Using the VMRC API

vCloud Director 1.5

With VMware vCloud Director, you can give users the ability to access virtual machine console functions from your web-based user interface. vCloud Director contains the VMware Remote Console browser plug-in, which can be loaded in supported web browsers. Applications running in the browser can use JavaScript to access virtual machine console functions by using the VMRC API.

The VMRC API contains various methods and that can be used to connect to, and communicate with, a virtual machine. The VMRC API also contains callback signals to notify users of changes to the virtual machine's state. You can use these methods and callbacks to give a user the ability to remotely manage a virtual machine from any system with the appropriate web browser and operating system.

This document contains the following topics:

Requirements for Using the VMRC API

To use the VMRC API, your web-based user interface must be able to load the VMRC browser plug-in. The plug-in is supported for the following web browsers and operating systems:

- Microsoft Internet Explorer on Microsoft Windows
- Mozilla Firefox on Microsoft Windows
- Mozilla Firefox on Linux

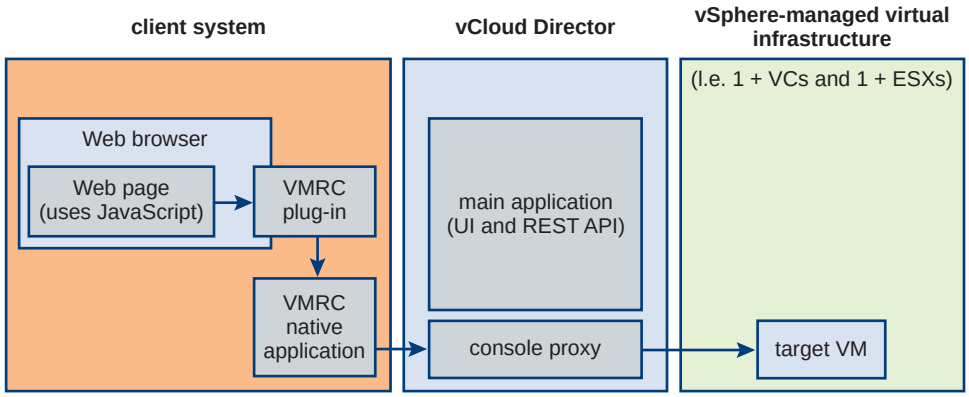
Your web-based user interface must support JavaScript to use the VMRC browser plug-in and the VMRC API. The browser plug-in is loaded using JavaScript, and you invoke API methods using JavaScript calls.

VMRC is supported by all ESX and vCenter Server configurations supported by vCloud Director 1.5.

VMRC Overview

A client system (typically a web-based user interface) uses VMRC to communicate with a virtual machine through a vCloud Director console proxy. The VMRC processes are bundled within a web browser plug-in. The client system must load this browser plug-in to use VMRC. The web interface on the client system can then use VMRC API JavaScript methods to connect to the target virtual machine through the browser plug-in. Figure 1 shows a diagram of the basic VMRC architecture.

Figure 1. Basic VMRC Architecture



Steps for Using the VMRC API

To use the VMRC API, the VMRC plug-in must be installed on the client system. Once the plug-in is installed, a web-based application must typically do the following things:

1 Load the VMRC browser plug-in using the appropriate JavaScript call and constant value.

2 Set JavaScript event handlers to respond to VMRC callback signals.

3 Start the VMRC plug-in using the `startup()` API method.

4 Use the VMRC API `connect()` method to connect to a target virtual machine.

5 Use the VMRC API methods to send commands to the target virtual machine.

6 Use the VMRC API `disconnect()` method to disconnect from the target virtual machine.

7 Shut down the VMRC browser plug-in using the `shutdown()` API method.

Using VMRC Property Value Constants

The VMRC API exposes a set of property value constants. Within your web-based interface's JavaScript code, you must use these property value constants when invoking VMRC API methods or handling callback signals. The VMRC API exposes the property value constants differently depending on which web browser you use with VMRC.

