Genesys Video Developer's Guide

Genesys Video Gateway 9.0.0

1/2/2022

Table of Contents

Genesys Video Gateway Developer's Guide

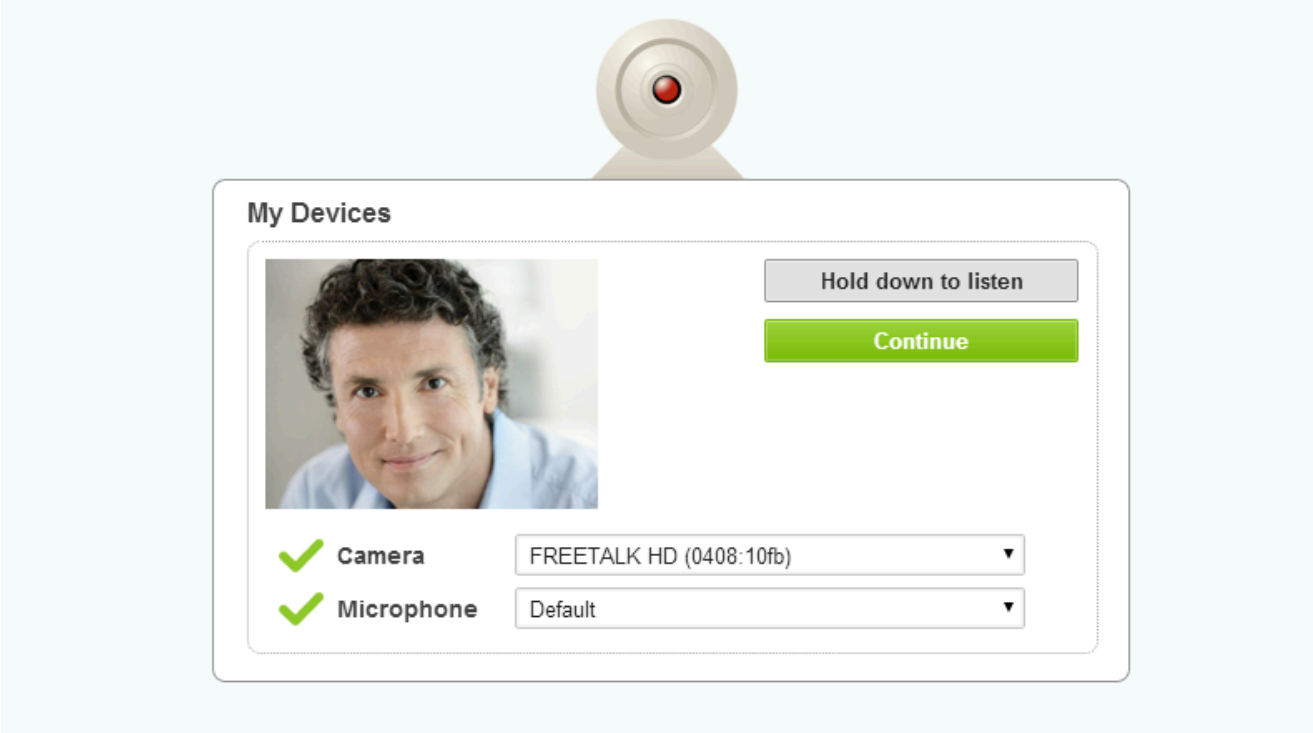
This developer's guide includes the following pages:

Overview

This guide provides an overview of the features built on the underlying SDKs and demonstrates some of the key platform features. The Client SDK provides you with the tools you need to create your own client applications.

