Virtual Channel SDK

For full documentation visit Teradici Support.

Introduction

The PCoIP Virtual Channel Software Development Kit (SDK) enables developers to build custom PCoIP Virtual Channel plug-ins for PCoIP sessions. You can implement PCoIP Virtual Channel functionality as a plug-in to send encrypted data between servers and client endpoints during an active PCoIP session.

The PCoIP Virtual Channel Application Programming Interface (API) is available as an optional add-on to solution developers who want to extend the types of traffic flowing through the PCoIP session, such as clipboard redirection, local printing, and customised device support.

Who Should Read This Guide?

This guide provides information for solution developers to create custom PCoIP solutions or PCoIP commercial products using the Virtual Channel SDK. Solution developers can build custom PCoIP Virtual Channel plug-ins that stream data between the desktop/agent and the client using a secure PCoIP session. In this guide you'll learn about:

- Implementing and Building PCoIP Virtual Channel plug-in files
- Installing and Updating PCoIP Virtual Channel plug-in files
- Troubleshooting Performance and Implementation issues

Overview of the PCoIP Virtual Channel

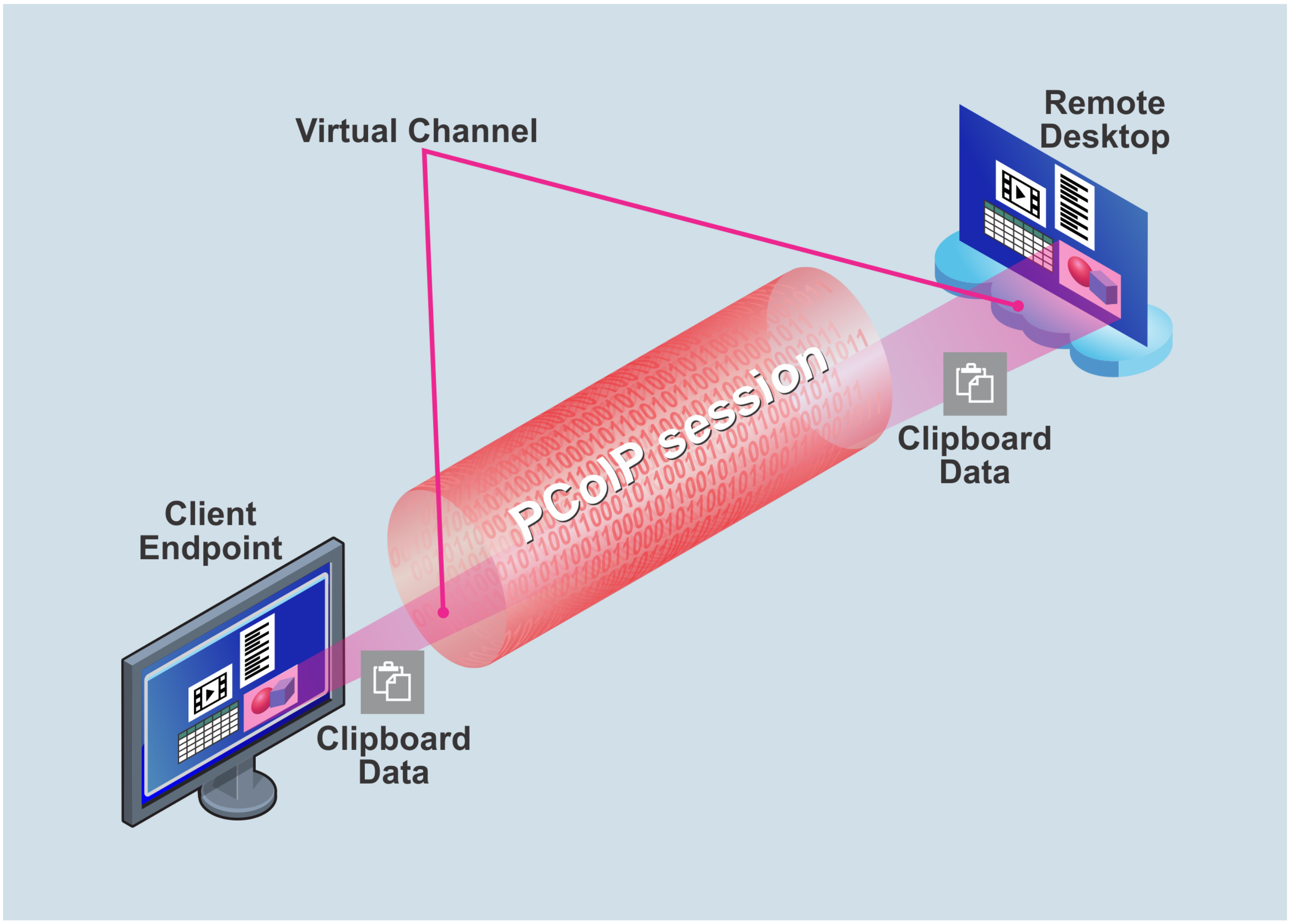
The PCoIP Virtual Channel Software Development Kit (SDK) enables developers to build custom PCoIP Virtual Channel plug-ins for PCoIP sessions. You can implement PCoIP Virtual Channel functionality as a plug-in to send encrypted data between servers and client endpoints during an active PCoIP session.

