width of the custom toast popup, in pixels. This value takes precedence over the service-client-api.toast.width configuration option.																						
succeeded	function	A function called when the operation succeeds.																						
failed	function	A function called when the operation fails.																						
Returns	A unique identifier																							

## triggerActivity

<b>Signature</b>	triggerActivity(succeeded, failed)											
<b>Description</b>	Triggers a fake activity to prevent the inactivity timer from closing the agent session.											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.		
Name	Type	Description										
succeeded	function	A function called when the operation succeeds.										
failed	function	A function called when the operation fails.										

## updateViewInApplicationMenuBar

<b>Signature</b>	updateViewInApplicationMenuBar(parameters, succeeded, failed) → {string}											
<b>Description</b>	Creates a custom view in the <b>Application Menu bar</b> region.											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>name</td> <td>string</td> <td>A unique name for the custom view of the <b>Application Menu bar</b> that is to be created or updated. If a view with the given name already exists, it will be updated, otherwise, a new view will be created.</td> </tr> <tr> <td>iconUrl</td> <td>string</td> <td>The URL of the icon you want to display in the custom view of the <b>Application Menu bar</b> region. This parameter is</td> </tr> </tbody> </table>	Name	Type	Description	name	string	A unique name for the custom view of the <b>Application Menu bar</b> that is to be created or updated. If a view with the given name already exists, it will be updated, otherwise, a new view will be created.	iconUrl	string	The URL of the icon you want to display in the custom view of the <b>Application Menu bar</b> region. This parameter is		
Name	Type	Description										
name	string	A unique name for the custom view of the <b>Application Menu bar</b> that is to be created or updated. If a view with the given name already exists, it will be updated, otherwise, a new view will be created.										
iconUrl	string	The URL of the icon you want to display in the custom view of the <b>Application Menu bar</b> region. This parameter is										

Signature	updateViewInApplicationMenuBar(parameters, succeeded, failed) → {string}		
	Name	Type	Description
			mandatory if <b>label</b> is not provided.
	label	string	The main textual content to be displayed for the custom view. This parameter is mandatory if <b>iconUrl</b> is not provided.
	shortLabel	string	Optional. A shorter version of the <b>label</b> that will be used in the shortened mode if <b>iconUrl</b> is not available.
	tooltip	string	Optional. The tooltip content to be shown when the mouse is hovered on the custom view.
	labelColor	string	Optional. The color of the label text in case-insensitive hex color code format, for example, #FFFFFF.
	backgroundColor	string	Optional. The background color of the region where icon and title are displayed. The format of the background color is case-

Signature	updateViewInApplicationMenuBar(parameters, succeeded, failed) → {string}		
	<b>Name</b>	<b>Type</b>	<b>Description</b>
			insensitive hex color code, for example, #FFFFFF. By default, it is usually the same color as the navigation bar.
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.
Returns	View name if successful.		

### Sample request

```
genesys.wwe.service.system.updateViewInApplicationMenuBar({
  name: "view1",
  iconUrl: "https://cdn1.iconfinder.com/data/icons/free-social-media-12/32/
RSS_social_media-128.png",
  label: "Main content text",
  shortLabel: "Short text",
  tooltip: "Tooltip text",
  labelColor: "#FFFFFF",
  backgroundColor: "#000000"
}, succeeded, failed)'
```

### Sample response

```
{
  "request": "system.updateViewInApplicationMenuBar",
  "data": "view1",
  "userAgent": "WWE Server",
  "protocolVersion": 2
}
```

### updateToast

Signature	updateToast( <i>id</i> , <i>parameters</i> , succeeded, failed) → {boolean}
Description	Updates the specified toast.

Signature	updateToast( <i>id</i> , <i>parameters</i> , succeeded, failed) → {boolean}			
<b>Parameters</b>	Name	Type	Description	
	id	string	The identifier of the toast to update. The identifier is returned by the popupToast method.	
		object	Name	
title			string	The title
iconUrl			string	The URL of the icon you want to display in the title bar of the custom toast popup.
subject			string	Optional. The subject.
message			string	Optional. The message.
keyValues			object	Optional. JSON object used to fill the key value pair list. For

Signature	updateToast( <i>id</i> , <i>parameters</i> , <i>succeeded</i> , <i>failed</i> ) → {boolean}														
	Name	Type	Description												
			<table border="1"> <thead> <tr> <th data-bbox="1243 380 1312 428">Name</th> <th data-bbox="1312 380 1380 428">Type</th> <th data-bbox="1380 380 1463 428">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="1243 428 1312 667"></td> <td data-bbox="1312 428 1380 667"></td> <td data-bbox="1380 428 1463 667">example: {"key1" : "value one", "key2" : "value two", "key3" : "value three"}.</td> </tr> <tr> <td data-bbox="1243 667 1312 1514"></td> <td data-bbox="1312 667 1380 1514">Array</td> <td data-bbox="1380 667 1463 1514">Each character string in this array becomes a button. All buttons are displayed as buttons, not hyperlinks, in the following order: [Button 2] [Button 3] ... [Button N] [Button 1].</td> </tr> <tr> <td data-bbox="1243 1514 1312 1837"></td> <td data-bbox="1312 1514 1380 1837">boolean</td> <td data-bbox="1380 1514 1463 1837">If set to true, displays <b>Show</b> and <b>Dismiss</b> buttons and</td> </tr> </tbody> </table>	Name	Type	Description			example: {"key1" : "value one", "key2" : "value two", "key3" : "value three"}.		Array	Each character string in this array becomes a button. All buttons are displayed as buttons, not hyperlinks, in the following order: [Button 2] [Button 3] ... [Button N] [Button 1].		boolean	If set to true, displays <b>Show</b> and <b>Dismiss</b> buttons and
Name	Type	Description													
		example: {"key1" : "value one", "key2" : "value two", "key3" : "value three"}.													
	Array	Each character string in this array becomes a button. All buttons are displayed as buttons, not hyperlinks, in the following order: [Button 2] [Button 3] ... [Button N] [Button 1].													
	boolean	If set to true, displays <b>Show</b> and <b>Dismiss</b> buttons and													

