# 平台



## 自定义嵌入式选项卡

| 客户端 | | | |
|---|---|---|---|
| **Windows** | **Mac** | **iPhone 和 iPad** | **Android** |
| 是 | 是 | 是 | 是 |

| 部署 | | | |
|---|---|---|---|
| **场内** | **Webex Messenger** | **组消息模式** | **VDI 软终端** |
| 是 | 是 | 是 | 是 |

自定义嵌入式选项卡在客户端界面中显示 HTML 内容。您可以为 Cisco Jabber 创建自定义嵌入式选项卡定义。现在，您可以通过编程方式将自定义选项卡调整为使用当前的客户端主题。

**注释**

- Jabber 嵌入式浏览器不支持通过启用 SSO 的网页与弹出窗口共享 cookie。弹出窗口中的内容可能无法加载。
- 为远程 Jabber 客户端需要访问的任何企业内部 Web 服务配置 HTTP 服务器允许列表（白名单）。有关详细信息，请参阅《*Mobile and Remote Access Through Cisco Expressway* 部署指南》。

# 自定义嵌入式选项卡定义

您可以使用 jabber-config.xml 文件配置自定义嵌入式选项卡。以下 XML 代码段显示自定义选项卡定义的结构：

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

Cisco Jabber Windows 版本使用 Chromium 嵌入式框架在自定义嵌入式选项卡上显示内容。

Cisco Jabber Mac 版本使用 Safari WebKit 渲染引擎来显示嵌入式选项卡的内容。

下表说明了用于自定义嵌入式选项卡定义的参数：

| 参数 | 说明 |
|---|---|
| browser-plugin | 包含自定义嵌入式选项卡的所有定义。<br>该值包括所有自定义选项卡定义。 |
| page | 包含一个自定义嵌入式选项卡定义。 |
| refresh | 控制何时刷新内容。<br>• true—每当用户选择该选项卡时刷新内容。<br>• false（默认值）—在用户重新启动客户端或登录时刷新内容。<br>此参数是可选的，并且是页面元素的属性。 |
| preload | 控制何时加载内容。<br>• true—在客户端启动时加载内容。<br>• false（默认值）—在用户选择该选项卡时加载内容。<br>此参数是可选的，并且是页面元素的属性。 |
| tooltip | 定义自定义嵌入式选项卡的悬停文本。<br>此参数是可选的。如果不指定悬停文本，客户端将使用自定义选项卡。<br>该值为 Unicode 字符串。 |

| 参数 | 说明 |
| --- | --- |
| 图标 | 指定选项卡的图标。您可以按如下方式指定本地或托管图标：<br>• 本地图标 — 指定 URL 如下：`file://file_path/icon_name`<br>• 托管图标 — 指定 URL 如下：`http://path/icon_name`<br>您还可以使用 *%JabberTheme%* 变量更改本地图标和托管图标，以匹配当前的Jabber主题，如下所示：`http://path/icon_name_%JabberTheme%.jpg`<br>当请求图标时，Jabber 客户端会解释 *%JabberTheme%*。可能的值包括：<br>• 默认值—默认的 Jabber 主题<br>• 深色—Jabber“深色”主题<br>• 独特—Jabber“高对比度”主题<br>• 高对比度—Windows 高对比度主题<br>如果 URL 中不包含 *%JabberTheme%*，则图标不会随主题一起更改。在文件名中包含每个图标的主题名称。如果下载自定义图标失败或者您没有所选主题的图标，Jabber 客户端将使用默认图像。<br>您可以使用客户端浏览器可以渲染的任何图标，包括 .JPG、.PNG 和 .GIF 格式。<br>此参数是可选的。如果您未指定图标，客户端会从HTML页面加载首选图标。如果没有可用的 favicon，客户端会加载默认图标。 |
| url | 指定嵌入式选项卡内容所在的 URL。<br>客户端使用浏览器渲染引擎来显示嵌入式选项卡的内容。因此，您可指定浏览器支持的任何内容。<br>对于 Cisco Jabber Mac 版本，URL 元素必须包含 HTTP 或 HTTPS。<br>此参数为必选。<br>**注释** 如果目标网页需要 Windows 集成身份验证，则默认情况下，Jabber 会提示用户输入登录凭证。为避免提示，您可以在 Windows 注册表中配置验证服务器白名单。<br>将列入白名单的 URL 加入到以下位置：<br>• `HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Google\Chrome\AuthServerWhitelist`<br>• `HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Google\Chrome\AuthNegotiateDelegateWhitelist`<br>例如，假设您将 `AuthServerWhitelist` 和 `AuthNegotiateDelegateWhitelist` 策略设置为 `*example.com,*foobar.com,*baz`。以 'example.com'、'foobar.com' 或 'baz' 结尾的任何 URL 均位于允许列表中。如果没有 '*' 前缀，则 URL 必须完全匹配。 |

| 参数 | 说明 |
| --- | --- |
| 机构内部 | 指定您的网页是网络的内部页面还是外部页面。<br>• true（默认值）— 您的网页是网络的内部页面。<br>• false — 您的网页是网络的外部页面。 |

