VMware HTML Console SDK Programming Guide

VMware HTML Console SDK 1.0.0

For vSphere 5.1 and later (General Availability)
and vCloud Director (Technical Preview)

September 2015

VMware HTML Console SDK Programming Guide

Table of Contents

HTML Console (WMKS) SDK Overview

The HTML Console SDK is a JavaScript library implemented based on webmks (WMKS), which can provide mouse, keyboard, touch input processing and handling as well as screen updates and cursor changes to a desktop session from a browser. Applications running in the browser can use JavaScript to access virtual machine console functions by using the WMKS SDK API.

The WMKS SDK API contains various methods that can be used to connect to and communicate with a virtual machine. It also triggers events to notify users of changes to the virtual machine's state. You can use these methods and callbacks to give end users the ability to remotely manage a virtual machine from any system with the appropriate web browser and operating system.

Supported Browsers (including on iOS and Android)

- Internet Explorer 10+
- Firefox 24+
- Chrome 30+
- Safari 6.1+

Steps for using the WMKS SDK API

To use the WMKS SDK API, a web-based application must typically do the following things:

1. Load the WMKS SDK JavaScript library files.
2. Create the WMKS core object by using the factory method createWMKS() with a given div element ID in the document DOM and the create options.
3. Register JavaScript callbacks to respond to WMKS triggered Events.
4. Use the WMKS SDK connect() method to connect to a target virtual machine.
5. Use the WMKS SDK methods to interact with the connected virtual machine.
6. Use the WMKS SDK method disconnect() to disconnect the connection (if active) and remove the widget from the associated element.

Placing the console on a web page

The WMKS SDK is available in the Downloads section of www.vmware.com : Drivers and Tools for vSphere, vCloud Director, vRealize Automation, and vCloud Air.

A minified wmks.min.js and an un-minified wmks.js are both provided in each build. The WMKS SDK uses jQuery widget to display the CM console, and therefore the jQuery related library needs to be included first. Here is an example:

1. Include jQuery and jQuery UI Javascript components in script tags, and the corresponding jQuery css file in link stylesheet tag.
2. Include wmks.js (or wmks.min.js) in a script tag, and the wmks-all.css in link stylesheet tag.
3. Provide a div with an appropriate id.
4. Create the WMKS core object with factory method WMKS.createWMKS().
5. Bind the callbacks for WMKS events.
6. Connect with the appropriate destination URL.

```
<link rel="stylesheet" type="text/css" href="wmks-all.css" />

<script type="text/javascript" src="jquery-1.8.3.min.js"></script>
<script type="text/javascript" src="jquery-ui.1.8.16.min.js"></script>
<script type="text/javascript" src="wmks.js" type="text/javascript"></script>
<script>
        var wmks = WMKS.createWMKS("wmksContainer",{})
                    .register(WMKS.CONST.Events.CONNECTION_STATE_CHANGE,
                    function(event,data){
                        if(data.state == cons.ConnectionState.CONNECTED)
                        {
                         console.log("connection state change : connected");
                        }
                    });
        wmks.connect("ws://127.0.0.1:8080");
</script >
<div  id="wmksContainer" style="position:absolute;width:100%;height:100%"></div>
```

Connecting to a Remote VM

There are two ways to connect to a remote VM

1) Direct access to a VM console, and enable access to the 'static' Console (VNC) port of a VM for a websocket connection by adding:

```
RemoteDisplay.vnc.WebSocket.enabled = "TRUE"
RemoteDisplay.vnc.WebSocket.port = "8080"
```

2) WMKS is also usable as a component in delivering a pure web-based console connection to vCenter Server-managed VMs by using an intermediary proxy to handle the authentication requirements.

In both cases, connect to the VM using the following method:

```
wmks.connect(url);
```

The URL should be in this scheme:

```
<ws | wss> :// <host:port>/ <path> /? <authentication info>
```

Disconnect and Destroy

Call this when you are done with the WMKS component. Destroying the WMKS will also result in a disconnect (if active) and remove the widget from the associated element.

The factory method of the WMKS

createWMKS()

This should be the first method when you use the WMKS SDK. Using this method generates the widget, which can display the remote screen, and then return the WMKS core object which can use all the WMKS SDK APIs to connect to a VM and perform operations.