Device Check

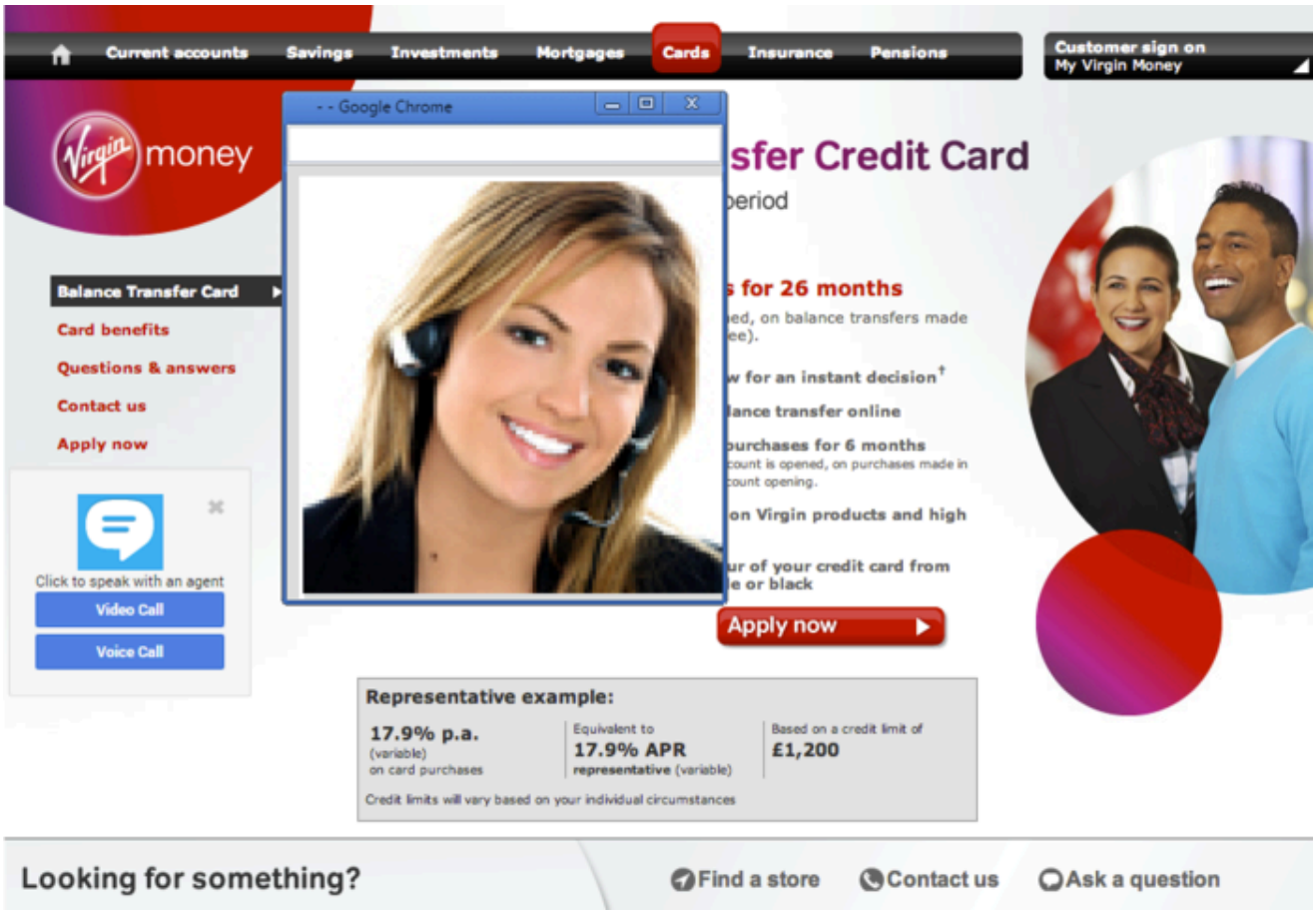
Client applications can be configured so that for a user, a webcam and microphone check is performed.



Click to Call

By adding `/user/demo/index2.jsp?destination=XXXXXXX` to your platform's domain, and replacing XXXXXX with a SIP address of the form `sip:genesys@sip-server-address` the call will be sent to an agent logged in at a SIP soft-phone such as Bria or Genesy SIP Endpoint.