In Internet Explorer, the property value constants are exposed as dictionaries corresponding to each property value class name. For example, a value of `VMRC_CS_CONNECTED` for the `VMRC_ConnectionState` variable is represented by the following JavaScript code in Internet Explorer:

```
vmrc.VMRC_ConnectionState("VMRC_CS_CONNECTED")
```

In Firefox, the property value constants directly correspond to the property name. A value of `VMRC_CS_CONNECTED` for the `VMRC_ConnectionState` variable is represented by the following JavaScript code in Firefox:

```
vmrc.VMRC_ConnectionState.VMRC_CS_CONNECTED
```

The VMRC Reference Implementation contains example JavaScript code that abstracts these differences in handling property value constants. You can use the example code to ensure that the constants can be accessed in a uniform manner from your JavaScript code. To obtain the sample code from the reference implementation, go to https://www.vmware.com/support/vcd/doc/vcd-1.5-vmrc-api-example.zip.

Loading the VMRC Browser Plug-In

You load the VMRC browser plug-in by creating an object instance of the plug-in. When you create this object instance, you must set the instance class ID (in Internet Explorer) or type attribute (in Firefox) to a specific value. Example 1 shows how to create an object instance using HTML, and how to set the class ID or type values. The object instance is assigned an ID value of "vmrc".

Example 1. Creating an Object Instance of the VMRC Browser Plug-In

```
<!--[if IE]>
<object id="vmrc" classid="CLSID:A24C4C22-E2B7-4701-9DF1-E51BDC809850"
                style="width: 100%; height: 100%;"></object>
<![endif] -->
<!--[if !IE]><!-->
<object id="vmrc" type="application/x-vmware-remote-console-2011-1"
                style="width: 100%; height: 100%;"></object>
<!--<![endif]-->
```

The code in Example 1 can be included anywhere in the HTML code that makes up your web-based user interface. Subsequent VMRC API examples in this document refer to the plug-in object instance using the value `vmrc` as set in the HTML example code.

Setting Handlers for VMRC Events

The VMRC API generates events when there is a change in the state of the VMRC session. These events include changes in connection state, changes in screen size on the target virtual machine, or events sent in response to messages from the VMRC plug-in. The complete list of VMRC events can be found in the API reference file included with the VMRC SDK.

For your web interface to respond to VMRC events, you must bind the events to JavaScript handler methods. The VMRC API provides different binding mechanisms for each supported browser.

The handler method that you provide for each event must accept the same parameters that the event provides. For example, the VMRC event `onScreenSizeChange` provides two integer parameters (the new screen width and the new screen height). The handler method that you provide for the `onScreenSizeChange` event must likewise accept the two integer values for width and height as parameters. For more information on events and the parameters they provide, see "VMRC Events" on page 8.

Setting Handlers in Internet Explorer

In Internet Explorer, handler methods are bound using the `attachEvent()` method. You set the handler method by calling `attachEvent()` with the event name (as a string) and a pointer to the JavaScript handler method as parameters.

For example, to set a handler method for the `onConnectionStateChange` event, you must call `attachEvent()` as follows:

```
vmrc.attachEvent("onConnectionStateChange", onConnectionStateChangeHandler);
```

The first parameter to `attachEvent()` is the event name, defined by the VMRC API. The second parameter is the JavaScript handler method that you define.

Setting Handlers in Firefox

In Firefox, each event corresponds to a property of the VMRC plug-in object instance. You bind a handler method to a VMRC event by setting the value of the corresponding event property to the handler method.

For example, to set a handler method for the `onConnectionStateChange` event, you must set the VMRC object instance's `onConnectionStateChange` property as follows:

```
vmrc["onConnectionStateChange"] = onConnectionStateChangeHandler;
```

Abstracting Event Handler Setup

The VMRC Reference Implementation contains sample code that shows how to abstract the setting of event handlers into a single function that can be used with both supported browsers. To obtain the reference implementation and sample code files, go to https://www.vmware.com/support/vcd/doc/vcd-1.5-vmrc-api-example.zip.

