

The list of methods and their parameters

The following table contains a complete list of methods and parameters.

Grey denotes parameters described above or below in the table.

Depending on the direction and destination of the call, different subsets of parameters for the same invocation can be used. For example, in case of the invocation of `ConnectionStatusEvent` event, `sipLogin`, `sipPassword` and other corresponding parameters are passed. In case of an error, the same event `ConnectionStatusEvent` will have only two parameters: `status` and `info` when sending to a client, and `status`, `info`, `nodeId`, `sessionId`, `appKey` when sending to the backend server.

| connect | Establishes connection with the WCS S server |
|-----------------------|---|
| urlServer | This parameter is used by WCS JavaScript API to connect to the server. |
| appKey | This parameter passes the REST - URL of the given application to WCS. To view and add applications use the command line interface (CLI). |
| sipRegisterRequired | If this parameter is true, registration on the SIP server is performed by invoking <code>SIP_REGISTER</code> . If the parameter is false, registration on the SIP server is not performed. In this case, a web page cannot accept incoming SIP calls, but still can make outgoing calls if the SIP server allows outgoing calls without SIP registration. |
| sipLogin | SIP login of a user |
| sipAuthenticationName | SIP name of a user used for SIP authentication. Can be different from sipLogin. |
| sipPassword | SIP password. Used for SIP authentication. |
| sipVisibleName | SIP user name displayed to other users receiving an incoming call from this user. |
| sipDomain | SIP domain. FQDN or IP address. |
| sipOutboundProxy | SIP proxy server. FQDN or IP address. Can be different from sipDomain. |
| | |

| | |
|------------------------------|--|
| sipPort | SIP port the SIP server uses to handle S IP traffic. |
| sipContactParams | A string of custom parameters added to the <code>SIP Connect</code> header of the <code>REGISTER</code> query |
| status | |
| mediaProviders | Array of available types of media on WCS JavaScript API: <code>['WebRTC', 'Flash']</code> |
| restClientConfig | A JSON-object describing web-server interaction control configuration. If the object isn't passed, the default values are used. See also: <code>[restClientConfig](restClientConfig_object_description.en.md)</code> |
| width | Maximal video width, in pixels |
| height | Maximal video height, in pixels |
| custom | Custom object used to authenticate client on backend server |
| ConnectionStatusEvent | Connection status change |
| sipRegisterRequired | |
| sipLogin | |
| sipPassword | |
| sipVisibleName | |
| sipDomain | |
| sipOutboundProxy | |
| sipPort | |
| sipContactParams | |
| status | WCS Server connection status: <code>PENDING</code> , <code>ESTABLISHED</code> , <code>FAILED</code> , <code>DISCONNECTED</code> |
| info | Additional information can be added to this field. For example, if <code>status: "FAILED"</code> , the <code>info</code> contains the description of the reason |
| authToken | A key being used for connection to the same session if it is still active on WCS |
| mediaProviders | |
| nodeId | |

| | |
|--------------------------------|--|
| sessionId | |
| appKey | |
| custom | Custom object received in <code>/connect</code> hook |
| RegistrationStatusEvent | SIP registration status change |
| status | Registration status: <code>REGISTERED</code> , <code>UNREGISTERED</code> , <code>FAILED</code> |
| info | |
| sipMessageRaw | Original SIP-message with headers. SIP Response to the <code>REGISTER</code> Request |
| nodeId | |
| sessionId | |
| appKey | |
| call | Outgoing call |
| callId | Unique id of the call |
| callee | A callee in the SIP URI format, tel URI or a telephone number. |
| caller | A caller in the SIP URI format. |
| visibleName | A label displayed to the callee. |
| hasVideo | If true, this is a video call. |
| inviteParameters | Parameters added to the <code>SIP INVITE</code> request |
| isMsrp | If true, this is not a voice call, but establishing of MSRP-connection to transmit data. |
| status | |
| incoming | If true, it is an incoming call from SIP side. |
| mediaProvider | Media technology used by WCS JavaScript API, possible values: <code>WebRTC</code> , <code>Flash</code> |
| sdp | SDP, created on WCS JavaScript API side, will be placed when <code>mediaProvider</code> is <code>WebRTC</code> |
| | |

