VMware HTML Console SDK プログラミング ガイド

vSphere 5.5 以降（一般公開）
および vCloud Director（技術プレビュー）用

2016 年 2 月

VMware HTML Console SDK プログラミング ガイド

目次

HTML Console (WMKS) SDK の概要

HTML Console SDK は、webmks (WMKS) に基づいて実装された JavaScript ライブラリです。WMKS は、マウス、キーボード、タッチ入力の処理、ブラウザからのデスクトップ セッションに対する画面更新やカーソル変更を可能にします。ブラウザで実行されるアプリケーションは、HTML Console SDK API を使用することによって、JavaScript で仮想マシン コンソールの機能にアクセスできます。

HTML Console SDK API には、仮想マシンに接続して通信するために使用できるさまざまなメソッドが含まれています。また、イベントをトリガして、ユーザーに仮想マシンの状態の変更を通知します。これらのメソッドとコールバックを使用して、適切な Web ブラウザとオペレーティング システムを備えているあらゆるシステムから仮想マシンをリモートで管理する機能をエンド ユーザーに提供できます。

サポートされているブラウザ（iOS、Android を含む）

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

HTML Console SDK API を使用する手順

Web ベースのアプリケーションで HTML Console SDK API を使用するには、通常は次のことを行う必要があります。

1. HTML Console SDK JavaScript ライブラリ ファイルをロードします。
2. ファクトリ メソッド createWMKS() を使用し、ドキュメント DOM 内の特定の div 要素 ID と作成オプションを指定して、WMKS コア オブジェクトを作成します。
3. WMKS でトリガされたイベントに応答するための JavaScript コールバックを登録します。
4. HTML Console SDK の connect() メソッドを使用して、ターゲット仮想マシンに接続します。
5. HTML Console SDK のメソッドを使用して、接続した仮想マシンと通信します。
6. HTML Console SDK の disconnect() メソッドを使用して接続を切断し（アクティブな場合）、ウィジェットを関連付けられている要素から削除します。

Web ページにコンソールを配置する

HTML Console SDK は、www.vmware.com/jp/ の vSphere、vCloud Director、vRealize Automation、および vCloud Air のドライバとツールのダウンロード セクションから入手できます。

各ビルドで、圧縮されている wmks.min.js と圧縮されていない wmks.js の両方が提供されています。HTML Console SDK は、jQuery ウィジェットを使用して仮想マシン コンソールを表示するため、jQuery 関連のライブラリを最初にインクルードする必要があります。例を次に示します。

1. jQuery および jQuery UI Javascript コンポーネントを script タグでインクルードし、対応する jQuery css ファイルを link stylesheet タグでインクルードします。
2. wmks.js（または wmks.min.js）を script タグでインクルードし、wmks-all.css を link stylesheet タグでインクルードします。
3. 適切な id の div を用意します。
4. ファクトリ メソッド WMKS.createWMKS() を使用して WMKS コア オブジェクトを作成します。
5. WMKS イベント用のコールバックをバインドします。
6. 適切な接続先 URL を使用して接続します。

```
 <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>
```

リモート仮想マシンに接続する

リモート仮想マシンに接続する方法は次のとおりです。

WMKS は、中間プロキシを使用して認証要求を処理することで、vCenter Server で管理されている仮想マシンへの純粋な Web ベースのコンソール接続を提供する場合のコンポーネントとして使用できます。

次のメソッドを使用して、仮想マシンに接続します。

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

URL は次のスキームに従っている必要があります。

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

切断と破棄

WMKS コンポーネントの作業が完了したら、これを呼び出します。WMKS を破棄すると切断も行われ（アクティブな場合）、ウィジェットが関連付けられている要素から削除されます。

WMKS のファクトリ メソッド

createWMKS()

これは、HTML Console SDK を使用するときの最初のメソッドである必要があります。このメソッドを使用すると、リモート画面を表示可能なウィジェットが生成され、WMKS コア オブジェクトが返されます。このオブジェクトによってすべての HTML Console SDK API を使用できるようになり、仮想マシンに接続したり、操作を実行できます。

