Genesys Mobile Services API Reference

Chat API Version 2 via CometD

4/3/2026

Chat API Version 2 via CometD

Introduced in 8.5.109

Client applications can use the GMS Chat API Version 2 to initiate an anonymous and unauthenticated chat session from a client application. GMS then routes the interaction to an agent. The Agent application, usually a desktop, uses another distinct Genesys API to participate in the chat session.

See also the Chat Version 2 CometD Sample page. Link to video

Versions	Changes
8.5.109	Genesys Mobile Enterprise supports the Digital Channels Chat API V2 over CometD channel. Most of the time, requests, responses, parameter names, and error codes are identical between the REST and CometD APIs.
8.5.110	GMS supports CometD WebSocket Transport. To enable Secure WebSocket, deploy GMS with an SSL certificate.
8.5.112	• Digital Channels Chat API V2 over CometD now supports File Transfer operations. • Chat API with CometD usability is enhanced and request parameters such as `alias`, `userId`, and `chatId` are now deprecated. To ensure backward compatibility, GMS nodes will still include those keys with bogus values in notification events.
8.5.114	• Chat V2 API over CometD now supports Firebase Cloud Messaging push notifications, that allow delivering notifications of new chat events in the outgoing chat session to a mobile terminal, while the main application is offline or in background. • Chat V2 API over CometD now supports native push notifications through Apple Push Notification Service. APNS support is limited to Legacy Binary Provider API.
8.5.201	Chat V2 API over CometD now supports Read receipt.

Versions	Changes
8.5.219	The push_notification_to_active_client option was introduced to stop sending mobile and HTTP push notifications in chat sessions when the CometD connection is established. If you configure this option to `true`, mobile and HTTP push notifications for new events in chat sessions are only sent when there is no active CometD connection.

Prerequisites for the CometD API

Server	Minimum version	Configuration details
GMS	8.5.109.05+	• GMS can run in chat only or in full mode. • To enable the notification mode, set to `true` the option enable_notification_mode in the `chat` section. • If you use a load balancer in front of several GMS nodes, enable the sticky sessions. The option enable_notification_mode is not dynamic. You cannot modify it at runtime.
Chat Server	8.5.108+	• All the connected Chat Server instances need to be deployed in N+1 configuration with session restoration. • Enable the notification mode by setting to `true` the option `flex-push-enabled` in the `settings` section. • Configure a digital channel

Important

Configuration changes to the connected Chat Server instance disable this instance for serving notifications.

- Do all changes when GMS is offline. Note that you don't need to stop the Chat Server to process the configuration changes.

- If you absolutely need to keep GMS online, remove the connection to the Chat Server instance from the GMS cluster, do configuration changes on the instance, and add the connection to the Chat Server again in the GMS cluster.

How to use the CometD API

When you are using the CometD API, both the client and server applications exchange messages with a JSON payload.

- The client application can request an operation from the GMS server, and the GMS server responds with notifications.
- GMS also delivers unsolicited notifications when an event happens during the chat session.

All operations are asynchronous, which means that notifications respond and confirm the requested operations.

CometD in GMS supports:

- HTTP JSON Long Polling
- JSONP with Callback
- Web Socket

CometD in GMS supports WebSockets unless there are some other limitations, such as a Load Balancer. In this scenario, long polls need to end up on the same GMS Node and require that you enable sticky sessions.

To implement your client, use a CometD library such as CometD Javascript or CometD-Faye.

CometD selects the best available transport. When implementing your application, you do not need to take into account the transport, as it doesn't modify GMS API queries and their order.

Important

Use a standard CometD client library to implement your client application. Chat V2 over CometD is tested and intended to be used only with CometD libraries.

Handshake with GMS

The first step is a handshake with the GMS node, using the full GMS URI; for instance: `http://<gms-hostname>:8080/genesys/cometd`

Subscribe for a channel

After a successful handshake, you need to either subscribe to or add a listener for your channel. The channel name depends on the digital channel that you created before. It includes the chat service name and is formatted as follows: `/service/chatV2/<chat service>`.

For instance, if you create and configure the chat service named `customer-support`, the associated chat channel is `/service/chatV2/customer-support`

Example of handshake and channel listener:

```
var gmsCometURL = "http://my.gms.host:8080/genesys/cometd";
var chatChannel = "/service/chatV2/customer-support";

function handleChatEvent( message) { ... }

cometd.addListener( chatChannel, handleChatEvent )
cometd.configure({
    url: gmsCometURL,
    logLevel: 'debug'
});
cometd.handshake();
```

Example of handshake and subscription:

```
var gmsCometURL = "http://my.gms.host:8080/genesys/cometd";
var chatChannel = "/service/chatV2/customer-support";

function handleChatEvent( message ) { ... }

function _handleHandshake( handshake ) {
    if ( true === handshake.successfull ) {
        cometd.batch( function() { cometd.subscribe( chatChannel, handleChatEvent ); } );
    }
}

cometd.addListener( '/meta/handshake', _handleHandshake );
cometd.configure({
    url: gmsCometURL,
    logLevel: 'debug'
});
cometd.handshake();
```

Receive Notifications

To start receiving notifications, you can either:

- Use the `requestChat` operation to request a new chat for your channel.
- Request notifications for an existing chat through the `requestNotifications` operation. In this scenario, you must specify the channel where the chat was created.