The PCoIP Virtual Channel Application Programming Interface (API) is available as an optional add-on to solution developers who want to extend the types of traffic flowing through the PCoIP session, such as clipboard redirection, local printing, and custom device support.

About the PCoIP Virtual Channel API/SDK

Solution developers can write plug-ins that stream data between the agent and the client using a secure PCoIP session. For example, the PCoIP copy-and-paste function uses a virtual channel to transport clipboard data between a remote desktop client and the clipboard on a local PC, as shown next.

An example use case of the PCoIP Virtual Channel



Virtual channel traffic travels over the PCoIP session and is therefore secure since it is encrypted and authenticated like other PCoIP traffic.

The virtual channel does not set up a new socket or port number and the channel can be opened and closed dynamically.

Both reliable and unreliable transport options are available. If PCoIP Virtual Channel traffic can be compressed and still meet WAN criteria, the API will automatically apply lossless compression. The API periodically checks if traffic can be compressed. Plug-ins can explicitly disable compression.

During the session, the PCoIP Virtual Channel API works to:

- Transfer data in either direction using streaming or datagram transfers.
- Receive notifications of relevant events via callbacks.
- Log messages in the standard PCoIP session logs.

This SDK includes sample plug-ins which demonstrate how to implement plug-ins and use the PCoIP Virtual Channel API. These sample plug-ins use the PCoIP Virtual Channel API to transfer data between client and server endpoints:

- *plugin_vchan_sample_stream* uses the reliable PCoIP Virtual Channel streaming data API.
- *plugin_vchan_sample_dgram* uses the reliable PCoIP Virtual Channel datagram API.
- *plugin_vchan_sample_u_echo* uses the unreliable PCoIP Virtual Channel API to send unreliable datagrams in parallel with the reliable data transfer.

Guidelines for Implementing PCoIP Virtual Channel Plug-Ins

The following section lists guidelines, best practices, recommendations, and warnings for solution developers who wish to implement plug-ins:

Do not modify the ~*\VChan_SDK\inc* folder

This folder is used directly within the PCoIP executable and must not be modified.

- On Windows Agents you can build PCoIP Virtual Channel plugins with Visual Studio 2015 and higher. However, as the PCoIP Agent installation package only installs Visual Studio 2015 re-distributable libraries, it is your responsibility to ensure that required re-distributables are installed, if you choose to build with a different version.
- You are responsible for providing a mechanism to change run-time variables within plug-ins. The PCoIP Virtual Channel SDK does not provide a built-in mechanism to enable a user or customer to change behavior, for example, using a registry key or config file. -You can build a plug-in implemented as a Windows *.dll* file, a Linux *.so* file, an Android *.mk* file, or an iOS or Android library. This enables you to use the PCoIP Virtual Channel API to access virtual channel functionality, including:
    - `init` and `exit` functions, for both server and client, that are called at session startup and session exit when the plug-in is coded and built. This is a pre-requisite to enable the subsequent installation of the plug-in on server and client.
    - `init` and `exit` functions within the plug-in and the exposed API that the plug-in uses to access the PCoIP Virtual Channel plug-in services.
    - `init` functions that allocate resources, such as threads, semaphores, and so on, to implement the autonomous processing and functionality for server and client endpoints.
    - `plug-in exit` functions that release resources and tidy up the processing to prepare for terminating the session.

Build plug-ins for all deployed host and client operating systems

A PCoIP Virtual Channel plug-in should be built for all host and client operating systems on which it is to be deployed.

