Omnissa Horizon RDP Virtual Channel Bridge SDK Programming Guide

For Horizon 7, Horizon 8, and Horizon Cloud Service

Omnissa Horizon RDP Virtual Channel Bridge SDK 4.0

You can find the most up-to-date technical documentation on the Omnissa website at: https://docs.omnissa.com/

Contents

Omnissa Horizon RDP Virtual Channel Bridge SDK Programming Guide

Omnissa Horizon® *RDP Virtual Channel Bridge SDK Programming Guide* provides information about developing applications using Microsoft Windows Remote Desktop Services (RDS) virtual channels on Omnissa Horizon.

With the virtual channel enhancements, the server and client components of the applications can communicate over the PCoIP and Blast virtual channels, in addition to the Remote Desktop Protocol (RDP) virtual channels, with minimal changes to the applications.

Intended Audience

This document is intended for third-party software developers who want to enhance the functionality of their applications that run on virtual desktops provisioned by Omnissa Horizon 7, Omnissa Horizon 8, or Omnissa Horizon Cloud Service. With the enhancements, the RDS virtual channels can provide a better user experience because the applications can run over PCoIP and Blast virtual channels.

NOTE: Within this document, file paths, registry keys, and similar code-related items are revised to reflect the SDK's use with the Omnissa stack. References to items that were for Horizon versions 2406 and earlier versions contain asterisks (***) to redact the previous name.

Introduction

Microsoft Windows Remote Desktop Services (RDS), previously known as Terminal Services (TS), is a set of software services that gives a user access to applications or desktops over a network. RDS uses the desktop remoting protocol, Remote Desktop Protocol (RDP). Omnissa Horizon supports RDP, PCoIP and Blast protocols, to provide a user with the remote access capability. PCoIP and Blast protocols provide an optimized desktop experience for the delivery of the entire desktop environment, including applications, images, audio, and video content.

RDP, PCoIP, and Blast support a feature called a virtual channel, which is a software extension to an RDS application. A virtual channel has a server and a client component. With virtual channels, you can add user experience enhancements to RDS applications without changing the client or server software, or the display protocol.

There are many third-party off-the-shelf plug-ins that are written for RDP virtual channels. Converting these plug-ins to work over PCoIP or Blast require a significant effort. In addition, it becomes necessary to maintain two versions of the plug-in, one for each protocol. The Omnissa Horizon RDP Virtual Channel Bridge, also called RDPVCBridge, helps solve both of these problems by providing a bridge between the RDP virtual channels and the PCoIP or Blast virtual channels. With RDPVCBridge, an application can work over the RDP, PCoIP, or Blast protocols. RDPVCBridge detects whether the application is running in an RDP, PCoIP, or Blast context and determines which protocol to use accordingly.

Benefits of Using the RDPVCBridge SDK

RDS provides the Windows Terminal Services (WTS) API for virtual channels. Omnissa PCoIP or Horizon Blast provides a different API for virtual channels. Without the RDPVCBridge SDK, you must rewrite RDP virtual channel plug-ins to use the PCoIP or Blast virtual channel API. The effort to rewrite the plug-ins can be significant. The

RDPVCBridge SDK makes it much easier to enhance RDS virtual channel applications to support RDP and PCoIP or Blast.

What's New in Omnissa Horizon RDP Virtual Channel Bridge SDK 4.0

The following list summarizes the new features and changes found in version 4.0 of the Omnissa Horizon RDP Virtual Channel Bridge SDK.

- This version of the SDK adds support for the new Omnissa-branded stack and also has backward compatibility with the older version as fallback.
- Make files for Linux and Mac are supplied with the sample.

Supported Platforms and Applications

The RDPVCBridge SDK supports multiple platforms and applications.

- The RDPVCBridge SDK was first released in 2013 and supports all versions of Horizon 7 and Horizon 8 that have been released since then. The RDPVCBridge SDK also supports Horizon Cloud Service on Microsoft Azure.
- The RDPVCBridge SDK supports all versions of Windows, Linux, and Mac clients that Horizon supports.
- The RDPVCBridge SDK does not support Zero clients.

Virtual Channel Security

This topic describes the security features of virtual channels which run over Horizon session connections.

Virtual channels run over the session connection that is established by the remote protocol and rely on security offered by the protocol. The communication over these supported protocols is highly secure and based on industry-recommended security practices. The endpoints negotiate the actual session encryption algorithm that is used by the selected protocol.