If the `requestChat` and `requestNotifications` operations succeed, GMS starts sending unsolicited notifications through CometD connection about the chat session activity, for example, when an agent connects or leaves chat session, sends a message, and so on.

- Unsolicited notifications have exactly one event.
- Notifications sent as a response to a request may have one or no events, except for notifications sent in response to the `requestChat` and `requestNotifications` requests which may have 0, 1, or more events.
- Your client application should read the `secureKey` parameter from each received notification response, and reuse it with each subsequent request message as its value can change, for example, when the chat session is moved to another Chat Server instance.
- Each chat event has an index matching the event order. Some gaps are possible because GMS does not deliver all session events to client applications.

> **Important**
>
> You cannot have multiple `requestChat` requests in one CometD instance. A separate CometD connection must be created for each new chat session.

See Digital Channels Chat V2 Response Format for additional details about the notification structure. Here is a notification example:

```
[
    {
        "data": {
            "messages": [
                {
                    "from": {
                        "nickname": "a1001",
                        "participantId": 3,
                        "type": "Agent"
                    },
                    "index": 8,
                    "text": "hello",
                    "type": "Message",
                    "utcTime": 1590051516000
                }
            ],
            "chatEnded": false,
            "statusCode": 0,
            "alias": "0",
            "secureKey": "zpcRUAk",
            "userId": "deprecated",
            "chatId": "deprecated",
            "nextPosition": 9,
            "monitored": false,
            "channel": "/service/chatV2/customer-support"
        }
    }
]
```

When your client application processes events, it needs to check the index of each event received and compare it with the index of the last processed event. If the index is identical or smaller, the event can be ignored. If the index of the newly received event is greater than the index of the last

processed event, the client application should update the UI with the new event, as appropriate, and remember the new index as the last processed event.

> **Important**
> Starting with release 8.5.112, the request parameters such as `alias`, `userId`, and `chatId` are deprecated.

Handling disconnections

In case of disconnection, the client application should perform a new handshake if necessary, and use the `requestNotifications` operation with the following information:

- The latest known transcript position
- Identical `userId`, `chatId`, and `secureKey` parameters

If the chat session is still ongoing, the client will receive a notification with a new (possibly the same) alias to use, and the list of the chat events that happened since the last known position.

> **Important**
> Starting with release 8.5.112, the request parameters such as `alias`, `userId`, and `chatId` are deprecated.

Authentication

If you enable Basic Authentication for the chat service accessed through the CometD API, you must provide the authentication credentials in the `auth` parameter of the `requestChat` operation. There are two ways to provide credentials in an `auth` object:

- In an open form containing the `username` and `password` fields
- In an encoded form using encoded fields, similar to the Basic Authentication header, which is a Base64-encoded composite string of "username:password"

Here is an open-form example of a request message with authentication credentials:

```
var channel ="/service/chatV2/customer-support";
var requestChatData ={
    "operation": "requestChat",
    "firstName": "Joan",
    "lastName": "Smith",
    "subject": "Savings Account",
    "userData": {
        "key1": "value1",
        "key2": "value2"
    },
```

```
    "auth": {
        "username": "user1",
        "password": "pass1"
    }
}
cometd.publish( channel, requestChatData);
```

Here is an encoded-form example of a request message with authentication credentials:

```
var channel ="/service/chatV2/customer-support";
var requestChatData ={
    "operation": "requestChat",
    "firstName": "Joan",
    "lastName": "Smith",
    "subject": "Savings Account",
    "userData": {
        "key1": "value1",
        "key2": "value2"
    },
    "auth": {
        "encoded": "dXNlcjE6cGFzczE="
    }
}
cometd.publish( channel, requestChatData);
```

(Optional) Enable Push Notifications for Mobile

CometD supports the following Push Notifications for Mobile:

- Firebase Cloud Messaging (FCM)
- Apple Push Notifications or iOS
- Custom HTTP Notifications

If push notifications are enabled, GMS sends these notifications independently from the CometD operations and status. The receiver, i.e. mobile application, needs to implement the appropriate logic for processing the push notifications – based on the current state of the mobile application.

Provide Push Notification Parameters in User Data

When your client application submits `requestChat` and `requestNotifications` queries, the request must include the `push_notification_*` parameters in its user data. These data identify the mobile terminal and your client app must provide the exact same data each time it re-establishes a CometD connection and issues the `requestNotifications` operation.

> **Important**
>
> The `push_notification_*` user data are detailed in this page in the requestChat and requestNotifications sections.

Apple Push Notifications

Warning

The Apple Push Notification service (APNs) will no longer support the legacy binary protocol as of March 31, 2021. Read more.

This means that the APN service will no longer be accessible for the legacy binary protocol. Genesys Mobiles Services will stop supporting the legacy binary protocol, which is the default protocol for Apple Push notifications. For further details, read End of Genesys Support for Apple Push Notification service (APNs) – Legacy binary protocol.

Genesys recommends that you migrate to a supported notification type such as:

- Firebase Cloud Notification
- Custom HTTP Notification

Refer to the Apple Notification documentation to configure your Apple Push Notification Service provider. You can define several different **provider** sections and associate any of them when starting a chat service with the `requestChat` and `requestNotifications` API queries.