Method	createWMKS(id, options)
Parameter1	**id**(string): the id of div element, this element would be the container of canvas which used to show remote screen.
Parameter2	**options**:(json object): create options would affect the behavior of the WMKS. Empty means "use all the default options."
Return Value	The WMKS core object, which could use all the WMKS API to connect to a VM and perform operations.
Example Call	var wmks =WMKS.createWMKS("container",{});

Create Options

Like webMKS embedded in the vSphere web client, WMKS also has multiple options that can be used to control its behavior. Here is the comparison between the SDK and webMKS embedded in the vSphere web client:

rescale

Boolean: default is true, and indicates whether to rescale the remote screen to fit the container's allocated size.

position

An enum: could be any value of WMKS.CONST.Position, default value is WMKS.CONST.Position.CENTER. It indicates which position (center or top left) the remote screen should be in the container.

changeResolution

Boolean: default is true. When the option changeResolution is true, WMKS would send the change resolution request to the connected VM, with the requested resolution (width & height) being the same as the container's allocated size.

These three options follow a specific order of procedure based on their priority.

1. Check **changeResolution**, and if true, webmks would send the change resolution request to the connected VM.
2. If the request fails, the operations **rescale** and **position** can be used to make the corresponding changes.

audioEncodeType

It's an enum: could be any value of WMKS.CONST.AudioEncodeType. It indicates which type of audio encode method is being used: vorbis, opus or aac.

useNativePixels

Boolean: default is false. Enables the use of native pixel sizes on the device. On iPhone 4+ or iPad 3+, turning this on will enable "Retina mode", which provides more screen space for the guest, making everything appear smaller.

useUnicodeKeyboardInput

Boolean: default is false. For all the user input, WMKS could send two kinds message to server: keyboard scan codes, or Unicode. Here *true* means client should use Unicode if possible.

useVNCHandshake

Boolean: default is true. Enables a standard VNC handshake. This should be used when the endpoint is using standard VNC authentication. Set to false if connecting to a proxy that authenticates through authd and does not perform a VNC handshake.

sendProperMouseWheelDeltas

Boolean: default is false. Previous versions of the library would normalize mouse wheel event deltas into one of three values: [-1, 0, 1]. When this is set to *true*, the actual deltas from the browser are sent to the server.

reverseScrollY

Boolean: default is false. If this flag set to *true*, it would send the opposite value of the mouse wheel to the connected VM.

retryConnectionInterval

Integer: default value is -1. The interval (in milliseconds) for reattempting to connect when the first attempt to set up a connection between web client and server fails. A value less than 0 means "won't retry."

ignoredRawKeyCodes

Array: default is empty. All these keycodes would be ignored and not send to the server.

fixANSIEquivalentKeys

Boolean: default is false. Enables fixing of any non-ANSI US layouts keyCodes to match ANSI US layout keyCodes equivalents. It attempts to fix any keys pressed where the client's international keyboard layout has a key that is also present on the ANSI US keyboard, but is in a different location or doesn't match the SHIFT or NO SHIFT status of an ANSI US keyboard.

keyboardLayoutId

String: default value is 'en-US' (U.S. English). For Japanese, keyboardLayoutId is 'ja-JP_106/109'. This provides different language keyboard setups for the guest. Users need to determine what the keyboard layout is, and keep the remote desktop and local desktop keyboard layout the same. In this version of the HTML console SDK, VMware does not support number pads (on Num Lock) and mobile devices.

Internet Explorer/Firefox/Chrome are supported on Windows, and Chrome/Safari on Mac are supported.

VCDProxyHandshakeVmxPath

String: default value is null. Pass to the VNC protocol when connecting. vncProtocol will respond to a connection request for a VMX path with the provided VCDProxyHandshakeVmxPath when connecting to **vCloud Director**.

enableUint8Utf8

Boolean: default value is false. If the project does not have working support for the binary protocol, the enableUint8Utf8 option is needed to enable the old uint8utf8 protocol.

Handling WMKS Events

List of WMKS Events

WebMKS generates events when the state of the currently connected virtual machine changes, or in response to messages from the currently connected virtual machine. All the WMKS events are listed in WMKS.CONST.Events.