In addition, you can increase the security of virtual channels by configuring a list of allowed channels. This configuration allows only the channels in the list to be opened by legitimate requests and prevents all other channels from being opened. To create the allow list, add the channels as registry entries to the .reg file included with the SDK. For more information, see Omnissa Knowledge Base (KB) article 84156.

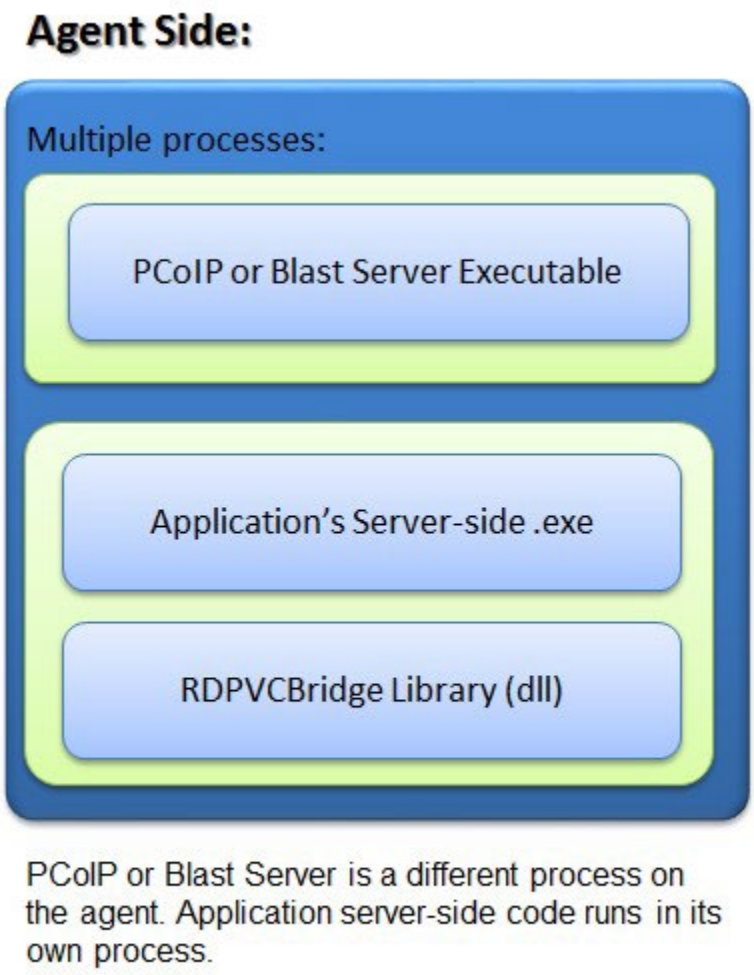
For detailed information about the types of security offered by the supported protocols, see Understanding Client Connections in the Omnissa Documentation.

To configure the cipher suites and protocols used by the client, follow the client-specific procedure described in Configuring Security Protocols and Cipher Suites for Specific Client Types in the Omnissa Documentation.