- If you do not add provider information to the user data, the system will use the default options that you configured in the push section of the GMS configuration.
- As stated above, default notifications are submitted in English only. To support additional languages, you must provision the apple.title and apple.alertMessage.body options at the provider service event level, or intercept notifications on the terminal, process them in the background to change text, and present a new notification to the user.

Firebase Cloud Messaging (FCM)

Introduced in 8.5.114

Configure Firebase Cloud Messaging in GMS

Refer to the Push Notification Service documentation to configure your Firebase Cloud Messaging provider. You can define several different **provider** sections and associate any of them when starting a chat service with the `requestChat` and `requestNotifications` API queries.

- If you do not add provider information to the user data, the system will use the default Firebase Cloud Messaging options that you configured in the push section of the GMS configuration.
- If you specify a non-default provider in the user data of the query, the system will use the Firebase Cloud Messaging options that are configured in the [push.provider.{ProviderName}] section, where {ProviderName} is the string to provide in the `push_notification_provider` user data of the `requestChat` operation.

Firebase Push Notification Limitations

- Only one session per client is supported. If the same user tries to login to the mobile app from two devices, only one device will be able to manage the chat session in progress.
- The client app must create the chat session over Chat V2 with CometD in order to receive notifications. Furthermore, the client app must use Chat V2 API with CometD whenever it is sent in the foreground. On background switchovers, Genesys recommends to perform a CometD disconnection or to unload the app from memory to preserve the device's battery.
- As stated above, default notifications are submitted in English only. To support additional languages, you must provision the fcm.title and fcm.body options at the provider service event level, or intercept notifications on the terminal, process them in the background to change text, and present a new notification to the user.

Push Notification Events

You can receive the following notifications with CometD.

Event Name	Title default value	Body default value	Comments
`ParticipantJoined`	"Chat participants changed"	"%s joined chat"	%s is substituted with the party name
`ParticipantLeft`	"Chat participants changed"	"%s left chat"	
`Message`	"%s sent new message"	The message sent by the participant.	
`PushUrl`	"%s suggests Web page"	URL pushed by the participant.	
`FileUploaded`	"%s suggests to download file"	"Open app for more info"	
`IdleAlert`	"Chat will close soon"	"Please use app if you want to continue"	
`IdleClose`	"Chat is over"	"Your chat session was ended due to inactivity"	

> **Important**
>
> The `FileDeleted`, `CustomNotice`, and `Notice` events are not pushed even if they are available in the chat transcript.

CometD API

Request Chat

Modified in 8.5.112

Use this method to request a new chat session and start receiving the associated notifications. If the `requestChat` operation is successful, you will receive a notification response data object that contains the parameters required for subsequent messages and operations:

- `chatId`—ID for the new chat session
- `userId`—User ID used for the chat session
- `secureKey`—Secure key for this session
- `alias`—Chat host alias used for the session

See also the Subscribe and Authentication sections for further use cases and details.

Operation

Operation	**requestChat**		
Parameter	**Type**	**Mandatory**	**Description**
nickname	string	yes	Customer's nickname. Genesys recommends that you supply either `nickname` or both `firstName` and `lastName` parameters. For example: "JohnDoe"
lastName	string	no	Last name of the customer. Genesys

Operation	requestChat		
			recommends that you supply either `nickname` or both `firstName` and `lastName` parameters. For example: "Doe"
subject	string	no	The subject as entered by the customer. For example: "Help with account"
emailAddress	string	no	The email address of the customer. For example: "jdoe@gmail.com"
userData	JSON-formatted string	no	Any attached data that the client wants to add to the chat session. For example: `{ "key":"value", ... }`

> **Important**
>
> Starting in 8.5.112, the request parameters such as `alias`, `userId`, and `chatId` are deprecated.

Starting in 8.5.114, you can add the following user data (that must be identical in the `requestNotifications` query) to enable Firebase Cloud Messaging or APNS push notifications.

User Data Key	Value	Required	Description
push_notification_deviceid	ID	Yes	Unique deviceId or token received from FCM or APNS. Your client app must provide the exact same data each time it re-establishes a CometD connection and issues the `requestNotifications` operation.
push_notification_type	fcm ios customhttp	Yes	• "fcm" for Google Firebase Messaging client • "ios" for Apple Push Notification Service • "customhttp" for Custom HTTP Service

User Data Key	Value	Required	Description
			Your client app must provide the exact same data each time it re-establishes a CometD connection and issues the `requestNotifications` operation.
push_notification_debug	"true" "false" (default)	No	If true, debug mode for Firebase Cloud Messaging or Apple notification service will be used. Note if your Apple certificate is available for debug mode only, set this flag to `true`; otherwise, the device will not receive notifications.
push_notification_language	ISO language code	No	Reserved for now. Defines the language for the client application. The language is en_US by default.
push_notification_provider	string	No	Defines the name of the GMS provider configuration section to use for this chat session. If you do not provide a name, the system will use the default provider defined in the push section. **Important** Genesys recommends to use short strings that include short words and exclude any special symbols or punctuation.

