omnissa™

Horizon SDK for WebRTC Redirection Programming Guide

May 30, 2025

Contents

Introduction

Omnissa provides several software development kit (SDK) products, each of which targets different developer communities and platforms.

With the Omnissa Horizon® SDK for WebRTC Redirection, you can enable media-optimized WebRTC Redirection for Electron applications that communicate between a client and a remote desktop over a Horizon connection using the Blast Extreme or PCoIP display protocol.

This SDK includes resources such as files, a sample Electron application, and documentation to help you develop applications that use the Omnissa Horizon WebRTC Redirection API.

About this Programming Guide

This document provides information about how to enable WebRTC Redirection of media streams in unified communication (UC) applications using the Omnissa Horizon WebRTC Redirection Application Programming Interface (API).

Intended Audience

This guide is intended for software developers who want to enable WebRTC Redirection for UC applications used remotely over a Omnissa Horizon 8 or Omnissa Horizon Cloud Service on Microsoft Azure connection.

> **Note:** This SDK only supports Electron-based UC applications running on Omnissa VDI deployments. It does not support CEF, PWA, or browser-based WebRTC applications. To implement media optimization for browser-based WebRTC applications, use the Browser Content Redirection feature.

How to Download

This SDK can be downloaded through Github

Note: The GitHub link is the only valid and supported source. It is not recommended to download any packages from npmjs.com.

About the Omnissa Horizon WebRTC Redirection API

The Omnissa Horizon WebRTC Redirection API specifies how the client side and the desktop side of an Electron application can communicate over a Horizon connection to accomplish WebRTC-based media redirection. Most of the interactions with the API are asynchronous.

Application Software Components

Any Electron software that uses the Horizon WebRTC Redirection API must include:

- The Electron application itself
- Horizon SDK for WebRTC Redirection, installed as JavaScript code in the application

Throughout this document, “Application” refers to the Electron-based app using the SDK.

Horizon WebRTC Redirection API Components

- **Basic WebRTC APIs**: Provide calls to retrieve the media device list and capture local streams.
- **Teleconnection APIs**: Provide functionality similar to W3C peer-to-peer communications APIs. See W3C WebRTC API.
- **MediaStream APIs**: Align with standard Web MediaStream APIs. See MDN MediaStream docs.

Supported Versions of Horizon Software

To work with the Horizon SDK for WebRTC Redirection, your Horizon deployment must meet certain system and setup requirements. For detailed information, see the “Omnissa Horizon SDK for WebRTC Redirection Setup Guide”.

Horizon SDK for WebRTC Redirection Program Flow

A typical Horizon WebRTC Redirection program flow involves two phases: 1. Initialization of the Application 2. Initialization of the Horizon SDK for WebRTC Redirection

Application Initialization

The desktop-side Application startup is initiated by the user.

On launch, the Application: - Calls the system API to get the current Windows session ID. - Looks up these registry values:

Registry Key	Description
`HKCU\Software\Omnissa\HTML5 Redirection Server\$(session id)\secureWebport`	Local Horizon Agent Web Socket port
`HKCU\Volatile Environment\$(session id)\ViewClient_Client_ID`	Horizon View client ID

> These values are used by the Application to help initialize the SDK.

SDK Initialization

The Application initializes the Horizon SDK by implementing three asynchronous functions under the `window` namespace:

Function	Description
`getHorizonClientID()`	Returns a Promise resolving to the Horizon View client ID.
`getHorizonWSSPort()`	Returns a Promise resolving to the WebSocket port.
`getWindowReference()`	Returns a Promise resolving to the current app window handle as a hex string.

Example: Calling an async function

```
let port = await window.getHorizonWSSPort();
```

You may optionally configure your app to provide a log object to the SDK during initialization. If no log is provided or if it’s invalid, the SDK will fall back to using its built-in logging, viewable in the browser console.

> A valid logger should include methods like `error`, `info`, and `warn`, similar to the standard `console` object.

Basic WebRTC API Reference

This chapter describes the Basic WebRTC API functions in the Horizon SDK for WebRTC Redirection. These Basic WebRTC APIs can be accessed directly.

Note: For reference information about the RTCPeerConnection APIs in the Horizon SDK for WebRTC Redirection, see https://www.w3.org/TR/webrtc/#peer-to-peer-connections. For reference information about the MediaStream APIs in the SDK, see https://developer.mozilla.org/en-US/docs/Web/API/MediaStream.

`enumerateDevices()`

Gets the media device list and returns it to the application.
This function is similar to `MediaDevices.enumerateDevices()`.