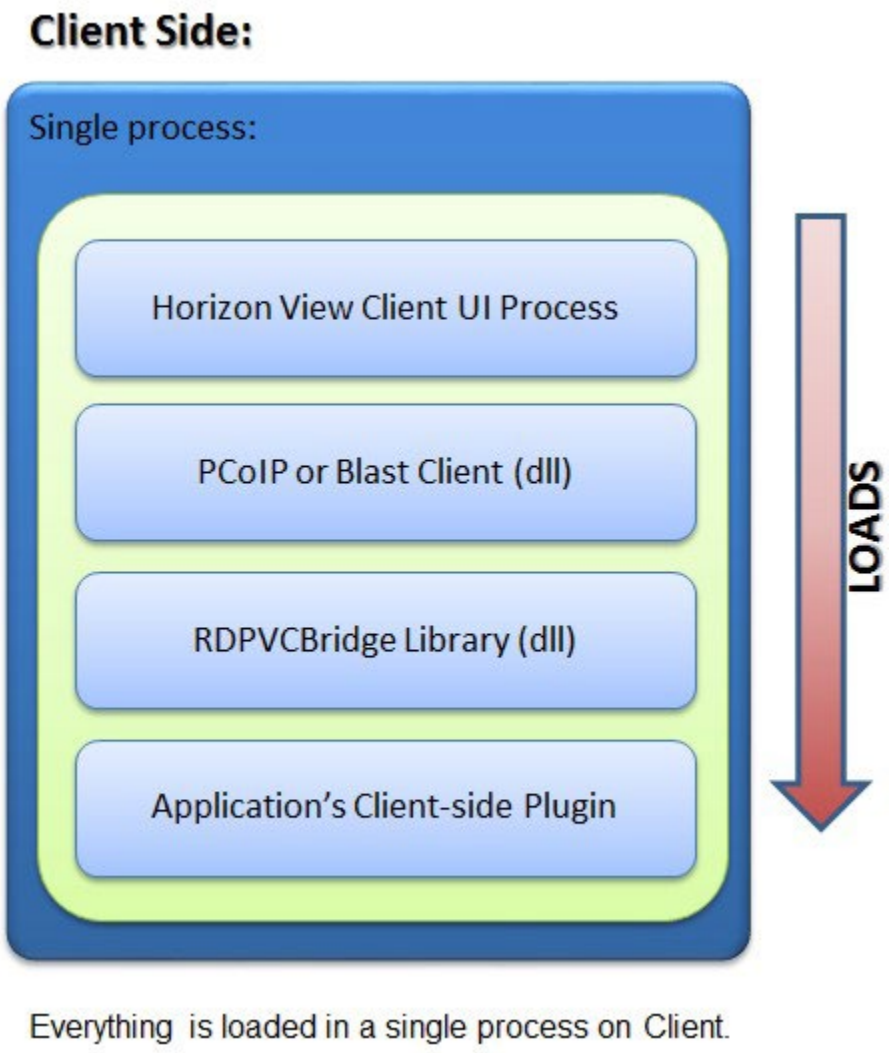
For information about the security features of Horizon Cloud Service, see the Horizon Cloud Service Security Overview technical white paper, available from Omnissa Tech Zone.

How to Use RDPVCBridge SDK

Implementing RDPVCBridge only requires changes to be made on the server-side virtual channel application. There are no changes required on the existing client-side plug-ins.



The preceding image is a pictorial view of the application, PCoIP or Blast Server, and RDPVCBridge on the agent side. The application and RDPVCBridge run in the same process. The PCoIP or Blast Server runs in its own process.



The preceding image is a pictorial view of the application's client-side plug-in, RDPVCBridge, PCoIP or Blast Client, and Omnissa Horizon Client on the client side. All the software components run in the same process.

Prerequisites

You must have the Horizon environment set up and running before you implement RDPVCBridge. For information about setting up Horizon 7 or 8, see the *Horizon Administration* guide which is part of the Omnissa Horizon Documentation. For information about setting up Horizon Cloud Service, see the Omnissa Horizon Cloud Service next-gen Documentation.

To get a better understanding of the required changes to an application, ensure that you have the **vchan-ping** sample application, which is included in the RDPVCBridge SDK, working before you implement RDPVCBridge for other applications.

RDPVCBridge - Server Side

On the server side, changes to the top-level header file of the virtual channel plug-in are required.

Make the following changes to the top-level header file:

1 Define the symbol **MAP_RDPVCBRIDGE**.

Defining the **MAP_RDPVCBRIDGE** symbol creates macros that map the Windows WTS* function calls to Horizon VDP_* function calls.

2 Include the **vdp_rdpvcbridge.h** header file from the RDPVCBridge SDK.

Example:

```
#define MAP_RDPVCBRIDGE
#include "vdp_rdpvcbridge.h"
```

3 Include the source file vdp_rdpvcbridge_import.cpp in your application. This file will find, load and fetch the entry points for the RDPVCBridge SDK.

4 Recompile your application.

5 Deploy the application.

RDPVCBridge - Client Side

On the client side, you do not need to change or recompile any library (.dll, .so, or .dylib) of your virtual channel application. The changes that you must make depend on the type of operating system.

Windows

1 Deploy the application plug-in.

2 Register the application plug-in in the registry.

3 Add a registry key to enable PCoIP or Blast for the plug-in. The registry key has two values:

- **Name** - The value is the complete absolute path of the plug-in.
- Client versions 2412 and later **set Horizon Enabled** - The value is 1.
- Client versions 2406 and earlier **set View Enabled** - The value is 1.

Note The **Horizon/View Enabled** registry entry, with value 1, loads the plug-in by rdpvcbridge for both PCoIP and Blast protocols.