Example

The following is an example of a request.

```
var channel ="/service/chatV2/customer-support";
var requestChatData ={
    "operation": "requestChat",
    "firstName": "Joan",
    "lastName": "Smith",
    "subject": "Savings Account",
    "userData": {
        "key1": "value1",
        "key2": "value2"
    }
```

```
}
cometd.publish(channel, requestChatData);
```

The following is an example of the notification response.

```
{
    "statusCode": 0,
    "alias": "117",
    "userId": "007555677B20000A",
    "secureKey": "4ee15d7e1c343c8e",
    "messages": [
        {
            "from": {
                "nickname": "Joan Smith",
                "participantId": 1,
                "type": "Client"
            },
            "index": 1,
            "type": "ParticipantJoined",
            "utcTime": 1432845088000
        }
    ],
    "chatId": "00048aAPEQJ8000U",
    "nextPosition": 2
}
```

Request Notifications

Modified in 8.5.112

Use this operation to request notifications to be delivered for the existing chat session, after you start a new chat session or after the CometD channel has been disconnected and reconnected. See Digital Channels Chat V2 Response Format for all of the notification details.

Operation

Operation	**requestNotifications**		
Parameter	**Type**	**Mandatory**	**Description**
chatId **Deprecated in 8.5.112**	Long	yes	Chat ID that GMS sent in response to the `requestChat` operation. For example: "00048aAPEQJ800U"
userId **Deprecated in 8.5.112**	Long	yes	User ID that GMS sent in response to the `requestChat` operation. For example: "007555677B20000A"
secureKey	Long	yes	Secure key that GMS sent in response to the `requestChat` operation. For example: "4ee15d7e1c343c8e"

Operation	requestNotifications		
alias **Deprecated in 8.5.112**	string	yes	Chat host alias.
transcriptPosition	integer	no	The transcript event position from which the client application would like to receive the previous events. If you set this option to 0 or if you don't set this option, the client will receive all the events.

> **Important**
>
> Starting in 8.5.112, the request parameters such as `alias`, `userId`, and `chatId` are deprecated.

Starting in 8.5.114, you can add the following user data (that must be identical in the `requestChat` query) to enable Firebase Cloud Messaging or APNS push notifications.

User Data Key	Value	Required	Description
push_notification_deviceid	ID	Yes	Unique deviceId or token received from FCM or APNS. Your client app must provide the exact same data each time it re-establishes a CometD connection and issues the `requestNotifications` operation.
push_notification_type	fcm ios customhttp	Yes	• "fcm" for Google Firebase Messaging client • "ios" for Apple (APNS) client • "customhttp" for Custom HTTP Service Your client app must provide the exact same data each time it re-establishes a CometD connection and issues the `requestNotifications` operation.

User Data Key	Value	Required	Description
push_notification_debug	"true" "false" (default)	No	If true, debug mode for Firebase Cloud Messaging or Apple notification service will be used. Note if your Apple certificate is available for debug mode only, set this flag to true, otherwise, the device will not receive notifications.
push_notification_language	ISO language code	No	Reserved for now. Defines the language for the client application. The language is en_US by default.
push_notification_provider	string	No	Defines the name of the GMS provider configuration section to use for this chat session. If you do not provide a name, the system will use the default provider defined in the push section. **Important** Genesys recommends to use short strings that include short words and exclude any special symbols or punctuation.

Example

Example of request:

```
var channel ="/service/chatV2/customer-support";

var requestNotificationData ={
    "operation": "requestNotifications",
    "alias": "117",
    "chatId": "00048aAPEQJ8000U",
    "userId": "007555677B20000A",
    "secureKey": "4ee15d7e1c343c8e",
    "transcriptPosition": "2"
}
cometd.publish( channel, requestNotificationData );
```

Example of received notification:

```
{
    "messages": [
        {
            "from": {
```

```
                "nickname": "Joan Smith",
                "participantId": 1,
                "type": "Client"
            },
            "index": 2,
            "text": "Hello, ...",
            "messageType": "text",
            "type": "Message",
            "utcTime": 1432845541000
        },
    ],
    "chatEnded": false,
    "statusCode": 0,
    "alias": "117",
    "secureKey": "3e2a69d421ae2672",
    "userId": "007555677CB4000D",
    "chatId" : "00048aAPEQJ8000U",
    "nextPosition": 2
}
```

Send message

Use this operation to deliver or send messages from the client application to the chat session.

Operation	sendMessage		
Parameter	**Type**	**Mandatory**	**Description**
message	string	yes	Text message to send. For example: "I need help with my account."
chatId **Deprecated in 8.5.112**	long	yes	The chat ID. For example: "00048aAPEQJ800U"
userId **Deprecated in 8.5.112**	long	yes	The user ID. For example: "007555677B20000A"
secureKey	long	yes	Secure key. For example: "4ee15d7e1c343c8e"
alias **Deprecated in 8.5.112**	string	yes	Chat host alias. For example: "117"
messageType	string	no	Any arbitrary type that the user wants to set. For example: "text"

Example

Example of request:

```
var channel ="/service/chatV2/customer-support";
var sendData ={
    "operation": "sendMessage",
    "message": "Hello, ...",
```

```
    "chatId": "00048aAPEQJ8000U",
    "userId": "007555677B20000A",
    "secureKey": "4ee15d7e1c343c8e"
}
cometd.publish( channel, sendData );
```

Example of notification response:

```
{
    "messages": [
        {
            "from": {
                "nickname": "Joan Smith",
                "participantId": 1,
                "type": "Client"
            },
            "index": 2,
            "text": "Hello, ...",
            "messageType": "text",
            "type": "Message",
            "utcTime": 1432845541000
        },
    ],
    "chatEnded": false,
    "statusCode": 0,
    "alias": "117",
    "secureKey": "3e2a69d421ae2672",
    "userId": "007555677CB4000D",
    "chatId" : "00048aAPEQJ8000U",
    "nextPosition": 3
}
```

Read receipt

Introduced in: 8.5.201.04

Use this operation to acknowledge that the user has read a given message or event.