A mock-up of a customer web page can look like this:



This demonstration presents a video/voice calling client to the visitor.

Note: By adding `bg=XXXXXX.jpg` to the end of the URL and uploading the image to the images directory under `user/demo` (you will need direct access to the server to do this), it is possible to very simply swap out the `Virgin Money` image shown above to another image of your choice.

Click-to-Call Widget

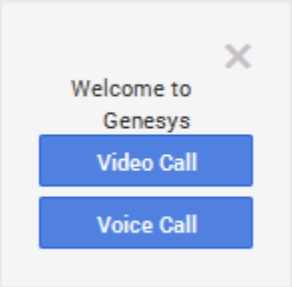
Introduction

Use the Click-to-Call (C2C) Widget to add a small customizable widget to a web page. This widget displays up to two buttons, each of which allows you to make a call. One button creates a video call, while the other is for voice-only calls. When you select a button, it opens a new window that makes the appropriate type of call.

The code for the widget is located on the server at `/opt/zenon/public_html/ZMobile_JS/ZMobile_JS/lib/SDK/sdk/C2CWidget.js`. The demo code is located on the server at `/opt/zenon/public_html/user/demo/index2.jsp` and `/opt/zenon/public_html/user/demo/index3.jsp`.

A patch that includes some improvements is available, along with a Readme that explains how to apply the patch. Click here to download the patch.

You can customize the widget by passing parameters to it. For example, the voice-only and video call buttons can be disabled. This is what the default widget looks like:



Supported Parameters

The C2C widget supports the following parameters:

uiParent	The C2C widget will add HTML to an element with this ID. This is meant to be used with a element.
widgetStyle	This is CSS that can be applied to the widget. If not specified, a default style is used instead.
closeStyle	This is CSS that can be applied to the close button of the widget. If not specified, a default style is used instead. If the close button is disabled, then this has no effect.

topImage	The C2C widget can have an image displayed on the top section. This specifies the image to be shown. By default, there is no image.
text	The C2C widget can have text displayed on the top section, below an image. This specifies the text to be shown. The default value is "Welcome to Genesys".
videoCallButtonImage	An image can be used as the video call button instead of a default button. This specifies which image to use.
voiceCallButtonImage	An image can be used as the voice call button instead of a default button. This specifies which image to use.
buttonStyle	This is CSS that can be applied to the buttons. It has no effect if *videoCallButtonImage* and *voiceCallButtonImage* are not specified.
iconStyle	This is CSS that can be applied to the button icons. It has no effect if *videoCallButtonImage* and *voiceCallButtonImage* are not specified.
enableVideoButton	If this is false, then the video call button is disabled. Otherwise, the button is enabled.
enableVoiceButton	If this is false, then the voice call button is disabled. Otherwise, the button is enabled.
disableCloseButton	If this is true, then the close button will be disabled on the widget. Otherwise, the widget will have a close button.
hideLeftPosn	This, along with *hideTopPosn*, specifies where the widget will go when the close button is pressed. This parameter specifies where on the screen the left side of the widget will be when hidden.
hideTopPosn	This, along with *hideLeftPosn*, specifies where the widget will go when the close button is pressed. This parameter specifies where on the screen the top side of the widget will be when hidden.
showLeftPosn	This, along with *showTopPosn*, specifies where the widget will go when the widget is revealed. This parameter specifies where on the screen the left side of the widget will be when shown.
showTopPosn	This, along with *showLeftPosn*, specifies where the widget will go when the widget is revealed. This parameter specifies where on the screen the top side of the widget will be when shown.
showC2CImmediately	If this is true, then the C2C widget is shown immediately. Otherwise, the C2C widget's initial position will be the same as the element specified by *uiParent*.
callPageToOpen	This is the URL of the page that will be opened when the voice-only or video call button is clicked. A new window will be opened and then go to this page. This is intended to be specific pages (for