メソッド	createWMKS(id, options)
パラメータ 1	**id**（文字列）：div 要素の id。この要素が、リモート画面を表示するために使用されるキャンバスのコンテナになります。
パラメータ 2	**options**（json オブジェクト）：WMKS の動作に影響を与える作成オプションです。空は、「すべてデフォルト オプションを使用する」を意味します。
戻り値	WMKS コア オブジェクト。このオブジェクトによってすべての WMKS API を使用できるようになり、仮想マシンに接続したり、操作を実行できます。
呼び出しの例	var wmks =WMKS.createWMKS(“container”,{});

作成オプション

vSphere Web クライアントに組み込まれている webMKS のように、WMKS には、その動作を制御するために使用できるいくつかのオプションがあります。ここでは、この SDK と vSphere Web クライアントに組み込まれている webMKS を比較します。

rescale

真偽値：デフォルトは true であり、コンテナの割り当てサイズに適合するようにリモート画面を再スケーリングするかどうかを示します。

position

列挙値：WMKS.CONST.Position のいずれかの値を指定できます。デフォルト値は WMKS.CONST.Position.CENTER です。これは、コンテナ内でリモート画面をどの位置（中心または左上）に表示するかを示します。

changeResolution

真偽値：デフォルトは true です。この changeResolution オプションが true の場合、WMKS は接続先の仮想マシンに解像度変更要求を送信します。このとき、要求される解像度（幅と高さ）は、コンテナの割り当てサイズと同じになります。

これらの 3 つのオプションは、優先順位に基づいて特定の処理順序に従います。

1. **changeResolution** をチェックし、true の場合、webmks は接続先の仮想マシンに解像度変更要求を送信します。
2. この要求が失敗した場合、オプション **rescale** および **position** を使用して、対応する変更を行うことができます。

audioEncodeType

列挙値であり、WMKS.CONST.AudioEncodeType のいずれかの値を指定できます。これは、使用されている audio エンコード メソッドのタイプ（vorbis、opus、または aac）を示します。

useNativePixels

真偽値：デフォルトは false です。デバイスのネイティブのピクセル サイズを使用できるようにします。iPhone 4 以降または iPad 3 以降でこれをオンにすると、「Retina モード」が有効になります。このモードでは、より広い表示スペースをゲストに提供し、すべてがより小さく見えるようになります。

useUnicodeKeyboardInput

真偽値：デフォルトは false です。すべてのユーザー入力に対して、WMKS は 2 種類のメッセージ（キーボード スキャン コードまたは Unicode）をサーバに送信できます。**true** は、クライアントが可能な限り Unicode を使用することを意味します。

useVNCHandshake
真偽値：デフォルトは true です。標準の VNC ハンドシェイクを有効にします。エンドポイントで標準の VNC 認証を使用している場合、これを使用する必要があります。authd を通じて認証するプロキシに接続しようとしており、VNC ハンドシェイクを実行しない場合は、false に設定してください。

sendProperMouseWheelDeltas
真偽値：デフォルトは false です。以前のバージョンのライブラリでは、マウス ホイール イベントの差分値は [-1, 0, 1] の 3 つの値のいずれかに正規化されていました。このオプションを **true** に設定すると、ブラウザからの実際の差分値がサーバに送信されます。

reverseScrollY
真偽値：デフォルトは false です。このフラグを **true** に設定すると、接続先の仮想マシンにマウス ホイールの反対の値を送信します。

retryConnectionInterval
整数値：デフォルト値は -1 です。Web クライアントとサーバの間の接続を確立する最初の試行が失敗した場合に、接続を再試行するまでの間隔です（単位：ミリ秒）。0 よりも小さい値は、「再試行しない」を意味します。

ignoredRawKeyCodes
配列：デフォルトは空です。ここに指定されたすべてのキーコードは無視され、サーバに送信されません。

fixANSIEquivalentKeys
真偽値：デフォルトは false です。ANSI US 以外のレイアウトのキーコードを修正して ANSI US レイアウトの等価のキーコードに一致させる処理を有効にします。これが有効になると、クライアントの国際キーボード レイアウト上のキーが ANSI US キーボードにもあるものの、それらの場所が異なっている場合や、それらの Shift キー押下の有無の状態が一致していない場合に、押されたキーの修正を試みます。

keyboardLayoutId
文字列：デフォルト値は「en-US」（US 英語）です。これは、ゲスト用にさまざまな言語のキーボードを設定する方法を提供します。ユーザーは、キーボード レイアウトの種類を決定し、リモート デスクトップとローカル デスクトップのキーボード レイアウトが同じになるようにする必要があります。このバージョンの HTML Console SDK では、VMware はモバイル デバイスをサポートしていません。ゲスト OS に送信されるコードは 2 種類あります。国際マッピングに対応しているほうが vScancode です。