Parameters

None

Return Values

Type	Description
Promise	Resolves to an array of `MediaDeviceInfo` objects or rejects with a related `DOMException`

Code Example

```
let _deviceList = await HorizonWebRtcRedirectionAPI.enumerateDevices();
```

`getCallConfig(index)`

Helper function that processes call configurations and returns the associated `pcfId`.

Use this to retrieve configuration data when handling multiple call sessions or scenarios indexed by ID.

Parameters

Name	Type	Description
`index`	Number	Index of the call configuration to retrieve

Return Values

Type	Description
Object	Object containing the `pcfId` field

Code Example

```
let config = HorizonWebRtcRedirectionAPI.getCallConfig(0);
console.log(config.pcfId);
```

getDisplayMedia(constraints)

Captures the contents of the remote desktop as a media stream and returns the stream to the application.
This function is similar to `MediaDevices.getDisplayMedia()`.

Parameters

Name	Description
`constraints`	*(Required)* An object that specifies details about the media stream. More info

Return Values

Type	Description
Promise	Resolves to a `MediaStream` or rejects with a related `DOMException`

Code Example

```
let _stream = await HorizonWebRtcRedirectionAPI.getDisplayMedia(_constraints);
```

getReceiverCapabilities(kind)

Acquires the `RTCRtpReceiver` capabilities on the client side.
This function is similar to `RTCRtpReceiver.getCapabilities()`.

Parameters

Name	Description
`kind`	*(Required)* The type of capabilities to acquire. Allowed values: `"audio"` or `"video"`

Return Values

Type	Description
`RTCRtpCapabilities`	The capabilities object. MDN Reference

Code Example

```
let _caps = HorizonWebRtcRedirectionAPI.getReceiverCapabilities("video");
```

getScreenInfo()

Gets information about the monitors connected to the client system.

Parameters

None

Return Values

Type	Description
Promise	Resolves to an array of screen info objects or rejects with an error

Code Example

```
let _screenInfo = await HorizonWebRtcRedirectionAPI.getScreenInfo();
```

getUserMedia(constraints)

Retrieves the media stream from the local media device and returns it to the application.
This function is similar to `MediaDevices.getUserMedia()`.

Parameters

Name	Description
`constraints`	*(Required)* An object specifying the details of the media stream. More info

Return Values

Type	Description
Promise	Resolves to a `MediaStream` or rejects with a related `DOMException`

Code Example

```
let _stream = await HorizonWebRtcRedirectionAPI.getUserMedia(_constraints);
```

getUserMediaEx(constraints, callConfig)

Retrieves the media stream from the local media device and returns it to the application.
This function is similar to `MediaDevices.getUserMedia()`.

Parameters

Name	Description
`constraints`	*(Required)* An object specifying the details of the media stream. More info
`callConfig`	*(Optional)* An opaque object specifying the call configuration object. The object can be retrieved through getCallConfig(). If undefined, getUserMediaEx() is reduced to getUserMedia().

Return Values

Type	Description
Promise	Resolves to a `MediaStream` or rejects with a related `DOMException`

Code Example

```
let callConfig = HorizonWebRtcRedirectionAPI.getCallConfig(configIndex);
let _stream = await HorizonWebRtcRedirectionAPI.getUserMediaEx(_constraints, call
Config);
```

initSDK(appLogger, appName, eventCallback, sdkConfig)

Initializes the SDK so the application can access API methods.

Parameters

Name	Description
`appLogger`	*(Optional)* An object with functions like `error`, `info`, and `warn` (compatible with `console`)
`appName`	*(Required)* A string that names the application
`eventCallback`	*(Optional)* A callback function to receive events from the SDK
`sdkConfig`	*(Optional)* A configuration object provided the following options. `numOfCallConfigs`: The default value for the numOfCallConfigs is 1 if the numOfCallConfigs is undefined. After initSDK() called, use the getCallConfigs(index) to retrieve the corresponding callConfig object. The callConfig object can be passed to newPeerConnectionEx() and getUserMediaEx() in order to allow different audio headset used for different RTCPeerConnection object.

Return Values

Value	Description
`true`	Initialization successful; SDK APIs can be used
`false`	Initialization failed; app will fall back to standard Electron behavior

Code Example

```
HorizonWebRtcRedirectionAPI.initSDK(myAppLogger, "My App Name", myCallbackFn);
```

```
let sdkConfig = { numOfCallConfigs: 2 };
HorizonWebRtcRedirectionAPI.initSDK(myAppLogger, "My App Name", myCallbackFn, sdk
Config);
```

`isFeatureSupported(feature)`

Checks whether the specified feature is supported by the SDK.

