VMware Horizon RDP Virtual Channel Bridge SDK Programming Guide

For Horizon 7, Horizon 8, and Horizon Cloud Service on Microsoft Azure

VMware Horizon RDP Virtual Channel Bridge SDK 3.3

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

VMware Horizon RDP Virtual Channel Bridge SDK Programming Guide

VMware Horizon® RDP Virtual Channel Bridge SDK Programming Guide provides information about developing applications using Microsoft Windows Remote Desktop Services (RDS) virtual channels on VMware 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 a VMware Horizon 7, VMware Horizon 8, or VMware Horizon Cloud Service on Microsoft Azure virtual desktop. With the enhancements, the RDS virtual channels can provide a better user experience because the applications can run over PCoIP and Blast virtual channels.

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). VMware Horizon supports RDP and PCoIP or 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 VMware 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. VMware PCoIP or Blast provides a different API for virtual channels. Without the RDPVCBridge SDK, you must rewrite RDP virtual channel plug-ins to use the VMware 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 VMware Horizon RDP Virtual Channel Bridge SDK 3.3

The following list summarizes the new features and changes found in version 3.3 of the VMware Horizon RDP Virtual Channel Bridge SDK.

- This version of the SDK no longer ships with a copy of `vdp_rdpvcbridge.dll`. To preserve compatibility with previous and future releases of Horizon, you must use the copy of `vdp_rdpvcbridge.dll` that is installed with the Horizon agent and client software. For more information, see How Components Are Delivered and Supported.
- This version of the SDK includes a `.cpp` file that replaces the import library from previous versions. The `.cpp` file provides API entry points and the code for loading `vdp_rdpvcbridge.dll`.

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 VMware Knowledge Base (KB) article 84156.

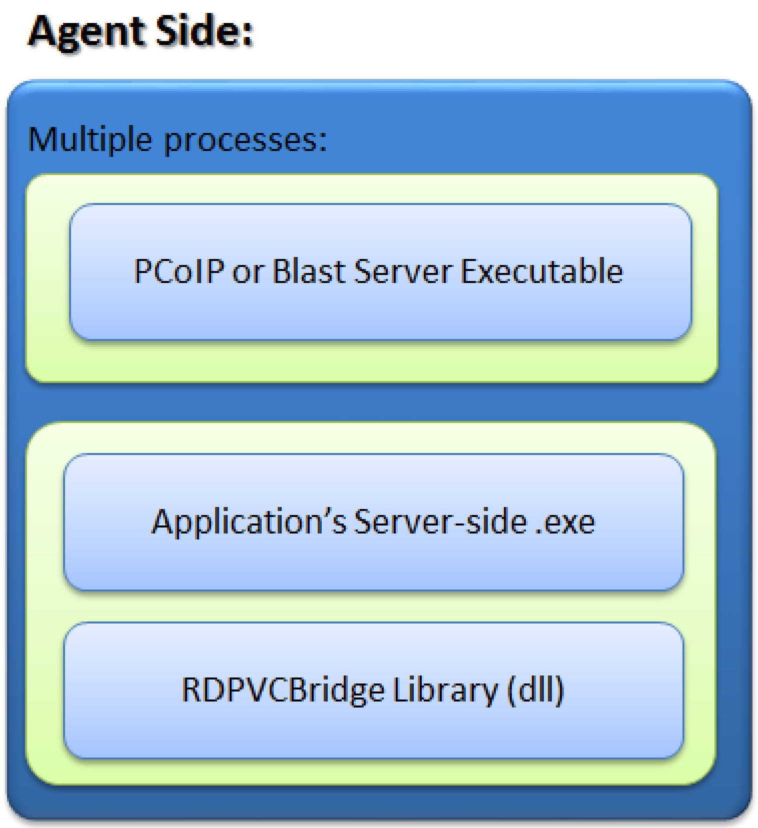
For detailed information about the types of security offered by the supported protocols, see the “Understanding Client Connections” topic in the *Horizon Architecture Planning* guide, which is part of the VMware Horizon Documentation.

To configure the cipher suites and protocols used by the client, follow the client-specific procedure described in the “Configuring Security Protocols and Cipher Suites for Specific Client Types” topic in the *Horizon Client and Agent Security* guide, which is part of the VMware Horizon Documentation.

