Platform

Custom Embedded Tabs

Applies to: All clients

Custom embedded tabs display HTML content in the client interface. You can create custom embedded tab definitions for Cisco Jabber.

Note

- The Jabber embedded browser doesn't support cookie sharing with pop-ups from SSO-enabled web pages. The content on the pop-up window may fail to load.
- Configure the HTTP server allow list (whitelist) for any web services inside the enterprise that remote Jabber clients need to access. See the *Mobile and Remote Access Through Cisco Expressway Deployment Guide* for more information.

Custom Embedded Tab Definitions

You can configure a custom embedded tab using the `jabber-config.xml` file. The following XML snippet shows the structure for custom tab definitions:

```
<jabber-plugin-config>
 <browser-plugin>
  <page refresh="" preload="" internal="">
   <tooltip></tooltip>
   <icon></icon>
   <url></url>
  </page>
 </browser-plugin>
</jabber-plugin-config>
```

Cisco Jabber for Windows uses the Chromium Embedded Framework to display the content on the custom embedded tabs.

Cisco Jabber for Mac uses the Safari WebKit rendering engine to display the content of the embedded tab.

The following table describes the parameters for custom embedded tab definitions:

Parameter	Description
browser-plugin	Contains all definitions for custom embedded tabs. The value includes all custom tab definitions.
page	Contains one custom embedded tab definition.
refresh	Controls when the content refreshes. • true—Content refreshes each time users select the tab. • false (default)—Content refreshes when users restart the client or sign in. This parameter is optional and is an attribute of the page element.
preload	Controls when the content loads. • true—Content loads when the client starts. • false (default)—Content loads when users select the tab. This parameter is optional and is an attribute of the page element.
tooltip	Defines hover text for the custom embedded tab. This parameter is optional. If you don't specify the hover text, the client uses *Custom tab*. The value is string of unicode characters.
icon	Specifies an icon for the tab. You can specify a local or hosted icon as follows: • Local icon—Specify the URL as follows: `file://`*`file_path`*`/`*`icon_name`* • Hosted icon—Specify the URL as follows: `http://`*`path`*`/`*`icon_name`* You can use any icon that the client browser can render, including .JPG, .PNG, and .GIF formats. This parameter is optional. If you don't specify an icon, the client loads the favicon from the HTML page. If no favicon is available, the client loads the default icon.

Parameter	Description
url	Specifies the URL where the content for the embedded tab resides. The client uses the browser rendering engine to display the content of the embedded tab. For this reason, you can specify any content that the browser supports. For Cisco Jabber for Mac, the URL element must contain HTTP or HTTPS. This parameter is required. **Note** If the target web page requires Windows integrated authentication, Jabber prompts the user for sign-in credentials by default. To avoid the prompt, you can configure the authentication server whitelist in Windows Registry. Add the whitelisted URLs to these locations: • `HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Google\Chrome\AuthServerWhitelist` • `HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Google\Chrome\AuthNegotiateDelegateWhitelist` For example, suppose you set the `AuthServerWhitelist` and `AuthNegotiateDelegateWhitelist` policies to `*example.com,*foobar.com,*baz`. Any URLs ending in either 'example.com', 'foobar.com', or 'baz' are in the permitted list. Without the '*' prefix, the URL has to match exactly.
internal	Specifies if your web page is an internal or an external page to your network. • true (default)—Your web page is an internal page to your network. • false—Your web page is an external page to your network.

User Custom Tabs

You can allow users to specify a tab name and URL for a custom embedded tab through the client user interface. Users cannot set the other parameters for a custom embedded tab.

Set AllowUserCustomTabs to **`true`** before users can customize their tabs:

```
<Options>
  <AllowUserCustomTabs>true</AllowUserCustomTabs>
</Options>
```

Note The default value for AllowUserCustomTabs is **`true`**.

