VMware HTML Console SDK Programmierhandbuch

Für vSphere 5.5 und höher (Allgemeine Verfügbarkeit)
und vCloud Director (Tech Preview)

Februar 2016

VMware HTML Console SDK Programmierhandbuch

Inhaltsverzeichnis

Unterstützte Browser (auch auf iOS und Android) 4

Schritte für die Verwendung der HTML Console SDK API 4

Platzieren der Konsole auf einer Website 5
Herstellen der Verbindung zu einer Remote-VM 6
Trennen und löschen 6

Die werkseitige Methode der WMKS 6

createWMKS() 6
Erstellungsoptionen 7
rescale 7
position 7
changeResolution 7
audioEncodeType 7
useNativePixels 7
useUnicodeKeyboardInput 7
useVNCHandshake 8
sendProperMouseWheelDeltas 8
reverseScrollY 8
retryConnectionInterval 8
ignoredRawKeyCodes 8
fixANSIEquivalentKeys 8
keyboardLayoutId 8
VCDProxyHandshakeVmxPath 9

Umgang mit WMKS-Ereignissen 10

Liste der WMKS-Ereignisse 10
connectionstatechange 10
screensizechange 10
fullscreenchange 10
error 10
Keyboardledschanged 11
heartbeat 11
audio 11
copy 11
toggle 11
Festlegen von Handlern für WMKS-Ereignisse 11
register() 12
unregister() 12

Methoden des HTML Console SDK-Objekts 12

Allgemeine APIs 12
getVersion() 12
getConnectionState() 13

APIs, die sich auf den Lebenszyklus beziehen 13
connect() *13*
disconnect() *13*
destroy() *13*
APIs, die sich auf die Anzeige beziehen 13
setRemoteScreenSize() *13*
getRemoteScreenSize() *14*
updateScreen() *14*
APIs, die sich auf den Vollbildmodus beziehen 15
canFullScreen() *15*
isFullScreen() *15*
enterFullScreen() *15*
exitFullScreen() *15*
APIs, die sich auf die Eingabe beziehen 16
sendInputString() *16*
sendKeyCodes() *16*
sendCAD() *16*
APIs, die sich auf Mobilgeräte beziehen 16
enableInputDevice() *16*
disableInputDevice() *17*
showKeyboard() *17*
hideKeyboard() *17*
toggleExtendedKeypad() *17*
toggleTrackpad() *18*
toggleRelativePad () *18*
APIs, die sich auf Optionen beziehen 19
setOption() *19*
Anhang 19
In WMKS verwendete Konstanten 19

HTML Console (WMKS) SDK Übersicht

Beim HTML Console SDK handelt es sich um eine JavaScript-Bibliothek, die auf WebMKS (WMKS) basiert und Maus-, Tastatur- und Berührungseingaben sowie Bildschirm-Updates und Cursoränderungen bei einer Desktopsitzung über einen Browser verarbeitet. Anwendungen, die im Browser ausgeführt werden, können JavaScript verwenden, um mit der HTML Console SDK API auf Funktionen der virtuellen Maschinenkonsole zuzugreifen.

Die HTML Console SDK API enthält verschiedene Methoden, die für die Verbindung zu und die Kommunikation mit einer virtuellen Maschine verwendet werden können. Außerdem löst sie Ereignisse aus, um Benutzer über Veränderungen des Status der virtuellen Maschine zu informieren. Sie können diese Methoden und Callbacks verwenden, um Endbenutzern die Remote-Verwaltung einer virtuellen Maschine von einem System mit dem entsprechenden Webbrowser und Betriebssystem zu ermöglichen.