| OnCallEvent | Incoming call |
|------------------|--|
| callId | |
| callee | |
| caller | |
| visibleName | |
| hasVideo | |
| inviteParameters | |
| sipMessageRaw | SIP INVITE message the incoming call even is based upon |
| incoming | |
| status | |
| mediaProvider | |
| sdp | |
| nodeId | |
| sessionId | |
| appKey | |
| CallStatusEvent | Call status change |
| callId | |
| incoming | If true, the call is incoming |
| status | Call status: TRYING, RING, SESSION_PROGRESS, BUSY, ESTABLISHED, HOLD, FINISH, FAILED |
| info | |
| sipMessageRaw | Original message corresponding to the message being sent. For example, in case of TRYING, this would be SIP 100 TRYING response, in case of ESTABLISHED, this would be SIP 200 OK response, and in case of HOLD, this would be SIP 200 OK response to re-INVITE, and so on |
| sipStatus | Response status received from SIP side |
| caller | |
| callee | |

| | |
|---------------|------------------------------|
| hasVideo | |
| visibleName | |
| mediaProvider | |
| nodeId | |
| sessionId | |
| appKey | |
| answer | Answer incoming call |
| callId | |
| incoming | |
| sipStatus | |
| caller | |
| callee | |
| hasVideo | |
| visibleName | |
| mediaProvider | |
| sdp | |
| status | |
| nodeId | |
| sessionId | |
| appKey | |
| hangup | Hangs up the call |
| callId | |
| hasVideo | |
| nodeId | |
| sessionId | |
| appKey | |
| hold | Puts the call on hold |
| callId | |

| | |
|----------------------------|---|
| hasVideo | |
| nodeId | |
| sessionId | |
| appKey | |
| unhold | Unhold the call |
| callId | |
| hasVideo | |
| nodeId | |
| sessionId | |
| appKey | |
| transfer | Transfer the call |
| callId | |
| target | The number or the SIP URI of the subscriber the call is transferred to. |
| nodeId | |
| sessionId | |
| appKey | |
| TransferStatusEvent | Call transfer status change |
| callId | |
| incoming | If true, the transfer was initiated by the other side. |
| status | Call transfer status: ACCEPTED , TRYING , COMPLETED , FAILED . If the status is not recognized, then status received from SIP side will be passed |
| info | |
| sipMessageRaw | |
| hasVideo | |
| nodeId | |
| sessionId | |
| appKey | |

| | |
|--------------------|---|
| sendDTMF | Sends DTMF signal |
| callId | |
| dtmf | A symbol to pass in DTMF as text: 0-9, *, # |
| type | The type of the DTMF signal: <code>INFO</code> , <code>INFO_RELAY</code> , <code>RFC2833</code> |
| nodeId | |
| sessionId | |
| appKey | |
| sendMessage | Sends a message |
| id | The unique id of the message. |
| from | The number of the SIP URI of the sender. |
| to | The number, the login or the SIP URI of the recipient. |
| body | The text of the message. |
| contentType | <p><code>text/plain</code>, - the message is sent as SIP MESSAGE with the <code>Content-Type: text/plain</code> header and with text in the body of the message.</p> <p><code>message/cpim</code>, - the message is sent as SIP MESSAGE with the <code>Content-Type:message/cpim</code> header and a text/plain message in the body of the CPIM-message.</p> <p><code>multipart/mixed</code>, - the message is sent as SIP MESSAGE with the <code>Content-Type:multipart/mixed</code> header and CPIM messages in the body, each one containing one text/plain message.</p> |
| isImdnRequired | If the flag is set to true, for <code>message/cpim</code> and <code>multipart/mixed</code> message types, the information asking for a delivery notification via IMDN will be added to the body of the CPIM message. |
| recipients | The list of recipients separated by commas. SIP URI, tel URI or SIP logins of recipients must be specified and separated by commas. The field is used only if <code>ContentType</code> is set to <code>multipart/mixed</code> . This field works correctly only when the SIP-server supports sending messages to multiple subscribers based on multipart/mixed. WCS sends a multipart/mixed message with multiple recipients to the SIP-server. If yo |