All the event handlers have two parameters: **event** and **data**. **Event** is a jQuery event, and all the events' special parameters are stored in **data**.

connectionstatechange

The connectionstatechange event is generated in response to a change in the connection state, such as from 'disconnected' to 'connecting', from 'connecting' to 'connected'.

Parameters in **data**:

- state: could be any value in WMKS.CONST. ConnectionState can be:
  1. "connecting"
  2. "connected"
  3. "disconnected"
- If state is WMKS.CONST. ConnectionState. CONNECTING, there are two more parameters ***vvc*** and ***vvcSession*** in data.
- If state is WMKS.CONST. ConnectionState. DISCONNECTED, there are two more parameters ***reason*** and ***code*** in data.

screensizechange

The screensizechange event is generated in response to changes in the screen size of the currently connected virtual machine.
Parameters in data:

- width: the width(in pixels) of the currently connected virtual machine.
- Height: the Height (in pixels) of the currently connected virtual machine.

fullscreenchange

The fullscreenChange event is generated when the WMKS console exits or enters fullscreen mode.
Parameters in data:

- isFullScreen: Boolean, true mean enter fullscreen, false means exit fullscreen.

error

The error event is generated when an error occurs, such as authentication failure, websocket error or protocol error.

Parameters in data:

- errorType: could be any value in WMKS.CONST.Events.ERROR:
  1. AUTHENTICATION_FAILED: "authenticationfailed",
  2. WEBSOCKET_ERROR: "websocketerror",
  3. PROTOCOL_ERROR: "protocolerror"

Keyboardledschanged

The keyboardledschanged event is generated when the remote VM's keyboard LED lock state changes.

Parameters in data:

- Data is an integer: 1 means Scroll Lock, 2 means Num Lock, 4 means Caps Lock.

heartbeat

The heartbeat event is generated when a server side heartbeat message is received.

- Data is an integer that indicates the interval of the heartbeat.

audio

The audio event is generated when an audio message is received from the server.
Parameters in data:

- sampleRate
- numChannels
- containerSize
- sampleSize
- length
- audioTimestampLo
- audioTimestampHi
- frameTimestampLo
- frameTimestampHi
- flags
- data

copy

The copy event is generated when the server sends a cut text event.

- Data is the string being copied.

toggle

The toggle event is generated when show or hide keyboard, extended keyboard or trackpad.
Parameters in data:

- type: could be "KEYBOARD", "EXTENDED_KEYPAD", "TRACKPAD"
- visibility: Boolean, true mean "show", and false means "hide".

Setting Handlers for WMKS Events

In WMKS SDK, provide register() and unregister method to add and remove handlers for WMKS events.

register()

This method used to register event handler for the WMKS.

Method	register(eventName, eventHandler)
Parameter1	eventName(Constant, any value of WebMKS.Events)
Parameter2	eventHandler(javascript callback function)
Return Value	None
Example Call	var connectionStateHandler = function(event, data){}; wmks.register(WebMKS.Events.CONNECTION_STATE_CHANGE,

connectionStateHandler);

unregister()

This method is used to unregister the event handler for WMKS.
If this method take no parameters, it would remove all the event handlers.
If there is only one parameter, it would remove all the event handlers to that given eventName.

Method	unregister(eventName, eventHandler)
Parameter1	eventName(Constant, any value of WebMKS.Events)
Parameter2	eventHandler(javascript callback function)
Return Value	None
Example Call	wmks.unregister(WebMKS.Events.CONNECTION_STATE_CHANGE, connectionStateHandler);

Methods of the WMKS SDK Object

After invoking the method createWMKS(), you would get a core WMKS object which can use WMKS API to connect to a VM and perform operations. In this example, we will name the WMKS object "wmks".

General APIs

General-purpose API methods provide information about WMKS. These methods can be called at any time, including before you connect to target VM.

getVersion()

Retrieves the current version number of the WMKS SDK.

Method	getVersion()
Parameters	None
Return Value	String. Contains the full version number of WMKS SDK.
Example Call	var version = wmks.getVersion();

getConnectionState()

Retrieves the current connection state.