Operation	**readReceipt**		
Parameter	**Type**	**Mandatory**	**Description**
secureKey	long	yes	Secure key. For example: "4ee15d7e1c343c8e"
transcriptPosition	integer	yes	The index of the event that the client app acknowledges having read. This is not the same index as `nextPosition`, but rather the index of the event from the "messages" array.

Example

Example of request:

```
var channel = "/service/chatV2/customer-support"
...
var readReceiptData = {
    "operation":"readReceipt",
    "transcriptPosition" : "5",
    "secureKey": "4ee15d7e1c343c8e"
}
cometd.publish( channel, updateNicknameData );
```

Example of notification response:

```
{
    "messages":[],
    "chatEnded":false,
    "statusCode":0,
    "secureKey": "4ee15d7e1c343c8e"
    "nextPosition" : 9
}
```

Start Typing

Use this operation to indicate that the client has started typing a message. You can include a partial message, if desired, for typing preview.

Operation

Operation	**startTyping**		
Parameter	**Type**	**Mandatory**	**Description**
chatId **Deprecated in 8.5.112**	long	yes	The chat ID. For example: "00048aAPEQJ800U"
userId **Deprecated in 8.5.112**	long	yes	The user ID. For example: "007555677B20000A"
secureKey	long	yes	Secure key. For example: "4ee15d7e1c343c8e"
alias **Deprecated in 8.5.112**	string	yes	Chat host alias. For example: "117"
message	string	no	Optional preview typing message to send. For example: "I need help with my account."

Example

Example of request:

```
var channel ="/service/chatV2/customer-support";
var startTypingData ={
    "operation": "startTyping",
    "message": "Hello, ...",
    "chatId": "00048aAPEQJ8000U",
    "userId": "007555677B20000A",
    "secureKey": "4ee15d7e1c343c8e"
}
cometd.publish( channel, sendData );
```

Example of notification response:

```
{
    "messages": [
        {
            "from": {
                "nickname": "John Smith",
                "participantId": 1,
                "type": "Client"
            },
            "index": 3,
            "text": "hello, I ha",
            "type": "TypingStarted",
            "utcTime": 1493509617000
        }
    ],
    "chatEnded": false,
    "statusCode": 0,
    "alias": "117",
    "secureKey": "3e2a69d421ae2672",
    "userId": "007555677CB4000D",
    "chatId" : "00048aAPEQJ8000U",
    "nextPosition": 4
}
```

Stop Typing

Use this operation to indicate that the client has stopped typing a message. You can include a partial message, if desired, for typing preview.

Operation

Operation	**stopTyping**		
Parameter	**Type**	**Mandatory**	**Description**
chatId **Deprecated in 8.5.112**	long	yes	The chat ID. For example: "00048aAPEQJ800U"
userId **Deprecated in 8.5.112**	long	yes	The user ID. For example: "007555677B20000A"
secureKey	long	yes	Secure key. For example: "4ee15d7e1c343c8e"
alias **Deprecated in**	string	yes	Chat host alias. For example: "117"

Operation	stopTyping		
8.5.112			
message	string	no	Optional preview typing message to send. For example: "I need help with my account."

Example

Example of request message:

```
var channel ="/service/chatV2/customer-support";
var sendData ={
    "operation": "stopTyping",
    "message": "Hello, ...",
    "chatId": "00048aAPEQJ8000U",
    "userId": "007555677B20000A",
    "secureKey": "4ee15d7e1c343c8e"
}
cometd.publish( channel, sendData );
```

Example of notification response data object:

```
{
    "messages": [
        {
            "from": {
                "nickname": "John Smith",
                "participantId": 1,
                "type": "Client"
            },
            "index": 4,
            "text": "hello, I have a question",
            "type": "TypingStopped",
            "utcTime": 1493509618000
        }
    ],
    "chatEnded": false,
    "statusCode": 0,
    "alias": "117",
    "secureKey": "3e2a69d421ae2672",
    "userId": "007555677CB4000D",
    "chatId" : "00048aAPEQJ8000U",
    "nextPosition": 5
}
```

Disconnect

Use this operation to finish the chat session. The notification response does not include a `secureKey` or `userId` parameter. Your application needs to clean up `chatId`, `userId`, and `secureKey` values and forget about them. Any subsequent operation with these IDs will generate error notification responses.