For example, you have a virtual channel application called **VChanApp** and **vchan-app-client.dll** is the client-side plug-in. The location of **vchan-app-client.dll** is **C:\VChanApp\vchan-app-client.dll**.
You must add the following registry key and values:

```
[HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Default\AddIns\VChanApp]
"Name"="C:\\VChanApp\\vchan-app-client.dll"
"Horizon Enabled"=dword:0000001
```

Linux

1. Deploy the application plug-in.
2. Place the client-side shared-object file in the appropriate folder:
   - Client versions 2412 and later - **/lib/.omnissa/rdpvcbridge** or **~/.omnissa/rdpvcbridge**
   - Client versions 2406 and earlier – **/lib/.*****/rdpvcbridge** or **~/.*****/rdpvcbridge**

Mac

3. Deploy the application plug-in.
4. Place the client-side shared-object file in the appropriate folder:
   - Client versions 2412 and later - **/Library/.omnissa/rdpvcbridge** or **~/.omnissa/rdpvcbridge**
   - Client versions 2406 and earlier – **/Library/.*****/rdpvcbridge** or **~/.*****/rdpvcbridge**

RDPVCBridge SDK Source Files

The RDPVCBridge SDK contains the following source files.

- **vdp_rdpvcbridge.h** - The header file that maps WTS function calls to VDP_ function calls.
- **vdp_rdpvcbridge_import.cpp** - C++ source code for the import library that you can use with your application.

The **vdp_rdpvcbridge.dll** file is delivered with Horizon on both the agent and client sides. Use the version of vdp_rdpvcbridge.dll that is included with Horizon Agent and Horizon Client. Do not use the versions of vdp_rdpvcbridge.dll that came with previous versions of the RDPVCBridge SDK.

Remote Desktop Services API Support

RDPVCBridge supports a subset of RDS API functions.

Client-side API Functions

- VirtualChannelClose: https://docs.microsoft.com/en-us/windows/win32/api/cchannel/nccchannel-virtualchannelclose
- VirtualChannelEntry: https://docs.microsoft.com/en-us/windows/win32/api/cchannel/nccchannel-virtualchannelentry
- VirtualChannelInit: https://docs.microsoft.com/en-us/windows/win32/api/cchannel/nccchannel-virtualchannelinit
- VirtualChannelInitEvent: https://docs.microsoft.com/en-us/windows/win32/api/cchannel/nccchannel-channel_init_event_fn
- VirtualChannelOpen: https://docs.microsoft.com/en-us/windows/win32/api/cchannel/nccchannel-virtualchannelopen

- VirtualChannelOpenEvent: https://docs.microsoft.com/en-us/windows/win32/api/ cchannel/nc-cchannel-channel_open_event_fn
- VirtualChannelWrite: https://docs.microsoft.com/en-us/windows/win32/api/cchannel/nccchannel-virtualchannelwrite

Server-side API Functions

- GetSystemMetrics (SM_REMOTESESSION): https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-getsystemmetrics
- WTSFreeMemory: https://docs.microsoft.com/en-us/windows/win32/api/wtsapi32/nfwtsapi32-wtsfreememory
- WTSQuerySessionInformationA: https://docs.microsoft.com/en-us/windows/win32/api/ wtsapi32/nf-wtsapi32-wtsquerysessioninformationa
- WTSQuerySessionInformationW: https://docs.microsoft.com/en-us/windows/win32/api/ wtsapi32/nf-wtsapi32-wtsquerysessioninformationw
- WTSRegisterSessionNotification: https://docs.microsoft.com/en-us/windows/win32/api/ wtsapi32/nf-wtsapi32-wtsregistersessionnotification
- WTSUnRegisterSessionNotification: https://docs.microsoft.com/en-us/windows/win32/api/ wtsapi32/nf-wtsapi32-wtsunregistersessionnotification
- WTSVirtualChannelClose: https://docs.microsoft.com/en-us/windows/win32/api/wtsapi32/nfwtsapi32-wtsvirtualchannelclose
- WTSVirtualChannelOpen: https://docs.microsoft.com/en-us/windows/win32/api/wtsapi32/nfwtsapi32-wtsvirtualchannelopen
- WTSVirtualChannelOpenEx: https://docs.microsoft.com/en-us/windows/win32/api/ wtsapi32/nf-wtsapi32-wtsvirtualchannelopenex
- WTSVirtualChannelPurgeInput: https://docs.microsoft.com/en-us/windows/win32/api/ wtsapi32/nf-wtsapi32-wtsvirtualchannelpurgeinput
- WTSVirtualChannelPurgeOutput: https://docs.microsoft.com/en-us/windows/win32/api/ wtsapi32/nf-wtsapi32-wtsvirtualchannelpurgeoutput
- WTSVirtualChannelQuery: https://docs.microsoft.com/en-us/windows/win32/api/ wtsapi32/nf-wtsapi32-wtsvirtualchannelquery
- WTSVirtualChannelRead: https://docs.microsoft.com/en-us/windows/win32/api/wtsapi32/nfwtsapi32-wtsvirtualchannelread
- WTSVirtualChannelWrite: https://docs.microsoft.com/en-us/windows/win32/api/wtsapi32/nfwtsapi32-wtsvirtualchannelwrite