Unterstützte Browser (auch auf iOS und Android)

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

Schritte für die Verwendung der HTML Console SDK API

Um die HTML Console SDK API verwenden zu können, muss eine webbasierte Anwendung normalerweise Folgendes tun:

1. Die Dateien der HTML Console SDK JavaScript-Bibliothek laden.
2. Die WMKS-Kernobjekte mit der werkseitigen Methode „createWMKS()“ mit einer bestimmten „div element ID“ im Dokument-DOM und die Erstellungsoptionen erstellen.
3. JavaScript-Callbacks registrieren, die auf Ereignisse reagieren, die von WMKS ausgelöst werden.
4. Die HTML Console SDK-Methode „connect()“ verwenden, um eine Verbindung zu einer virtuellen Zielmaschine herzustellen.
5. Die HTML Console SDK-Methoden verwenden, um mit der verbundenen virtuellen Maschine zu kommunizieren.
6. Die HTML Console SDK-Methode „disconnect()“ verwenden, um die Verbindung zu trennen (falls sie aktiv ist) und das Widget vom zugewiesenen Element zu entfernen.

Platzieren der Konsole auf einer Website

Das HTML Console SDK ist im Bereich „Downloads“ auf www.vmware.com/de verfügbar: Treiber und Tools für vSphere, vCloud Director, vRealize Automation und vCloud Air.

Eine verkleinerte wmks.min.js und eine nicht verkleinerte wmks.js werden in jedem Build bereitgestellt. Das HTML Console SDK verwendet das jQuery-Widget, um die VM-Konsole anzuzeigen, weshalb die mit jQuery in Verbindung stehende Bibliothek zuerst eingebunden werden muss. Hier ein Beispiel:

1. Schließen Sie jQuery und jQuery UI Javascript-Komponenten in Skript-Tags und die entsprechende jQuery css-Datei in ein Link-Stylesheet-Tag ein.
2. Schließen Sie wmks.js (oder wmks.min.js) in ein Skript-Tag und wmks-all.css in ein Link-Stylesheet-Tag ein.
3. Stellen Sie eine Div mit einer entsprechenden ID bereit.
4. Erstellen Sie das WMKS-Kernobjekt mit der werkseitigen Methode „WMKS.createWMKS()“.
5. Binden Sie die Callbacks für WMKS-Ereignisse ein.
6. Stellen Sie eine Verbindung mit der entsprechenden Ziel-URL her.

```
 <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>
```

Herstellen der Verbindung zu einer Remote-VM

Möglichkeiten, eine Verbindung zu einer Remote-VM herzustellen:

WMKS kann als Komponente für die Bereitstellung einer reinen webbasierten Konsolenverbindung zu von vCenter verwalteten VMs verwendet werden, indem ein zwischengeschalteter Proxy die Authentifizierungsanforderungen verarbeitet.

Verwenden Sie die folgende Methode, um eine Verbindung zur VM herzustellen:

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

Die URL sollte folgendes Schema aufweisen:

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

Trennen und löschen

Rufen Sie das auf, wenn Sie mit der WMKS-Komponente fertig sind. Durch das Löschen der WMKS wird auch die Verbindung getrennt (falls aktiv) und das Widget vom verknüpften Element entfernt.

Die werkseitige Methode der WMKS

createWMKS()

Das sollte die erste Methode sein, wenn Sie das HTML Console SDK verwenden. Durch die Verwendung dieser Methode wird das Widget generiert, das den Remote-Bildschirm anzeigen kann. Anschließend wird das WMKS-Kernobjekt geliefert, das alle HTML Console SDK APIs zum Herstellen einer Verbindung zu einer VM und zum Durchführen von Operationen verwenden kann.

Methode	createWMKS(id, options)
Parameter1	**id**(string): die ID des Div-Elements; dieses Element ist der Container der Anzeige für den Remote-Bildschirm.
Parameter2	**options**:(json object): Erstellungsoptionen wirken sich auf das Verhalten der WMKS aus. Leer bedeutet „verwende alle standardmäßigen Optionen.“
Rückgabewert	Das WMKS-Kernobjekt, das alle WMKS APIs zum Herstellen einer Verbindung zu einer VM und zum Durchführen von Operationen verwenden kann.
Beispiel eines Aufrufs	var wmks =WMKS.createWMKS(“container”,{});

Erstellungsoptionen

Wie webMKS, das in den vSphere-Web-Client integriert ist, verfügt auch WMKS über mehrere Optionen, die zur Steuerung des Verhaltens verwendet werden können. Dies ist ein Vergleich zwischen dem SDK und webMKS, das in den vSphere-Web-Client integriert ist.

