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.

Digital Channels Developer's Guide

9/11/2025

Table of Contents

Contents

Find information about how to develop using the Digital Channels APIs.

Related documentation:

-

Digital Channels provides APIs you can use to build custom chat applications and deliver messages between Digital Channels and third-party services such as SMS or email.

Developer tasks

Explore these topics for examples of how to use the Digital Channels APIs.

- Genesys Widgets: Close the chat when an agent disconnects
- Persistent Chat across Different Domains

Digital Channels APIs

Learn more about the Digital Channels APIs on the Genesys Multicloud CX Developer Center.

- Consumer Messaging API
- Third-Party Messaging API

Genesys Widgets: Close the chat when an agent disconnects

Contents

Learn how to finalize asynchronous chat interactions so that customers can no longer type in the Genesys Widget after an agent marks a chat as done.

Related documentation:

-

Digital Channels supports an asynchronous chat model that lets your customers join or rejoin a chat session at their convenience. This divides a chat session into multiple segments, which are represented by separate interactions that agents can handle within your contact center. During these interactions, an agent connects to a chat session and conducts the conversation with a customer. Each interaction can be ended only from the agent side or, for chat bot conversations, a workflow. The interaction ends when the agent sends a reply to the customer that isn't likely to generate an immediate response, or when the customer goes offline (disconnects from chat or doesn't reply in a timely manner). The next interaction in the chat is started by either a new message from the customer or when an agent requests the creation of a new interaction to follow up with customer. From the customer's perspective, these interactions are one large seamless and ongoing conversation.

There are some edge cases when a customer might initiate a new segment of conversation by sending a follow up that isn't actionable for the agent. For example, after being offline the customer comes back and sends a thank you message after the agent has already closed the segment. This follow up from the customer creates a new segment (interaction) to be routed to an agent.

The sections below describe a solution you can implement to modify the Chat Widget behavior to prevent these non-actionable messages, by either notifying the customer or closing the chat window.

Note: It's possible that only one interaction is expected for the whole chat session, which resembles non-asynchronous chat behavior. In this case, you must also adjust the Chat Widget's behavior accordingly.

How to implement the solution

To handle this scenario in your Chat Widget, first set up the widget to respond to state changes from Digital Channels using the Consumer Messaging API.

When Digital Channels detects a state change, it adds a System message to the chat session history and returns it in the response of a long poll or history GET /chat/sessions/{sessionId}/messages request.

For example:

```
{
  "status": {
    "code": "SUCCESS"
  },
  "data": {
```

```
    "messages": [
      {
        "type": "System",
        "from": {
          "participantId": "00000000-0000-0000-0000-000000000000",
          "type": "System"
        },
        "visibility": "All",
        "id": "dba9117a-38d4-4c06-ad67-665409093314",
        "event": "InteractionStateChange",
        "data": {
          "interactionState": "Finalizing"
        },
        "utcTime": "2021-07-09T10:25:38.829Z",
        "streamId": "1625826338845-0",
        "index": "1625826338845-0"
      }
    ],
    "lastDeliveredIndex": "1625826338845-0",
    "chatEnded": false
  },
  "operationId": "69be0dc50e-de8a9-093ca-52684-51ce7aa716db743",
  "referenceId": "1d9c9d1e-e518-446c-9889-57fadc98fd16"
}
```


The widget can capture this message by using the Widgets Bus API to subscribe to the WebChatService.messageReceived event. The way you process this event depends on whether the chat is asynchronous. See the table below for details:

State	Notify customer	Close the chat
Finalizing OR Completed	Inject a message into the widget that says the chat interaction is over. This leaves the option for the customer to start a new interaction. Example message: "Thank you for your query. Please do not respond to this automated message if you do not have additional questions."	End the chat on the widget side. This removes the participant in Digital Channels.

Deploying the widget example

The code below shows an example of how you could handle the event change. If the chat is asynchronous, it posts a message to the customer when the agent closes the segment; otherwise, it closes the chat window when the agent closes the segment.