| | |
|---------------------------|--|
| | ur SIP-server doesn't support such sending, leave this field blank and try sending several individual messages. |
| nodeId | |
| sessionId | |
| appKey | |
| OnMessageEvent | Incoming message |
| id | |
| from | |
| to | |
| body | |
| contentType | |
| isImdnRequired | If the incoming message has this flag, an IMDN delivery notification will be sent. |
| sipMessageRaw | A SIP MESSAGE message that corresponds to the OnMessageEvent event of the incoming message. |
| status | |
| nodeId | |
| sessionId | |
| appKey | |
| MessageStatusEvent | Message status change |
| id | |
| from | |
| to | |
| contentType | |
| isImdnRequired | |
| body | |
| status | Message status: RECEIVED , ACCEPTED , FAILED , IMDN_DELIVERED , IMDN_FAILED , IMDN_NOTIFICATION_SENT |
| info | |

| | |
|--------------------------------|--|
| sipMessageRaw | SIP message corresponding to the status: - ACCEPTED , - SIP 200 OK Response to SIP MESSAGE Request - FAILED , - SIP 4xx Response from the SIP-server - DELIVERED , - received a SIP MESSAGE delivery notification with the status Delivered - DELIVERY_FAILED , - received a delivery notification with the status Delivery Failed |
| nodeId | |
| sessionId | |
| appKey | |
| sendIMDN | Sends IMDN delivery notification |
| messageId | |
| nodeId | |
| sessionId | |
| appKey | |
| subscribe | SIP subscribe - subscribe to notification of the SIP-server: RFC3265 |
| event | Event type: reg |
| expires | Time interval in seconds. During this interval the WCS-server performs re-SUBSCRIBE. |
| terminate | |
| nodeId | |
| sessionId | |
| appKey | |
| SubscriptionStatusEvent | SIP-subscription status change |
| event | |
| expires | |
| terminate | If true, the subscription should be deactivated |
| requestBody | XML received from SIP side |
| status | Subscription status: Active , Terminated |

| | |
|------------------------|---|
| info | |
| sipMessageRaw | SIP message changing the status of the subscription: - Active, - SIP 200 OK Response on SUBSCRIBE request - Terminated, - SIP 200 OK Response on SUBSCRIBE request with expires:0 - Terminated, - SIP NOTIFY request with the terminated status in the body of the NOTIFY message |
| nodeId | |
| sessionId | |
| appKey | |
| sendXcapRequest | Send an XCAP request |
| url | URL for the XCAP request |
| nodeId | |
| sessionId | |
| appKey | |
| XcapStatusEvent | Receiving XCAP response |
| url | |
| xcapResponse | The body of the XCAP response |
| publishStream | Publishing the stream to the server |
| name | The name of the published stream. Must be unique. If a stream with such name already published, the publishing of the stream is prohibited. |
| mediaSessionId | Identifier of media session |
| published | If true, the stream is being published |
| hasVideo | If true, the stream has video |
| status | |
| sdp | SDP received from client |
| nodeId | |
| sessionId | |
| appKey | |
| record | If true, the published stream is being recorded |