Method	getConnectionState()
Parameters	None
Return Value	Constant. Any value in WMKS. CONST.ConnectionState.
Example Call	var state = wmks.getConnectionState();

Lifecycle-related APIs

connect()

Connects the WMKS to a remote virtual machine by the WebSocket URL, and sets up the UI.

Method	connect()
Parameter	WebSocket URL, type is string in format: <ws | wss> :// <host:port>/ <path> /? <authentication info>
Return Value	none
Example Call	wmks.connect("ws://localhost:8080");

disconnect()

Disconnects from the remote virtual machine and tear down the UI.

Method	disconnect()
Parameter	None
Return Value	None
Example Call	wmks.disconnect();

destroy()

Terminates the connection (if active) with the VM and removes the widget from the associated element. Consumers should call this before removing the element from the DOM.

Method	disconnect()
Parameter	None
Return Value	None
Example Call	wmks.destroy();

Display-related APIs

setRemoteScreenSize()

Sends a request to set the screen resolution of the currently connected VM. Here, if the parameters width and height that are passed are larger than those of the WMKS widget allocated size, the sizing would be normalized.

Note: this method could work properly only when the option changeResultion set to *true*.

Method	setRemoteScreenSize(width,height)
Parameter1	width(int)represents the desired width of the connected vm, in pixel
Parameter2	height(int) represents the desired height of the connected vm, in pixel
Return Value	None
Example Call	wmks.setRemoteScreeSize(800,600);

getRemoteScreenSize()

Retrieves the screen width and height in pixels of currently connected VM.

Method	getRemoteScreenSize()
Parameter	None
Return Value	Object in format of {width:, height:}
Example Call	var size = wmks.getRemoteScreenSize();

updateScreen()

Changes the resolution or rescale the remote screen to match the currently allocated size.

The behavior of updateScreen depends on the option changeResolution, rescale, and position:

1) If the option changeResolution is true, it would send the change resolution request to the connected VM, the request resolution(width & height) is the same as the container's allocated size.

2) Check rescale option: if true, rescale the remote screen to fit the container's allocated size.

3) Check position option: If the remote screen's size is not the same as the container's allocated size, then the remote screen will be placed in the center or top left of the container based on its value.

Method	updateScreen()
Parameter	None
Return Value	None
Example Call	wvmrc.updateScreen();

Full-screen-related APIs

canFullScreen()

Indicates if the fullscreen feature is enabled on this browser. Fullscreen mode is disabled on Safari as it does not support keyboard input in fullscreen for security reasons.

Method	canFullScreen()
Parameter	None
Return Value	Boolean. True mean can, false not.
Example Call	wmks.canFullScreen();

isFullScreen()

Informs if the browser is in full screen mode.

Method	isFullScreen()
Parameter	None
Return Value	Boolean. True mean in full screen mode, false not.
Example Call	wmks.isFullScreen();

enterFullScreen()

Forces the browser enter full screen mode if supported. In the fullscreen mode, only the remote screen would be displayed.

Method	enterFullScreen()
Parameter	None
Return Value	None
Example Call	wmks.enterFullScreen();

exitFullScreen()

Forces the browser to exit full screen mode.

Method	exitFullScreen()
Parameter	None
Return Value	None
Example Call	wmks.exitFullScreen();

Input-related APIs

sendInputString()

Sends a string as keyboard input to the server.

Method	sendInputString(string)
Parameters	String
Return Value	None.
Example Call	wmks.sendInputString("test");

sendKeyCodes()

Sends a series of special key codes to the VM. This takes an array of special key codes and sends keydowns for each in the order listed. It then sends keyups for each in reverse order. This can be used to send key combinations such as Control-Alt-Delete.

Method	sendKeyCodes ()
Parameters	Array. Each element in the array could be a keycode or a Unicode. Keycode for none printable key, Unicode for printable character. If using Unicode, negative it, such as -118 means "v"

Return Value	None.
Example Call	wmks.sendKeyCodes([17,18,46]) ; // Ctrl + Alt + Delete

sendCAD()

Sends a Control-Alt-Delete key sequence to the currently connected virtual machine.

Method	sendCAD()
Parameters	None.
Return Value	None.
Example Call	wmks.sendCAD();

Mobile-related APIs

enableInputDevice()

Enables the input device (keyboard, extended keyboard, trackpad) on the mobile device, and initial them for use.