Windows では Internet Explorer/Firefox/Chrome がサポートされており、Mac では Chrome/Safari がサポートされています。
ユーザーが使用する言語を選択するために、html で select リストを追加するか、他の何らかの方法を使用してください。js ファイルでは、次のように setOption API を使用してください。

```
<select id="selectLanguage">
        <option value="en-US">English</option>
        <option value="ja-JP_106/109">Japanese</option>
        <option value="de-DE">German</option>
        <option value="it-IT">Italian</option>
        <option value="es-ES">Spanish</option>
        <option value="pt-PT">Portuguese</option>
</select>
```

```
$('#selectLanguage').change(function(){
   if(!wmks) return;
   var keyboardLayoutId = $(this).find(":selected").val();
   wmks.setOption('keyboardLayoutId',keyboardLayoutId);
 });
```

VCDProxyHandshakeVmxPath

文字列：デフォルト値は null です。接続するときに VNC プロトコルに渡します。**vCloud Director** に接続するときに、vncProtocol は指定された VCDProxyHandshakeVmxPath を使用して、VMX パスへの接続リクエストに応答します。

enableUint8Utf8

真偽値：デフォルト値は false です。プロジェクトで正しく動作するバイナリ プロトコルがサポートされていない場合、古い uint8utf8 プロトコルを有効にするは、enableUint8Utf8 オプションが必要です。

WMKS のイベントの処理

WMKS のイベントのリスト

WebMKS は、現在接続されている仮想マシンの状態が変化したとき、または現在接続されている仮想マシンからのメッセージに応答して、イベントを生成します。WMKS のすべてのイベントは、WMKS.CONST.Events にリストされています。

すべてのイベント ハンドラに、**event** と **data** の 2 つのパラメータがあります。**event** は jQuery のイベントであり、イベントのすべての特殊なパラメータは **data** に格納されます。

connectionstatechange

connectionstatechange イベントは、「切断」から「接続中」、「接続中」から「接続済み」のような、接続状態の変化に応答して生成されます。

data 内のパラメータ：

- state：WMKS.CONST の任意の値を取ります。ConnectionState は次の値を取ります。
  1. "connecting"
  2. "connected"
  3. "disconnected"
- 状態が WMKS.CONST.ConnectionState.CONNECTING の場合、data にはさらに 2 つのパラメータ **vvc**、**vvcSession** があります。
- 状態が WMKS.CONST.ConnectionState.DISCONNECTED の場合、data にはさらに 2 つのパラメータ **reason**、**code** があります。

screensizechange

screensizechange イベントは、現在接続されている仮想マシンの画面サイズの変化に応答して生成されます。

data 内のパラメータ：

- width：現在接続されている仮想マシンの幅（ピクセル単位）。
- Height：現在接続されている仮想マシンの高さ（ピクセル単位）。

fullscreenchange

fullscreenChange イベントは、WMKS コンソールが全画面モードに切り替わったときや、全画面モードが終了したときに生成されます。

data 内のパラメータ：

- isFullScreen：真偽値。true は全画面への切り替えを意味し、false は全画面の終了を意味します。

error

error イベントは、認証失敗、websocket エラー、プロトコル エラーなどのエラーが発生したときに生成されます。

data 内のパラメータ：

- errorType：以下の WMKS.CONST.Events.ERROR 内のいずれかの値を取ります。
  1. AUTHENTICATION_FAILED："authenticationfailed"
  2. WEBSOCKET_ERROR："websocketerror"
  3. PROTOCOL_ERROR："protocolerror"

keyboardledschanged

keyboardledschanged イベントは、リモート仮想マシンのキーボード LED ロック状態が変化したときに生成されます。

data 内のパラメータ：

- data は整数値であり、1 は Scroll Lock、2 は Num Lock、4 は Caps Lock を意味します。

heartbeat

heartbeat イベントは、サーバ側のハートビート メッセージを受信したときに生成されます。

- data は、ハートビートの間隔を示す整数値です。

audio

audio イベントは、サーバからオーディオ メッセージを受信したときに生成されます。
data 内のパラメータ：

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

copy

copy イベントは、サーバがテキスト切り取りイベントを送信したときに生成されます。

- data は、コピーされる文字列です。

toggle

toggle イベントは、キーボード、拡張キーボード、またはトラックパッドの表示/非表示が切り替えられたときに生成されます。
data 内のパラメータ：

- type："KEYBOARD"、"EXTENDED_KEYPAD"、"TRACKPAD" のいずれかの値です。
- visibility：真偽値。true は「表示」を意味し、false は「非表示」を意味します。