Custom Icons

To achieve optimal results, your custom icon should conform to the following guidelines:

- Dimensions: 20 x 20 pixels
- Transparent background

- PNG file format

Chats and Calls from Custom Tabs

You can use protocol handlers to start chats and calls from custom embedded tabs. Make sure the custom embedded tab is an HTML page.

Use the `XMPP:` or `IM:` protocol handler to start chats.
Use the `TEL:` protocol handler to start audio and video calls.

UserID Tokens

You can specify the `${UserID}` token as part of the value for the url parameter. When users sign in, the client replaces the `${UserID}` token with the username of the logged in user.

Tip You can also specify the `${UserID}` token in query strings; for example, `www.cisco.com/mywebapp.op?url=${UserID}`.

The following is an example of how you can use the `${UserID}` token:

1. You specify the following in your custom embedded tab:
   ```
   <url>www.cisco.com/${UserID}/profile</url>
   ```
2. Mary Smith signs in. Her username is msmith.
3. The client replaces the `${UserID}` token with Mary's username as follows:
   ```
   <url>www.cisco.com/msmith/profile</url>
   ```

JavaScript Notifications

You can implement JavaScript notifications in custom embedded tabs. This topic describes the methods the client provides for JavaScript notifications. This topic also gives you an example JavaScript form that you can use to test notifications. It's beyond the scope of this documentation to describe how to implement JavaScript notifications for asynchronous server calls and other custom implementations. Refer to the appropriate JavaScript documentation for more information.

Notification Methods

The client includes an interface that exposes the following methods for JavaScript notifications:

- SetNotificationBadge—You call this method from the client in your JavaScript. This method takes a string value that can have any of the following values:
  - Empty—An empty value removes any existing notification badge.
  - A number 1–999
  - Two-digit alphanumeric combinations, for example, A1
- onPageSelected()—The client invokes this method when users select the custom embedded tab.

- onPageDeselected()—The client invokes this method when users select another tab.

Note Not applicable for Jabber for iPhone and iPad

- onHubResized()—The client invokes this method when users resize or move the client hub window.
- onHubActivated()—The client invokes this method when the client hub window activates.
- onHubDeActivated()—The client invokes this method when the client hub window deactivates.

Subscribe to Presence in Custom Tabs

You can use the following JavaScript methods to subscribe to the presence of a contact and receive presence updates from the client:

- SubscribePresence()—Specify a string value using the IM address of a user for this method.
- OnPresenceStateChanged—This method enables users to receive updates from the client on the presence of a contact. You can specify one of the following values as the string:
  - IM address
  - Basic presence (Available, Away, Offline, Do Not Disturb)
  - Rich presence (In a meeting, On a call, or a custom presence state)

Note

- Subscriptions for people not on your contact list expire after 68 minutes. After the subscription expires, you must resubscribe to see their presence data.
- Jabber for iPad and iPhone only supports OnPresenceStateChanged.

Get Locale Information in Custom Tabs

You can use the following JavaScript methods to retrieve the current locale information of a contact from the client:

- GetUserLocale()—This method enables users to request locale information from the client.
- OnLocaleInfoAvailable—This method enables users to receive locale information from client. You can use a string value that contains the client locale information.

Note Jabber for iPad and iPhone only supports OnLocaleInfoAvailable.