Method	enableInputDevice (deviceType)
Parameters	deviceType:(Constant) Any value in WMKS.CONST.InputDeviceType.
Return Value	None.
Example Call	wmks.enableInputDevice(WMKS. CONST .InputDeviceType. KEYBOARD) ;

disableInputDevice()

Disables the input device (keyboard, extended keyboard, trackpad) on the mobile device, and destroys them.

Method	disbleInputDevice (deviceType)
Parameters	deviceType:(Constant) Any value in WMKS.CONST.InputDeviceType.
Return Value	None.
Example Call	wmks.disableInputDevice(WMKS. CONST .InputDeviceType. KEYBOARD) ;

showKeyboard()

Shows the keyboard on a mobile device.

Method	showKeyboard()
Parameters	None
Return Value	None.
Example Call	wmks.showKeyboard();

hideKeyboard()

Hides the keyboard on a mobile device.

Method	hideKeyboard()
Parameters	None.
Return Value	None.
Example Call	wmks.hideKeyboard();

toggleExtendedKeypad

Shows/hides the extendedKeypad on the mobile device depend on the current visibility.

Method	toggleExtendedKeypad()
Parameters	A map, could include minToggleTime(ms) such as {minToggleTime: 50} if the user try to call this toggle method too frequency, the duration less then the minToggleTime, it would not been executed.
Return Value	None.
Example Call	wmks.toggleExtendedKeypad();

toggleTrackpad

Shows/hides the trackpad on the mobile device depend on the current visibility.

Method	toggleTrackpad()
Parameters	A map, could include minToggleTime(ms) such as {minToggleTime: 50} if the user try to call this toggle method too frequency, the duration less then the minToggleTime, it would not been executed.
Return Value	None.
Example Call	wmks.toggleTrackpad();

Option-related APIs

setOption()

Changes the option value. Only options listed below can use this method:

- rescale
- position
- changeResolution
- useNativePixels
- reverseScrollY
- fixANSIEquivalentKeys
- sendProperMouseWheelDeltas
- keyboardLayoutId
- VCDProxyHandshakeVmxPath
- enableUint8Utf8

Method	setOption(optionName, optionValue)
Parameter1	optionName(string of the option name)
Parameter2	optionValue
Return Value	None
Example Call	wmks.setOption("changeResolution",false);

Appendix

Constants used in WMKS

- **Position**: WMKS would display the remote screen of VM in equal proportion. Therefore after rescale, the remote screen size maybe still not the same with the container's size. Here ***Position*** provides two possible options for put the screen at which position of the container.

- **ConnectionState**: There are 3 possible connection states when trying to connect to the remote VM.

- **Events**: All the event names that WMKS can trigger are listed here.

- **ErrorType**: possible error types in the lifecycle of WMKS.

- **InputDeviceType**: WMKS SDK supports viewing VM consoles on mobile devices, and this field lists the possible input devices.

(Please see next page for list of constants)

```
Position: {
   CENTER:   0,
   LEFT_TOP: 1
},

ConnectionState: {
   CONNECTING: "connecting",
   CONNECTED: "connected",
   DISCONNECTED: "disconnected"
},

Events: {
   CONNECTION_STATE_CHANGE: "connectionstatechange",
   REMOTE_SCREEN_SIZE_CHANGE: "screensizechange",
   FULL_SCREEN_CHANGE: "fullscreenchange",
   ERROR: "error",
   KEYBOARD_LEDS_CHANGE: "keyboardledschanged",
   HEARTBEAT: "heartbeat",
   AUDIO: "audio",
   COPY: "copy",
   TOGGLE: "toggle"
},

ErrorType: {
   AUTHENTICATION_FAILED: "authenticationfailed",
   WEBSOCKET_ERROR: "websocketerror",
   PROTOCOL_ERROR: "protocolerror"
},

AudioEncodeType: {
   VORBIS: "vorbis",
   OPUS: "opus",
   AAC: "aac"
},

InputDeviceType: {
   KEYBOARD: 0,
   EXTENDED_KEYBOARD: 1,
   TRACKPAD: 2
}
```