rescale

Boolean: Der Standardwert ist „true“ und zeigt an, ob der Remote-Bildschirm neu skaliert werden soll, sodass er der zugewiesenen Größe des Containers entspricht.

position

Ein Zähler: Das kann ein beliebiger Wert von WMKS.CONST.Position sein. Der Standardwert ist WMKS.CONST.Position.CENTER. Er gibt an, welche Position (Mitte oder oben links) der Remote-Bildschirm im Container haben sollte.

changeResolution

Boolean: Der Standardwert ist „true“. Wenn die Option „changeResolution“ „true“ ist, sendet WMKS die Anfrage zur Änderung der Auflösung an die verbundene VM, wobei die angeforderte Auflösung (Breite und Höhe) mit der zugewiesenen Größe des Containers identisch ist.

Diese drei Optionen folgen basierend auf ihrer Priorität einer bestimmten Ausführungsreihenfolge.

1. Prüfen von **changeResolution** und falls „true“, sendet webmks die Anfrage zur Änderung der Auflösung an die verbundene VM.
2. Wenn die Anfrage fehlschlägt, können die Operationen **rescale** und **position** verwendet werden, um die entsprechenden Änderungen vorzunehmen.

audioEncodeType

Es ist ein Zähler: Das kann ein beliebiger Wert von WMKS.CONST.AudioEncodeType sein. Er gibt an, welcher Typ der Audioverschlüsselungsmethode verwendet wird: vorbis, opus oder aac.

useNativePixels

Boolean: Der Standardwert ist „false“. Aktiviert die Verwendung nativer Pixelgrößen auf dem Gerät. Auf dem iPhone 4+ oder iPad 3+ wird dadurch der „Retina-Modus“ aktiviert, der mehr Bildschirmplatz für den Gast bereitstellt, sodass alles kleiner erscheint.

useUnicodeKeyboardInput

Boolean: Der Standardwert ist „false“. Für alle Benutzereingaben kann WMKS zwei Arten von Meldungen an den Server senden: Tastaturscancodes oder Unicode. Hier bedeutet *true*, dass der Client Unicode verwenden soll, falls möglich.

useVNCHandshake

Boolean: Der Standardwert ist „true“. Aktiviert einen standardmäßigen VNC-Handshake. Dieser sollte verwendet werden, wenn das Endgerät standardmäßige VNC-Authentifizierung verwendet. Legen Sie diesen Wert auf „false“ fest, wenn Sie eine Verbindung zu einem Proxy herstellen, der sich über „authd“ authentifiziert und keinen VNC-Handshake durchführt.

sendProperMouseWheelDeltas

Boolean: Der Standardwert ist „false“. Frühere Versionen der Bibliothek normalisieren Mausrad-Ereignisdeltas in einen von drei Werten: [-1, 0, 1]. Wenn dieser Wert auf *true* festgelegt ist, werden die tatsächlichen Deltas vom Browser an den Server gesendet.

reverseScrollY

Boolean: Der Standardwert ist „false“. Wenn dieses Kennzeichen auf *true* festgelegt ist, wird der entgegengesetzte Wert des Mausrads an die verbundene VM gesendet.

retryConnectionInterval

Ganzzahl: Der Standardwert ist -1. Das Intervall (in Millisekunden) für einen Neuversuch der Verbindungsherstellung, wenn der erste Versuch, eine Verbindung zwischen dem Web-Client und dem Server herzustellen, fehlschlägt. Ein Wert kleiner 0 bedeutet, „nicht erneut versuchen.“

ignoredRawKeyCodes

Array: Der Standardwert ist leer. All diese Keycodes werden ignoriert und nicht an den Server gesendet.

fixANSIEquivalentKeys

Boolean: Der Standardwert ist „false“. Ermöglicht es, Nicht-ANSI-US-Layout-Keycodes so zu gestalten, dass sie ANSI-US-Layout-Keycodes entsprechen. Dabei wird versucht, gedrückte Tasten anzupassen, bei denen das internationale Tastaturlayout des Client eine Taste aufweist, die auch auf der ANSI-US-Tastatur vorhanden ist, sich aber an einer anderen Position befindet oder nicht dem UMSCHALT- oder NICHT-UMSCHALT-Status einer ANSI-US-Tastatur entspricht.