The sample uses the global CXBus instance, but you can also use other methods of accessing Widgets Bus.

```
// Widgets config and initialization code is omitted for clarity
// ...
CXBus.loadPlugin('widgets-core');
```

```
CXBus.subscribe('WebChatService.messageReceived', (event) => {
  // taking the latest interaction state change event when fetching the history, for long-
poll there will be only one
  const lastInteractionStateChange = event.data.originalMessages
    .filter(message => message.type === 'System' && message.event ===
'InteractionStateChange')
    .pop();

  // nothing to do if no event found
  if (!lastInteractionStateChange) return;

  // digging for current interaction state
  const interactionState = lastInteractionStateChange.data.interactionState;
  if (!interactionState) return;

  // processing the state
  if (interactionState === 'Finalizing' || interactionState === 'Completed') {
    // different handling for async and regular chat
    const isAsync = window._genesys.widgets.webchat.transport.async &&
window._genesys.widgets.webchat.transport.async.enabled;
    if (isAsync) {
      CXBus.command('WebChat.injectMessage', {
        type: 'text',
        text: 'Thank you for your query. Please do not respond to this automated message if
you do not have additional questions.',
        bubble: { time: false }
      });
    } else {
      CXBus.command('WebChat.endChat');
    }
  }
});
```

Persistent Chat across Different Domains

Contents

Learn how to make chat conversations persistent across different domains.

Related documentation:

-

> **Important**
>
> - Mozilla Firefox browsers do not support this functionality since it requires Cookie Partitioning.
> - Starting from 2024, Chromium has announced its end of support for Cookie Partitioning for both Chrome and Edge browsers. If you're using this feature with the aforementioned browsers, Genesys recommends you update your integration to avoid usage of third-party cookies.

Digital channels supports chat persistence across different domains, allowing your customers to continue chat conversations with agents across the organization's sites they visit. It also allows your organization to have a consistent group of agents to support customers visiting different domains.

The following sections describe how to implement a cross domain persistent chat.

Prerequisites

Before setting up persistent chat across different domains, ensure you configure the WebChatService.

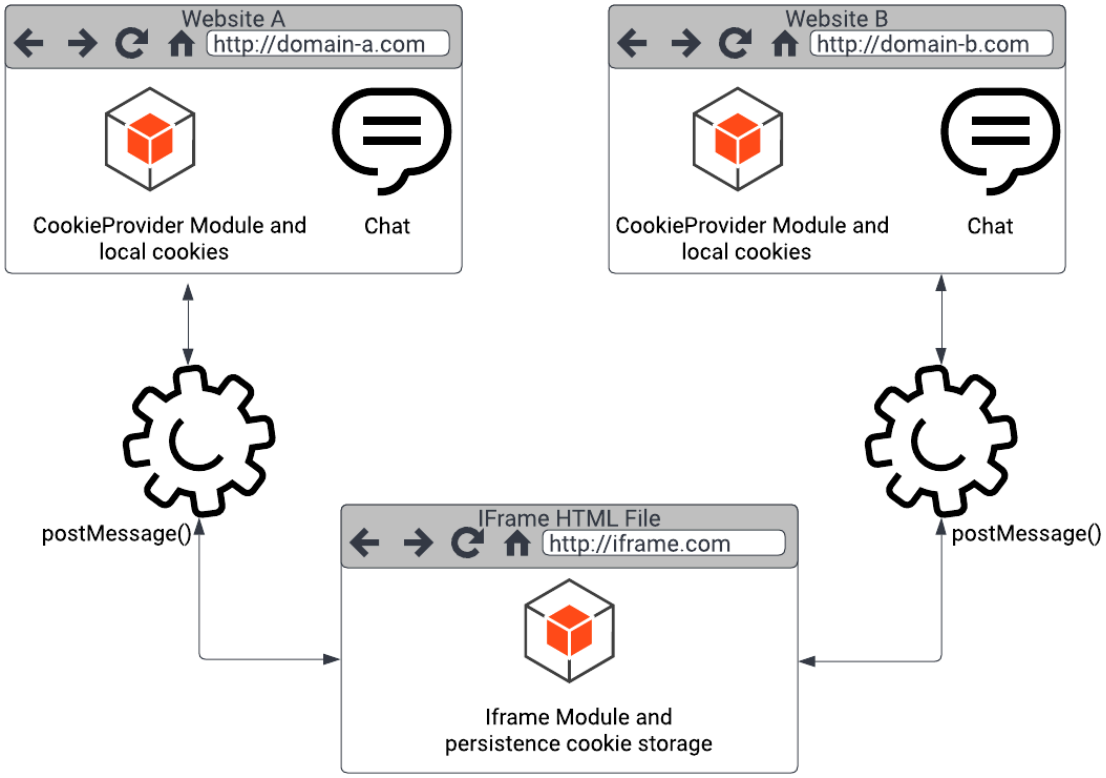
Third-party Cookie Notification

Ensure that the third-party cookies are enabled on your customer's browser and you can suggest them to enable cookies with your custom messages on your site or in the Widget.