Signature	updateToast( <i>id</i> , <i>parameters</i> , succeeded, failed) → {boolean}																				
	<table border="1"> <thead> <tr> <th data-bbox="824 321 1032 369">Name</th> <th data-bbox="1032 321 1240 369">Type</th> <th data-bbox="1240 321 1446 369">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="824 369 1032 1157"></td> <td data-bbox="1032 369 1240 1157"></td> <td data-bbox="1240 369 1446 1157"> <table border="1"> <thead> <tr> <th data-bbox="1256 390 1312 426">Name</th> <th data-bbox="1312 390 1367 426">Type</th> <th data-bbox="1367 390 1446 426">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="1256 426 1312 1136"></td> <td data-bbox="1312 426 1367 1136"></td> <td data-bbox="1367 426 1446 1136">pops up the current iframe if the <b>Show</b> button is pushed. If set to false, displays "OK" or custom buttons based on the parameter's buttons.</td> </tr> </tbody> </table> </td> </tr> <tr> <td data-bbox="824 1157 1032 1283">succeeded</td> <td data-bbox="1032 1157 1240 1283">function</td> <td data-bbox="1240 1157 1446 1283">A function called when the operation succeeds.</td> </tr> <tr> <td data-bbox="824 1283 1032 1409">failed</td> <td data-bbox="1032 1283 1240 1409">function</td> <td data-bbox="1240 1283 1446 1409">A function called when the operation fails.</td> </tr> </tbody> </table>			Name	Type	Description			<table border="1"> <thead> <tr> <th data-bbox="1256 390 1312 426">Name</th> <th data-bbox="1312 390 1367 426">Type</th> <th data-bbox="1367 390 1446 426">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="1256 426 1312 1136"></td> <td data-bbox="1312 426 1367 1136"></td> <td data-bbox="1367 426 1446 1136">pops up the current iframe if the <b>Show</b> button is pushed. If set to false, displays "OK" or custom buttons based on the parameter's buttons.</td> </tr> </tbody> </table>	Name	Type	Description			pops up the current iframe if the <b>Show</b> button is pushed. If set to false, displays "OK" or custom buttons based on the parameter's buttons.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.
Name	Type	Description																			
		<table border="1"> <thead> <tr> <th data-bbox="1256 390 1312 426">Name</th> <th data-bbox="1312 390 1367 426">Type</th> <th data-bbox="1367 390 1446 426">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="1256 426 1312 1136"></td> <td data-bbox="1312 426 1367 1136"></td> <td data-bbox="1367 426 1446 1136">pops up the current iframe if the <b>Show</b> button is pushed. If set to false, displays "OK" or custom buttons based on the parameter's buttons.</td> </tr> </tbody> </table>	Name	Type	Description			pops up the current iframe if the <b>Show</b> button is pushed. If set to false, displays "OK" or custom buttons based on the parameter's buttons.													
Name	Type	Description																			
		pops up the current iframe if the <b>Show</b> button is pushed. If set to false, displays "OK" or custom buttons based on the parameter's buttons.																			
succeeded	function	A function called when the operation succeeds.																			
failed	function	A function called when the operation fails.																			
Returns	true if the toast has been updated; false if the toast identifier has not been found.																				

# Voice namespace

## Contents

- **1 Methods**
  - 1.1 answer
  - 1.2 dial
  - 1.3 dialEx
  - 1.4 hangUp
  - 1.5 hold
  - 1.6 resume
  - 1.7 pauseCallRecording
  - 1.8 resumeCallRecording
  - 1.9 startCallRecording
  - 1.10 stopCallRecording
  - 1.11 isMicrophoneMute
  - 1.12 muteMicrophone
  - 1.13 unmuteMicrophone
  - 1.14 isSpeakerMute
  - 1.15 muteSpeaker
  - 1.16 unmuteSpeaker

Learn about the Voice namespace methods in the Service Client API.

## Methods

The Voice namespace includes the following methods:

- answer
- dial
- dialEx
- hangUp
- hold
- resume
- pauseCallRecording
- resumeCallRecording
- startCallRecording
- stopCallRecording
- isMicrophoneMute
- muteMicrophone
- unmuteMicrophone
- isSpeakerMute
- muteSpeaker
- unmuteSpeaker

### answer

<b>Signature</b>	answer('interactionId', succeeded, failed)												
<b>Description</b>	Answers the incoming call.												
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Argument</th><th>Description</th></tr></thead><tbody><tr><td>interactionId</td><td>string</td><td></td><td>The interaction identifier</td></tr><tr><td>succeeded</td><td>function</td><td></td><td>A function</td></tr></tbody></table>	Name	Type	Argument	Description	interactionId	string		The interaction identifier	succeeded	function		A function
Name	Type	Argument	Description										
interactionId	string		The interaction identifier										
succeeded	function		A function										

<b>Signature</b>	answer('interactionId', succeeded, failed)			
	<b>Name</b>	<b>Type</b>	<b>Argument</b>	<b>Description</b>
				called when the operation succeeds.
	failed	function		A function called when the operation fails.

## dial

<b>Signature</b>	dial(destination, userData, succeeded, failed)			
<b>Description</b>	Calls the destination in the same way Workspace Web Edition calls the destination from Team Communicator.			
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Argument</b>	<b>Description</b>
	destination	string		The call destination number.
	userData	object		The attached user data key/value object that is updated with each interaction event.
	succeeded	function		A function called when the operation succeeds.
	failed	function		A function called when the operation fails.

## dialEx

<b>Signature</b>	dialEx( <i>destination, userData, extensions, succeeded, failed</i> )																										
<b>Description</b>	Calls the destination with the attached data and extensions.																										
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>destination</td> <td>string</td> <td></td> <td>The call destination number.</td> </tr> <tr> <td>userData</td> <td>object</td> <td></td> <td>The attached user data key/value object. Set an undefined or empty JSON object if you don't want to set any user data.</td> </tr> <tr> <td>extensions</td> <td>object</td> <td></td> <td>The extensions key/value object. Set an undefined or empty JSON object if you don't want to set any extensions.</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td></td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td></td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>			Name	Type	Argument	Description	destination	string		The call destination number.	userData	object		The attached user data key/value object. Set an undefined or empty JSON object if you don't want to set any user data.	extensions	object		The extensions key/value object. Set an undefined or empty JSON object if you don't want to set any extensions.	succeeded	function		A function called when the operation succeeds.	failed	function		A function called when the operation fails.
Name	Type	Argument	Description																								
destination	string		The call destination number.																								
userData	object		The attached user data key/value object. Set an undefined or empty JSON object if you don't want to set any user data.																								
extensions	object		The extensions key/value object. Set an undefined or empty JSON object if you don't want to set any extensions.																								
succeeded	function		A function called when the operation succeeds.																								
failed	function		A function called when the operation fails.																								