Using the VMRC Plug-In: Startup, Invoking Methods, and Shutdown

Once your web-based interface has loaded the VMRC browser plug-in and set handlers for the appropriate VMRC callback signals, you can use the VMRC API methods to interface with a remote virtual machine. To use the VMRC API, you must perform the following sequence of steps:

1 Initialize the VMRC plug-in.

2 Connect to the remote host of the target virtual machine.

3 Invoke VMRC API methods that correspond to console commands on the target virtual machine.

4 Disconnect from the remote host.

5 Shut down the VMRC plug-in.

Initializing the VMRC Plug-In

Before using any of the other API methods, you must first initialize the VMRC plug-in. Initializing the VMRC plug-in is a two-step process:

1 Use the `isReadyToStart()` API method to determine whether the plug-in has been successfully loaded and is ready to be started.

2 Use the `startup()` API method to start the processes in the VMRC plug-in.

Using isReadyToStart()

The `isReadyToStart()` method takes no parameters and returns a boolean value. A return value of `true` indicates that the VMRC plug-in has been loaded and is ready to start. You may call `startup()` once the `isReadyToStart()` method returns a value of `true`.

NOTE The VMRC plug-in does not generate an event when the plug-in is ready to start. You must poll the return value of the `isReadyToStart()` method to determine when the VMRC plug-in has loaded and can be started.

A call to the `isReadyToStart()` method might appear as follows:

```
var ret = vmrc.isReadyToStart();
```

Starting the VMRC Plug-In

You must use the `startup()` API method to complete the initialization of the VMRC plug-in. The `startup()` method accepts four parameters:

Parameter Name	Type	Description/Notes
`mode`	`VMRC_Mode` property value constant; VMRC_Mode.VMRC_MKS or VMRC_Mode.VMRC_DEVICES	The `mode` parameter is specified at startup, and determines the operational mode of the VMRC plug-in instance for the lifetime of its connection. The plugin can operate in two modes, only one of which is currently supported: **VMRC_Mode.VMRC_MKS:** In MKS mode, guest screen contents are displayed and the user can interact with the guest. **VMRC_Mode.VMRC_DEVICES**: In Devices mode, the user can manage remote device backings and their mappings to virtual machine devices. This mode is not currently supported.
`msgmode`	`VMRC_MessageMode` property value constant; VMRC_MessageMode.VMRC_EVENT_MESSAGES is the only valid value	The `msgmode` parameter is specified at startup, and determines which messages are delivered from the plug-in to the containing document element. Only the value VMRC_EVENT_MESSAGES is currently supported.
`persistent`	boolean	Only the `false` value is currently supported.
`advancedcofig`	string	A string containing advanced configuration options. For vCloud Director 1.5, this value must be set to the following string: “userbrowserproxy=true;tunnelmks=true”

The `startup()` method returns a boolean value. The value is `true` for success and `false` for failure.

An example call of the `startup()` method in the Firefox browser might appear as follows:

```
var ret = vmrc.startup(vmrc.VMRC_Mode.VMRC_MKS, vmrc.VMRC_MessageMode.VMRC_EVENT_MESSAGES,
                false, “usebrowserproxy=true;tunnelmks=true”);
```

Connecting to a Remote Host

After initializing the VMRC plug-in, you can use the `connect()` method to connect to a remote host and access a particular virtual machine on that host. To use the `connect()` method, you need to obtain three pieces of information:

- The hostname or IP address of the remote host.
- The screen ticket.
- The virtual machine ID.

You can obtain this information by using the VMware vCloud Director API. For more information on the vCloud Director API, go to https://www.vmware.com/support/pubs/vcd_pubs.html.

The host, ticket, and virtual machine ID are passed as parameters to the `connect()` method. In addition, the `connect()` method accepts four internal parameters. These are reserved for future use to pass additional connection-related information. In vCloud Director 1.5, these must be passed as empty strings (“”).

Parameter Name	Type	Description/Notes
`host`	string	The hostname or IP address provided by the vCloud Director API.
`ticket`	string	The screen ticket provided by the vCloud Director API.
`internal1`	string	Reserved for future use; pass as empty string (“”).