- You can implement a PCoIP Virtual Channel plug-in as a single symmetric plug-in, one that is deployed on both client and host, or two asymmetric plug-ins, one implementing client functionality and one host functionality. Note that even a symmetric plug-in may need to be build for different operating systems for the host and client.
- Do not mix streaming and datagram API calls on a single channel.
- To avoid degrading the performance of virtual channel handling, avoid processing within a callback as that blocks or requires extensive processing. Do not use PCoIP Virtual Channel plug-ins to call any PCoIP Virtual Channel APIs in the context of the event/channel callback. For example, don't call `pcoip_vchan_recv()` in a `RECV_RDY` event callback. Use callback implementations only to set an event (or equivalent) to trigger a separate thread to do processing and data transfers.
- If you use the datagram API, use the supplied API call to check the maximum datagram size. Never exceed the maximum datagram size. If you need larger transfers, use the streaming API.
- Set up unreliable transport virtual channels by requesting a reliable virtual channel with the appropriate configuration option. For details, see the API documentation and the sample `u_echo` virtual channel plug-in.
- To avoid potential channel name conflicts, prefix the channel names in your implementations with the company name, for example, *ABCompany**_.
- There is no developer PCoIP Virtual Channel support for PCoIP Zero Clients or PCoIP Remote Workstation Card.
- Every time you install or upgrade PCoIP software, you must also reinstall the plug-ins.
- Although PCoIP Virtual Channel plug-in speed can exceed 10 Mbps, overall throughput rates depend on network bandwidth and other PCoIP traffic, especially video and audio traffic.
- All plug-ins in the PCoIP Virtual Channel directory are processed; you cannot specify order.
- Teradici recommends using the PCoIP Virtual Channel API to log PCoIP Virtual Channel related messages.

- Due to an iOS limitation, the client library can only link with PCoIP Virtual Channel plug-ins that are built as static libraries. You must implement the two functions in the static plug-ins:
    - `pcoip_vchan_plugin_client_init_<plugin_name>`
    - `pcoip_vchan_plugin_client_exit_<plugin_name>` Where `<plugin_name>` is a unique name that identifies the plug-in. For other platforms, implement `pcoip_vchan_plugin_client_init` and `pcoip_vchan_plugin_client_exit` instead.
- On Windows, the PCoIP Virtual Channel plug-in is implemented as a ***.dll*** file that is loaded by the PCoIP Virtual Channel Loader process at session startup using a Windows Registry entry. The PCoIP Virtual Channel Loader process is run as one of several different users depending on the deployment. This may affect the privilege level available to you. The current schema is:
    - For VMware Horizon View Agent: The process runs as SYSTEM in the user's session number with impersonation rights.
    - For other platforms, such as Amazon Workspaces, the process runs as USER in the user's session number.

API HTML Documenation

You can view the HTML documentation that Doxygen automatically generates from the source files here:

`\VChan_SDK\SDK\doc\html\index.html` .

Plug-In Sample Files for PCoIP Virtual Channel

This following table lists the source and header files that build the PCoIP Virtual Channel plug-ins. You can find the files in *~\VChan_SDK\samples*.

Description of File or Folder	File or Folder Path
The header file for both streaming and datagram samples.	*~\VChan_SDK\samples\common\inc\tera_plugin_vchan_sample.h*
The source file for the datagram API sample.	*~\VChan_SDK\samples\sample_dgram\src\tera_plugin_vchan_sample_dgram.c*
The source file for the streaming API sample.	*~\VChan_SDK\samples\sample_stream\src\tera_plugin_vchan_sample_stream.c*
The source file for the unreliable datagram API sample.	*~\VChan_SDK\samples\sample_stream\src\tera_plugin_vchan_sample_u_echo.c*
The header and source file for a thin operating system abstraction layer that enables the same source code to be used for both Windows and Linux plug-in samples.	*~\VChan_SDK\samples\common\inc\tera_plugin_os_abstraction.h*
The folder that contains the documentation for the PCoIP Virtual Channel sample implementation. The doxy file that constructs the documentation with the doxygen system (available as a free download).	*~\VChan_SDK\samples\doc*

Description of File or Folder	File or Folder Path
The doxy file that constructs the documentation with the doxygen system (available as a free download). All source and header files are annotated with doxygen markup to document the API and its usage in the samples.	*~\VChan_SDK\samples\doc\plugin_vchan_sample.doxy*
The folder that contains the header files that define the PCoIP Virtual Channel plug-in interface and associated declarations.*	*~\VChan_SDK\inc*
The folder that contains the top level documentation of the PCoIP Virtual Channel API (including this user guide).	*~\VChan_SDK\doc*
The doxy file that constructs the documentation with the doxygen system (available as a free download). All source and header files are annotated with doxygen markup to document the API and its usage in the samples.	*~\VChan_SDK\vchan_api_inc.doxy*

Do not modify the ~\VChan_SDK\inc folder

This folder is used directly within the PCoIP executable and must not be modified.