WMKS のイベントのハンドラを設定する

HTML Console SDK では、WMKS のイベントのハンドラを追加および削除するための register() メソッドと unregister() メソッドを用意しています。

register()

このメソッドは、WMKS のイベント ハンドラを登録するために使用します。

メソッド	register(eventName, eventHandler)
パラメータ 1	eventName（定数。WebMKS.Events のいずれかの値）
パラメータ 2	eventHandler（javascript コールバック関数）
戻り値	なし
呼び出しの例	var connectionStateHandler = function(event, data){}; wmks.register(WebMKS.Events.CONNECTION_STATE_CHANGE, connectionStateHandler);

unregister()

このメソッドは、WMKS のイベント ハンドラの登録を解除するために使用します。このメソッドにパラメータを指定しなかった場合、すべてのイベント ハンドラが削除されます。

パラメータを 1 つのみ指定した場合、その eventName に対するすべてのイベント ハンドラが削除されます。

メソッド	unregister(eventName, eventHandler)
パラメータ 1	eventName（定数。WebMKS.Events のいずれかの値）
パラメータ 2	eventHandler（javascript コールバック関数）
戻り値	なし
呼び出しの例	wmks.unregister(WebMKS.Events.CONNECTION_STATE_CHANGE, connectionStateHandler);

HTML Console SDK オブジェクトのメソッド

createWMKS() メソッドを呼び出した後、コア WMKS オブジェクトを取得します。このオブジェクトによって WMKS API を使用することができ、仮想マシンに接続したり、操作を実行できます。ここの例では、WMKS オブジェクトに「wmks」という名前を付けています。

全般的な API

全般的な目的の API メソッドは、WMKS についての情報を提供します。これらのメソッドは、ターゲット仮想マシンに接続する前も含め、いつでも呼び出すことができます。

getVersion()

HTML Console SDK の現在のバージョン番号を取得します。

メソッド	getVersion()
パラメータ	なし
戻り値	文字列。HTML Console SDK の完全なバージョン番号を含んでいます。
呼び出しの例	var version = wmks.getVersion();

getConnectionState()

現在の接続状態を取得します。

メソッド	getConnectionState()
パラメータ	なし
戻り値	定数。WMKS. CONST.ConnectionState のいずれかの値。
呼び出しの例	var state = wmks.getConnectionState();

ライフサイクル関連の API

connect()

WebSocket URL を使用して WMKS をリモート仮想マシンに接続し、ユーザー インターフェイスをセットアップします。

メソッド	connect()
パラメータ	WebSocket URL。次の形式の文字列です。 <ws | wss> ://<host:port>/ <path> /?<authentication info>
戻り値	なし
呼び出しの例	wmks.connect("ws://localhost:8080");

disconnect()

リモート仮想マシンから切断し、ユーザー インターフェイスを破棄します。

メソッド	disconnect()
パラメータ	なし
戻り値	なし
呼び出しの例	wmks.disconnect();

destroy()
仮想マシンとの接続を終了し（アクティブな場合）、ウィジェットを関連付けられている要素から削除します。利用者は、要素を DOM から削除する前にこれを呼び出す必要があります。

メソッド	destroy()
パラメータ	なし
戻り値	なし
呼び出しの例	wmks.destroy();

表示関連の API

setRemoteScreenSize()
現在接続されている仮想マシンの画面解像度を設定する要求を送信します。渡されたパラメータ width と height が WMKS ウィジェットの割り当てサイズよりも大きい場合、サイズは正規化されます。