Non-WTS API Support

RDPVCBridge supports the following non-WTS API functions.

Client-Side API Function

- VDP_IsNestedClient

  This API is called by the client plug-in that was loaded by the client-side RDPVCBridge.

  API Signature:

  ```
  BOOL VDP_IsNestedClient(const DWORD sessionID, BOOL* isNestedClient);
  ```

  Example:

  ```
  BOOL isSis;
  BOOL ok = VDP_IsNestedClient(WTS_CURRENT_SESSION, &isSis);
  if (!ok) {
    LOG_MESSAGE("VDP_IsNestedClient() failed\n");
  } else if (!isSis) {
    LOG_MESSAGE("This client is not in a nested session\n");
  } else {
    LOG_MESSAGE("This client is in a nested session\n");
  }
  ```

Agent-side API Functions

- VDP_GetSDKVersion

  This function returns the current version of Horizon stored in the DLL properties, for example, "8.1.0".

  API Signature:

  ```
  BOOL VDP_GetSDKVersion(const DWORD sessionID,
                         PCHAR localVersionBuffer,
                         const ULONG localVersionBufferSize,
                         PULONG pLocalVersionResultStrSize,
                         PULONG pLocalVersionNum,
                         PCHAR  remoteVersionBuffer,
                         const ULONG remoteVersionBufferSize,
                         PULONG pRemoteVersionResultStrSize,
                         PULONG pRemoteVersionNum);
  ```

  Example:

```
char   localVersionStr[1024];
ULONG  localVersionStrLen = ARRAYSIZE(localVersionStr);
ULONG  localResultStrSize = 0;
ULONG  localVersionNum = 0;
char  remoteVersionStr[1024];
ULONG  remoteVersionStrLen = ARRAYSIZE(remoteVersionStr);
ULONG  remoteResultStrSize = 0;
ULONG  remoteVersionNum = 0;

BOOL ok = VDP_GetSDKVersion(WTS_CURRENT_SESSION,
                    localVersionStr, localVersionStrLen,
                    &localResultStrSize,  &localVersionNum,
                    remoteVersionStr, remoteVersionStrLen,
                    &remoteResultStrSize, &remoteVersionNum);
If (!ok) {
  LOG_MESSAGE("VDP_GetSDKVersion () failed\n");
} else {
   LOG_MESSAGE("Local SDK version: v%d - v%s\n", localVersionNum, localVersionStr);

    // Note: remoteVersionNum will be 0 if not in a Horizon session
    if (remoteVersionNum != 0) {
     LOG_MESSAGE("RemoteSDK version: v%d - v%s\n",
                        remoteVersionNum, remoteVersionStr);
    }
}
```

- VDP_IsOmnissaHorizonInstalled
- VDP_IsLegacyHorizonInstalled

The Agent queries the local system to determine which version of Horizon is installed; Omnissa or Legacy. You must check VDP_IsOmnissaHorizonInstalled first and then VDP_IsLegacyHorizonInstalled second. Reversing the order may lead to incorrect results due to drivers installed by Horizon to allow applications that are designed to run on legacy Horizon to run on Omnissa Horizon.

