VMware HTML Console SDK Guia de programação

Para o vSphere 5.5 e versões posteriores (disponibilidade geral)
e para o vCloud Director (pré-visualização técnica)

Fevereiro de 2016

VMware HTML Console SDK
Guia de programação

Índice

Descrição geral do SDK HTML Console SDK (WMKS)

O HTML Console SDK consiste numa biblioteca de JavaScript implementada com base no webmks (WMKS) que permite o processamento e controlo de introduções por rato, teclado e toque, bem como atualizações do ecrã e alterações do cursor numa sessão de ambiente de trabalho a partir de um browser. As aplicações executadas no browser podem utilizar o JavaScript para aceder às funções da consola da máquina virtual utilizando a API do HTML Console SDK.

A API do HTML Console SDK inclui vários métodos que podem ser utilizados para ligar e comunicar com uma máquina virtual. Também aciona eventos para notificar os utilizadores das alterações no estado da máquina virtual. Pode utilizar estes métodos e as chamadas de retorno para permitir que os utilizadores finais procedam à gestão remota de uma máquina virtual a partir de qualquer sistema com o browser e o sistema operativo adequados.

Browsers suportados (incluindo em iOS e Android)

- Internet Explorer 10 ou superior
- Firefox 24 ou superior
- Chrome 30 ou superior
- Safari 6.1 ou superior

Passos para utilizar a API do HTML Console SDK

Geralmente, para utilizar a API do HTML Console SDK, é necessário que a aplicação baseada na Web faça o seguinte:

1. Carregue os ficheiros da biblioteca de JavaScript do HTML Console SDK.
2. Crie o objeto central do WMKS utilizando o método de fábrica createWMKS() com a ID de um dado elemento div no documento DOM e as opções de criação.
3. Registe chamadas de retorno de JavaScript para responder a eventos acionados pelo WMKS.
4. Utilize o método connect() do HTML Console SDK para estabelecer ligação a uma máquina virtual de destino.
5. Utilize os métodos do HTML Console SDK para interagir com a máquina virtual ligada.
6. Utilize o método disconnect() do HTML Console SDK para interromper a ligação (se estiver ativa) e remover o widget do elemento associado.

Colocar a consola numa página Web

O HTML Console SDK está disponível na secção Downloads (Transferências) em www.vmware.com: Controladores e ferramentas para o vSphere, o vCloud Director, o vRealize Automation e o vCloud Air.

Estão disponíveis um ficheiro wmks.min.js reduzido e um ficheiro wmks.js não reduzido em cada compilação. O HTML Console SDK utiliza o widget jQuery para apresentar a consola da VM e, por isso, é necessário incluir a biblioteca relacionada de jQuery primeiro. Por exemplo:

1. Inclua os componentes de Javascript jQuery e jQuery UI nas tags script e o ficheiro css jQuery correspondente na tag link stylesheet.
2. Inclua o wmks.js (ou o wmks.min.js) numa tag script e o wmks-all.css na tag link stylesheet.
3. Forneça um div com uma ID adequada.
4. Crie o objeto central do WMKS utilizando o método de fábrica WMKS.createWMKS().
5. Associe as chamadas de retorno dos eventos do WMKS.
6. Estabeleça ligação ao URL de destino adequado.

```
<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>
```

Estabelecer ligação a uma VM remota

Formas de estabelecer ligação a uma VM remota:

É possível utilizar o WMKS como um componente para fornecer uma ligação perfeita baseada na Web da consola às VMs geridas pelo vCenter Server, utilizando um proxy como intermediário para lidar com os requisitos de autenticação.

Estabeleça ligação à VM através do seguinte método:

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

O URL deve estar no seguinte esquema:

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

Desligar e destruir

Siga este procedimento quando já não precisar do componente WMKS. Destruir o WMKS resulta também na interrupção da ligação (se estiver ativa) e remove o widget do elemento associado.

O método de fábrica do WMKS

createWMKS()

Este deve ser o método principal quando utiliza o HTML Console SDK. A utilização deste método gera o widget, que pode apresentar o ecrã remoto e, em seguida, devolver o objeto central do WMKS que pode utilizar todas as APIs do HTML Console SDK para estabelecer ligação a uma VM e realizar operações.