keyboardLayoutId

String: Der Standardwert ist „en-US“ (US-Englisch). Dieser stellt dem Gast Tastaturlayouts für unterschiedliche Sprachen zur Verfügung. Benutzer müssen das Tastaturlayout bestimmen und dafür sorgen, dass das Tastaturlayout des Remote-Desktop und des lokalen Desktop identisch ist. In dieser Version des HTML Console SDK unterstützt VMware mobile Gerät nicht. Es werden zwei Arten von Codes an das Gastbetriebssystem gesendet. Für die internationale Zuweisung unterstützen wir vScancode.

Internet Explorer/Firefox/Chrome werden auf Windows unterstützt und Chrome/Safari auf Mac.
Bitte fügen Sie eine Auswahlliste in HTML hinzu oder verwenden Sie eine andere Möglichkeit, damit Benutzer die verwendete Sprache auswählen können. Verwenden Sie in der js-Datei die „setOption“-API wie folgt:

```
<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

String: Der Standardwert ist Null. Beim Verbinden an das VNC-Protokoll weiterleiten. vncProtocol antwortet auf eine Verbindungsanfrage für einen VMX-Pfad mit dem bereitgestellten VCDProxyHandshakeVmxPath, wenn eine Verbindung zu **vCloud Director** hergestellt wird.

enableUint8Utf8

Boolean: Der Standardwert ist „false“. Wenn das Projekt keine funktionierende Unterstützung für das binäre Protokoll hat, wird die Option „enableUint8Utf8“ benötigt, um das alte uint8utf8-Protokoll zu aktivieren.

Umgang mit WMKS-Ereignissen

Liste der WMKS-Ereignisse

WebMKS generiert Ereignisse, wenn sich der Status der aktuell verbundenen virtuellen Maschine ändert oder als Reaktion auf Meldungen von der aktuell verbundenen virtuellen Maschine. Alle WMKS-Ereignisse werden in WMKS.CONST.Events aufgelistet.

Alle Ereignis-Handler haben zwei Parameter: **event** und **data**. **Event** ist ein jQuery-Ereignis und alle Sonderparameter der Ereignisse werden in **data** gespeichert.

connectionstatechange

Das Ereignis „connectionstatechange“ wird als Reaktion auf eine Änderung des Verbindungsstatus generiert, bspw. von „getrennt“ zu „verbunden“ oder von „verbinden“ zu „verbunden“.

Parameter in **data**:

- state: kann ein beliebiger Wert in WMKS.CONST sein. ConnectionState kann Folgendes sein:
  1. „verbinden“
  2. „verbunden“
  3. „getrennt“
- Wenn der Status WMKS.CONST. ConnectionState. CONNECTING ist, gibt es zwei weitere Parameter ***vvc*** und ***vvcSession*** in „data“.
- Wenn der Status WMKS.CONST. ConnectionState. DISCONNECTED ist, gibt es zwei weitere Parameter ***reason*** und ***code*** in „data“.

screensizechange

Das Ereignis „screensizechange“ wird als Reaktion auf Änderungen der Bildschirmgröße der aktuell verbundenen virtuellen Maschine generiert.
Parameter in „data“:

- width: die Breite (in Pixel) der aktuell verbundenen virtuellen Maschine.
- Height: die Höhe (in Pixel) der aktuell verbundenen virtuellen Maschine.

fullscreenchange

Das Ereignis „fullscreenChange“ wird generiert, wenn die WMKS-Konsole den Vollbildmodus verlässt oder aktiviert.
Parameter in „data“:

- isFullScreen: Boolean, „true“ bedeutet, Vollbildmodus aktivieren, „false“ bedeutet, Vollbildmodus verlassen.

error

Das Ereignis „error“ wird generiert, wenn ein Fehler auftritt, beispielsweise ein Authentifizierungsfehler, Websocket-Fehler oder Protokollfehler.

Parameter in „data“:

- errorType: kann ein beliebiger Wert in WMKS.CONST.Events.ERROR sein:
  1. AUTHENTICATION_FAILED: "authenticationfailed",
  2. WEBSOCKET_ERROR: "websocketerror",
  3. PROTOCOL_ERROR: "protocolerror“

Keyboardledschanged

Das Ereignis „keyboardledschanged“ wird generiert, wenn sich der Tastatur-LED-Sperrstatus der Remote-VM ändert.

Parameter in „data“:

- „data“ ist eine Ganzzahl: 1 bedeutet Scroll Lock, 2 bedeutet Num Lock, 4 bedeutet Caps Lock.

heartbeat

Das Ereignis „heartbeat“ wird generiert, wenn eine serverseitige Heartbeat-Meldung empfangen wird.

- „data“ ist eine Ganzzahl, die das Intervall des Heartbeats angibt.

audio

Das Ereignis „audio“ wird generiert, wenn eine Audiomeldung vom Server empfangen wird.

Parameter in „data“:

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

copy

Das Ereignis „copy“ wird generiert, wenn der Server ein Ereignis mit ausgeschnittenem Text sendet.

- „data“ ist die Zeichenfolge, die kopiert wird.

toggle

Das Ereignis „toggle“ wird generiert, wenn die Tastatur, die erweiterte Tastatur oder das Trackpad angezeigt oder ausgeblendet wird.

Parameter in „data“:

- type: kann „KEYBOARD“, „EXTENDED_KEYPAD“, „TRACKPAD“ sein
- visibility: Boolean, „true“ bedeutet „anzeigen“ und „false“ bedeutet „ausblenden“.

Festlegen von Handlern für WMKS-Ereignisse

Stellen Sie im HTML Console SDK Methoden zum Registrieren und Aufheben der Registrierung bereit, um Handler für WMKS-Ereignisse hinzuzufügen und zu entfernen.

register()

Diese Methode dient zum Registrieren des Ereignis-Handlers für WMKS.

Methode	register(eventName, eventHandler)
Parameter1	eventName(Konstante, ein beliebiger Wert von WebMKS.Events)
Parameter2	eventHandler(Javascript-Callback-Funktion)
Rückgabewert	Keine
Beispiel eines Aufrufs	var connectionStateHandler = function(event, data){}; wmks.register(WebMKS.Events.CONNECTION_STATE_CHANGE, connectionStateHandler);

unregister()

Diese Methode dient zum Aufheben der Registrierung des Ereignis-Handlers für WMKS. Wenn diese Methode keine Parameter verwendet, werden alle Ereignis-Handler entfernt. Wenn es nur einen Parameter gibt, werden alle Ereignis-Handler für diesen speziellen eventName entfernt.

Methode	unregister(eventName, eventHandler)
Parameter1	eventName(Konstante, ein beliebiger Wert von WebMKS.Events)
Parameter2	eventHandler(Javascript-Callback-Funktion)
Rückgabewert	Keine
Beispiel eines Aufrufs	wmks.unregister(WebMKS.Events.CONNECTION_STATE_CHANGE, connectionStateHandler);

Methoden des HTML Console SDK-Objekts

Nach dem Aufrufen der Methode createWMKS() erhalten Sie ein WMKS-Kernobjekt, das die WMKS API zum Herstellen einer Verbindung zu einer VM und zum Ausführen von Operationen verwenden kann. In diesem Beispiel bezeichnen wir das WMKS-Objekt als „wmks“.

Allgemeine APIs

Allgemeine API-Methoden liefern Informationen über WMKS. Diese Methoden können jederzeit aufgerufen werden, auch vor dem Herstellen einer Verbindung zur Ziel-VM.

getVersion()

Ruft die aktuelle Versionsnummer des HTML Console SDK ab.

Methode	getVersion()
Parameter	Keine
Rückgabewert	String. Enthält die vollständige Versionsnummer des HTML Console SDK.
Beispiel eines Aufrufs	var version = wmks.getVersion();

getConnectionState()

Ruft den aktuellen Verbindungsstatus ab.

Methode	getConnectionState()
Parameter	Keine