	example, */user/demo/sdki.html*) hosted by the ZAS, but could in theory be used with any web page. If another web page is to be used, it should support the parameters that will be passed to the newly created window.
callWindowLeft	This specifies the left position of the newly opened window.
callWindowTop	This specifies the top position of the newly opened window.
callWindowWidth	This specifies the width of the newly opened window.
callWindowHeight	This specifies the height of the newly opened window.
service	This specifies which service will be used when retrieving configurations. It passes the value of this parameter to the *service* parameter of the newly opened window.
destination	This specifies the destination of the call. It passes the value of this parameter to the *destination* parameter of the newly opened window.
customSipHeader	This can be specified to add a custom SIP header to the call. It passes the value of this parameter to the *arg1* parameter of the newly opened window.
forceFlash	This can be specified to force the call to use flash. It passes the value of this parameter to the *forceFlash* parameter of the newly opened window.
windowName	This can be used to specify the window name. It has no direct effect on the functionality, but the window name can be used to get a reference to the newly opened window.
IeCompatibilityModeErrorMessage	If a user is using Internet Explorer in compatibility mode, this message will be shown to the user.
webRTCVideoWidth	If the user is using WebRTC, this specifies the width of the video.
webRTCVideoHeight	If the user is using WebRTC, this specifies the height of the video.
flashVideoWidth	If the user is using Flash, this specifies the width of the video.
flashVideoHeight	If the user is using Flash, this specifies the height of the video.
videoFPS	This specifies the frame rate of the video.
webRTCEnableSFU	This specifies whether SFU will be used for WebRTC.
webRTCVideoBandwidth	This specifies the video bandwidth of a WebRTC call.
flashMinVideoBandwidth	This specifies the minimum video bandwidth of a flash call.
flashMaxVideoBandwidth	This specifies the maximum video bandwidth of a

	flash call.

Requirements

The C2C widget requires that you include jQuery, Zenon JavaScript SDK, and Underscore on the client page that uses it. If no styles are specified, the widget uses default styles. In that case, you must also include the default CSS.

Client SDK

The Client SDK is available for Javascript. The Javascript SDK works in all desktop browsers using Flash as well as WebRTC.

Classes and methodologies from the low level API of the CallManager to the higher level Model-View-Controller classes which implement the UI are listed in the Javascript Class Reference. You can download the API Reference from the following page:

- API Reference

Using the Client SDK

This is a tutorial for creating a page where a call is made upon loading and the results displayed in the center of the screen. By default, a video call is made and the video and audio that the server receives is sent sent to genesys@gvp.genrtc.net.

Support CSS and scripts

To use the Client SDK on a browser page, you must include the following CSS and JavaScript in your HTML source:

```
<link rel="stylesheet" type="text/css" href="//<yourdomain.com>/user/demo/css/zsdk.css">
<script id="zsdkjs" type="text/javascript" src="//<yourdomain.com>/ZMobile_JS/ZMobile_JS/lib/
SDK/sdk/zcontact/build/zsdk.js?v=3.1"></script>
```

> **Important**
>
> Some components require that other libraries, such as jQuery, be included as well.

Set some basics

In this section, we will gather information such as where the call should go and what kind of call it should be, and save it in some variables.

```
// Developer ID
ConnectionManager.partner = "home";

// Destination to dial.
var destination = GeneralUtils.getURLParameter("destination");
if (!destination)
```

```
        destination = “sip:genesys@gvp.genrtc.net”;

// Passing Custom SIP header
// Appears to SIP-Server in INVITE as:
// X-Custom-2: val2
// X-Custom-1: val1
CallManager._arg1 = "ch%23X-Custom-1%23val1%23ch%23X-Custom-2%23val2";

// Voice or Video call
var videoCall = true;
if (GeneralUtils.getURLParameter("type")=="VOICE")
videoCall = false;
```

ConnectionManager.partner is used to tell the server which customer is calling, and from that it figures out which settings it should load. You should not need to change this value.