For information about the security features of Horizon Cloud Service on Microsoft Azure, see the VMware Horizon Cloud Service on Microsoft Azure Security Considerations technical white paper, available from VMware Digital Workspace Tech Zone.

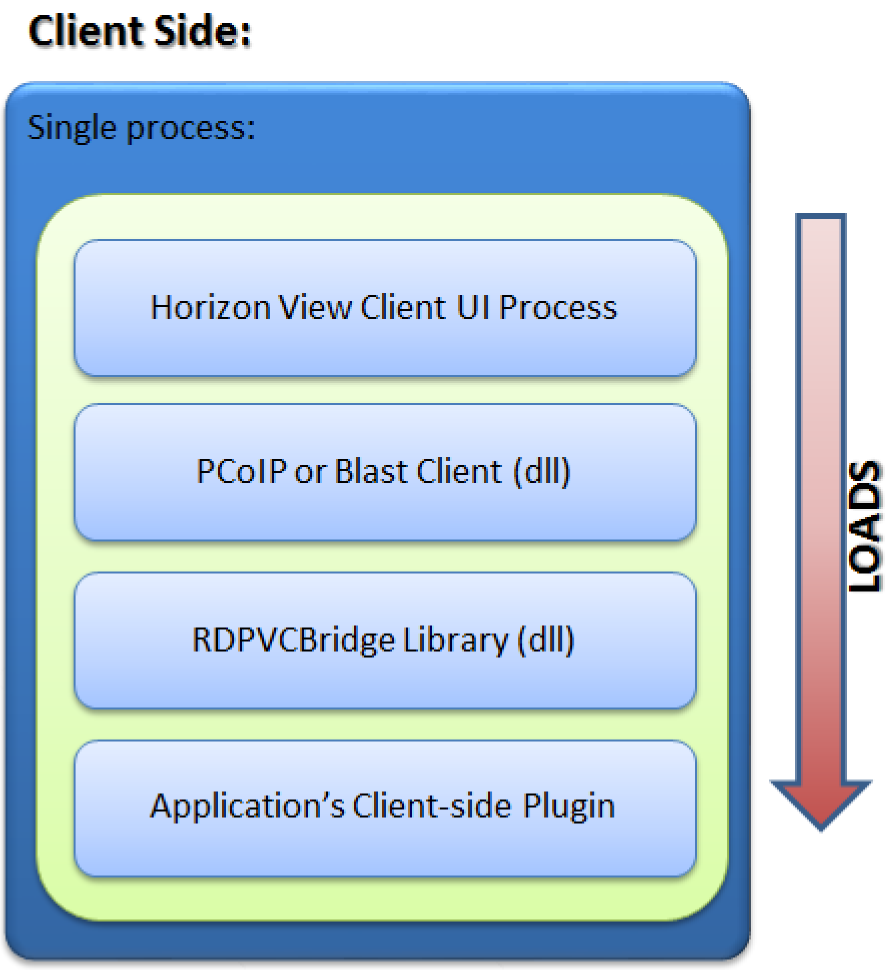
How to Use RDPVCBridge SDK

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.


Agent Side:
Multiple processes:
PCoIP or Blast Server Executable
Application's Server-side .exe
RDPVCBridge Library (dll)

PCoIP or Blast Server is a different process on the agent. Application server-side code runs in its own process.

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.


Client Side:
Single process:
Horizon View Client UI Process
PCoIP or Blast Client (dll)
RDPVCBridge Library (dll)
Application's Client-side Plugin
LOADS

Everything is loaded in a single process on Client.

The preceding image is a pictorial view of the application's client-side plug-in, RDPVCBridge, PCoIP or Blast Client, and VMware 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 VMware Horizon Documentation. For information about setting up Horizon Cloud Service on Microsoft Azure, see the VMware Horizon Cloud Service 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 7 VDP_* function calls.

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

  Example:

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

3 Compile the source file `vdp_rdpvcbridge_import.cpp` into your application.

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.
  - **View Enabled** - The value is 1.