Operation

Operation	disconnect		
Parameter	**Type**	**Mandatory**	**Description**
chatId **Deprecated in 8.5.112**	long	yes	The chat ID. For example: "00048aAPEQJ800U"
userId **Deprecated in 8.5.112**	long	yes	The user ID. For example: "007555677B20000A"
secureKey	long	yes	Secure key. For example: "4ee15d7e1c343c8e"
alias **Deprecated in 8.5.112**	string	yes	Chat host alias. For example: "117"

Example

Example of request message:

```
var channel ="/service/chatV2/customer-support";
var disconnectData ={
    "operation": "disconnect",
    "alias": "117",
    "chatId": "00048aAPEQJ8000U",
    "userId": "007555677B20000A",
    "secureKey": "4ee15d7e1c343c8e"
}
cometd.publish( channel, disconnectData );
```

Example of notification response:

```
{
   "messages":[],
   "chatEnded":true,
   "statusCode":0,
   "alias":"117",
   "chatId" : "00048aAPEQJ8000U",
   "nextPosition":1
}
```

Push URL

Use this operation to push a web page to the agent.

Operation

Operation	pushUrl		
Parameter	**Type**	**Mandatory**	**Description**
chatId **Deprecated in**	long	yes	The chat ID. For example:

Operation	pushUrl		
8.5.112			"00048aAPEQJ800U"
userId **Deprecated in 8.5.112**	long	yes	The user ID. For example: "007555677B20000A"
secureKey	long	yes	Secure key. For example: "4ee15d7e1c343c8e"
alias **Deprecated in 8.5.112**	string	yes	Chat host alias. For example: "117"
pushUrl	string	yes	URL to push to the agent. For example: "http://www.genesys.com"

Example

Example of request.

```
var channel ="/service/chatV2/customer-support";
var pushUrlData ={
    "operation": "pushUrl",
    "pushUrl": "http://www.genesys.com",
    "chatId": "00048aAPEQJ8000U",
    "userId": "007555677B20000A",
    "secureKey": "4ee15d7e1c343c8e"
}
cometd.publish( channel, pushUrlData );
```

Example of notification response:

```
{
    "messages": [
        {
            "from": {
                "nickname": "John Smith",
                "participantId": 1,
                "type": "Client"
            },
            "index": 2,
            "text": "http://www.google.com",
            "type": "PushUrl",
            "utcTime": 1493250733000
        }
    ],
    "chatEnded": false,
    "statusCode": 0,
    "alias": "117",
    "secureKey": "3e2a69d421ae2672",
    "userId": "007555677CB4000D",
    "chatId" : "00048aAPEQJ8000U",
    "nextPosition": 3
}
```

Update Nickname

Use this operation to update the customer's nickname.

Operation

Operation	updateNickname		
Parameter	**Type**	**Mandatory**	**Description**
chatId **Deprecated in 8.5.112**	long	yes	The chat ID. For example: "00048aAPEQJ800U"
userId **Deprecated in 8.5.112**	long	yes	The user ID. For example: "007555677B20000A"
secureKey	long	yes	Secure key. For example: "4ee15d7e1c343c8e"
alias **Deprecated in 8.5.112**	string	yes	Chat host alias. For example: "117"
nickname	string	yes	New nickname. For example: "John Doe 2"

Example

Example of request:

```
var channel = "/service/chatV2/customer-support"
...
var updateNicknameData = {
"operation":"updateNickname",
"nickname" : "MyNewNickname",
"chatId" : "00048aAPEQJ8000U",
"userId":"007555677B20000A",
"secureKey":"4ee15d7e1c343c8e"
}
cometd.publish( channel, updateNicknameData );
```

Example of notification response:

```
{
"messages":[{
     "from":{"nickname":"MyNewNickname","participantId":1,"type":"Client"},
     "index":8,
     "text":"MyNewNickname",
     "type":"NicknameUpdated",
     "utcTime":1493936549000
     }],
"chatEnded":false,
"statusCode":0,
"alias":"117",
"secureKey":"3e2a69d421ae2672",
"userId":"007555677CB4000D",
"chatId" : "00048aAPEQJ8000U",
"nextPosition" : 9
```

```
}
```

Custom Notice

Use this operation to send a custom notice to the agent. The agent will need a customized desktop that can process this notice.

Operation

Operation	**customNotice**		
Parameter	**Type**	**Mandatory**	**Description**
chatId **Deprecated in 8.5.112**	long	yes	The chat ID. For example: "00048aAPEQJ800U"
userId **Deprecated in 8.5.112**	long	yes	The user ID. For example: "007555677B20000A"
secureKey	long	yes	Secure key. For example: "4ee15d7e1c343c8e"
alias **Deprecated in 8.5.112**	string	yes	Chat host alias. For example: "117"
message	string	no	Optional message.

Example

Example of request message:

```
var channel = "/service/chatV2/customer-support"
...
var customNoticeData = {
 "operation":"customNotice",
 "message" : "ORDER UPDATE",
 "chatId" : "00048aAPEQJ8000U",
 "userId":"007555677B20000A",
 "secureKey":"4ee15d7e1c343c8e"
}
cometd.publish( channel, customNoticeData );
```

Example of notification response:

```
{
    "messages": [
        {
            "from": {
                "nickname": "John Smith",
                "participantId": 1,
                "type": "Client"
            },
            "index": 8,
            "text": "ORDER UPDATE",
            "type": "CustomNotice",
```

```
            "utcTime": 1493510733000
        }
    ],
    "chatEnded": false,
    "statusCode": 0,
    "alias": "117",
    "secureKey": "3e2a69d421ae2672",
    "userId": "007555677CB4000D",
    "chatId" : "00048aAPEQJ8000U",
    "nextPosition": 9
}
```

Update User Data

Use this operation to modify or add attached data to the chat session.