Método	createWMKS(id, options)
Parâmetro 1	**id**(string): a ID do elemento div; este elemento é o contentor da tela que apresentava o ecrã remoto.
Parâmetro 2	**options**:(json object): as opções de criação afetam o comportamento do WMKS. Vazio significa "utilizar todas as opções predefinidas".
Valor devolvido	O objeto central do WMKS, que pode utilizar todas as APIs do WMKS para estabelecer ligação a uma VM e realizar operações.
Chamada de exemplo	var wmks =WMKS.createWMKS(“container”,{});

Opções de criação

Tal como o webMKS incorporado no cliente Web vSphere, também o WMKS tem várias opções que podem ser utilizadas para controlar o respetivo comportamento. Segue-se uma comparação entre o SDK e o webMKS incorporado no cliente Web vSphere:

rescale

Boleano: a predefinição é *true* e indica se o ecrã remoto deve ser redimensionado de modo a ajustar-se ao tamanho atribuído do contentor.

position

Uma enum: pode ser qualquer valor de WMKS.CONST.Position; o valor predefinido é WMKS.CONST.Position.CENTER. Indica a posição (centro ou superior esquerda) que o ecrã remoto deve ocupar no contentor.

changeResolution

Boleano: a predefinição é *true*. Quando a opção changeResolution está definida para *true*, o WMKS envia o pedido de alteração da resolução à VM ligada, sendo que a resolução solicitada (largura e altura) é equivalente ao tamanho atribuído do contentor.

Estas três opções respeitam uma ordem de processo específica com base na respetiva prioridade.

1. Verifique **changeResolution**. Se estiver definido para *true*, o webmks envia o pedido de alteração da resolução à VM ligada.
2. Se o pedido falhar, é possível utilizar as operações **rescale** e **position** para fazer as alterações correspondentes.

audioEncodeType

É uma enum: pode ser qualquer valor de WMKS.CONST.AudioEncodeType. Indica o tipo de método de codificação de áudio utilizado: vorbis, opus ou aac.

useNativePixels

Boleano: a predefinição é *false*. Permite utilizar os tamanhos de píxeis nativos no dispositivo. No iPhone 4 (ou superior) ou no iPad 3 (ou superior), a ativação desta opção ativa o "modo retina", que fornece mais espaço no ecrã ao convidado, fazendo com que tudo pareça mais pequeno.

useUnicodeKeyboardInput

Boleano: a predefinição é *false*. O WMKS pode enviar dois tipos de mensagem para o servidor para todas as introduções do utilizador: códigos de tecla premida ou Unicode. Neste caso, *true* significa que o cliente deve utilizar Unicode, se possível.

useVNCHandshake

Boleano: a predefinição é *true*. Ativa um procedimento de handshake VNC padrão. Esta opção deve ser utilizada quando o ponto final está a utilizar uma autenticação VNC padrão. Definido para *false* se estabelecer ligação a um proxy autenticado por authd e não realiza um procedimento de handshake VNC.

sendProperMouseWheelDeltas

Boleano: a predefinição é *false*. As versões anteriores da biblioteca normalizavam os deltas dos eventos da roda do rato para um de três valores: [-1, 0, 1]. Quando esta opção está definida para *true*, os deltas reais do browser são enviados para o servidor.

reverseScrollY

Boleano: a predefinição é *false*. Se este sinalizador estiver definido para *true*, envia o valor oposto da roda do rato para a VM ligada.

retryConnectionInterval

Número inteiro: o valor predefinido é -1. O intervalo (em milissegundos) para voltar a tentar estabelecer ligação quando a primeira tentativa de configuração de uma ligação entre o cliente Web e o servidor falha. Um valor inferior a 0 significa "não voltará a tentar".

ignoredRawKeyCodes

Matriz: a predefinição é vazio. Todos estes códigos de teclas são ignorados e não são enviados para o servidor.

fixANSIEquivalentKeys

Boleano: a predefinição é *false*. Permite corrigir quaisquer códigos de teclas de esquemas não ANSI US de modo a corresponderem aos códigos de teclas de esquemas ANSI US equivalentes. Tenta corrigir quaisquer teclas premidas quando o esquema de teclado internacional do cliente tem uma tecla que também existe no teclado ANSI US, mas está noutra posição ou não corresponde ao estado de ativação da tecla Shift de um teclado ANSI US.

keyboardLayoutId

Cadeia: o valor predefinido é "en-US" (Inglês dos E.U.A.). Esta opção disponibiliza ao convidado diferentes configurações de idioma para o teclado. Os utilizadores têm de determinar qual é o esquema do teclado e utilizar o mesmo para o ambiente de trabalho remoto e para o ambiente de trabalho local. Nesta versão do HTML Console SDK, o VMware não suporta dispositivos móveis. São enviados dois tipos de códigos para o SO convidado. Aquele que suportamos para o mapeamento internacional é o vScancode.