## hangUp

<b>Signature</b>	hangUp('interactionId', succeeded, failed)																		
<b>Description</b>	Releases the incoming call.																		
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td></td> <td>The interaction identifier</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td></td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td></td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Argument	Description	interactionId	string		The interaction identifier	succeeded	function		A function called when the operation succeeds.	failed	function		A function called when the operation fails.		
Name	Type	Argument	Description																
interactionId	string		The interaction identifier																
succeeded	function		A function called when the operation succeeds.																
failed	function		A function called when the operation fails.																

## hold

<b>Signature</b>	hold('interactionId', succeeded, failed)																		
<b>Description</b>	Holds the incoming call.																		
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td></td> <td>The interaction identifier</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td></td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td></td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Argument	Description	interactionId	string		The interaction identifier	succeeded	function		A function called when the operation succeeds.	failed	function		A function called when the operation fails.		
Name	Type	Argument	Description																
interactionId	string		The interaction identifier																
succeeded	function		A function called when the operation succeeds.																
failed	function		A function called when the operation fails.																

## resume

<b>Signature</b>	resume('interactionId', succeeded, failed)			
<b>Description</b>	Resumes the held call.			
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Argument</b>	<b>Description</b>
	interactionId	string		The interaction identifier
	succeeded	function		A function called when the operation succeeds.
	failed	function		A function called when the operation fails.

## pauseCallRecording

<b>Signature</b>	pauseCallRecording('interactionId', succeeded, failed)			
<b>Description</b>	Pause the call recording.			
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Argument</b>	<b>Description</b>
	interactionId	string		The interaction identifier
	succeeded	function		A function called when the operation succeeds.
	failed	function		A function called when the operation fails.

## resumeCallRecording

<b>Signature</b>	resumeCallRecording('interactionId', succeeded, failed)																		
<b>Description</b>	Resumes the call recording.																		
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td></td> <td>The interaction identifier</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td></td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td></td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Argument	Description	interactionId	string		The interaction identifier	succeeded	function		A function called when the operation succeeds.	failed	function		A function called when the operation fails.		
Name	Type	Argument	Description																
interactionId	string		The interaction identifier																
succeeded	function		A function called when the operation succeeds.																
failed	function		A function called when the operation fails.																

## startCallRecording

<b>Signature</b>	startCallRecording('interactionId', succeeded, failed)																		
<b>Description</b>	Starts the call recording.																		
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td></td> <td>The interaction identifier</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td></td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td></td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Argument	Description	interactionId	string		The interaction identifier	succeeded	function		A function called when the operation succeeds.	failed	function		A function called when the operation fails.		
Name	Type	Argument	Description																
interactionId	string		The interaction identifier																
succeeded	function		A function called when the operation succeeds.																
failed	function		A function called when the operation fails.																

### stopCallRecording

<b>Signature</b>	stopCallRecording('interactionId', succeeded, failed)																		
<b>Description</b>	Stops the call recording.																		
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td></td> <td>The interaction identifier</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td></td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td></td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Argument	Description	interactionId	string		The interaction identifier	succeeded	function		A function called when the operation succeeds.	failed	function		A function called when the operation fails.		
Name	Type	Argument	Description																
interactionId	string		The interaction identifier																
succeeded	function		A function called when the operation succeeds.																
failed	function		A function called when the operation fails.																

### isMicrophoneMute

<b>Signature</b>	isMicrophoneMute(succeeded, failed)														
<b>Description</b>	Get the mute state of the microphone of the SIP Endpoint.														
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>succeeded</td> <td>function</td> <td></td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td></td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Argument	Description	succeeded	function		A function called when the operation succeeds.	failed	function		A function called when the operation fails.		
Name	Type	Argument	Description												
succeeded	function		A function called when the operation succeeds.												
failed	function		A function called when the operation fails.												

### muteMicrophone

<b>Signature</b>	muteMicrophone(succeeded, failed)
<b>Description</b>	Mute the microphone of the SIP Endpoint.

---

<b>Signature</b>	muteMicrophone(succeeded, failed)			
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Argument</b>	<b>Description</b>
	succeeded	function		A function called when the operation succeeds.
	failed	function		A function called when the operation fails.

### unmuteMicrophone

<b>Signature</b>	unmuteMicrophone(succeeded, failed)			
<b>Description</b>	Unmute the microphone of the SIP Endpoint.			
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Argument</b>	<b>Description</b>
	succeeded	function		A function called when the operation succeeds.
	failed	function		A function called when the operation fails.

### isSpeakerMute

<b>Signature</b>	isSpeakerMute(succeeded, failed)			
<b>Description</b>	Get the mute state of the speaker of the SIP Endpoint.			
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Argument</b>	<b>Description</b>
	succeeded	function		A function called when the operation succeeds.

<b>Signature</b>	isSpeakerMute(succeeded, failed)		
	<b>Name</b>	<b>Type</b>	<b>Argument</b>
	failed	function	A function called when the operation fails.

### muteSpeaker

<b>Signature</b>	muteSpeaker(succeeded, failed)		
<b>Description</b>	Mute the speaker of the SIP Endpoint.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Argument</b>
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.

### unmuteSpeaker

<b>Signature</b>	unmuteSpeaker(succeeded, failed)		
<b>Description</b>	Unmute the speaker of the SIP Endpoint.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Argument</b>
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.