API Signature:

```
BOOL VDP_IsOmnissaHorizonInstalled();
BOOL VDP_IsLegacyHorizonInstalled();
```

Example:

```
If (VDP_IsOmnissaHorizonInstall()) {
  LOG_MESSAGE(“Omnissa Horizon installed\n”);
} else if (VDP_IsLegacyHorizonInstalled()) {
  LOG_MESSAGE(“Legacy Horizon installed\n”);
} else {
  LOG_MESSAGE(“Horizon is not installed\n”);
}
```

- VDP_IsHorizonSession

The Agent queries the local system to verify if it is running in a Horizon session and returns a TRUE value if it is. The sessions supported are those using any of the protocols that Horizon supports: Blast, PCoIP, or RDP. The Agent's query returns a FALSE when the local system is not running in a Horizon session, including not being in

a remote session or establishing a remote session using a different application, such as Microsoft Remote Desktop Connection.

API Signature:

```
BOOL VDP_IsHorizonSession(DWORD sessionID);
```

Example:

```
if (!VDP_IsHorizonSession(WTS_CURRENT_SESSION)) {
    LOG_MESSAGE("We are not currently in a Horizon session\n");
} else {
    LOG_MESSAGE("Invoked from an ongoing Horizon session\n");
}
```

- VDP_IsBlastSession

  The Agent queries the local system to verify if it is running in a Blast session and returns a TRUE value if it is.

  API Signature:

  ```
  BOOL VDP_IsBlastSession(DWORD sessionID);
  ```

  Example:

  ```
  if (!VDP_IsBlastSession(WTS_CURRENT_SESSION)) {
      LOG_MESSAGE("We are not currently in a Horizon Blast session\n");
  } else {
      LOG_MESSAGE("Invoked from an ongoing Horizon Blast session\n");
  }
  ```

- VDP_IsPCoIPSession

  The Agent queries the local system to verify if it is running in a PCoIP session and returns a TRUE value if it is.

  API Signature:

  ```
  BOOL VDP_IsPCoIPSession(DWORD sessionID);
  ```

  Example:

  ```
  if (!VDP_IsPCoIPSession(WTS_CURRENT_SESSION)) {
      LOG_MESSAGE("We are not currently in a Horizon PCoIP session\n");
  } else {
      LOG_MESSAGE("Invoked from an ongoing Horizon PCoIP session\n");
  }
  ```

- VDP_IsRDPSession

  The Agent queries the local system to verify if it is running in an RDP session and returns a TRUE value if it is. This includes both Horizon and Microsoft RDP sessions.

  API Signature:

  ```
  BOOL VDP_IsRDPSession(DWORD sessionID);
  ```

Example:

```
if (!VDP_IsRDPSession(WTS_CURRENT_SESSION)) {
    LOG_MESSAGE("We are not currently in an RDP session! \n");
} else if (VDP_IsHorizonSession(WTS_CURRENT_SESSION)) {
    LOG_MESSAGE("Invoked from an ongoing Horizon RDP session\n");
} else {
    LOG_MESSAGE("Invoked from an ongoing Mircosoft RDP session\n");
}
```

- VDP_IsDesktopSession

  The Agent queries the local system to verify if it is running in a remote desktop session and returns a TRUE value if it is.

  API Signature:

  ```
  BOOL VDP_IsDesktopSession(DWORD sessionID);
  ```

  Example:

  ```
  if (VDP_IsDesktopSession(WTS_CURRENT_SESSION)) {
    LOG_MESSAGE("Invoked from an ongoing Horizon remote desktop session\n");
  }
  ```

- VDP_IsApplicationSession

  The Agent queries the local system to verify if it is running in a remote application session and returns a TRUE value if it is.

  API Signature:

  ```
  BOOL VDP_IsApplicationSession(DWORD sessionID);
  ```

  Example:

  ```
  if (VDP_IsApplicationSession(WTS_CURRENT_SESSION)) {
    LOG_MESSAGE("Invoked from an ongoing Horizon remote application session\n");
  }
  ```

- VDP_IsNestedSession

  When this function is called in the Agent's context (the ideal use case), the Agent sends a message to the Client asking if the Client is running in an ongoing Horizon session.

  API Signature:

  ```
  BOOL VDP_IsNestedSession(const DWORD sessionID, BOOL* isNestedSession);
  ```

Example:

```
BOOL isSis;
BOOL ok = VDP_IsNestedSession(WTS_CURRENT_SESSION, &isSis);
if (!ok) {
  LOG_MESSAGE("VDP_IsNestedSession() failed\n");
} else if (!isSis) {
  LOG_MESSAGE("The remote client is not in nested session\n");
}  else {
  LOG_MESSAGE("The remote client is in a nested session\n");
}
```

Dynamic Virtual Channel API Support

RDS Dynamic Virtual Channel (DVC) API extends the existing Virtual Channel API. The existing API is now referred to as Static Virtual Channel (SVC) API. RDPVCBridge supports DVC API in addition to SVC API. See Microsoft documentation for more details on Dynamic Virtual Channels and the DVC API.