Rückgabewert	Konstante. Ein beliebiger Wert in WMKS. CONST.ConnectionState.
Beispiel eines Aufrufs	var state = wmks.getConnectionState();

APIs, die sich auf den Lebenszyklus beziehen

connect()

Verbindet WMKS mithilfe der WebSocket-URL mit einer virtuellen Remote-Maschine und richtet die Benutzeroberfläche ein.

Methode	connect()
Parameter	WebSocket-URL, Typ ist eine Zeichenfolge im Format: <ws | wss> :// <host:port>/ <path> /? <authentication info>
Rückgabewert	keine
Beispiel eines Aufrufs	wmks.connect("ws://localhost:8080");

disconnect()

Trennt die Verbindung zur virtuellen Remote-Maschine und löscht die Benutzeroberfläche.

Methode	disconnect()
Parameter	Keine
Rückgabewert	Keine
Beispiel eines Aufrufs	wmks.disconnect();

destroy()

Beendet die Verbindung (falls aktiv) zur VM und entfernt das Widget vom dazugehörigen Element. Anwender sollten diese Funktion aufrufen, bevor sie das Element aus dem DOM entfernen.

Methode	destroy()
Parameter	Keine
Rückgabewert	Keine
Beispiel eines Aufrufs	wmks.destroy();

APIs, die sich auf die Anzeige beziehen

setRemoteScreenSize()

Sendet eine Anfrage zum Einrichten der Bildschirmauflösung an die aktuell verbundene VM. Hier gilt: Wenn die gemeldeten Parameter Breite und Höhe größer sind als die zugewiesene Größe des WMKS-Widgets, wird die Größe normalisiert.

Hinweis: Diese Methode funktioniert nur dann korrekt, wenn die Option „changeResolution“ auf *true* festgelegt ist.

Methode	setRemoteScreenSize(width,height)
Parameter1	width(int) gibt die gewünschte Höhe der verbundenen VM in Pixel an.
Parameter2	height(int) gibt die gewünschte Höhe der verbundenen VM in Pixel an.
Rückgabewert	Keine
Beispiel eines Aufrufs	wmks.setRemoteScreeSize(800,600);

getRemoteScreenSize()

Ruft die Bildschirmbreite und -höhe der aktuell verbundenen VM in Pixel ab.

Methode	getRemoteScreenSize()
Parameter	Keine
Rückgabewert	Objekt im Format {Breite:, Höhe:}
Beispiel eines Aufrufs	var size = wmks.getRemoteScreenSize();

updateScreen()

Ändert die Auflösung oder skaliert den Remote-Bildschirm neu, sodass er mit der aktuell zugewiesenen Größe übereinstimmt.

Das Verhalten von „updateScreen“ ist abhängig von den Optionen „changeResolution“, „rescale“ und „position“:

1) Wenn die Option „changeResolution“ „true“ ist, sendet WMKS die Anfrage zur Änderung der Auflösung an die verbundene VM; die angeforderte Auflösung (Breite und Höhe) ist mit der zugewiesenen Größe des Containers identisch.

2) Prüfen der Option „rescale“: wenn sie „true“ ist, wird der Remote-Bildschirm neu skaliert, sodass er mit der zugewiesenen Größe des Containers übereinstimmt.

3) Prüfen der Option „position“: Wenn die Größe des Remote-Bildschirms nicht mit der zugewiesenen Größe des Containers identisch ist, wird der Remote-Bildschirm basierend auf diesem Wert in der Mitte oder oben links im Container platziert.

Methode	updateScreen()
Parameter	Keine
Rückgabewert	Keine
Beispiel eines Aufrufs	wvmrc.updateScreen();

APIs, die sich auf den Vollbildmodus beziehen

canFullScreen()

Zeigt an, ob die Vollbildfunktion auf diesem Browser aktiviert ist. Der Vollbildmodus ist in Safari deaktiviert, da hier aus Sicherheitsgründen keine Tastatureingaben im Vollbildmodus unterstützt werden.

Methode	canFullScreen()
Parameter	Keine
Rückgabewert	Boolean. „true“ bedeutet: kann im Vollbildmodus sein, „false“ bedeutet: kann nicht im Vollbildmodus sein.