O Internet Explorer, o Firefox e o Chrome são suportados no Windows, e o Chrome e o Safari são suportados no Mac.

Adicione uma lista de seleção em html ou utilize outras formas de os utilizadores escolherem o idioma utilizado. No ficheiro js, utilize a API setOption da seguinte forma:

```
<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

Cadeia: o valor predefinido é *null*. Passe para o protocolo VNC ao estabelecer a ligação. O vncProtocol responde a um pedido de ligação para um caminho VMX com o VCDProxyHandshakeVmxPath fornecido ao ligar ao **vCloud Director**.

enableUint8Utf8

Boleano: o valor predefinido é *false*. Se o projeto não tiver suporte ativo para o protocolo binário, é necessária a opção enableUint8Utf8 para ativar o protocolo uint8utf8 antigo.

Processar eventos do WMKS

Lista de eventos do WMKS

O WebMKS gera eventos quando o estado da máquina virtual atualmente ligada se altera ou em resposta às mensagens da máquina virtual atualmente ligada. Todos os eventos do WMKS se encontram enumerados em WMKS.CONST.Events.

Todos os processadores de eventos têm dois parâmetros: **event** e **data**. **Event** consiste num evento jQuery e todos os parâmetros especiais dos eventos são guardados em **data**.

connectionstatechange

O evento connectionstatechange é gerado em resposta a uma alteração do estado da ligação, por exemplo, de "desligado" para "a ligar", de "a ligar" para "ligado".

Parâmetros de **data**:

- state: pode ser qualquer valor de WMKS.CONST. O ConnectionState pode ser:
  1. "a ligar"
  2. "ligado"
  3. "desligado"
- Se o estado for WMKS.CONST. ConnectionState. A LIGAR, existem mais dois parâmetros ***vvc*** e ***vvcSession*** em data.
- Se o estado for WMKS.CONST. ConnectionState. DESLIGADO, existem mais dois parâmetros ***reason*** e ***code*** em data.

screensizechange

O evento screensizechange é gerado em resposta às alterações do tamanho do ecrã da máquina virtual atualmente ligada.
Parâmetros de data:

- width: a largura (em píxeis) da máquina virtual atualmente ligada.
- height: a altura (em píxeis) da máquina virtual atualmente ligada.

fullscreenchange

O evento fullscreenChange é gerado quando a consola do WMKS sai ou entra no modo de ecrã completo.
Parâmetros de data:

- isFullScreen: Boleano: *true* significa entrar em ecrã completo, *false* significa sair do ecrã completo.

error

O evento error é gerado quando ocorre um erro, por exemplo, uma falha de autenticação, um erro de WebSocket ou um erro de protocolo.

Parâmetros de data:

- errorType: pode ser qualquer valor de WMKS.CONST.Events.ERROR:
  1. AUTHENTICATION_FAILED: "authenticationfailed",
  2. WEBSOCKET_ERROR: "websocketerror",
  3. PROTOCOL_ERROR: "protocolerror"

Keyboardledschanged

O evento keyboardledschanged é gerado quando o estado de bloqueio dos LED do teclado da VM remota é alterado.

Parâmetros de data:

- Data é um número inteiro: 1 significa Scroll Lock, 2 significa Num Lock, 4 significa Caps Lock.

heartbeat

O evento heartbeat é gerado quando é recebida uma mensagem de heartbeat do lado do servidor.

- Data é um número inteiro que indica o intervalo do heartbeat.

audio

O evento audio é gerado quando é recebida uma mensagem de áudio do servidor.
Parâmetros de data:

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

copy

O evento copy é gerado quando o servidor envia um evento de texto cortado.

- Data corresponde à cadeia a ser copiada.

toggle

O evento toggle é gerado ao mostrar ou ocultar o teclado, o teclado expandido ou o trackpad.
Parâmetros de data:

- type: pode ser "KEYBOARD", "EXTENDED_KEYPAD", "TRACKPAD"
- visibility: Boleano: *true* significa "mostrar" e *false* significa "ocultar".

Configurar processadores para eventos do WMKS