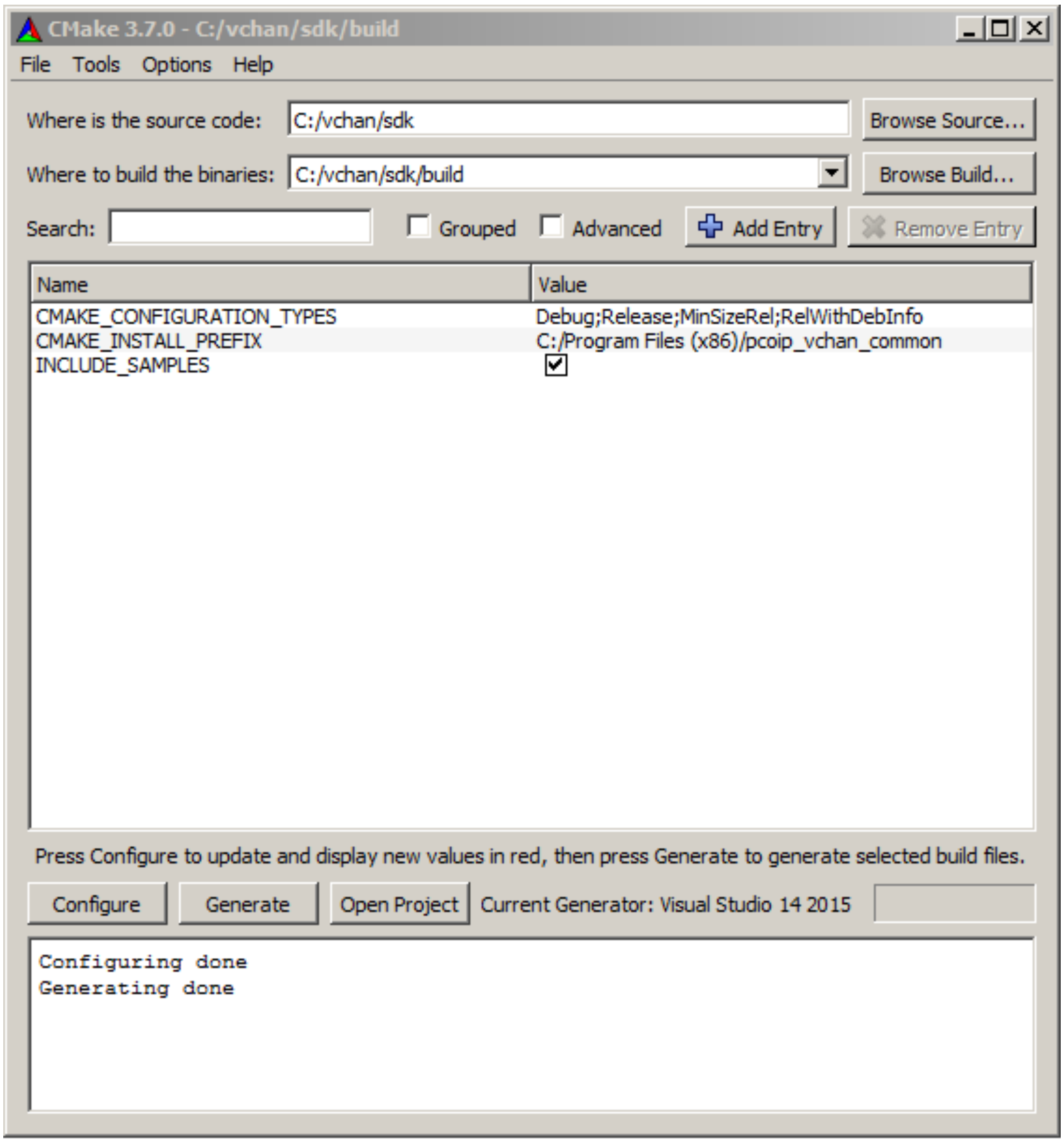
Building PCoIP Virtual Channel Plug-In Samples

Use CMake (Cross-Platform Make) to build the sample plug-ins for Windows, macOS, or Linux. CMake is an open-source, cross-platform family of tools designed to build, test, and package software. You can download the latest version of CMake for platform specific installation file from www.cmake.org. Building the plug-ins for Windows requires Microsoft Visual Studio.