Currently supported features: - `"video"` - `"screenshare"` - `"multimonitorscreenshare"` - `"datachannel"`

Parameters

Name	Description
`feature`	*(Required)* Name of the feature to check

Return Values

Value	Description
`true`	The feature is supported
`false`	The feature is not supported

Code Example

```
let _isSupported = HorizonWebRtcRedirectionAPI.isFeatureSupported("datachannel");
```

`newMediaStream(tracks)`

Creates a new `MediaStream` object for the application.
This is similar to the `MediaStream` constructor.

Parameters

Name	Description
`tracks`	*(Required)* Array of `MediaStreamTrack` objects created or returned by the SDK

Return Values

Type	Description
`MediaStream`	A new `MediaStream` object

Code Example

```
let _stream = await HorizonWebRtcRedirectionAPI.newMediaStream(_tracks);
```

`newPeerConnection(configuration)`

Creates a new `RTCPeerConnection` object.
This is similar to the `RTCPeerConnection` constructor.

Parameters

Name	Description
`configuration`	*(Optional)* Peer connection configuration object (e.g., ICE servers)

Return Values

Type	Description
`RTCPeerConnection`	A new peer connection object

Code Example

```
let _pc = HorizonWebRtcRedirectionAPI.newPeerConnection(_configurations);
```

`newPeerConnectionEx(arg1, arg2, callConfig)`

Creates a new `RTCPeerConnection` object.
This is similar to the `RTCPeerConnection` constructor.

Parameters

Name	Description
`configuration`	*(Optional)* Peer connection configuration object (e.g., ICE servers)
`callConfig`	*(Optional)* An opaque object specifying the call configuration object. The object can be retrieved through getCallConfig(). If undefined, newPeerConnectionEx() is reduced to newPeerConnection().

Return Values

Type	Description
`RTCPeerConnection`	A new peer connection object

Code Example

```
let callConfig = HorizonWebRtcRedirectionAPI.getCallConfig(configIndex);
let pc = HorizonWebRtcRedirectionAPI.newPeerConnectionEx(_configurations, {}, cal
lConfig);
```

onAudioCreated(audioElem, window)

Binds an audio element to the SDK for playback redirection on the client.

Parameters

Name	Description
`audioElem`	*(Required)* The actual HTML audio element
`window`	*(Optional)* The application window handle (hex string)

Return Values

None

Code Example

```
HorizonWebRtcRedirectionAPI.onAudioCreated(_myAudioElem, "6810070000000000");
```

onAudioDisposed(audioElem)

Cleans up a previously bound audio element when it’s destroyed.

Parameters

Name	Description
`audioElem`	*(Required)* The HTML audio element to unbind

Return Values

None

Code Example

```
HorizonWebRtcRedirectionAPI.onAudioDisposed(_myAudioElem);
```

onScreenSelected(screenId)

Selects the preferred monitor to use for screen sharing.

Parameters

Name	Description
`screenId`	*(Required)* ID returned from `getScreenInfo()`

Return Values

None

Code Example

```
HorizonWebRtcRedirectionAPI.onScreenSelected("monitor01");
```

onVideoCreated(videoElem, window)

Binds a video element to the SDK for display redirection on the client.

Parameters

Name	Description
`videoElem`	*(Required)* The HTML video element
`window`	*(Optional)* The app window handle (hex string)

Return Values

None

Code Example

```
HorizonWebRtcRedirectionAPI.onVideoCreated(_myVideoElem, "6810070000000000");
```

onVideoDisposed(videoElem)

Cleans up a previously bound video element when it’s removed.

Parameters

Name	Description
`videoElem`	*(Required)* The HTML video element to unbind

Return Values

None

Code Example

```
HorizonWebRtcRedirectionAPI.onVideoDisposed(_myVideoElem);
```

onWindowSessionConnected(state)

Notifies the SDK about the session’s connection status.

Parameters

Name	Description
`state`	*(Required)* Boolean — `true` if connected, `false` if disconnected

Return Values

None

Code Example

```
HorizonWebRtcRedirectionAPI.onWindowSessionConnected(true);
```

pauseRingtone(id)

Pauses a ringtone that’s currently playing on the client.

Parameters

Name	Description
`id`	*(Required)* Unique identifier for the ringtone (string or number)

Return Values

None

Code Example

```
HorizonWebRtcRedirectionAPI.pauseRingtone("audio01");
```

playRingtone(id, src, isLoop)

Plays a ringtone on the client system using a local or remote audio source.

Parameters

Name	Description
`id`	*(Required)* Unique audio identifier
`src`	*(Required)* URL to the ringtone file
`isLoop`	*(Required)* `true` to loop the sound; `false` to play once

Return Values

None

Code Example

```
HorizonWebRtcRedirectionAPI.playRingtone("audio01", "https://example.com/ringtone.mp3", true);
```

setPrimarySinkId(sinkId)

Sets the default output device for audio playback on the client.