No HTML Console SDK, forneça os métodos register() e unregister para adicionar e remover processadores para os eventos do WMKS.

register()

Este método é utilizado para registar o processador de eventos do WMKS.

Método	register(eventName, eventHandler)
Parâmetro 1	eventName(Constante, qualquer valor de WebMKS.Events)
Parâmetro 2	eventHandler(função de chamada de retorno de javascript)
Valor devolvido	Nenhum
Chamada de exemplo	var connectionStateHandler = function(event, data){}; wmks.register(WebMKS.Events.CONNECTION_STATE_CHANGE, connectionStateHandler);

unregister()

Este método é utilizado para anular o registo do processador de eventos para o WMKS.
Se este método não tiver parâmetros, remove todos os processadores de eventos.
Se tiver apenas um parâmetro, remove todos os processadores de eventos desse eventName.

Método	unregister(eventName, eventHandler)
Parâmetro 1	eventName(Constante, qualquer valor de WebMKS.Events)
Parâmetro 2	eventHandler(função de chamada de retorno de javascript)
Valor devolvido	Nenhum
Chamada de exemplo	wmks.unregister(WebMKS.Events.CONNECTION_STATE_CHANGE, connectionStateHandler);

Métodos do objeto do HTML Console SDK

Após invocar o método createWMKS(), recebe um objeto central do WMKS que pode utilizar a API do WMKS para estabelecer ligação a uma VM e realizar operações. Neste exemplo, vamos atribuir o nome "wmks" ao objeto do WMKS.

APIs gerais

Os métodos API gerais fornecem informações sobre o WMKS. Estes métodos podem ser utilizados a qualquer momento, incluindo antes de estabelecer ligação a uma VM de destino.

getVersion()

Obtém o número da versão atual do HTML Console SDK.

Método	getVersion()
Parâmetros	Nenhum
Valor devolvido	Cadeia. Contém o número completo da versão do HTML Console SDK.
Chamada de exemplo	var version = wmks.getVersion();

getConnectionState()

Obtém o estado atual da ligação.

Método	getConnectionState()
Parâmetros	Nenhum
Valor devolvido	Constante. Qualquer valor no WMKS. CONST.ConnectionState.
Chamada de exemplo	var state = wmks.getConnectionState();

APIs relacionadas com o ciclo de vida

connect()

Liga o WMKS a uma máquina virtual remota através do URL WebSocket e configura a IU.

Método	connect()
Parâmetro	URL WebSocket; o tipo é a cadeia no formato: <ws | wss> :// <host:port>/ <path> /? <authentication info>
Valor devolvido	nenhum
Chamada de exemplo	wmks.connect("ws://localhost:8080");

disconnect()

Interrompe a ligação da máquina virtual remota e destrói a IU.

Método	disconnect()
Parâmetro	Nenhum
Valor devolvido	Nenhum
Chamada de exemplo	wmks.disconnect();

destroy()

Termina a ligação (se estiver ativa) à VM e remove o widget do elemento associado. Os consumidores devem chamar esta função antes de remover o elemento do DOM.

Método	destroy()
Parâmetro	Nenhum
Valor devolvido	Nenhum
Chamada de exemplo	wmks.destroy();

APIs relacionadas com o ecrã

setRemoteScreenSize()

Envia um pedido de definição da resolução de ecrã da VM atualmente ligada. Neste caso, se os parâmetros de largura e altura passados forem maiores do que os do tamanho atribuído do widget WMKS, o dimensionamento é normalizado.

Nota: este método apenas funciona devidamente se a opção changeResolution estiver definida para *true*.

Método	setRemoteScreenSize(width,height)
Parâmetro 1	width(int), representa a largura desejada da VM ligada, em píxeis
Parâmetro 2	height(int), representa a altura desejada da VM ligada, em píxeis
Valor devolvido	Nenhum
Chamada de exemplo	wmks.setRemoteScreeSize(800,600);

getRemoteScreenSize()

Obtém a largura e a altura do ecrã da VM atualmente ligada, em píxeis.

Método	getRemoteScreenSize()
Parâmetro	Nenhum
Valor devolvido	Objeto no formato {width:, height:}
Chamada de exemplo	var size = wmks.getRemoteScreenSize();

updateScreen()