Note The **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"
"View Enabled"=dword:0000001
```

Linux

1 Deploy the application plug-in.

2 Place the client-side shared-object file in the **/usr/lib/vmware/rdpvcbridge** folder.

Mac OS X

For instructions on using RDPVCBridge with a Mac client, see VMware Knowledge Base (KB) article 85188.

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/nc-cchannel-virtualchannelclose
- VirtualChannelEntry: https://docs.microsoft.com/en-us/windows/win32/api/cchannel/nc-cchannel-virtualchannelentry
- VirtualChannelInit: https://docs.microsoft.com/en-us/windows/win32/api/cchannel/nc-cchannel-virtualchannelinit
- VirtualChannelInitEvent: https://docs.microsoft.com/en-us/windows/win32/api/cchannel/nc-cchannel-channel_init_event_fn
- VirtualChannelOpen: https://docs.microsoft.com/en-us/windows/win32/api/cchannel/nc-cchannel-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/nc-cchannel-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/nf-wtsapi32-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/nf-wtsapi32-wtsvirtualchannelclose
- WTSVirtualChannelOpen: https://docs.microsoft.com/en-us/windows/win32/api/wtsapi32/nf-wtsapi32-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/nf-wtsapi32-wtsvirtualchannelread
- WTSVirtualChannelWrite: https://docs.microsoft.com/en-us/windows/win32/api/wtsapi32/nf-wtsapi32-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 == FALSE) {
   LOG_MESSAGE("The VDP_IsNestedClient() Failed! \n");
   return;
}
if (isSis == FALSE) {
   LOG_MESSAGE("This Client is Not in Nested Session. \n");
} else {
   LOG_MESSAGE("This Client is in 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);
```

- `VDP_IsViewSession`

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_IsViewSession(DWORD sessionID);
```

Example:

```
    if (!VDP_IsViewSession(WTS_CURRENT_SESSION))
        {
        LOG_ERROR("We are not currently in View Session! \n");
        return false;
        }
    else
        {
        LOG_MESSAGE("Invoked from an ongoing View 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 == FALSE) {
   LOG_MESSAGE("The VDP_IsNestedSession() Failed! \n");
   return;
}
if (isSis == FALSE) {
   LOG_MESSAGE("The Remote-side Client is Not in Nested Session. \n");
} else {
   LOG_MESSAGE("The Remote-side Client is in 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.

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 solution file for building it on Windows, and a `Makefile` for building it on Linux.

Setting up `vchan-ping` on the Client

- On 32-bit Windows systems:

  a Copy the file **vchan-ping-client.dll** to the client.

  b Register the DLL using the following command:

  ```
  regsvr32.exe vchan-ping-client.dll
  ```

- On 64-bit Windows systems:

  a Copy the file **vchan-ping-client.dll** to the client.

  b Register the DLL using the following command:

  ```
  regsvr32.exe vchan-ping-client.dll
  ```

- On Linux systems:

  Copy the file **libvchanpingclient.so** to the folder **/usr/lib/vmware/rdpvcbridge**.

Setting up `vchan-ping` on the Agent

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

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

3 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.

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`.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\VMware, Inc.\VMware 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	`C:\Users\`*<USER-NAME>*`\AppData\Local\Temp\vmware-`*<USER-NAME>*`\vmware-rdpvcbridge-Client-`*<pid>*`.log`
Agent-side application log file when the application is running under the user account	`C:\Users\`*<USER-NAME>*`\AppData\Local\Temp\vmware-`*<USER-NAME>*`\vmware-rdpvcbridge-`*<APP_NAME>*`-`*<pid>*`.log`
Agent-side application log file when the application is running under the system account	`C:\ProgramData\VMware\VDM\logs\vmware-rdpvcbridge-`*<APP_NAME>*`-`*<pid>*`.log`

Linux

1 Create a configuration file at `~/.vmware/config`.

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

```
rdpvcbridge.rdpvcbridgeLogger.logLevel=trace
```

3 Restart the Horizon Client.

The log file is located at `/tmp/vmware-`*<USER-NAME>*`/vmware-rdpvcbridge-Client-`*<pid>*`.log`.

Mac OS X

1 Create a configuration file at `~/Library/Preferences/VMware Horizon View/config`.

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

```
rdpvcbridge.rdpvcbridgeLogger.logLevel=trace
```

3 Restart the Horizon Client.

The log file is located at `/Users/`*<USER-NAME>*`/Library/Logs/VMware/vmware-rdpvcbridge-Client-`*<pid>*`.log`.

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 VMware View Agent Direct Connect (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 VMware Knowledge Base (KB) articles https://kb.vmware.com/s/article/78714 and https://kb.vmware.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 VMware Horizon RDP Virtual Channel Bridge SDK.

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

- VMware 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-SDK-Support@vmware.com for more information.
- VMware 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.