注：このメソッドは、オプション changeResolution が **true** に設定されている場合のみ適切に機能します。

メソッド	setRemoteScreenSize(width,height)
パラメータ 1	width（整数）は、接続されている仮想マシンの目的の幅をピクセル単位で指定します。
パラメータ 2	height（整数）は、接続されている仮想マシンの目的の高さをピクセル単位で指定します。
戻り値	なし
呼び出しの例	wmks.setRemoteScreeSize(800,600);

getRemoteScreenSize()
現在接続されている仮想マシンの画面の幅と高さ（ピクセル単位）を取得します。

メソッド	getRemoteScreenSize()
パラメータ	なし
戻り値	{width:, height:} の形式のオブジェクト
呼び出しの例	var size = wmks.getRemoteScreenSize();

updateScreen()
現在割り当てられているサイズに一致するように解像度を変更するか、リモート画面を再スケーリングします。

updateScreen の動作は、オプション changeResolution、rescale、position に応じて次のように異なります。

1) changeResolution オプションが true の場合、接続先の仮想マシンに解像度変更要求を送信します。要求される解像度（幅と高さ）は、コンテナの割り当てサイズと同じになります。

2) rescale オプションをチェックし、これが true の場合、コンテナの割り当てサイズに適合するようにリモート画面を再スケーリングします。

3) position オプションをチェックします。リモート画面のサイズがコンテナの割り当てサイズと同じではない場合、リモート画面はこの値に基づいてコンテナの中心または左上に配置されます。

メソッド	updateScreen()
パラメータ	なし
戻り値	なし
呼び出しの例	wvmrc.updateScreen();

全画面関連の API

canFullScreen()

このブラウザ上で全画面機能が有効になっているかどうかを示します。Safari では、セキュリティ上の理由から全画面でのキーボード入力がサポートされていないため、全画面モードは無効になります。

メソッド	canFullScreen()
パラメータ	なし
戻り値	真偽値。true は「可」を意味し、false は「不可」を意味します。
呼び出しの例	wmks.canFullScreen();

isFullScreen()

ブラウザが全画面モードになっているかどうかを知らせます。

メソッド	isFullScreen()
パラメータ	なし
戻り値	真偽値。true は全画面モードであることを意味し、false は全画面モードでないことを意味します。
呼び出しの例	wmks.isFullScreen();

enterFullScreen()

サポートされている場合、ブラウザは強制的に全画面モードになります。全画面モードでは、リモート画面のみが表示されます。

メソッド	enterFullScreen()
パラメータ	なし
戻り値	なし
呼び出しの例	wmks.enterFullScreen();

exitFullScreen()

ブラウザの全画面モードは強制的に終了されます。

メソッド	exitFullScreen()
パラメータ	なし
戻り値	なし
呼び出しの例	wmks.exitFullScreen();

入力関連の API

sendInputString()

キーボード入力文字列をサーバに送信します。

メソッド	sendInputString(string)
パラメータ	文字列
戻り値	なし
呼び出しの例	wmks.sendInputString("test");

sendKeyCodes()

一連の特殊なキー コードを仮想マシンに送信します。これは、特殊なキー コードの配列を使用し、リストされている順序でキーダウンを送信します。次に、逆の順序でキーアップを送信します。これは、Ctrl + Alt + Delete のようなキーの組み合わせを送信するために使用できます。

メソッド	sendKeyCodes ()
パラメータ	配列。配列の各要素はキーコードまたは Unicode にすることができます。印字できないキーにはキーコードを使用し、印字できる文字には Unicode を使用します。Unicode を使用する場合は負にします。たとえば、-118 は「v」を意味します。
戻り値	なし
呼び出しの例	wmks.sendKeyCodes([17,18,46]) ; // Ctrl + Alt + Delete

sendCAD()

Ctrl + Alt + Delete キー シーケンスを現在接続されている仮想マシンに送信します。

メソッド	sendCAD()
パラメータ	なし
戻り値	なし
呼び出しの例	wmks.sendCAD();

モバイル関連の API

enableInputDevice()

モバイル デバイス上の入力デバイス（キーボード、拡張キーボード、トラックパッド）を有効にし、使用できるように初期化します。

メソッド	enableInputDevice (deviceType)
パラメータ	deviceType：（定数）WMKS.CONST.InputDeviceType のいずれかの値。
戻り値	なし
呼び出しの例	wmks.enableInputDevice(WMKS.CONST .InputDeviceType. KEYBOARD) ;

disableInputDevice()