Building the plug-ins for other platforms requires the SCons software construction tool which in turn requires Python, and a gcc compiler or a corresponding toolchain which supports the pthreads library (for the operating system primitives used in the plug-ins). On all platforms you can use CMake to configure and generate platform and compiler specific build files and build the target plug-ins on the chosen platforms.

Using CMake to Configure and Generate Platform and Compiler Build Files

The following image shows the configuration information for CMake with Microsoft Visual Studio on Windows:



Where is the source code: C:/vchan/sdk

Where to build the binaries: C:/vchan/sdk/build

Search:

Name	Value
CMAKE_CONFIGURATION_TYPES	Debug;Release;MinSizeRel;RelWithDebInfo
CMAKE_INSTALL_PREFIX	C:/Program Files (x86)/pcoip_vchan_common
INCLUDE_SAMPLES	✓

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 14 2015

```
Configuring done
Generating done
```

You can configure other platforms in a similar way to take the source codes and generate the build files in the designated binaries folder. It is possible to configure CMake on the command-line user interface, for information refer to CMake documentation at the CMake wiki page.

Completing the CMake process

You must first **Configure**, and then **Generate** as a two-pass routine to complete the CMake process.

To build the target plug-ins on your chosen platform, go to the output build files folder and build the plug-in with the designated compiler.

Binary compatibility for Windows plug-ins

For Windows Clients, please ensure that the PCoIP Virtual Channel plug-ins are built as 64-bit .dlls with Visual Studio Code 2015.

For Windows Agents, if the PCoIP Virtual Channel plug-ins are not built with Visual Studio 2015, you need to ensure that the required re-distributables are installed.

Binary compatibility for macOS plug-ins

To maintain binary compatibility, ensure that PCoIP Virtual Channel plug-ins on macOS are built as 64-bit dylibs with Xcode 5.0 or above.

Installing, Updating, and Uninstalling PCoIP Virtual Channel Plug-Ins

In this section, you'll learn how to install the PCoIP Virtual Channel plug-in on a Windows endpoint, a Linux endpoint, a macOS endpoint, and an iOS or Android endpoint. You'll also learn how to update and uninstall PCoIP Virtual Channel plug-ins.

The topics include:

- Installing PCoIP Virtual Channel Plug-Ins
- Updating PCoIP Virtual Channel Plug-Ins
- Uninstalling PCoIP Virtual Channel Plug-Ins

Creating a custom application installer

For developers of custom plug-ins, Teradici recommends that you create an installer that will automatically install and uninstall the plug-ins.

Installing PCoIP Virtual Channel Plug-Ins

You can deploy custom PCoIP Virtual Channel plug-ins as identical (symmetric) or asymmetric plug-in .dll files. You can create one .dll file to handle both receiving and transmitting functions and deploy the same .dll file on both agent and client sides.