Beispiel eines Aufrufs	wmks.canFullScreen();

isFullScreen()

Informiert darüber, ob sich der Browser im Vollbildmodus befindet.

Methode	isFullScreen()
Parameter	Keine
Rückgabewert	Boolean. „true“ bedeutet: ist im Vollbildmodus, „false“ bedeutet: ist nicht im Vollbildmodus.
Beispiel eines Aufrufs	wmks.isFullScreen();

enterFullScreen()

Zwingt den Browser, den Vollbildmodus auszuführen, wenn dieser unterstützt wird. Im Vollbildmodus wird nur der Remote-Bildschirm angezeigt.

Methode	enterFullScreen()
Parameter	Keine
Rückgabewert	Keine
Beispiel eines Aufrufs	wmks.enterFullScreen();

exitFullScreen()

Zwingt den Browser, den Vollbildmodus zu verlassen.

Methode	exitFullScreen()
Parameter	Keine
Rückgabewert	Keine
Beispiel eines Aufrufs	wmks.exitFullScreen();

APIs, die sich auf die Eingabe beziehen

sendInputString()

Sendet eine Zeichenfolge als Tastatureingabe an den Server.

Methode	sendInputString(string)
Parameter	String.
Rückgabewert	Keine.
Beispiel eines Aufrufs	wmks.sendInputString(“test”);

sendKeyCodes()

Sendet eine Reihe von Sonder-Keycodes an die VM. Dafür wird ein Array spezieller Keycodes verwendet und es werden für jeden Keycode in der aufgeführten Reihenfolge Keydowns gesendet. Anschließend werden in umgekehrter Reihenfolge für jeden Keycode Keyups gesendet. Somit können Tastenkombinationen wie Strg-Alt-Entf gesendet werden.

Methode	sendKeyCodes ()
Parameter	Array. Jedes Element im Array kann ein Keycode oder ein Unicode sein. Keycode für nicht druckbaren Schlüssel, Unicode für druckbares Zeichen. Bei der Verwendung von Unicode bedeutet ein negativer Wert wie -118 „v“.
Rückgabewert	Keine.
Beispiel eines Aufrufs	wmks.sendKeyCodes([17,18,46]) ; // Ctrl + Alt + Delete

sendCAD()

Sendet eine Strg-Alt-Entf-Tastensequenz an die aktuell verbundene virtuelle Maschine.

Methode	sendCAD()
Parameter	Keine.
Rückgabewert	Keine.
Beispiel eines Aufrufs	wmks.sendCAD();

APIs, die sich auf Mobilgeräte beziehen

enableInputDevice()

Aktiviert das Eingabegerät (Tastatur, erweiterte Tastatur, Trackpad) auf dem Mobilgerät und initialisiert die Verwendung.

Methode	enableInputDevice (deviceType)
Parameter	deviceType:(Constant) Ein beliebiger Wert in WMKS.CONST.InputDeviceType.
Rückgabewert	Keine.
Beispiel eines Aufrufs	wmks.enableInputDevice(WMKS. CONST .InputDeviceType. KEYBOARD) ;

disableInputDevice()

Deaktiviert das Eingabegerät (Tastatur, erweiterte Tastatur, Trackpad) auf dem Mobilgerät und löscht es.

Methode	disableInputDevice (deviceType)
Parameter	deviceType:(Constant) Ein beliebiger Wert in WMKS.CONST.InputDeviceType.
Rückgabewert	Keine.
Beispiel eines Aufrufs	wmks.disableInputDevice(WMKS. CONST .InputDeviceType. KEYBOARD) ;

showKeyboard()

Zeigt die Tastatur auf einem Mobilgerät an.

Methode	showKeyboard()
Parameter	Keine.
Rückgabewert	Keine.
Beispiel eines Aufrufs	wmks.showKeyboard();

hideKeyboard()

Blendet die Tastatur auf einem Mobilgerät aus.

Methode	hideKeyboard()
Parameter	Keine.
Rückgabewert	Keine.
Beispiel eines Aufrufs	wmks.hideKeyboard();

toggleExtendedKeypad()

Das Anzeigen/Ausblenden der erweiterten Tastatur auf dem Mobilgerät ist abhängig von der aktuellen Sichtbarkeit.

Methode	toggleExtendedKeypad()