Operation

Operation	**updateData**		
Parameter	**Type**	**Mandatory**	**Description**
userId **Deprecated in 8.5.112**	long	yes	The user ID. For example: "007555677B20000A"
secureKey	long	yes	Secure key. For example: "4ee15d7e1c343c8e"
alias **Deprecated in 8.5.112**	string	yes	Chat host alias. For example: "117"
userData	JSON-formatted string	no	Any attached data that the client wants to add to the chat session. For example: { "key":"value", ... }

Example

Example of request message:

```
var channel = "/service/chatV2/customer-support"
...
var updateUDData = {
"operation":"updateData",
"chatId" : "00048aAPEQJ8000U",
"userId":"007555677B20000A",
"secureKey":"4ee15d7e1c343c8e",
"userData" : { "key3" : "value3", "key4" : "value4" }
}
cometd.publish( channel, updateUDData );
```

Example of notification response:

```
{
"messages":[],
```

```
"chatEnded":false,
"statusCode":0,
"alias":"117",
"secureKey":"3e2a69d421ae2672",
"userId":"007555677CB4000D",
"chatId" : "00048aAPEQJ8000U",
"nextPosition" : 9
}
```

Chat Session File Management

Updated in 8.5.112

Starting with release 8.5.112, the GMS Digital Channels API allows agents and customers to exchange files during their chat sessions using these requests:

- **Get Limits**—Check for allowable file types and file size—or other constraints on file uploads—before sending an upload request.
- **Upload File**—Upload a file.
- **Download File**—Download a previously uploaded file.
- **Delete File**—Delete a previously uploaded the file.

File Management requests are all POST requests submitted to a single URL endpoint:

```
POST /genesys/2/chat-ntf
```

Each request must include at least the `operation` and `secureKey` parameters.

Get Limits

Use this optional request to avoid wasting network and CPU overhead by checking for allowable file types or maximum file size—or other constraints on file uploads—before sending an upload request.

HTTP Header	Value
Content-Type	• multipart/form-data • application/x-www-form-urlencoded

Input Parameters

Parameter Name	Sample Value	Description	Required/Optional
operation	"fileGetLimits"	user ID	Required

Parameter Name	Sample Value	Description	Required/Optional
secureKey	"8b31761b2154884c"	secure key	Required

Output Parameters

Parameter	Description
download-attempts	Maximum number of times a specific file can be downloaded during this session
upload-max-files	Maximum number of files a client can upload during this session
upload-max-file-size	Maximum allowable file size of a single uploaded file
upload-max-total-size	Maximum allowable file size of all uploaded files
upload-need-agent	Indicates whether an agent needs to accept a chat session before an upload is allowed
upload-file-types	Colon-delimited list of file extensions indicating which types of files can be uploaded
used-upload-max-files	Current number of files uploaded during this session
used-upload-max-total-size	Current total size of all files uploaded during this session
used-download-attempts	Current number of downloaded files during this session
delete-file	Indicates whether **upload-max-files** is decremented after a file has been deleted

Example

```
http://localhost:8080/genesys/2/chat-ntf

operation=fileGetLimits
secureKey=8b31761b2154884c
```

Output

```
{
  "messages": [
        {
            "from": {
                "nickname": "JohnS",
                "participantId": 1,
                "type": "Client"
            },
            "text": "file-client-cfg-get",
            "type": "Notice",
            "utcTime": 1497840553000,
            "userData": {
                "download-attempts": "10",
                "upload-max-files": "3",
                "delete-file": "odd",
```

```
                "upload-max-file-size": "2097152",
                "used-download-attempts": "0",
                "used-upload-max-total-size": "0",
                "upload-need-agent": "true",
                "used-upload-max-files": "0",
                "upload-max-total-size": "5242880",
                "upload-file-types":
"bmp:csv:doc:docx:gif:htm:jpg:pdf:png:ppt:pptx:tif:txt:xls:xlsx"
            }
        }
    ],
  "chatEnded": false,
  "statusCode": 0,
  "secureKey": "799c5ff0faccd5bb"
}
```


Upload File

Use this request to upload a file into a chat session so it can be shared with an agent or a customer. You can then use the **file-id** value in the response to delete or download the file, as needed.

HTTP Header	Value
Content-Type	multipart/form-data

Input parameters

Parameter Name	Sample Value	Description	Required/ Optional	Source
operation	fileUpload	secure key	Required	Form
secureKey	"8b31761b2154884c"	secure key	Required	Form
file	"afile.txt"	file to be uploaded into the session	Required	Form
userData[file-description]	"March invoice"	This value can be used by the agent desktop and the client UI to provide additional file info	Optional	Form
userData[userKey1]	"User value 1"	Collection of key-value pairs that provides file metadata	Optional	Form

Example

```
http://localhost:8080/genesys/2/chat-ntf

operation=fileUpload
```

```
secureKey=8b31761b2154884c
file=<some file to upload>
```

Output

```
{
  "chatEnded": false,
  "statusCode": 0,
  "secureKey": "77cd1c487b67fefb",
  "userData": {
    "file-id": "00F057DB0FF10005"
  }
}
```

Download File

Use this request to download a file that was uploaded into a chat session either by an agent or a customer. The `fileId` parameter can be retrieved from the previous transcript event (see the response for Refresh Chat request) indicating that a file was upload.