Sample Virtual Channel Application

The RDPVCBridge SDK contains a sample virtual channel application called **vchan-ping**. This application creates a virtual channel and sends a simple string using the virtual channel four times to the Client.

vchan-ping has a client-side component and an agent-side component. The client-side component runs on a client machine. The agent-side component runs in a Horizon virtual desktop machine.

The RDPVCBridge SDK includes the source code of **vchan-ping**, the Visual Studio solution file for building it on Windows, and a Makefile for building it on Linux and Mac.

Setting up vchan-ping on the Client

- On Windows systems:
  - Build **vchan-ping-client.dll** using the Visual Studio solution file.
  - Copy the file **vchan-ping-client.dll** to the client.
  - Register the DLL using the following command:
    - regsvr32.exe vchan-ping-client.dll
- On Linux systems:

  Build **libvchanpingclient.so** using the supplied Makefile.

  Copy the file **libvchanpingclient.so** to the appropriate folder (the Makefile may have already done this):
  - Client versions 2412 and later - **~/.omnissa/rdpvcbridge or /lib/.omnissa/rdpvcbridge**
  - Client versions 2406 and earlier - **~/.*****/rdpvcbridge or /lib/.*****/rdpvcbridge**
- On Mac systems:

  Build **libvchanpingclient.dylib** using the supplied Makefile.

  Copy the file **libvchanpingclient.dylib** to the appropriate folder (the Makefile may have already done this):
  - Client versions 2412 and later - **~/.omnissa/rdpvcbridge or /Library/.omnissa/rdpvcbridge**
  - Client versions 2406 and earlier - **~/.*****/rdpvcbridge or /Library/.*****/rdpvcbridge**

Setting up vchan-ping on the Agent

1 Build **vchan-ping.exe** using the Visual Studio solution file.

2 Copy the files **vchan-ping.exe** to a folder.

3 Establish a Horizon connection from the client to the agent.

4 Run **vchan-ping.exe** on the agent from a command prompt.

Debugging and Collecting Logs

RDPVCBridge writes messages to log files to facilitate debugging.

On the agent-side, for every application that loads the DLL, a separate log file is created. On the client side, RDPVCBridge creates a single log file for all virtual channel plug-ins. Logging is set to INFO by default and can be changed to ERROR, WARN, INFO, DEBUG, or TRACE. Use the following information to change the logging level.

- Versions 2412 and later: the files will be named
  - Client: horizon-rdpvcbridge-Client-<sid>-<pid>__<date-time>.log
  - Server: horizon-rdpvcbridge-<appName>-<sid>-<pid>__<date-time>.log

- Versions 2406 and earlier: the files will be named
  - Client: *****-rdpvcbridge-Client-<sid>-<pid>__<date-time>.log
  - Server: *****-rdpvcbridge-<appName>-<sid>-<pid>__<date-time>.log

Windows

1 To change the default logging level, set the following registry entries on both the Agent and the Client systems. The following example changes the logging level to TRACE.

- Versions 2412 and later:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Omnissa\Horizon\rdpvcbridge]
"LogLevel"="trace"
```

- Versions 2406 and earlier:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\*****, Inc.\***** VDM\rdpvcbridge]
"LogLevel"="trace"
```

2 Disconnect from the Horizon desktop, if you are connected, and start the session again.

The log files can be found in the following locations.