# Outbound namespace

## Contents

- **1 Methods**
  - 1.1 getCampaigns
  - 1.2 getPreviewRecord
  - 1.3 callPreviewRecord
  - 1.4 rejectPreviewRecord
  - 1.5 cancelPreviewRecord
  - 1.6 startDirectPushPreview
  - 1.7 stopDirectPushPreview
  - 1.8 getListOfCallResults
  - 1.9 setCallResult
  - 1.10 getCallResult
  - 1.11 setDoNotCall
  - 1.12 removeDoNotCall
  - 1.13 rescheduleRecord
  - 1.14 cancelReschedule
  - 1.15 getChainedRecords
  - 1.16 getRecordFields
  - 1.17 updateRecordFields
- **2 Type definitions**
  - 2.1 Field

- Developer

Learn about the Outbound namespace methods in the Service Client API.

### Important

Depending on your environment, you might need to contact your Genesys representative to complete the configuration described on this page.

## Methods

The Outbound namespace includes the following methods:

- getCampaigns
- getPreviewRecord
- callPreviewRecord
- rejectPreviewRecord
- cancelPreviewRecord
- startDirectPushPreview
- stopDirectPushPreview
- getListOfCallResults
- setCallResult
- getCallResult
- setDoNotCall
- removeDoNotCall
- rescheduleRecord
- cancelReschedule
- getChainedRecords
- getRecordFields
- updateRecordFields

## getCampaigns

<b>Signature</b>	getCampaigns( <i>succeeded, failed</i> )											
<b>Description</b>	Get the details of all outbound campaigns (loaded or active) for the current agent.											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.		
Name	Type	Description										
succeeded	function	A function called when the operation succeeds.										
failed	function	A function called when the operation fails.										

## getPreviewRecord

<b>Signature</b>	getPreviewRecord( <i>campaignName, succeeded, failed</i> )														
<b>Description</b>	Get a preview record from Outbound Contact Server.														
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>campaignName</td> <td>string</td> <td>The name of the outbound campaign.</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	campaignName	string	The name of the outbound campaign.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.		
Name	Type	Description													
campaignName	string	The name of the outbound campaign.													
succeeded	function	A function called when the operation succeeds.													
failed	function	A function called when the operation fails.													

## callPreviewRecord

<b>Signature</b>	callPreviewRecord( <i>interactionId, recordHandle, succeeded, failed</i> )								
<b>Description</b>	Make a call using the preview record.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td>The unique identifier for the</td> </tr> </tbody> </table>	Name	Type	Description	interactionId	string	The unique identifier for the		
Name	Type	Description							
interactionId	string	The unique identifier for the							

---

<b>Signature</b>	callPreviewRecord( <i>interactionId, recordHandle, succeeded, failed</i> )		
	<b>Name</b>	<b>Type</b>	<b>Description</b>
			interaction.
	recordHandle	number	The record number in the chain to be dialed.
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.

### rejectPreviewRecord

<b>Signature</b>	rejectPreviewRecord( <i>succeeded, failed</i> )		
<b>Description</b>	Reject a pull preview, push preview, or direct push preview record.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.

### cancelPreviewRecord

<b>Signature</b>	cancelPreviewRecord( <i>succeeded, failed</i> )		
<b>Description</b>	Cancel a pull preview, push preview, or direct push preview record.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function

<b>Signature</b>	cancelPreviewRecord( <i>succeeded, failed</i> )		
	<b>Name</b>	<b>Type</b>	<b>Description</b>
			called when the operation fails.

### startDirectPushPreview

<b>Signature</b>	startDirectPushPreview( <i>succeeded, failed</i> )		
<b>Description</b>	Send a Dialing Mode Start request to Outbound Contact Server to start sending direct push preview records to the agent.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.

### stopDirectPushPreview

<b>Signature</b>	stopDirectPushPreview( <i>succeeded, failed</i> )		
<b>Description</b>	Send a Dialing Mode Stop request to Outbound Contact Server to stop sending direct push preview records to the agent.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.

### getListOfCallResults

<b>Signature</b>	getListOfCallResults( <i>succeeded, failed</i> )		
<b>Description</b>	Get the list of call results currently available in Workspace Web Edition.		

Signature	getListOfCallResults( <i>succeeded, failed</i> )		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.

### Sample request

```
genesys.wwe.service.outbound.getListOfCallResults(succeeded, failed)
```

### Sample response

```
{
  "request": "outbound.getListOfCallResults",
  "data": {
    "OK": 0,
    "GENERAL_ERROR": 3,
    "SYSTEM_ERROR": 4,
    "BUSY": 6,
    "NO_ANSWER": 7,
    "SIT_DETECTED": 8,
    "ANSWERING_MACHINE": 9,
    "ALL_TRUNKS_BUSY": 10,
    "SIT_INVALID_NUM": 11,
    "SIT_VACANT": 12,
    "SIT_OPERINTERCEPT": 13,
    "SIT_UNKNOWN": 14,
    "SIT_NO_CIRCUIT": 15,
    "SIT_REORDER": 16,
    "FAXDETECTED": 17,
    "ABANDONED": 21,
    "DROPPED": 26,
    "DROPPED_NO_ANSWER": 27,
    "UNKNOWN": 28,
    "SILENCE": 32,
    "ANSWER": 33,
    "NUTONE": 34,
    "NO_DIAL_TONE": 35,
    "NO_PROGRESS": 36,
    "NO_RINGBACK": 37,
    "NO_ESTABLISHED": 38,
    "PAGER_DETECTED": 39,
    "WRONG_PARTY": 40,
    "DIAL_ERROR": 41,
    "CALL_DROP_ERROR": 42,
    "SWITCH_ERROR": 43,
    "NO_FREE_PORT_ERROR": 44,
    "TRANSFER_ERROR": 45,
    "STALE": 46,
    "AGENT_CALLBACK_ERROR": 47,
    "GROUP_CALLBACK_ERROR": 48,
    "DO_NOT_CALL": 51,
  }
}
```

```

    "CANCEL_RECORD": 52,
    "WRONG_NUMBER": 53
  },
  "userAgent": "WWE Server",
  "protocolVersion": 2
}

```

### setCallResult

<b>Signature</b>	setCallResult( <i>interactionId</i> , <i>callResult</i> , <i>succeeded</i> , <i>failed</i> )																	
<b>Description</b>	Set the call result for this interaction.																	
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td>The unique identifier for the interaction. The interaction should have an active or completed call. "Do Not Call" must not be set for the interaction.</td> </tr> <tr> <td>callResult</td> <td>string</td> <td>The call result value, which must be a number.</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	interactionId	string	The unique identifier for the interaction. The interaction should have an active or completed call. "Do Not Call" must not be set for the interaction.	callResult	string	The call result value, which must be a number.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.		
Name	Type	Description																
interactionId	string	The unique identifier for the interaction. The interaction should have an active or completed call. "Do Not Call" must not be set for the interaction.																
callResult	string	The call result value, which must be a number.																
succeeded	function	A function called when the operation succeeds.																
failed	function	A function called when the operation fails.																