HTTP Header	Value
Content-Type	• multipart/form-data • application/x-www-form-urlencoded

Parameter Name	Sample Value	Description	Required/ Optional	Source
operation	fileDownload		Required	Form
secureKey	"8b31761b2154884c"	"secure key	Required	Form
fileId	"00048aAPEQJ8ABCD"	File to be downloaded from the session	Required	URL

Example

```
http://localhost:8080/genesys/2/chat-ntf

operation=fileDownload
secureKey=8b31761b2154884c
fileId=00048aAPEQJ8ABCD
```

Output

If the operation is successful, the contents of the file are downloaded. In case of an error, the server returns an HTML 500 response that includes the **referenceId** of the request.

The following HTML snippet shows how to download a file using an HTML form:

```
var form = $(
"<form method='POST' enctype='multipart/form-data' target='_blank'
action='http://GMS_HOST:GMS_PORT/genesys/2/chat-ntf'>
 <input name='secureKey' value='6b46a7a8910f21be' type='hidden'>
<input name='fileId' value='00F057C8B6B7004D' type='hidden'>
 <input name='operation' value='download' type='hidden'>/form>");

form.submit();
```

Delete File

Use this request to delete a file that was uploaded into a chat session by a customer if allowed by Chat Server settings. Web User has no permission to delete file uploaded by the agent.

HTTP Header	Value
Content-Type	• multipart/form-data • application/x-www-form-urlencoded

Parameter Name	Sample Value	Description	Required/ Optional	Source
operation	"fileDelete"	user ID	Required	Form
secureKey	"8b31761b2154884c"	secure key	Required	Form
fileId	"00048aAPEQJ8ABCD"	The file to be deleted.	Required	URL

Example

```
POST /genesys/2/chat-ntf

operation=fileDelete
fileId=007553863DC30029
secureKey=8b31761b2154884c
```

Output

```
{
  "chatEnded": false,
  "statusCode": 0,
  "secureKey": "ad437d2338d09d09"
}
```

Configure your Load-Balancer/Proxy Server for CometD

This section provides some load-balancer/proxy configuration examples that show how to make a load-balancer proxy work with GMS WebSockets.

Nginx Configuration Sample

To configure NGINX, edit the NGINX configuration file available in the <NGINX INSTALLATION DIR>/conf directory and add the below configuration.

```
http {
  map $http_upgrade $connection_upgrade {
          default upgrade;
          '' close;
      }
upstream nginixloadbalancer {
    server <your_domain_name>:<your_port>;
  }
server {
    listen 80;
    server_name  localhost;
    location / {
      proxy_pass http://nginixloadbalancer/;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection $connection_upgrade;
      proxy_set_header Host $host;
    }
  }
}
```

> **Important**
>
> Make sure to replace the <your_domain_name>:<your_port> text placeholder with the relevant domain and port. Use, for example, `gms.network.local:8080`.

Apache Configuration Sample

If your application uses Apache, edit the `httpd.conf` file (or alternate file).

1. Uncomment the following modules in the `httpd.conf` file.

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_wstunnel_module modules/mod_proxy_wstunnel.so
LoadModule rewrite_module modules/mod_rewrite.so
```

2. Add the below configuration at the end of the `httpd.conf` file.

```
<VirtualHost *:80>    ProxyRequests on  RequestHeader set X-Forwarded-Proto "http"
ProxyPass / http://{your domain name Eg: gms.network.local:8080}/
  RewriteEngine on
  RewriteCond %{HTTP:Upgrade} websocket [NC]
```

```
  RewriteCond %{HTTP:Connection} upgrade [NC]
  RewriteRule ^/?(.*) "ws://<your_domain_name>:<your_port>/$1" [P,L]
</VirtualHost>
```

> **Important**
>
> Make sure to replace the <your_domain_name>:<your_port> text placeholder with the relevant domain and port. Use, for example, gms.network.local:8080.

HA Proxy Configuration Sample

Edit the etc/haproxy/haproxy.cfg file and add some special docker directives:

```
global
    maxconn 2000
    pidfile /var/run/haproxy.pid
    log     127.0.0.1 local0
    log     127.0.0.1 local1 notice    # echo "" | nc -U /var/run/haproxy.sock
    stats   socket /usr/local/sbin/haproxy.sock mode 777resolvers dockerdns
  nameserver dockerowned 127.0.0.11:53
  timeout retry   1s
  resolve_retries 10
  hold valid 1sdefaults
  mode http
  log global
  option httplog
  option  http-server-close
  option  dontlognull
  option  redispatch
  option  contstats
  retries 3
  backlog 10000
  timeout client          25s
  timeout connect          5s
  timeout server          25s
# timeout tunnel available in ALOHA 5.5 or HAProxy 1.5-dev10 and higher
  timeout tunnel        3600s
  timeout http-keep-alive  1s
  timeout http-request    15s
  timeout queue           30s
  timeout tarpit          60s
  default-server inter 3s rise 2 fall 3
  option forwardforfrontend main
  bind :80
  maxconn 10000
  default_backend bk_gms  backend bk_gms
  server server1 <your_domain_name>:<your_port> cookie server1 resolvers dockerdns
```

> **Important**
>
> Make sure to replace the <your_domain_name>:<your_port> text placeholder with the relevant domain and port. Use, for example, gms.network.local:8080.