Parameters

Name	Description
`sinkId`	*(Required)* The ID of the audio output device

Return Values

None

Code Example

```
HorizonWebRtcRedirectionAPI.setPrimarySinkId("audioDevice01");
```

setSinkId(id, sinkId)

Sets the audio output device for a specific sound stream (e.g., a ringtone or alert).

Parameters

Name	Description
`id`	*(Required)* Audio stream or ringtone identifier
`sinkId`	*(Required)* Output device ID

Return Values

None

Code Example

```
HorizonWebRtcRedirectionAPI.setSinkId("audio01", "audioDevice01");
```

setVideoClipRegion(isAdd, clipRegion, window)

Adds or removes a clipped video display region on the client, associated with a specific application window.

Parameters

Name	Description
`isAdd`	*(Required)* Boolean — `true` to add a region, `false` to remove
`clipRegion`	*(Required)* DOM element (typically a video element)
`window`	*(Optional)* Window handle as a hex string

Return Values

None

Code Example

```
HorizonWebRtcRedirectionAPI.setVideoClipRegion(true, _myVideoElem, "6810070000000
000");
```

Setting Up the Sample Application

The Horizon SDK for WebRTC Redirection includes a sample folder containing a sample Electron application that shows examples of how to use the API. You can run the sample app on either your local machine or remote desktop.

The sample app itself does not contain any open source or third-party code. However, you must install certain NodeJS packages and perform some server setup tasks in order to run the app. Follow the procedures in this chapter to set up the sample app.

Set Up the Web Server (https)

1. Make sure you have NodeJS(latest stable version will be good enough) installed in your environment. For the httpServer you can also use any other statically hosting solution.
2. Copy the httpServer folder to the location you want to host.
3. Copy the folders ‘sdk’ into httpServer folder.
4. Gather a certificate, self signed is fine and place it into a folder called ‘certs’. The final hierarchy should look like this:

```
|--- httpServer
   |--- httpServer.js
   |--- index.html
   |--- common
      |--- constant.js
   |--- sdk
      |--- HorizonSDKforWebRTCRedir.js
   |--- app
      |--- appMain.js
      |--- callMgr.js
      |--- ui.js
      |--- utils.js
|--- certs
   |--- cert.pem
   |--- key.pem
```

5. Enter the ./httpServer folder and run ‘npm install’ then ‘npm start’ (or node httpServer.js)

Set Up the Signaling Server

You can set up the web server (https) and signaling server on either the same machine or on different machines. The servers can either share the same certificate-and-key pair or use different ones. 1. Make sure you have NodeJS(latest stable version will be good enough) installed in your environment. 2. Copy the ‘sigServer’ to the location you want to start. 3. Gather a certificate, self signed is fine and place it into a folder called ‘certs’. copy the certs folder next to the sigServer 4. The final hierarchy should look like this:

```
|--- sigServer
   |--- wsServer.js
   |--- user.js
   |--- package.json
   |--- common
      |--- constant.js
|--- certs
```

```
|--- cert.pem
|--- key.pem
```

5. Enter ./sigServer folder, execute: ‘npm install’ and then ‘npm start’ (or node wsServer.js)

Configure the Client and Run the App

Configure electron client and run

1. Make sure you have NodeJS(latest stable version will be good enough) installed in your environment.
2. Copy client folder to wherever you want
3. Set the link url for index page in ‘client/serverInfo.json’. depends on how you configure your web server in steps above. Example: “pageUlr”: “https://192.168.1.101/sample/index.html”
4. Set the Web Socket server ulr for ‘callServerUrl’ in ‘client/ serverInfo.json’, depends on how you configure your web socket server in steps above. Example: “callServerUrl”: “wss://192.168.1.200:8443”
5. The final hierarchy should look like this

```
|--- client
   |--- main.js
   |--- preload.js
   |--- package.json
   |--- serverInfo.json
```

6. Enter ./client folder, execute: ‘npm install’ and then ‘npm start’

Configure web client and run

1. Add the Horizon WebRTC SDK browser extension to your chrome browser.
2. Note the IP address of your httpServer and the signaling server that will be used in step 4
3. Configure the following registry keys.

```
[HKLM\SOFTWARE\Policies\Omnissa\Horizon\WebRTCRedirSDKWebApp]
"enabled"=dword:00000001<br>
"chrome_enabled"=dword:00000001<br>
"edge_chrome_enabled"=dword:00000001
[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Omnissa\Horizon\WebRTCRedirSDKWebAp
p\UrlAllowList]
"https://httpServerIpAddress:3000/*"=""
```

4. Close and reopen the VM then navigate to https://httpServerIpAddress:3000/webIndex.html?callServerUrl=wss://signalingServerIpAddress:8443