Altera a resolução ou redimensiona o ecrã remoto de modo a corresponder ao tamanho atualmente atribuído.

O comportamento de updateScreen depende das opções changeResolution, rescale e position:

1) Se a opção changeResolution estiver definida para *true*, envia o pedido de alteração da resolução à VM ligada, sendo que a resolução solicitada (largura e altura) é equivalente ao tamanho atribuído do contentor.

2) Verifique a opção rescale: se estiver definida para *true*, redimensione o ecrã remoto de modo a ajustar-se ao tamanho atribuído do contentor.

3) Verifique a opção position: se o tamanho do ecrã remoto não for equivalente ao tamanho atribuído do contentor, o ecrã remoto é colocado no centro ou na parte superior esquerda do contentor, com base no respetivo valor.

Método	updateScreen()
Parâmetro	Nenhum
Valor devolvido	Nenhum
Chamada de exemplo	wvmrc.updateScreen();

APIs relacionadas com o ecrã completo

canFullScreen()

Indica se a funcionalidade de ecrã completo está ativada neste browser. O modo de ecrã completo é desativado no Safari, uma vez que este não suporta introduções por teclado em ecrã completo por questões de segurança.

Método	canFullScreen()
Parâmetro	Nenhum
Valor devolvido	Boleano. *True* significa permitir, *false* significa não permitir.
Chamada de exemplo	wmks.canFullScreen();

isFullScreen()

Informa se o browser está no modo de ecrã completo.

Método	isFullScreen()
Parâmetro	Nenhum
Valor devolvido	Boleano. *True* significa que está em modo de ecrã completo, *false* significa que não está.
Chamada de exemplo	wmks.isFullScreen();

enterFullScreen()

Força o browser a entrar no modo de ecrã completo, se suportado. No modo de ecrã completo, apenas é apresentado o ecrã remoto.

Método	enterFullScreen()
Parâmetro	Nenhum
Valor devolvido	Nenhum
Chamada de exemplo	wmks.enterFullScreen();

exitFullScreen()

Força o browser a sair do modo de ecrã completo.

Método	exitFullScreen()
Parâmetro	Nenhum
Valor devolvido	Nenhum
Chamada de exemplo	wmks.exitFullScreen();

APIs relacionadas com a introdução

sendInputString()

Envia uma cadeia como introdução por teclado para o servidor.

Método	sendInputString(string)
Parâmetros	Cadeia
Valor devolvido	Nenhum.
Chamada de exemplo	wmks.sendInputString(“test”);

sendKeyCodes()

Envia uma série de códigos de teclas especiais para a VM. Esta função utiliza uma matriz de códigos de teclas especiais e envia eventos de keydown para cada um deles pela ordem listada. Em seguida, envia eventos de keyup para cada um deles, pela ordem inversa. Esta função pode ser utilizada para enviar combinações de teclas como Ctrl-Alt-Delete.

Método	sendKeyCodes ()
Parâmetros	Matriz. Cada elemento da matriz pode ser um código de teclas ou um Unicode. Código de teclas para teclas não imprimíveis, Unicode para carateres imprimíveis. Se utilizar Unicode, utilize números negativos; por exemplo, -118 significa "v"
Valor devolvido	Nenhum.
Chamada de exemplo	wmks.sendKeyCodes([17,18,46]) ; // Ctrl + Alt + Delete

sendCAD()

Envia uma sequência de teclas Ctrl-Alt-Delete para a máquina virtual atualmente ligada.

Método	sendCAD()
Parâmetros	Nenhum.
Valor devolvido	Nenhum.
Chamada de exemplo	wmks.sendCAD();

APIs relacionadas com dispositivos móveis

enableInputDevice()

Ativa o dispositivo de introdução (teclado, teclado expandido ou trackpad) do dispositivo móvel e inicia a respetiva utilização.

Método	enableInputDevice (deviceType)
Parâmetros	deviceType:(Constante) Qualquer valor em WMKS.CONST.InputDeviceType.
Valor devolvido	Nenhum.
Chamada de exemplo	wmks.enableInputDevice(WMKS. CONST .InputDeviceType. KEYBOARD) ;

disableInputDevice()

Desativa e destrói o dispositivo de introdução (teclado, teclado expandido ou trackpad) do dispositivo móvel.

Método	disableInputDevice (deviceType)
Parâmetros	deviceType:(Constante) Qualquer valor em WMKS.CONST.InputDeviceType.