モバイル デバイス上の入力デバイス（キーボード、拡張キーボード、トラックパッド）を無効にし、破棄します。

メソッド	disableInputDevice (deviceType)
パラメータ	deviceType：（定数）WMKS.CONST.InputDeviceType のいずれかの値。
戻り値	なし
呼び出しの例	wmks.disableInputDevice(WMKS.CONST .InputDeviceType. KEYBOARD) ;

showKeyboard()

モバイル デバイス上のキーボードを表示します。

メソッド	showKeyboard()
パラメータ	なし
戻り値	なし
呼び出しの例	wmks.showKeyboard();

hideKeyboard()
モバイル デバイス上のキーボードを非表示にします。

メソッド	hideKeyboard()
パラメータ	なし
戻り値	なし
呼び出しの例	wmks.hideKeyboard();

toggleExtendedKeypad()
モバイル デバイス上の拡張キーパッドの表示/非表示を、現在の表示状態に応じて切り替えます。

メソッド	toggleExtendedKeypad()
パラメータ	マップ。minToggleTime（ミリ秒）を指定できます。例：{minToggleTime: 50}。ユーザーがこの切り替えメソッドを非常に頻繁に、つまり minToggleTime よりも短い間隔で呼び出そうとした場合、何も実行されません。
戻り値	なし
呼び出しの例	wmks.toggleExtendedKeypad();

toggleTrackpad()
トラックパッド上の拡張キーパッドの表示/非表示を、現在の表示状態に応じて切り替えます。

メソッド	toggleTrackpad()
パラメータ	マップ。minToggleTime（ミリ秒）を指定できます。例：{minToggleTime: 50}。ユーザーがこの切り替えメソッドを非常に頻繁に、つまり minToggleTime よりも短い間隔で呼び出そうとした場合、何も実行されません。
戻り値	なし
呼び出しの例	wmks.toggleTrackpad();

toggleRelativePad ()
画面上の相対マウス トラック パッドの表示/非表示を切り替えます。この相対パッドを開いた後は、絶対マウス イベントではなく相対マウス イベントが送信されます。リモート ゲスト OS に VMware Tools がインストールされていない場合に使用できます。ユーザー インターフェイスに toggleRelativeMouse ボタンを追加し、toggleRelativePad() API にバインドしてください。そのボタンがクリックされると、相対トラック パッドが表示され、相対マウス イベントがリモート ゲストに送信されます。これは、ゲスト OS に VMware Tools がインストールされておらず、絶対マウス イベントを受け付けることができない場合に便利です。サーバからカーソル イメージが戻らない場合、ローカルのカーソル画像を表示する必要があります。したがって、css フォルダ内の wmks-all.css と img フォルダを製品に追加する必要があります。これらはすべて、wmkssdk フォルダの下にあります。

メソッド	toggleRelativePad ()
パラメータ	マップ。minToggleTime（ミリ秒）を指定できます。例：{minToggleTime: 50}。ユーザーがこの切り替えメソッドを非常に頻繁に、つまり minToggleTime よりも短い間隔で呼び出そうとした場合、何も実行されません。
戻り値	なし
呼び出しの例	wmks. toggleRelativePad ();

オプション関連の API

setOption()

オプションの値を変更します。このメソッドは、以下のオプションのみで使用できます。

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

メソッド	setOption(optionName, optionValue)
パラメータ 1	optionName（オプション名の文字列）
パラメータ 2	optionValue
戻り値	なし
呼び出しの例	wmks.setOption(“changeResolution”,false);

付録

WMKS で使用される定数

- **Position**：WMKS は仮想マシンのリモート画面を等しい比率で表示します。したがって、再スケーリングの後、リモート画面のサイズがコンテナのサイズと同じにならない場合があります。ここで、**Position** は、画面をコンテナのどの位置に置くかについて 2 つのオプションを提供しています。

- **ConnectionState**：リモート仮想マシンに接続しようとするとき、取りうる接続状態は 3 つあります。

- **Events**：WMKS によってトリガされる可能性のあるイベント名がすべてここにリストされます。

- **ErrorType**：WMKS のライフサイクルの中で発生する可能性のあるエラーのタイプです。

- **InputDeviceType**：HTML Console SDK により、モバイル デバイス上に仮想マシン コンソールを表示する機能がサポートされます。このフィールドに、使用可能な入力デバイスがリストされます。

(定数のリストについては、次のページを参照してください)

```
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
}
```