### getCallResult

<b>Signature</b>	getCallResult( <i>interactionId</i> , <i>succeeded</i> , <i>failed</i> )								
<b>Description</b>	Get the call result already set in an outbound record, if any.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td>The unique identifier for the</td> </tr> </tbody> </table>	Name	Type	Description	interactionId	string	The unique identifier for the		
Name	Type	Description							
interactionId	string	The unique identifier for the							

Signature	getCallResult( <i>interactionId</i> , <i>succeeded</i> , <i>failed</i> )		
	<b>Name</b>	<b>Type</b>	<b>Description</b>
			interaction.
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.

### Sample request

```
genesys.wwe.service.outbound.getCallResult(interactionId, succeeded, failed)
```

### Sample response

```
{
  "request": "outbound.getCallResult",
  "data": 6,
  "userAgent": "WWE Server",
  "protocolVersion": 2
}
```

### setDoNotCall

Signature	setDoNotCall( <i>interactionId</i> , <i>succeeded</i> , <i>failed</i> )		
Description	Set the interaction to "Do Not Call".		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	interactionId	string	The unique identifier for the interaction. The interaction should have an active or completed call.
	succeeded	function	A function called when the operation succeeds.
	failed	function	A function called when the operation fails.

## removeDoNotCall

<b>Signature</b>	removeDoNotCall( <i>interactionId</i> , <i>succeeded</i> , <i>failed</i> )														
<b>Description</b>	Remove "Do Not Call" from the interaction.														
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td>The unique identifier for the interaction. The interaction should have an active or completed call.</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	interactionId	string	The unique identifier for the interaction. The interaction should have an active or completed call.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.		
Name	Type	Description													
interactionId	string	The unique identifier for the interaction. The interaction should have an active or completed call.													
succeeded	function	A function called when the operation succeeds.													
failed	function	A function called when the operation fails.													

## rescheduleRecord

<b>Signature</b>	rescheduleRecord( <i>interactionId</i> , <i>recordHandle</i> , <i>rescheduleDate</i> , <i>callbackType</i> , <i>succeeded</i> , <i>failed</i> )								
<b>Description</b>	Set the schedule information on the record based on its time zone. You can perform this operation regardless of how the Workspace Web Edition options <code>privilege.outbound.can-reschedule</code> and <code>privilege.outbound.can-reschedule-before-call</code> are configured.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td>The unique identifier for the interaction. Note: For Preview and Push Preview modes, once the call is made the ID provided becomes the new</td> </tr> </tbody> </table>	Name	Type	Description	interactionId	string	The unique identifier for the interaction. Note: For Preview and Push Preview modes, once the call is made the ID provided becomes the new		
Name	Type	Description							
interactionId	string	The unique identifier for the interaction. Note: For Preview and Push Preview modes, once the call is made the ID provided becomes the new							

Signature	rescheduleRecord( <i>interactionId</i> , <i>recordHandle</i> , <i>rescheduleDate</i> , <i>callbackType</i> , <i>succeeded</i> , <i>failed</i> )		
	<b>Name</b>	<b>Type</b>	<b>Description</b>
			interaction ID that corresponds to the call.
	recordHandle	number	The record number in the chain to be dialed.
	rescheduleDate	string	<p>The date for which the callback is to be rescheduled, in MM/DD/YYYY HH:MM format. This date should be in the time zone of the record that is being rescheduled. This ensures the date is set correctly in cases where the agent and the customer are in different time zones. To calculate the correct hour and minute values, you can get the outbound record's time zone offset value from any of the interaction's events.</p> <p><b>Example</b></p> <p>An agent calls a customer and they ask to be called back one hour later. The agent and</p>

Signature	rescheduleRecord( <i>interactionId</i> , <i>recordHandle</i> , <i>rescheduleDate</i> , <i>callbackType</i> , <i>succeeded</i> , <i>failed</i> )																	
	<table border="1"> <thead> <tr> <th data-bbox="824 321 1032 369">Name</th> <th data-bbox="1032 321 1240 369">Type</th> <th data-bbox="1240 321 1450 369">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="824 369 1032 1167"></td> <td data-bbox="1032 369 1240 1167"></td> <td data-bbox="1240 369 1450 1167"> <p>customer have the following time zone information:</p> <ul style="list-style-type: none"> <li>• Agent's time zone - BST</li> <li>• Agent's current time - 2:30 PM</li> <li>• Customer's time zone - EDT</li> <li>• Customer's current time - 9:30 AM</li> </ul> <p>In this case, you would make the <b>rescheduleRecord</b> request with the <b>rescheduleDate</b> HH:MM set to a value of 10:30 and not 15:30.</p> </td> </tr> <tr> <td data-bbox="824 1167 1032 1318">callbackType</td> <td data-bbox="1032 1167 1240 1318">string</td> <td data-bbox="1240 1167 1450 1318">The type of callback. Valid values are CAMPAIGN or PERSONAL.</td> </tr> <tr> <td data-bbox="824 1318 1032 1444">succeeded</td> <td data-bbox="1032 1318 1240 1444">function</td> <td data-bbox="1240 1318 1450 1444">A function called when the operation succeeds.</td> </tr> <tr> <td data-bbox="824 1444 1032 1570">failed</td> <td data-bbox="1032 1444 1240 1570">function</td> <td data-bbox="1240 1444 1450 1570">A function called when the operation fails.</td> </tr> </tbody> </table>			Name	Type	Description			<p>customer have the following time zone information:</p> <ul style="list-style-type: none"> <li>• Agent's time zone - BST</li> <li>• Agent's current time - 2:30 PM</li> <li>• Customer's time zone - EDT</li> <li>• Customer's current time - 9:30 AM</li> </ul> <p>In this case, you would make the <b>rescheduleRecord</b> request with the <b>rescheduleDate</b> HH:MM set to a value of 10:30 and not 15:30.</p>	callbackType	string	The type of callback. Valid values are CAMPAIGN or PERSONAL.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.
Name	Type	Description																
		<p>customer have the following time zone information:</p> <ul style="list-style-type: none"> <li>• Agent's time zone - BST</li> <li>• Agent's current time - 2:30 PM</li> <li>• Customer's time zone - EDT</li> <li>• Customer's current time - 9:30 AM</li> </ul> <p>In this case, you would make the <b>rescheduleRecord</b> request with the <b>rescheduleDate</b> HH:MM set to a value of 10:30 and not 15:30.</p>																
callbackType	string	The type of callback. Valid values are CAMPAIGN or PERSONAL.																
succeeded	function	A function called when the operation succeeds.																
failed	function	A function called when the operation fails.																