Table 1-1. Location of Log Files on a Windows System

Log File Type	Location
Client-side plug-in log file	Versions 2412 and later - **C:\Users\<USER-NAME>\AppData\Local\Temp\omnissa-<USER-NAME>** Versions 2406 and earlier - **C:\Users\<USER-NAME>\AppData\Local\Temp*****-<USER-NAME>**
Agent-side application log file when the application is running under the user account	Versions 2412 and later - **C:\Users\<USER-NAME>\AppData\Local\Temp\omnissa-<USER-NAME>** Versions 2406 and earlier - **C:\Users\<USER-NAME>\AppData\Local\Temp*****-<USER-NAME>**

Agent-side application log file when the application is running under the system account	Versions 2412 and later - **C:\ProgramData\Omnissa\Horizon\logs** Versions 2406 and earlier - **C:\ProgramData******\VDM\logs**

Linux

1 Create a configuration file in the appropriate location:

- Versions 2412 and later - **~/.omnissa/config**
- Versions 2406 and earlier - **~/.*****/config**

2 Add the following line at the end of the configuration file.

```
Versions 2412 and later: rdpvcbridge.log.logLevel=trace
Versions 2406 and earlier: rdpvcbridge.rdpvcbridgeLogger.logLevel=trace
```

3 Restart the Horizon Client.

The log file is located in one of the following locations:

- Versions 2412 and later - **/tmp/omnissa-<USER-NAME>**
- Versions 2406 and earlier - **/tmp/*****-<USER-NAME>**

Mac

1 Create a configuration file in the appropriate location:

- Versions 2412 and later - **~/Library/Preferences/Omnissa Horizon View/config**
- Versions 2406 and earlier - **~/Library/Preferences/***** Horizon View/config**

2 Add the following at the end of the configuration file.

```
Versions 2412 and later: rdpvcbridge.log.logLevel=trace
Versions 2406 and earlier: rdpvcbridge.rdpvcbridgeLogger.logLevel=trace
```

3 Restart the Horizon Client.

The log file is located in one of the following locations:

- Versions 2412 and later - **~/Library/Logs/Omnissa**
- Versions 2406 and earlier - **~/Library/Logs/*****

Setting up a Developer Environment Using VADC

You can quickly set up an environment to test the RDPVCBridge SDK at a small scale by using Horizon Agent Direct Connect (also known as VADC). VADC allows you to connect Horizon Client to the Horizon server without going through Horizon Connection Server.

Prerequisites

- Install VMware Workstation.

Procedure

1. Start VMware Workstation.
2. Create a virtual machine (VM) running a version of Windows that supports Horizon Agent. For the list of supported operating systems, see the Omnissa Knowledge Base (KB) articles https://kb.omnissa.com/s/article/78714 and https://kb.omnissa.com/s/article/78715.

   Select hardware version 8 when prompted. Select bridged networking when prompted so that the VM has its own IP address.
3. Install VMware Tools.

   If you use the wizard to set up the VM from an ISO image file, this step is not necessary.
4. Shut down the VM.
5. Find the VMX file for the VM and open it in a text editor.
6. Add the following line to the file and save it.

   ```
   machine.id = "vdi.broker.asmanaged=1"
   ```
7. Power on the VM.
8. Install Horizon Agent.
9. Install the VADC plug-in.
10. (Optional) Shut down the VM and create a snapshot.

    The snapshot allows you to roll back your changes.
11. Connect the Horizon client to the VM using the VM's IP address.

Example

You now have a VM that is a Horizon desktop. The Horizon desktop is running with the PCoIP or Blast protocol. A Horizon client can connect to it directly over PCoIP or Blast. You can use the RDPVCBridge SDK to implement RDPVCBridge in this VM. With this setup, you do not need to deploy:

- The Active Directory and DNS servers.
- VMware vSphere.
- Horizon Connection Server.

Software Requirements

This topic describes the software prerequisites for the Omnissa Horizon RDP Virtual Channel Bridge SDK.

To use the RDPVCBridge SDK, verify that you have installed the following components in your system environment:

- Omnissa Horizon infrastructure
- Horizon Agent (on each remote desktop)
- Horizon Client (on each client system)

How Components Are Delivered and Supported

The agent-side components are included with Horizon Agent. The client-side components are included with Horizon Client.

You must use the version of the vdp_rdpvcbridge.dll that is installed with Horizon Agent and Horizon Client to ensure compatibility with previous and future releases of Horizon.

The RDPVCBridge SDK is available free for download and can be used immediately.

Support Policy

The RDPVCBridge SDK is free, public, and unsupported. However, we do offer Paid Development Consulting on request. Contact us at Horizon-View-RDPVCBridge-SDKSupport@omnissa.com for more information.

Omnissa might periodically release bug fixes:

- Bugs in the client-side components are fixed as part of a Horizon Client release.
- Bugs in the agent-side components are fixed as part of a Horizon release.
- The RDPVCBridge SDK is updated when new features are added to the SDK.