Overview of the solution

Suppose that a customer started a chat on domain-a.com and intends to continue on domain-b.com, Digital Channels persists the chat conversation, and the customer can continue the interaction with the agent across the two domains.

The solution combines a third-party JavaScript file that you include on the webpages with a hidden Iframe added to the webpages. The module uses a **window.postMessage()** function of HTML5 to establish connections among the webpages and the Iframe site. The function sets the cookies and saves them on the Iframe site.



To implement this solution, complete the configuration steps.

Step 1: Create and host an Iframe HTML file

Create a plain HTML page with no server-side dependencies and host this file on a webpage you own. Include the attached **iframe.min.js** file and use the following sample iframe.html file. Modify the **TRUSTED_DOMAINS** array with the list of websites you would like to implement the chat persistence with agent.

iframe.min.js

For example:

After hosting the HTML file, note down the direct URL of the site such as

https://www.iframe.com/iframe.html. This URL is used in Step 3.

Step 2: Include the Cookie Provider JavaScript file

Include the attached **cookie-provider.min.js** across all the webpages (in our example, it was domain-a.com and domain-b.com) where your Genesys chat widget is started.

cookie-provider.min.js

Step 3: Create an Instance of Cookie Provider or Add Cookie Provider Extension in Widget

To start chat persistence across different websites, you can either

Create an Instance of Cookie Provider or

Add Cookie Provider Extension in Widget

Create an Instance of Cookie Provider

Create an instance of CookieProvider in the script where you configure your chat widget. While creating an instance, the CookieProvider adds Iframe to your webpages. The following table describes the parameters of the CookieProvider class.

Parameter Name	Mandatory	Description
SESSION_COOKIE_NAME	Yes	Add your own session cookie name.
IFRAME_URL	Yes	Add the Iframe URL obtained from Step 1.
showThirdPartyCookieNotification	No	Add this parameter if you want to show a message that third-party cookies are disabled in the customer's browser. For more information about third-party cookie notifications, see Third-party Cookie Notification.

```
// provide your own SESSION_COOKIE_NAME (example 'customer-defined-session-cookie')  and
IFRAME_URL (example 'https://iframe.com/iframe.html')
// showThirdPartyCookieNotification is an optional parameter to show message if third party
cookies are disabled
let cookieProvider = new CookieProvider(SESSION_COOKIE_NAME, IFRAME_URL,
```

```
showThirdPartyCookieNotification);
//load cookie
cookieProvider.loadCookie();

// chat initializing
```

After creating the instance, you can call the **loadCookie** method to receive your newly created session cookie.

Save Cookie

To use the cookie across various websites, you must save the created cookie into the Iframe. To do this, call the **saveCookie** method after your chat session cookie is set. In the following example, the registered plugin is subscribed to the **WebChat.started** event, to save the cookie inside the callback function.

```
var cookieProviderPlugin = window._genesys.widgets.bus.registerPlugin("CookieProvider");
cookieProviderPlugin.subscribe("WebChat.started", function (e) {
    // call saveCookie when you want to save chat session cookie
    cookieProvider.saveCookie();
});
```

Third-party Cookie Notification

You can check if third-party cookies are unavailable on your customer's browser and suggest them to enable cookies with your custom messages.

> **Important**
>
> Mozilla Firefox browsers do not support this functionality.

```
// example of function to show if third party cookies are enabled
function showThirdPartyCookieNotification(isEnabled) {
    // you can show here your own notification to enable third-party cookies
    alert("Third-party cookies are " + isEnabled ? "enabled" : "disabled");
}

// you can call cookieProvider.cookieTest with callback
cookieProvider.cookieTest(showThirdPartyCookieNotification);
// or without parameter if you provide callback when created instance of CookieProvider
cookieProvider.cookieTest();
```