### Sample request

```
genesys.wwe.service.outbound.rescheduleRecord('1', 257, '05/27/2021 10:55', 'PERSONAL',
succeeded, failed)
```

## cancelReschedule

<b>Signature</b>	<code>cancelReschedule(<i>interactionId</i>, <i>succeeded</i>, <i>failed</i>)</code>												
<b>Description</b>	Remove the schedule information from the record.												
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>interactionId</code></td> <td>string</td> <td>The unique identifier for the interaction. Note: For Preview and Push Preview modes, once the call is made the ID provided becomes the new interaction ID that corresponds to the call.</td> </tr> <tr> <td><code>succeeded</code></td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td><code>failed</code></td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	<code>interactionId</code>	string	The unique identifier for the interaction. Note: For Preview and Push Preview modes, once the call is made the ID provided becomes the new interaction ID that corresponds to the call.	<code>succeeded</code>	function	A function called when the operation succeeds.	<code>failed</code>	function	A function called when the operation fails.
Name	Type	Description											
<code>interactionId</code>	string	The unique identifier for the interaction. Note: For Preview and Push Preview modes, once the call is made the ID provided becomes the new interaction ID that corresponds to the call.											
<code>succeeded</code>	function	A function called when the operation succeeds.											
<code>failed</code>	function	A function called when the operation fails.											

## getChainedRecords

<b>Signature</b>	<code>getChainedRecords(<i>interactionId</i>, <i>succeeded</i>, <i>failed</i>)</code>												
<b>Description</b>	Get the list of chained records for the interaction.												
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>interactionId</code></td> <td>string</td> <td>The unique identifier for the interaction.</td> </tr> <tr> <td><code>succeeded</code></td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td><code>failed</code></td> <td>function</td> <td>A function called when</td> </tr> </tbody> </table>	Name	Type	Description	<code>interactionId</code>	string	The unique identifier for the interaction.	<code>succeeded</code>	function	A function called when the operation succeeds.	<code>failed</code>	function	A function called when
Name	Type	Description											
<code>interactionId</code>	string	The unique identifier for the interaction.											
<code>succeeded</code>	function	A function called when the operation succeeds.											
<code>failed</code>	function	A function called when											

Signature	getChainedRecords( <i>interactionId</i> , <i>succeeded</i> , <i>failed</i> )		
	Name	Type	Description
			the operation fails.

### Sample request

```
genesys.wwe.service.outbound.getChainedRecords('1', succeeded, failed)
```

### Sample response

```
{
  "request": "outbound.getChainedRecords",
  "data": [
    {
      "records": [
        {
          Custom_Character: "c"
          Custom_Datetime: "2021-03-17 14:42:39"
          Custom_Float: "16.64"
          Custom_Integer: 0
          Custom_String_with_default: "Hi there!"
          Custom_VarChar: ""
          GSW_AGENT_ID: "+33298025000"
          GSW_APPLICATION_ID: 139
          GSW_ATTEMPTS: 0
          GSW_CALLING_LIST: "Calling List Custom"
          GSW_CALLING_LIST_DBID: 101
          GSW_CALL_ATTEMPT_GUID: "003DC7H6HG84DBRT1KMIF1TAES000031"
          GSW_CALL_RESULT: 28
          GSW_CAMPAIGN_GROUP_DBID: 101
          GSW_CAMPAIGN_GROUP_DESCRIPTION: ""
          GSW_CAMPAIGN_GROUP_NAME: "Outbound Campaign Custom@Agent Group Outbound"
          GSW_CAMPAIGN_NAME: "Outbound Campaign Custom"
          GSW_CHAIN_ID: 3
          GSW_CONTACT_MEDIA_TYPE: "voice"
          GSW_FROM: 0
          GSW_PHONE: "+33647005"
          GSW_PHONE_TYPE: 1
          GSW_RECORD_HANDLE: 283
          GSW_REFERENCE_ID: 3
          GSW_SWITCH_DBID: 101
          GSW_TZ_NAME: "ACT"
          GSW_TZ_OFFSET: 34200
          GSW_UNTIL: 86399
          GSW_USER_EVENT: "PreviewRecord"
          IW_BundleUid: "27458420-0348-4345-c693-45bd95b5c81f"
          IW_CaseUid: "a26f59d2-2979-43c5-5c1d-b0757f9ab077"
          InteractionSubtype: "OutboundNew"
          InteractionType: "Outbound"
          WWE_OUTBOUND_CAMP_TYPE: "PreviewRecord"
        },
        {
          Custom_Character: "c"
          Custom_Datetime: "2021-03-17 14:42:32"
          Custom_Float: "51.69"
          Custom_Integer: 0
        }
      ]
    }
  ]
}
```

```

Custom_String_with_default: "Hello General Kenobi"
Custom_VarChar: ""
GSW_AGENT_ID: "+33298025000"
GSW_APPLICATION_ID: 139
GSW_ATTEMPTS: 0
GSW_CALLING_LIST: "Calling List Custom"
GSW_CALLING_LIST_DBID: 101
GSW_CALL_ATTEMPT_GUID: "003DC7H6HG84DBRT1KMIF1TAES000031"
GSW_CALL_RESULT: 28
GSW_CAMPAIGN_GROUP_DBID: 101
GSW_CAMPAIGN_GROUP_DESCRIPTION: ""
GSW_CAMPAIGN_GROUP_NAME: "Outbound Campaign Custom@Agent Group Outbound"
GSW_CAMPAIGN_NAME: "Outbound Campaign Custom"
GSW_CHAIN_ID: 3
GSW_CONTACT_MEDIA_TYPE: "voice"
GSW_FROM: 0
GSW_PHONE: "+33647004"
GSW_PHONE_TYPE: 1
GSW_RECORD_HANDLE: 284
GSW_REFERENCE_ID: 4
GSW_SWITCH_DBID: 101
GSW_TZ_NAME: "ACT"
GSW_TZ_OFFSET: 34200
GSW_UNTIL: 86399
GSW_USER_EVENT: "ChainedRecord"
InteractionSubtype: "OutboundNew"
InteractionType: "Outbound"
    }
  ]
}
],
"userAgent": "WWE Server",
"protocolVersion": 2
}