| | |
|------------------------|---|
| custom | Custom object used to authenticate client on backend server |
| unPublishStream | Unpublishing the stream |
| name | |
| mediaSessionId | |
| published | |
| hasVideo | |
| status | |
| sdp | |
| nodeId | |
| sessionId | |
| appKey | |
| record | |
| custom | Custom object received in <code>/publishStream</code> hook |
| playStream | Play the stream |
| name | The name of the played stream. |
| mediaSession | |
| published | |
| hasVideo | |
| status | |
| sdp | |
| nodeId | |
| sessionId | |
| appKey | |
| custom | Custom object used to authenticate client on backend server |
| playHLS | Play the stream via HLS |
| name | |
| mediaSessionId | |

| | |
|--------------------------|---|
| mediaProvider | |
| nodeId | |
| sessionId | |
| appKey | |
| token | Client authentication token |
| playRTSP | Play the stream via RTSP |
| name | |
| mediaSessionId | |
| mediaProvider | |
| nodeId | |
| sessionId | |
| appKey | |
| rtspUrl | RTSP stream URL |
| User-Agent | Client user agent |
| stopStream | Stop playback of the stream |
| name | |
| mediaSessionId | |
| published | |
| hasVideo | |
| status | |
| sdp | |
| nodeId | |
| sessionId | |
| appKey | |
| custom | Custom object received in <code>/playStream</code> hook |
| StreamStatusEvent | Stream status change |
| name | |
| | |

| | |
|-----------------------------|---|
| status | Stream status: PUBLISHING , UNPUBLISHED , PLAYING , STOPPED , PL |
| mediaSessionId | |
| published | |
| hasVideo | |
| sdp | |
| info | |
| nodeId | |
| sessionId | |
| appKey | |
| record | |
| custom | Custom object received in /publishStream or /playStream hook |
| StreamKeepAliveEvent | Stream keep-alive REST request |
| nodeId | |
| appKey | |
| sessionId | |
| mediaSessionId | |
| name | |
| published | |
| hasVideo | |
| status | Stream status: PLAYING , PUBLISHING |
| info | |
| mediaProvider | Media technology used in WCS JavaScript API, possible values: WebRTC , Flash |
| record | |
| sendData | Sends data |
| operationId | Unique id of the data to send |
| payload | JSON object containing data |

| | |
|-----------------------------|--|
| nodeId | |
| sessionId | |
| appKey | |
| OnDataEvent | Receiving of input data |
| operationId | |
| payload | |
| nodeId | |
| sessionId | |
| appKey | |
| DataStatusEvent | Sent data status change |
| operationId | |
| status | ACCEPTED , FAILED |
| info | |
| nodeId | |
| sessionId | |
| appKey | |
| ErrorEvent | Unclassified error |
| info | Additional information about the error |
| sendBugReport | Sends an error report to save on the server |
| text | Brief custom description of the error |
| type | If the type is no_media , the server enables traffic dump before creating a bug report to make sure the traffic goes properly for that user. Sending bug reports of this type can help diagnose problems with sound going one side only |
| nodeId | |
| sessionId | |
| appKey | |
| BugReportStatusEvent | Error report sending confirmation with the name of the saved file as the output |

| | |
|---------------------------|---|
| filename | The name of the file on the server where the bug report was saved |
| nodeId | |
| sessionId | |
| appKey | |
| sendStreamEvent | Publishing stream event |
| info | Additional info |
| type | Stream event type: audioMuted, videoMuted, audioUnmuted, videoUnmuted |
| mediaSessionId | Publishing media session Id |
| nodeId | |
| sessionId | |
| appKey | |
| StreamEvent | Publishing stream event for subscribers |
| info | Additional info |
| type | Stream event type: audioMuted, videoMuted, audioUnmuted, videoUnmuted |
| mediaSessionId | Subscriber media session Id |
| nodeId | |
| sessionId | |
| appKey | |
| Context Parameters | Context parameters. Used for all calls from WCS to the Web-server |
| nodeId | Unique id of the WCS server instance |
| sessionId | Unique id of the client connected in that instance |
| appKey | Application id on the WCS server the user has established connection with |