Parameter Name	Type	Description/Notes
`internal2`	string	Reserved for future use; pass as empty string (“”).
`vmid`	string	The virtual machine ID provided by the vCloud Director API.
`internal3`	string	Reserved for future use; pass as empty string (“”).
`internal4`	string	Reserved for future use; pass as empty string (“”).

The `connect()` method returns a boolean value. The value is `true` for success and `false` for failure. Note that the return value pertains only to the attempt to connect (that is, the actual call to the `connect()` method). If the connection itself is successful, the VMRC plug-in will generate an `onConnectionStateChange` event. See “VMRC Events” on page 8 for more information.

An example call of the `connect()` method might appear as follows:

```
var ret = vmrc.connect(host, ticket, “”, “”, vmid, “”, “”);
```

Invoking VMRC API Methods on a Virtual Machine

You can use the different VMRC API methods to interact with the target virtual machine. Certain methods, such as those that provide version and support information for the VMRC plug-in, can be used at any time (including before `startup()` has been invoked). Other methods depend on the mode you choose when calling the startup() method for the VMRC plug-in (MKS mode or Devices mode).

NOTE Currently, only MKS mode (`VMRC_Mode.VMRC_MKS`) is supported with the VMRC plug-in.

General Methods

General methods provide information about VMRC and the APIs it supports. These methods can be called at any time and are not dependent on the VMRC plug-in `startup()` method.

getVersion()

The `getVersion()` method retrieves the current complete version number of the installed VMRC plug-in.

Method	`getVersion()`
Parameters	None
Return Value	`string`; contains the full version number for the VMRC plug-in
Example Call	`var version = vmrc.getVersion();`

getSupportedApi()

The `getSupportedApi()` method retrieves the names of supported APIs from the VMRC plug-in.

Method	`getSupportedApi()`
Parameters	None
Return Value	`string[]`; an array of strings, each containing the name of a supported API (such as “vsphere-2011”
Example Call	`var api = vmrc.getSupportedApi();`

MKS Mode Methods

MKS mode methods can only be used after invoking the `startup()` method with a parameter value of `VMRC_Mode.VMRC_MKS`. When using the VMRC plug-in in MKS mode, you can use methods to obtain screen information about the currently connected virtual machine, and the status of the current connection. You can also send a Control-Alt-Delete key sequence to the virtual machine.

You can use the following methods with the VMRC plug-in in MKS mode:

getConnectionState()

The `getConnectionState()` method retrieves the current connection state from the VMRC plug-in.

Method	`getConnectionState()`
Parameters	None
Return Value	Property Value Constant; valid values are VMRC_CS_CONNECTED or VMRC_CS_DISCONNECTED
Example Call	`var ret = vmrc.getConnectionState();`

screenWidth()

The `screenWidth()` method retrieves the screen width, in pixels, of the currently connected virtual machine.

Method	`screenWidth()`
Parameters	None
Return Value	`int`; represents screen width in pixels
Example Call	`var sw = vmrc.screenWidth();`

screenHeight()

The `screenHeight()` method retrieves the screen height, in pixels, of the currently connected virtual machine.

Method	`screenHeight()`
Parameters	None
Return Value	`int`; represents screen height in pixels
Example Call	`var sh = vmrc.screenHeight();`

setFullscreen()

The `setFullscreen()` method commands the VMRC plug-in to enter or exit full-screen mode.

Method	`setFullscreen(boolean fs)`
Parameters	`boolean`; set to `true` to enter full-screen mode, `false` to exit
Return Value	`boolean`; `true` for success or `false` for failure
Example Call	`var ret = vmrc.setFullscreen(true);`

sendCAD()

The `sendCAD()` method sends a Control-Alt-Delete key sequence to the currently connected virtual machine.

Method	`sendCAD()`
Parameters	None
Return Value	`boolean`; `true` for success or `false` for failure
Example Call	`var ret = vmrc.sendCAD();`

Devices Mode Methods

Devices mode (VMRC_Mode.VMRC_DEVICES) is not currently supported for use with the VMRC plug-in.