```

### getRecordFields

<b>Signature</b>	getRecordFields( <i>interactionId</i> , <i>succeeded</i> , <i>failed</i> ) → {Array.}												
<b>Description</b>	Get the list of outbound fields for an interaction. This method also returns information about whether a field is mandatory and if it can be edited.												
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td>The unique identifier for the interaction.</td> </tr> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when the operation</td> </tr> </tbody> </table>	Name	Type	Description	interactionId	string	The unique identifier for the interaction.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation
Name	Type	Description											
interactionId	string	The unique identifier for the interaction.											
succeeded	function	A function called when the operation succeeds.											
failed	function	A function called when the operation											

<b>Signature</b>	getRecordFields( <i>interactionId</i> , <i>succeeded</i> , <i>failed</i> ) → {Array.}		
	<b>Name</b>	<b>Type</b>	<b>Description</b>
			fails.
<b>Returns</b>	Array.		

### Sample request

```
genesys.wwe.service.outbound.getRecordFields('1', succeeded, failed)
```

### Sample response

```
{
  "request": "outbound.getRecordFields",
  "data": [
    {
      "name": "GWS_FROM",
      "displayName": "Call From",
      "value": "10.15",
      "isMandatory": true,
      "isEditable": false,
      "type": "time",
      "valueType": "string"
    },
    {
      "name": "GSW_CUSTOM_STRING",
      "value": "Custom message",
      "isMandatory": false,
      "isEditable": true,
      "fieldType": "var-char",
      "valueType": "string"
    },
    {
      "name": "GSW_PHONE_TYPE",
      "displayName": "Phone Type",
      "isEditable": true,
      "isMandatory": false,
      "options": {
        "3": "Business With Extension",
        "2": "Direct Business Phone",
        "10": "Email Address",
        "1": "Home Phone",
        "11": "Instant Messaging",
        "4": "Mobile",
        "7": "Modem",
        "0": "None",
        "6": "Pager",
        "9": "Pin Pager",
        "5": "Vacation Phone",
        "8": "Voice Mail"
      },
      "fieldType": "enum",
      "valueType": "number"
    }
  ],
  "userAgent": "WWE Server",
  "protocolVersion": 2
}
```

## updateRecordFields

<b>Signature</b>	updateRecordFields( <i>interactionId</i> , <i>recordData</i> , <i>succeeded</i> , <i>failed</i> )																	
<b>Description</b>	Update one or more outbound fields. The updated fields are sent to Outbound Contact Server when the record is marked done. <b>Note:</b> This operation fails if one of the updated fields does not comply with the data type or mandatory requirements.																	
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>interactionId</td> <td>string</td> <td>The unique identifier for the interaction.</td> </tr> <tr> <td>recordData</td> <td>string</td> <td>The record data to be updated. This must be an object containing the field names as properties and the values to be updated. The values should comply with the valueType property of the field as returned by getRecordFields. You can update custom fields and the following system fields:                             <ul style="list-style-type: none"> <li>• Call From</li> <li>• Call Until</li> <li>• Phone</li> <li>• Phone Type</li> </ul> </td> </tr> <tr> <td>succeeded</td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td>failed</td> <td>function</td> <td>A function called when</td> </tr> </tbody> </table>			Name	Type	Description	interactionId	string	The unique identifier for the interaction.	recordData	string	The record data to be updated. This must be an object containing the field names as properties and the values to be updated. The values should comply with the valueType property of the field as returned by getRecordFields. You can update custom fields and the following system fields: <ul style="list-style-type: none"> <li>• Call From</li> <li>• Call Until</li> <li>• Phone</li> <li>• Phone Type</li> </ul>	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when
Name	Type	Description																
interactionId	string	The unique identifier for the interaction.																
recordData	string	The record data to be updated. This must be an object containing the field names as properties and the values to be updated. The values should comply with the valueType property of the field as returned by getRecordFields. You can update custom fields and the following system fields: <ul style="list-style-type: none"> <li>• Call From</li> <li>• Call Until</li> <li>• Phone</li> <li>• Phone Type</li> </ul>																
succeeded	function	A function called when the operation succeeds.																
failed	function	A function called when																

Signature	updateRecordFields( <i>interactionId</i> , <i>recordData</i> , <i>succeeded</i> , <i>failed</i> )		
	<b>Name</b>	<b>Type</b>	<b>Description</b>
			the operation fails.