# 用户自定义选项卡

您可以允许用户通过客户端用户界面指定自定义嵌入式选项卡的选项卡名称和 URL。用户无法为自定义嵌入式选项卡设置其他参数。

将 AllowUserCustomTabs 设置为 **true**，然后用户才能自定义其选项卡：

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

**注释** AllowUserCustomTabs 的默认值为 **true**。

# 自定义图标

为取得最佳效果，您的自定义图标应该符合以下原则：

- 尺寸：20 x 20 像素
- 透明背景
- PNG 文件格式

您可以包含不同版本的图标，以匹配特定的主题。有关详细信息，请参阅 icon 参数的说明。

# 从自定义选项卡聊天和呼叫

您可以使用协议处理机从自定义嵌入式选项卡启动聊天和呼叫。确保自定义嵌入式选项卡为 HTML 页面。

使用 `XMPP:` 或 `IM:` 协议处理程序启动聊天。
使用 `TEL:` 协议处理程序启动音频和视频呼叫。

# 用户 ID 令牌

您可以指定 `${UserID}` 令牌作为 url 参数值的一部分。用户登录时，客户端会将 `${UserID}` 令牌替换为已登录用户的用户名。

提示 您还可以在查询字符串中指定 `${UserID}` 令牌；例如，`www.cisco.com/mywebapp.op?url=${UserID}`。

以下是如何使用 `${UserID}` 令牌的示例：

1. 您可以在自定义嵌入式选项卡中指定以下内容：

   ```
   <url>www.cisco.com/${UserID}/profile</url>
   ```

2. Mary Smith 登录。她的用户名是 msmith。
3. 客户端会将 `${UserID}` 令牌替换为 Mary 的用户名，如下所示：

   ```
   <url>www.cisco.com/msmith/profile</url>
   ```

# JavaScript 通知

您可以执行自定义嵌入式选项卡中的 JavaScript 通知。此主题说明客户端为 JavaScript 通知提供的方法。此主题还提供用于测试通知的示例 JavaScript 表格。此文档中未说明如何为异步服务器呼叫及其他自定义实施执行 JavaScript 通知。请参阅适当的 JavaScript 文档，以了解详细信息。

### 通知方法

客户端包括为 JavaScript 通知提供以下方法的界面：

- SetNotificationBadge—您可以在 JavaScript 中从客户端调用此方法。此方法使用可以具有任意以下值的字符串值：
  - 空—空值会删除任何现有的通知标志。
  - 数字 1 到 999
  - 两位字母数字组合，例如 A1
- onPageSelected()—当用户选择自定义嵌入式选项卡时，客户端会调用此方法。
- onPageDeselected()—当用户选择另一个选项卡时，客户端会调用此方法。

注释 不适用于 Jabber iPhone 和 iPad 版本

- onHubResized()—当用户调整客户端中央窗口大小或移动窗口时，客户端会调用此方法。
- onHubActivated()—客户端中央窗口激活后，客户端会调用此方法。
- onHubDeActivated()—客户端中央窗口停用后，客户端会调用此方法。

### 订阅自定义选项卡中的在线状态

您可以使用以下 JavaScript 方法订阅联系人的在线状态，并从客户端接收在线状态更新：

- SubscribePresence()—使用用户的 IM 地址为此方法指定字符串值。
- OnPresenceStateChanged—此方法使用户能够基于联系人的在线状态从客户端接收更新。您可以指定以下值之一作为字符串：
  - IM 地址
  - 基本在线状态（有空、离开、离线、免打扰）
  - 多样的在线状态（会议中、呼叫中或自定义在线状态）

注释

- 对于不在联系人列表中的人员的订阅将在68分钟后过期。订阅过期后，必须重订才能查看其在线状态数据。
- Jabber iPad 和 iPhone 版本仅支持 OnPresenceStateChanged。

### 在自定义选项卡中获取区域设置信息

您可以使用以下 JavaScript 方法从客户端检索联系人的当前区域设置信息：

- GetUserLocale()—此方法使用户能够从客户端请求区域设置信息。
- OnLocaleInfoAvailable—此方法使用户能够从客户端接收区域设置信息。您可以使用包含客户端区域设置信息的字符串值。

注释 Jabber iPad 和 iPhone 版本仅支持 OnLocaleInfoAvailable。

### 调整自定义选项卡以匹配客户端主题

您可以使用以下 JavaScript 方法返回当前客户端主题：

- QueryCurrentTheme()—您可以通过此方法获取当前的 Jabber 主题。在自定义选项卡页面，按如下方式使用此方法：`window.external.QueryCurrentTheme()`。
- OnThemeChanged(*theme*)—当 Jabber 中的主题更改时，此方法将被动接收新主题。

以下是主题的可能值：

- 默认值—默认的 Jabber 主题
- 深色—Jabber“深色”主题
- 独特—Jabber“高对比度”主题

- 高对比度—Windows 高对比度主题

### 示例 JavaScript

此示例显示了一个 HTML 页面，该页面使用 JavaScript 来显示可在其中输入数字 1 – 999 的表单：

```
<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>
```