Example JavaScript

This example shows an HTML page that uses JavaScript to display a form into which you can input a number 1–999:

```
<html>
        <head>
                <script type="text/javascript">
                                function OnPresenceStateChanged(jid, basicPresence,
localizedPresence)
                                {
                                         var cell = document.getElementById(jid);
                                         cell.innerText = basicPresence.concat(",
",localizedPresence);
                                }

                                function GetUserLocale()
                                {
                                         window.external.GetUserLocale();
                                }

                                function SubscribePresence()
                                {

window.external.SubscribePresence('johndoe@example.com');
                                }

                                function OnLocaleInfoAvailable(currentLocale)
                                {
                                      var cell = document.getElementById("JabberLocale");

                                         cell.innerText = currentLocale;
                                }

                                function onHubActivated()
                                {
                                         var cell = document.getElementById("hubActive");
                                         cell.innerText = "TRUE";
                                }

                                function onHubDeActivated()
                                {
                                         var cell = document.getElementById("hubActive");
                                         cell.innerText = "FALSE";
                                }

                                function onHubResized()
                                {
                                         alert("Hub Resized or Moved");
                                }

                                function OnLoadMethods()
                                {
                                         SubscribePresence();
                                         GetUserLocale();
                                }
                     </script>
               </head>

               <body onload="OnLoadMethods()">
                     <table>
                                <tr>
                                            <td>John Doe</td>
                                            <td id="johndoe@example.com">unknown</td>
                                </tr>
                    </table>
                    <table>
                                <tr>
                                            <td>Jabber Locale: </td>
```

```
                                                <td id="JabberLocale">Null</td>
                                    </tr>
                                    <tr>
                                                <td>Hub Activated: </td>
                                                <td id="hubActive">---</td>
                                    </tr>
                        </table>
                </body>

</html>
```

To test this example JavaScript form, copy the preceding example into an HTML page and then specify that page as a custom embedded tab.

Show Call Events in Custom Tabs

You can use the following JavaScript function to show call events in a custom tab:

OnTelephonyConversationStateChanged — An API in the telephony service enables the client to show call events in a custom embedded tab. Custom tabs can implement the `OnTelephonyConversationStateChanged` JavaScript function. The client calls this function every time a telephony conversation state changes. The function accepts a JSON string that the client parses to get call events.

The following snippet shows the JSON that holds the call events:

```
{
      "conversationId": string,
      "acceptanceState": "Pending" | "Accepted| | "Rejected",
      "state": "Started" | "Ending" | "Ended",
      "callType": "Missed" | "Placed" | "Received" | "Passive" | "Unknown",
      "remoteParticipants": [{participant1}, {participant2}, …, {participantN}],
      "localParticipant": {
      }
}
```

Each participant object in the JSON can have the following properties:

```
{
      "voiceMediaDisplayName": "<displayName>",
      "voiceMediaNumber": "<phoneNumber>",
      "translatedNumber": "<phoneNumber>",
      "voiceMediaPhoneType": "Business" | "Home" | "Mobile" | "Other" | "Unknown",
      "voiceMediaState": "Active" | "Inactive" | "Pending" | "Passive" | "Unknown",
}
```

The following is an example implementation of this function in a custom embedded tab. This example gets the values for the `state` and `acceptanceState` properties and shows them in the custom tab.

```
function OnTelephonyConversationStateChanged(json) {
      console.log("OnTelephonyConversationStateChanged");
      try {
        var conversation = JSON.parse(json);
        console.log("conversation id=" + conversation.conversationId);
        console.log("conversation state=" + conversation.state);
        console.log("conversation acceptanceState=" + conversation.acceptanceState);
        console.log("conversation callType=" + conversation.callType);
      }
      catch(e) {
        console.log("cannot parse conversation:" + e.message);
      }
    }
```

The following is an example implementation of this function with all possible fields:

```
function OnTelephonyConversationStateChanged(json) {
      console.log("OnTelephonyConversationStateChanged");
      try {
        var conversation = JSON.parse(json);
        console.log("conversation state=" + conversation.state);
        console.log("conversation acceptanceState=" + conversation.acceptanceState);
        console.log("conversation callType=" + conversation.callType);
        for (var i=0; i<conversation.remoteParticipants.length; i++) {
          console.log("conversation remoteParticipants[" + i + "]=");
          console.log("voiceMediaDisplayName=" +
conversation.remoteParticipants[i].voiceMediaDisplayName);
          console.log("voiceMediaNumber=" +
conversation.remoteParticipants[i].voiceMediaNumber);
          console.log("translatedNumber=" +
conversation.remoteParticipants[i].translatedNumber);
          console.log("voiceMediaPhoneType=" +
conversation.remoteParticipants[i].voiceMediaPhoneType);
          console.log("voiceMediaState=" +
conversation.remoteParticipants[i].voiceMediaState);
        }
        console.log("conversation localParticipant=");
        console.log("  voiceMediaDisplayName=" +
conversation.localParticipant.voiceMediaDisplayName);
       console.log("  voiceMediaNumber=" + conversation.localParticipant.voiceMediaNumber);

       console.log("  translatedNumber=" + conversation.localParticipant.translatedNumber);

        console.log("  voiceMediaPhoneType=" +
conversation.localParticipant.voiceMediaPhoneType);
        console.log("  voiceMediaState=" + conversation.localParticipant.voiceMediaState);
      }
      catch(e) {
        console.log("cannot parse conversation:" + e.message);
      }
    }
```