The *destination* variable is used to determine the destination of the call. The *videoCall* variable is used as a parameter for the *PlaceCall()* function. In this page, it specifies that a video call is to be made (instead of voice). In this example, custom SIP headers are specified. In general, any customer SIP headers may be assigned to *CallManager._arg1*. When a call is placed, the custom SIP headers will be included.

Create session with service

A session and a connection to the server must be established before a call can be made. This example assumes that login information and possibly a session ID are passed to the page. If a session has already been established, then the session ID can be used to locate and use the existing session. If it has not been established, then a session must be established by logging in.

```
var session = GeneralUtils.getURLParameter("session");
var login = GeneralUtils.getURLParameter("login");
var password = GeneralUtils.getURLParameter("password");
/* Authenticate user with service */
if (session)
        ConnectionManager.locateAndLogin('', '', '',session,'spm');  //  obtain session using
provisioning API or login directly
else
        ConnectionManager.locateAndLogin(login,password); // username and password
```

Initialize the calling SDK

To create a screen where the call actually takes place, we use the ZUI class. The ZUI class is a UI element that displays the call. There are settings to control the display, such as setting whether or not it is a video-only call. It is important to note that the *uiParent* parameter must be the ID of an existing HTML element with a # prepended. In this example, the HTML element is `<div id="myVideoClient"> </div>` and *uiParent* is *#myVideoClient*. The ID should be unique in the page. This element is usually a `<div>` element. After creating a ZUI, we have CallManager use that ZUI to display the video and output audio by using *setZUI()*:

```
var zui;
$(document).ready(function() {
```

```
zui=new ZUI({
                       "uiParent": "#myVideoClient",
                      "widget": "phone",
                     "video": videoCall,
                     "showTimer": true,
                      "allowDeny": "tomato dash",
                     "podTitle": "Phone Pod",
                      "showStatus": true,
                      "draggable": true,
                      "resizable": true,
                     "showControls": true,
                "showTechCheck":true,
                "flashBgColor":"#333333"
                });


CallManager.setZUI(zui,  videoCall ?  CallManager.CAM_MIC : CallManager.MIC, false);
});
```

Place the call

Now the page will wait for the server to be ready to place a call. This is done by listening for the ZWRTC_READY event on the ConnectionManager. A call is made to the destination that was specified earlier.

```
ConnectionManager.addListener('ZWRTC_READY', this, function(event) {
if (destination)
CallManager.placeCall(videoCall ? "VIDEO" : "VOICE", destination,  "Sample Call");
});
```

Send DTMF from web client

After a call is successfully placed and established, DTMF tones can be sent from the client application to the remote peer via the GVG platform. You can achieve this by using the `CallManager` class, which allows placing and receiving voice and video calls. Also, you must update the MCU configuration, as described below.

The supported DTMF tones are: `0,1,2,3,4,5,6,7,8,9,*,#`

The following simple example shows how a DTMF dial-pad can be created using Javascript, and then the DTMF tones can be sent using that dial-pad:

```
 <!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
. . .
<body>
. . .
<div id="dialingPad"> </div>
. . .
<script type = "text/javascript” >
// Send DTMF tone
function sendDTMF(tone) {
  console.log("Sending DTMF [" + tone + "]");
  CallManager.sendDTMF(tone);
```

```
};

// Create a phone like dialing pad for sending DTMF to remote peer.
function createDialingPad() {
  var tones = '1234567890*#';
  var dialingPad = document.getElementById('dialingPad');
  for (var i = 0; i < tones.length; ++i) {
    var tone = tones.charAt(i);
    dialingPad.innerHTML += '<button id="' +
      tone + '" onclick="sendDTMF(\'' + tone +
      '\')" style="height:40px; width: 30px">' + tone + '</button>';
    if ((i + 1) % 4 == 0) {
      dialingPad.innerHTML += '<br>';
    }
  }
};

createDialingPad();
</script>
. . .
</body>
</html>
```

Update MCU configuration

Updating the MCU configuration is required for sending DTMF to the remote peer as telephone-events, as per RFC 2833. In the Application Server, update the following configuration file:

`/opt/zenon/public_html/WEB-INF/infotypes_saypage.xml`

In this file, locate the parameter `SIPUserInputMode`, and replace the default value of `SIPINFO` with the value RFC2833.

```
<config>
    <name>SIPUserInputMode</name>
    <!-- <value>SIPINFO</value> -->
    <value>RFC2833</value>
    <type>replace</type>
</config>
```

Mute/un-mute speaker/microphone/camera

The ZUI class provides methods for muting/un-muting the local audio/video devices. In your HTML page, you can use those methods to mute/un-mute your camera or microphone once a call is established.

Speaker

- spkmute - mutes the speaker device
- spkunmute - unmutes the speaker device

Example:

```
<a href="#" onclick="zui.spkmute();">Disable Speaker</a>
<a href="#" onclick="zui.spkunmute();">Enable Speaker</a>
```

Microphone

- mute - mutes the microphone
- unmute - un-mutes the microphone

Example:

```
<a href="#" onclick="zui.mute();">Disable Microphone</a>
<a href="#" onclick="zui.unmute();">Enable Microphone</a>
```

Camera

- cammute - stops the camera
- camunmute - starts the camera

Example:

```
<a href="#" onclick="zui.cammute();">Stop Camera</a>
<a href="#" onclick="zui.camunmute();">Start Camera</a>
```

Full HTML

```
<!doctype html>
<html>
<head>
        <title>SDK Sample</title>
        <link rel="stylesheet" type="text/css" href="//<yourdomain.com>/user/demo/css/
zsdk.css">
          <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-
ui.css">
          <script src="//code.jquery.com/jquery-1.9.1.js"></script>
          <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
        <script id="zsdkjs" type="text/javascript" src="//<yourdomain.com>/ZMobile_JS/
ZMobile_JS/lib/SDK/sdk/zcontact/build/zsdk.js?v=3.1"></script>
</head>

<script type = "text/javascript" >

/* Basic steps */
        // Developer ID
        ConnectionManager.partner = "home";
        // Destination to dial.
        var destination = GeneralUtils.getURLParameter("destination");
        if (!destination)
                destination = "sip:genesys@gvp.genrtc.net";
        // Passing Custom SIP header
        CallManager._arg1 = "ch%23X-Custom-1%23val1%23ch%23X-Custom-2%23val2";
        // Voice or Video call
        var videoCall = true;
        if (GeneralUtils.getURLParameter("type")=="VOICE")
```

```
                videoCall = false;


/* Authenticate user with service */
        var session = GeneralUtils.getURLParameter("session");
        var login = GeneralUtils.getURLParameter("login");
        var password = GeneralUtils.getURLParameter("password");
        if (session)
                ConnectionManager.locateAndLogin('', '', '',session,'spm');
        else
                ConnectionManager.locateAndLogin(login,password);

/* Initialise the calling SDK with the DOM components to be used for rendering */
        var zui;
        $(document).ready(function() {

                zui=new ZUI({
                    "uiParent": "#myVideoClient",
                    "widget": "phone",
                    "video": videoCall,
                    "showTimer": true,
                    "allowDeny": "tomato dash",
                    "podTitle": "Phone Pod",
                    "showStatus": true,
                    "draggable": true,
                    "resizable": true,
                    "showControls": true,
                    "showTechCheck":true,
                    "flashBgColor":"#333333"
                });


                  CallManager.setZUI(zui,  videoCall ?  CallManager.CAM_MIC :
CallManager.MIC, false);
        });

/* Place the call */
        ConnectionManager.addListener('ZWRTC_READY', this, function(event) {
        if (destination)
            CallManager.placeCall(videoCall ? "VIDEO" : "VOICE", destination,  "Sample Call");
        });

</script>

<!-- HTML UI -->
<body>
        <center><h2>Test Call</h2></center>
        <div id="myVideoClient"> </div>
</body>
```