要测试此示例 JavaScript 表格，请将上面的示例复制到 HTML 页面中，然后将此页面指定为自定义嵌入式选项卡。

# 在自定义选项卡中显示呼叫事件

您可以使用以下 JavaScript 函数在自定义选项卡中显示呼叫事件：

OnTelephonyConversationStateChanged — 电话服务中的 API 使客户端能够在自定义嵌入式选项卡中显示呼叫事件。自定义选项卡可以实施 `OnTelephonyConversationStateChanged` JavaScript 功能。每当电话对话状态发生变化时，客户端都会调用此函数。该函数接受客户端解析的 JSON 字符串以获取呼叫事件。

以下代码段显示容纳呼叫事件的 JSON：

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

JSON 中的每个参与者对象均可具有以下属性：

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

以下是此函数在自定义嵌入式选项卡中的示例实施。此示例获取 `state` 和 `acceptanceState` 属性的值，并在自定义选项卡中显示这些值。

```
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);
      }
    }
```

以下是此函数的示例实施，其中包含所有可能的字段：

```
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);
      }
    }
```

# 自定义嵌入式选项卡示例

以下是具有一个嵌入式选项卡的配置文件的示例：

```
<?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>
```

# 在 Chromebook 中配置 Cisco Jabber Android 版本

| 客户端 | | | |
|---|---|---|---|
| **Windows** | **Mac** | **iPhone 和 iPad** | **Android** |
| — | — | — | 是 |

| 部署 | | | |
|---|---|---|---|
| 场内 | **Webex Messenger** | 组消息模式 | **VDI 软终端** |
| 是 | 是 | 是 | — |

**在 Chromebook 上配置 Cisco Jabber Android 版本的核对表**

| 任务 | 详细信息 |
|---|---|
| 请参阅支持的设备型号和操作系统 | 请参阅 *Android* 操作系统要求和支持的 *Chromebook* 型号 |
| 在公司网络中添加 MRA 配置 | 请参阅在公司网络中添加 *MRA* 配置 |
| 将 Chromebook 用户配置为 TAB 设备类型 | 请参阅将 *Chromebook* 用户配置为 *TAB* 设备类型 |
| 保持所需端口处于打开状态，以使您的用户访问 Chromebook 上的所有 Cisco Jabber 服务 | 请参阅配置端口 |

### Android 操作系统要求和支持的 Chromebook 型号

Chromebook 必须有 Chrome OS 53 或更高版本。用户可以从 Google Play 商店下载 Cisco Jabber Android 版本。

支持的 chromebook 型号：

- HP Chromebook 13 G1 笔记本 PC
- Google Chromebook 像素
- Samsung Chromebook Pro

### 在公司网络中添加 MRA 配置

从您的公司以及移动和远程访问 (MRA) 网络进行连接时，在 Chromebook 上使用 Cisco Jabber。要使用呼叫服务，Cisco Jabber 必须使用 MRA 网络登录。

要在您的用户在公司网络内操作时连接到 MRA 网络，请使用 "_collab-edge._tls.<domain>.com" SRV 记录配置您的内部域名服务器 (DNS)。有关 DNS 的完整详细信息，请参阅《*Cisco Jabber* 规划指南 *12.1*》中的服务发现部分。

### 将 Chromebook 用户配置为 TAB 设备类型

您可以将 Chromebook 用户配置为 TAB 设备类型。有关如何为用户配置软终端服务的完整详细信息，请参阅《*Cisco Jabber* 内部指南 *12.1*》的配置软终端部分。

### 配置端口

确保打开以下端口以在 Chromebook 上访问 Cisco Jabber 服务：

| 目的 | 协议 | 内部网络（源） | Expressway-E（目标） |
|---|---|---|---|
| XMPP(IM&P) | TCP | >=1024 | 5222 |
| HTTP 代理 (UDS) | TCP | >=1024 | 8443 |
| 媒体 | UDP | >=1024 | 36002 到 59999 |
| SIP 信令 | TLS | >=1024 | 5061 |

### 限制

在视频呼叫期间，如果用户切换到另一个应用程序，视频将停止。

# Cisco Jabber 移动应用程序升级

| 客户端 | | | |
|---|---|---|---|
| **Windows** | **Mac** | **iPhone 和 iPad** | **Android** |
| 是 | — | — | — |

| 部署 | | | |
|---|---|---|---|
| **场内** | **Webex Messenger** | **组消息模式** | **VDI 软终端** |
| 是 | 是 | 是 | 是 |

您可以为 Cisco Jabber Windows 版本用户启用通知，以促进用于移动应用程序的 Cisco Jabber（Android 版本和 iOS 版本）的使用。单击通知会将用户转到**设置**页面，其可在此处选择从 Google Play 或 iTunes Store 下载应用。添加新参数 EnablePromoteMobile 以控制这些通知。默认情况下会禁用此功能。

有关配置此参数的详细信息，请参阅《*Cisco Jabber* 参数参考指南》。