### Sample request

```
genesys.wwe.service.outbound.updateRecordFields(
  '1',
  {
    GSW_FROM: '10.15',
    GSW_UNTIL: '23:45',
    GSW_PHONE_TYPE: 9,
    GSW_CUSTOM_STRING: 'Custom message'
  },
  succeeded,
  failed
)
```

### Type definitions

The Outbound namespace includes the following object types:

- Field

#### Field

Description	Represents the JSON structure of a field.		
Type	Object		
<b>Properties</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	name	string	The name of the field. Use this name in updateRecordFields requests to set or update the value for the field.
	displayName	string	The name of the field as displayed in Workspace Web Edition. You can use this in a custom view,

Description	Represents the JSON structure of a field.		
	<b>Name</b>	<b>Type</b>	<b>Description</b>
			if required.
	value	string	The current value of the field.
	isEditable	boolean	Specifies whether the field is editable. If updateRecordFields contains a non-editable field, the operation fails.
	isMandatory	boolean	Specifies whether the field is mandatory. If updateRecordFields tries to set a null or empty value for a mandatory field, the operation fails.
	options	string	This property is present for fields of type 'enum'. Enums are displayed as dropdowns in Workspace Web Edition. See sample response for getRecordFields for details.
	fieldType	string	The data type of the field. Possible values are: <ul style="list-style-type: none"> <li>• int - Integer</li> <li>• float - Floating point</li> </ul>

Description	Represents the JSON structure of a field.								
	<table border="1"> <thead> <tr> <th data-bbox="833 306 1024 338">Name</th> <th data-bbox="1040 306 1235 338">Type</th> <th data-bbox="1252 306 1446 338">Description</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td data-bbox="1260 369 1438 989">                     number                      • char - Character                      • var-char - String                      • date - Date string (MM/DD/YYYY HH:MM)                      • time - Time string (HH:MM)                      • bool - Boolean                      • enum - Key/value pairs                 </td> </tr> </tbody> </table>	Name	Type	Description			number • char - Character • var-char - String • date - Date string (MM/DD/YYYY HH:MM) • time - Time string (HH:MM) • bool - Boolean • enum - Key/value pairs		
Name	Type	Description							
		number • char - Character • var-char - String • date - Date string (MM/DD/YYYY HH:MM) • time - Time string (HH:MM) • bool - Boolean • enum - Key/value pairs							
	valueType	string	The type of value that should be used in updateRecordFields. Possible values are: <ul style="list-style-type: none"> <li>• string</li> <li>• number</li> <li>• boolean</li> </ul> For example, an enum field may have to be updated with a value type of number. See the sample request for updateRecordFields.						

# Auth Namespace

## Contents

- [1 Methods](#)
  - [1.1 getJwtToken](#)

- Developer

Learn about the Auth namespace methods and type definitions in the Service Client API.

### Important

Depending on your environment, you might need to contact your Genesys representative to complete the configuration described on this page.

## Methods

The Auth namespace includes the following methods:

- `getJwtToken`

### getJwtToken

To use the `auth.getJwtToken` endpoint, you must explicitly define the full endpoint name in the `service-client-api-accepted-web-content-origins` option. For example: `service-client-api.accepted-web-content-origins = https://genesyspureengage.github.io (*, auth.getJwtToken)`

<b>Signature</b>	<code>getJwtToken(succeeded, failed) → {JSON object}</code>											
<b>Description</b>	Get the JWT access token for the current session. If the token is already generated and still valid, it is returned; otherwise a new token is returned.											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>succeeded</code></td> <td>function</td> <td>A function called when the operation succeeds.</td> </tr> <tr> <td><code>failed</code></td> <td>function</td> <td>A function called when the operation fails.</td> </tr> </tbody> </table>	Name	Type	Description	<code>succeeded</code>	function	A function called when the operation succeeds.	<code>failed</code>	function	A function called when the operation fails.		
Name	Type	Description										
<code>succeeded</code>	function	A function called when the operation succeeds.										
<code>failed</code>	function	A function called when the operation fails.										
<b>Returns</b>	JSON data object with the token and its expiration date in ISO 8601 date format. <pre>"data": {   "expiration":</pre>											

<b>Signature</b>	<code>getJwtToken(<i>succeeded, failed</i>) → {JSON object}</code>
	<pre>"2020-04-14T13:26:51.846Z",   "jwtToken": "" }</pre>

# Messenger namespace

## Contents

- [1 Methods](#)
  - [1.1 broadcastMessage](#)

- Developer

Learn about the Messenger namespace methods in the Service Client API.

## Methods

The Messenger namespace includes the following methods:

- broadcastMessage

### broadcastMessage

<b>Signature</b>	broadcastMessage( <i>channel, message, succeeded, failed</i> )																	
<b>Description</b>	Send a message to other web applications that use the Service Client API and have subscribed to the specified channel.																	
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>channel</td><td>string</td><td>The channel to send the message on.</td></tr><tr><td>message</td><td>object</td><td>The message (any JSON object) to broadcast on the channel.</td></tr><tr><td>succeeded</td><td>function</td><td>A function called when the operation succeeds.</td></tr><tr><td>failed</td><td>function</td><td>A function called when the operation fails.</td></tr></tbody></table>	Name	Type	Description	channel	string	The channel to send the message on.	message	object	The message (any JSON object) to broadcast on the channel.	succeeded	function	A function called when the operation succeeds.	failed	function	A function called when the operation fails.		
Name	Type	Description																
channel	string	The channel to send the message on.																
message	object	The message (any JSON object) to broadcast on the channel.																
succeeded	function	A function called when the operation succeeds.																
failed	function	A function called when the operation fails.																

### Samples

```
// Add a new message broadcaster:
genesys.wwe.service.messenger.broadcastMessage("my-channel", { foo: "A foo text.", bar: 1234
}, succeeded, failed)
// The operation "broadcastMessage" from the service "messenger" takes a channel name and any
JSON-compliant object.

// In order to receive this message, you must "subscribe" to "my-channel":
```

```
genesys.wwe.service.subscribe([ "messenger:my-channel" ], function(message) {
  console.log("message: " + message.data); }, this);
// It is possible to subscribe to several channels:
genesys.wwe.service.subscribe([ "messenger:my-channel", "messenger:my-channel2" ],
  function(message) {
    console.log("message: " + message.data + ", channel: " + message.event);
  }, this);
```

When a message is broadcast to your channel, you receive an event called `messenger:` with the message in the `data` attribute. For example, here's the event for the broadcast in the sample above:

```
{
  "event": "messenger:my-channel",
  "data": {
    "foo": "A foo text.",
    "bar": 1234
  },
  "userAgent": "WWE Server",
  "protocolVersion": 2
}
```