Disconnecting from a Remote Host

If you have finished performing operations on the target virtual machine, or otherwise want to close the connection, you can use the `disconnect()` method to terminate the connection to the remote host.

The `disconnect()` method accepts no parameters and returns a boolean value. The value is `true` for success and `false` for failure.

An example call to the `disconnect()` method might appear as follows:

```
var ret = vmrc.disconnect();
```

Shutting Down the VMRC Plug-In

You can shut down the VMRC browser plug-in by invoking the `shutdown()` method. Shutting down the VMRC browser plug-in stops the corresponding VMRC peer processes within the plug-in.

The `shutdown()` method accepts no parameters and returns a boolean value. The value is `true` for success and `false` for failure.

An example call to the `shutdown()` method might appear as follows:

```
var ret = vmrc.shutdown();
```

VMRC Events

The VMRC plug-in generates events when the state of the currently connected virtual machine changes, or in response to messages from the currently connected virtual machine. Each event provides a set of parameters with additional information on the changed state of the virtual machine.

You can respond to these events by attaching JavaScript handler methods to be called in response to the events. These handler methods should accept the same parameters as the events they are handling. See "Setting Handlers for VMRC Events" on page 3 for more information.

The VMRC plug-in can invoke the following events:

onConnectionStateChange()

This event is invoked in response to a change in the connection state. The `onConnectionStateChange()` event provides five parameters:

Parameter Name	Type	Description/Notes
`cs`	string	The new connection state. Valid values include VMRC_CS_CONNECTED or VMRC_CS_DISCONNECTED.
`host`	string	The remote hostname for the connection.
`vmId`	string	The virtual machine ID for the current connection.
`userRequested`	boolean	A boolean value denoting whether the change in connection state was requested by the user.
`reason`	string	A string containing information on the connection state change.

onScreenSizeChange()

This event is invoked in response to changes in the guest screen size. The `onScreenSizeChange()` event provides two parameters:

Parameter Name	Type	Description/Notes
`width`	int	The new screen width, in pixels.
`height`	int	The new screen height, in pixels.

onFullscreenChange()

This event is invoked when the VMRC plug-in exits or enters full-screen mode. The `onFullscreenChange()` event provides one parameter:

Parameter Name	Type	Description/Notes
`fs`	boolean	If `true`, the plug-in has entered full-screen mode. If `false`, the plug-in has exited full-screen mode.

onGrabStateChange()

This event is invoked when the VMRC plug-in grab state changes. The `onGrabStateChange()` event provides one parameter:

Parameter Name	Type	Description/Notes
`gs`	Property Value Constant	The new grab state, as denoted by a property value constant. Values can include: VMRC_GS_GRABBED VMRC_GS_UNGRABBED_HARD VMRC_GS_UNGRABBED_SOFT In a soft-ungrab state, input is redirected to the guest when the user mouses over the guest window.

onMessage()

This event is invoked in response to messages from the VMRC plug-in. The `onMessage()` event provides two parameters:

Parameter Name	Type	Description/Notes
`type`	Property Value Constant	The message type. Values can include: VMRC_MESSAGE_INFO VMRC_MESSAGE_WARNING VMRC_MESSAGE_ERROR VMRC_MESSAGE_HINT
`message`	string	The message content.

Known Issues

The following known issues have been observed in the current version of the VMRC plug-in:

- Calling the `connect()` method with empty login data results in an "Unspecified JavaScript Error" instead of returning `false`.
- Calling `getPhysicalClientDevices()`, or any other device methods, while in MKS mode can cause the VMRC plug-in to crash.

NOTE VMRC_DEVICES mode and device methods are currently not supported in VMRC.

- In the Firefox browser, calling `setFullscreen()` returns `true` even if called after a successful disconnection.
- Scroll bars rendered by the VMRC plug-in may not appear in the correct location.
- In Windows, a connection may remain open to a remote host that has been powered off and then powered on again. This functionality is not supported.