Parameter	Eine Zuordnung kann minToggleTime(ms) wie {minToggleTime: 50} enthalten; wenn der Benutzer versucht, diese Umschaltmethode zu häufig aufzurufen, und die Dauer kürzer als die minToggleTime ist, wird dieser Befehl nicht ausgeführt.
Rückgabewert	Keine.
Beispiel eines Aufrufs	wmks.toggleExtendedKeypad();

toggleTrackpad()

Das Anzeigen/Ausblenden des Trackpads auf dem Mobilgerät ist abhängig von der aktuellen Sichtbarkeit.

Methode	toggleTrackpad()
Parameter	Eine Zuordnung kann minToggleTime(ms) wie {minToggleTime: 50} enthalten; wenn der Benutzer versucht, diese Umschaltmethode zu häufig aufzurufen, und die Dauer kürzer als die minToggleTime ist, wird dieser Befehl nicht ausgeführt.
Rückgabewert	Keine.
Beispiel eines Aufrufs	wmks.toggleTrackpad();

toggleRelativePad ()

Zeigt das relative Maus-Trackpad auf dem Bildschirm an oder blendet es aus. Sendet das relative Mausereignis anstelle des absoluten Mausereignisses, nachdem dieses relative Pad geöffnet wurde. Kann verwendet werden, wenn auf dem Remote-Gastbetriebssystem keine VMware-Tools installiert sind. Fügen Sie eine Schaltfläche „toggleRelativeMouse“ auf der Benutzeroberfläche hinzu und binden Sie sie an die toggleRelativePad() API. Nach dem Anklicken wird das relative Trackpad angezeigt und relative Mausereignisse werden an den Remote-Gast gesendet. Das ist hilfreich, wenn das Gastbetriebssystem keine VMware-Tools installiert und keine absoluten Mausereignisse akzeptiert. Denn wenn kein Cursorbild vom Server gesendet wird, müssen wir das lokale Cursorbild anzeigen. Aus diesem Grund muss wmks-all.css in den Ordnern „css“ und „img“ im Produkt hinzugefügt werden. Sie befinden sich alle im Ordner „wmkssdk“.

Methode	toggleRelativePad ()
Parameter	Eine Zuordnung kann minToggleTime(ms) wie {minToggleTime: 50} enthalten; wenn der Benutzer versucht, diese Umschaltmethode zu häufig aufzurufen, und die Dauer kürzer als die minToggleTime ist, wird dieser Befehl nicht ausgeführt.
Rückgabewert	Keine.
Beispiel eines Aufrufs	wmks. toggleRelativePad ();

APIs, die sich auf Optionen beziehen

setOption()

Ändert den Optionswert. Nur die nachfolgend aufgeführten Optionen können diese Methode verwenden:

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

Methode	setOption(optionName, optionValue)
Parameter1	optionName(Zeichenfolge des Optionsnamens)
Parameter2	optionValue
Rückgabewert	Keine.
Beispiel eines Aufrufs	wmks.setOption("changeResolution",false);

Anhang

In WMKS verwendete Konstanten

- **Position**: WMKS zeigt den Remote-Bildschirm der VM in gleicher Proportion an. Aus diesem Grund ist die Größe des Remote-Bildschirms nach der Neuskalierung noch immer nicht mit der Größe des Containers identisch. Hier bietet ***Position*** zwei mögliche Optionen für die Platzierung des Bildschirms im Container.

- **ConnectionState**: Es gibt 3 mögliche Verbindungsstatus, wenn versucht wird, eine Verbindung zur Remote-VM herzustellen.

- **Events**: Hier werden alle Ereignisnamen aufgelistet, die WMKS auslösen kann.

- **ErrorType**: mögliche Fehlertypen im Lebenszyklus von WMKS.

- **InputDeviceType**: Das HTML Console SDK unterstützt das Anzeigen von VM-Konsolen auf Mobilgeräten. In diesem Feld sind die möglichen Eingabegeräte aufgelistet.

(Eine Liste der Konstanten finden Sie auf der nächsten Seite.)

```
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
}
```