Valor devolvido	Nenhum.
Chamada de exemplo	wmks.disableInputDevice(WMKS. CONST .InputDeviceType. KEYBOARD) ;

showKeyboard()

Mostra o teclado num dispositivo móvel.

Método	showKeyboard()
Parâmetros	Nenhum
Valor devolvido	Nenhum.
Chamada de exemplo	wmks.showKeyboard();

hideKeyboard()

Oculta o teclado num dispositivo móvel.

Método	hideKeyboard()
Parâmetros	Nenhum.
Valor devolvido	Nenhum.
Chamada de exemplo	wmks.hideKeyboard();

toggleExtendedKeypad()

Mostra/oculta o teclado expandido no dispositivo móvel consoante a visibilidade atual.

Método	toggleExtendedKeypad()
Parâmetros	Os mapas podem incluir um minToggleTime(ms), por exemplo, {minToggleTime: 50}, se o utilizador tentar chamar este método de ativação/desativação com demasiada frequência, a duração inferior ao minToggleTime não é executada.
Valor devolvido	Nenhum.
Chamada de exemplo	wmks.toggleExtendedKeypad();

toggleTrackpad()

Mostra/oculta o trackpad no dispositivo móvel consoante a visibilidade atual.

Método	toggleTrackpad()
Parâmetros	Os mapas podem incluir um minToggleTime(ms), por exemplo, {minToggleTime: 50}, se o utilizador tentar chamar este método de ativação/desativação com demasiada frequência, a duração inferior ao minToggleTime não é executada.
Valor devolvido	Nenhum.
Chamada de exemplo	wmks.toggleTrackpad();

toggleRelativePad ()

Mostra/oculta o trackpad do rato relativo no ecrã. Irá enviar o evento do rato relativo em vez do evento do rato absoluto após abrir este trackpad relativo. Pode ser utilizado quando o VMware Tools não está instalado no SO convidado remoto. Adicione um botão toggleRelativeMouse na IU e associe-o à API toggleRelativePad(). Após clicar no botão, o trackpad relativo é apresentado e envia eventos do rato relativos para o convidado remoto. Esta função é útil quando o SO convidado não instala o VMware Tools e não pode aceitar eventos do rato absolutos. Porque se não houver uma imagem de cursor proveniente do servidor, temos de apresentar uma imagem de cursor local. Por isso, os wmks-all.css das pastas css e img têm de ser adicionados no produto. Encontram-se todos na pasta wmkssdk.

Método	toggleRelativePad ()
Parâmetros	Os mapas podem incluir um minToggleTime(ms), por exemplo, {minToggleTime: 50}, se o utilizador tentar chamar este método de ativação/desativação com demasiada frequência, a duração inferior ao minToggleTime não é executada.
Valor devolvido	Nenhum.
Chamada de exemplo	wmks. toggleRelativePad ();

APIs relacionadas com opções

setOption()

Altera o valor da opção. Apenas as opções enumeradas abaixo podem utilizar este método:

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

Método	setOption(optionName, optionValue)
Parâmetro 1	optionName (cadeia do nome da opção)
Parâmetro 2	optionValue
Valor devolvido	Nenhum
Chamada de exemplo	wmks.setOption(“changeResolution”,false);

Apêndice

Constantes utilizadas no WMKS

- **Position**: o WMKS apresenta o ecrã remoto da VM em proporção idêntica. Assim, após o redimensionamento, o tamanho do ecrã remoto pode continuar a não ser equivalente ao tamanho do contentor. Neste caso, ***Position*** disponibiliza duas opções possíveis para colocar o ecrã numa posição do contentor.

- **ConnectionState**: existem 3 estados de ligação possíveis ao tentar estabelecer ligação à VM remota.

- **Events**: todos os nomes de eventos que o WMKS pode acionar encontram-se listados aqui.

- **ErrorType**: tipos de erro possíveis no ciclo de vida do WMKS.

- **InputDeviceType**: o HTML Console SDK suporta a visualização das consolas da VM em dispositivos móveis e este campo indica os dispositivos de introdução possíveis.

(Consulte a lista de constantes na página seguinte)

```
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
}
```