Custom Embedded Tab Example

The following is an example of a configuration file with one embedded tab:

```
<?xml version="1.0" encoding="utf-8"?>
<config version="1.0">
 <Client>
  <jabber-plugin-config>
   <browser-plugin>
     <page refresh ="true" preload="true">
     <tooltip>Cisco</tooltip>
     <icon>https://www.cisco.com/web/fw/i/logo.gif</icon>
     <url>https://www.cisco.com</url>
    </page>
   </browser-plugin>
  </jabber-plugin-config>
 </Client>
</config>
```

Configure Cisco Jabber for Android on Chromebook

Checklist to Configure Cisco Jabber for Android on Chromebook

Task	Details
See the device models and OS supported	See *Android OS Requirement and Chromebook Models Supported*
Add MRA configuration in corporate network	see *Add MRA Configuration in Corporate Network*
Configure the Chromebook user as a TAB device type	see *Configure Chromebook Users as TAB Device Type*
Keep the required ports open so that your users access all Cisco Jabber services on Chromebook	see *Configure Ports*

Android OS Requirement and Chromebook Models Supported

Chromebook must have Chrome OS version 53 or later. Users can download Cisco Jabber for Android from Google Play Store.

The chromebook models supported:

- HP Chromebook 13 G1 Notebook PC
- Google Chromebook Pixel
- Samsung Chromebook Pro

Add MRA Configuration in Corporate Network

Use Cisco Jabber on Chromebook while connected from your corporate and Mobile and Remote Access (MRA) network. To use call services, Cisco Jabber must be signed in using MRA Network.

To connect to MRA network when your users are operating within the corporate network, configure your internal Domain Name Server (DNS) with the "`_collab-edge._tls.<domain>.com`" SRV record. For complete details on DNS, see the section *Service Discovery* from the *Cisco Jabber Planning Guide 12.1*.

Configure Chromebook Users as TAB Device Type

You can configure Chromebook users as TAB device type. For complete details on how to configure softphone service for a user, see the section *Configure Softphone* from the *Cisco Jabber On-premises Guide 12.1*.

Configure Ports

Make sure these ports are open to access Cisco Jabber services on Chromebook:

Purpose	Protocol	On-premises Network (Source)	Expressway-E (Destination)
XMPP(IM&P)	TCP	>=1024	5222
HTTP proxy(UDS)	TCP	>=1024	8443

Purpose	Protocol	On-premises Network (Source)	Expressway-E (Destination)
Media	UDP	>=1024	36002 to 59999
SIP signaling	TLS	>=1024	5061

Limitations

During a video call, the video stops if the users switch to another app.

Cisco Jabber Mobile App Promotion

Applies to: Cisco Jabber for Windows and Cisco Jabber Softphone for VDI

You can enable a notification for Cisco Jabber for Windows users to promote the use of the Cisco Jabber for Mobile App (Android and iOS). Clicking the notification takes the user to the **Settings** page where they can choose to download the app from Google Play or the iTunes Store. A new parameter EnablePromoteMobile is added to control these notifications. This feature is disabled by default.

For more information on configuring this parameter, See the *Parameter Reference Guide for Cisco Jabber*.