Add Cookie Provider Extension in Widget

The following option is an alternate solution to enable chat persistence across

different websites. In this approach, instead of creating an instance of CookieProvider, add the CookieProviderExtension in the extensions field of your Widget's configuration. As a part of an extension, you can call the callback function in initCookieProviderExtension to check whether third-party cookies are enabled in your customers' browser.

```
// add it in you init configuration
if (!window._genesys) window._genesys = {};

// EXAMPLE of options value
const IFRAME_URL = "http://domain-a/iframe.html";
const SESSION_COOKIE_NAME = "customer-defined-session-cookie";

// EXAMPLE of function to show notification for third-party cookies
function showThirdPartyCookieNotification(isEnable) {
  if (!isEnable) {
    alert("please enable your third-party cookies");
  }
}

window._genesys = {
    widgets: {
        extensions: {
            CookieProviderExtension: initCookieProviderExtension(SESSION_COOKIE_NAME,
IFRAME_URL, showThirdPartyCookieNotification)
       }
    }
}

// or add it separately after initialization
if(!window._genesys.widgets.extensions){
    window._genesys.widgets.extensions = {};
}

window._genesys.widgets.extensions["CookieProviderExtension"] =
initCookieProviderExtension(SESSION_COOKIE_NAME, IFRAME_URL,
showThirdPartyCookieNotification);
```

Complete Chat Initialization Sample

The following code sample shows the chat configuration with chat persistence across different websites.

```
const IFRAME_URL = "http://127.0.0.1:5500/iframe.html";
const SESSION_COOKIE_NAME = "customer-defined-session-cookie";

if (!window._genesys) window._genesys = {};

window._genesys = {
  widgets: {
    webchat: {
      transport: {
        type: "",
        dataURL: "", // Provided by Genesys
        endpoint: "", // Provided by Genesys
        headers: {
          "x-api-key": "", // Provided by Genesys
        },
        async: {
```

```
          enabled: true,
          getSessionData: function (sessionData, Cookie, CookieOptions) {
            // Note: You don't have to use Cookies. You can, instead, store in a secured
location like a database.
            Cookie.set(
              SESSION_COOKIE_NAME,
              JSON.stringify(sessionData),
              CookieOptions
            );
          },
          setSessionData: function (Open, Cookie, CookieOptions) {
            // Retrieve from your secured location.
            return Cookie.get(SESSION_COOKIE_NAME);
          },
        },
      },
    },
    // you need add this option only if you use alternative connection
    extensions: {
      CookieProviderExtension
    }
  },
};

const widgetScriptElement = document.createElement("script");
const widgetBaseUrl = "https://apps.mypurecloud.de/widgets/9.0/";


// provide your own SESSION_COOKIE_NAME (example 'customer-defined-session-cookie')  and
IFRAME_URL (example 'https://iframe.com/iframe.html')
let cookieProvider = new CookieProvider(SESSION_COOKIE_NAME, IFRAME_URL);
//load cookie
cookieProvider.loadCookie();


// chat initializing
widgetScriptElement.setAttribute("src", widgetBaseUrl + "cxbus.min.js");
widgetScriptElement.addEventListener("load", async function () {
  await CXBus.configure({
    debug: true,
    pluginsPath: widgetBaseUrl + "plugins/",
  });
  await CXBus.loadPlugin("widgets-core");
  await CXBus.command("WebChat.open");

  // set cookie to iframe when chat started
   var cookieProviderPlugin = window._genesys.widgets.bus.registerPlugin("CookieProvider");
   cookieProviderPlugin .subscribe("WebChat.started", function (e) {
    //save cookie
    cookieProvider.saveCookie();
  });
});

document.head.appendChild(widgetScriptElement);
```

Limitations

Chat persistence does not work when a browser is configured to disable third-party cookies. Each browser manages and defines third-party cookies and data differently.

Mozilla Firefox and Apple Safari browsers block the third-party cookies and data by default, but they allow you to set cookies manually. Google Chrome will stop supporting third-party cookies during 2024.

The following documents provide guidelines and settings for third-party cookies across different browsers.

Browser	URL
Google Chrome	https://support.google.com/chrome/answer/95647
Mozilla Firefox	https://support.mozilla.org/en-US/kb/third-party-cookies-firefox-tracking-protection
Apple Safari	https://support.apple.com/en-gb/guide/safari/sfri11471/mac
Opera	https://blogs.opera.com/news/2015/08/how-to-manage-cookies-in-opera/