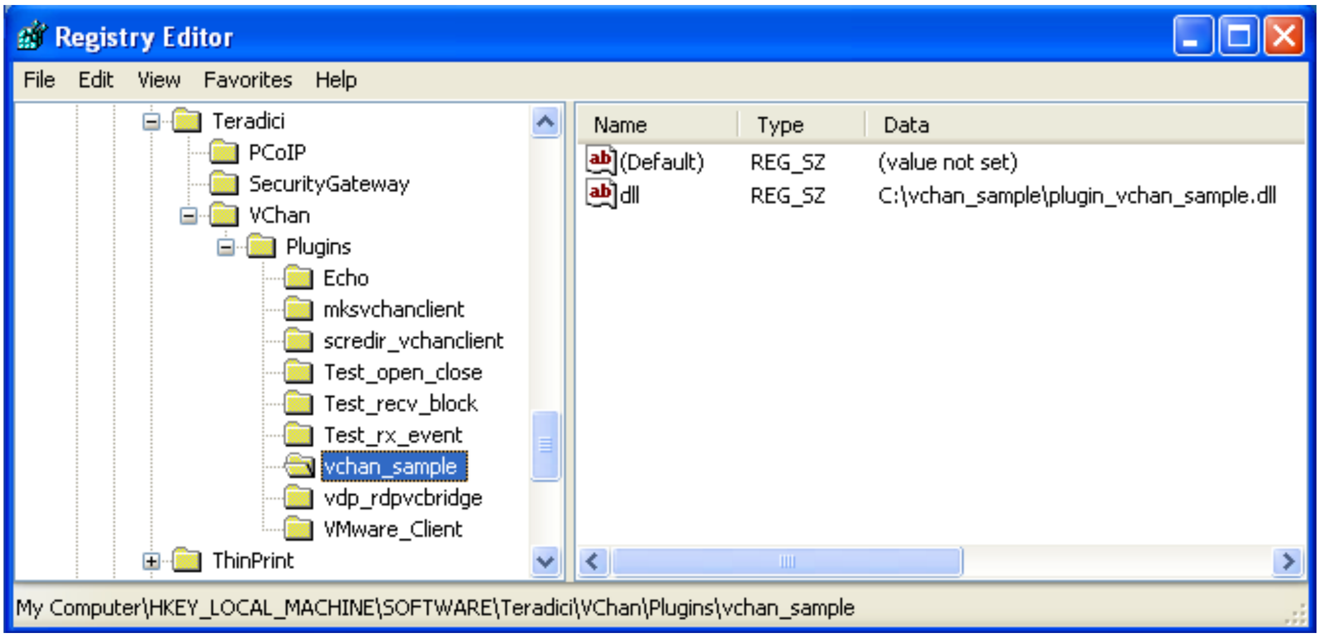
Use the general steps in this example to install any of the sample PCoIP Virtual Channel plug-ins, for example, ***plugin_vchan_sample_dgram.dll***, ***plugin_vchan_sample_stream.dll***, or ***plugin_vchan_sample_u_echo.dll***. Substitute the appropriate name in the description.

Installing PCoIP Virtual Channel Plug-Ins on a Windows Endpoint

Use the following procedure to install any of the sample PCoIP Virtual Channel plug-ins on a Windows endpoint where the PCoIP Client or Agent is installed.

To enable agent side PCoIP Virtual Channel functionality on Windows:

1. Use regedit to add a registry key for the plug-in and create ***vchan_sample*** key at:
   - HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Teradici\VChan\Plugins
2. Add a new string value called *dll* in the above key.
3. Set the associated data value of the dll string to the path and file name for the plug-in dll file, as shown next. In the following example, the data value of the *dll* string is set to C:*\vchan_sample\plugin_vchan_sample.dll*.



Name	Type	Data
(Default)	REG_SZ	(value not set)
dll	REG_SZ	C:\vchan_sample\plugin_vchan_sample.dll

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Teradici\VChan\Plugins\vchan_sample

To enable client side PCoIP Virtual Channel functionality on Windows:

- Copy the *.dll* file that implements the client component functionality of the plug-in into the *vchan plugins* folder relative to the client executable. For example, C:\Program Files (x86)\Teradici\PCoIP Client\bin\vchan_plugins

Installing PCoIP Virtual Channel Plug-Ins on a Linux Endpoint

Use the following procedure to install any of the sample PCoIP Virtual Channel plug-ins on a Linux endpoint where the PCoIP Client or Agent is installed.

To enable agent side PCoIP Virtual Channel functionality on Linux:

- Copy the shared object *(.so)* files that implement the agent side of the plug-in into the /usr/lib/pcoip-agent/vchan_plugins folder.

Standard Utilities

Use the following standard utilities to extract information from the .so file:

```
- To list all the APIs exposed by the .so file, use nm *.so sudo
- To list all Teradici APIs, use nm *.so sudo | grep tera_
- To list all the other shared objects that the given shared object depends on, use dll *.so
sudo
```

To enable client side PCoIP Virtual Channel functionality on Linux:

- Copy the shared object (.so) files that implement the client component functionality of the plug-in into the /usr/lib/x86_64-linux-gnu/pcoip-client/vchan_plugins folder.

Installing PCoIP Virtual Channel Plug-Ins on a macOS Endpoint

Use the following procedure to install any of the sample PCoIP Virtual Channel plug-ins on a Mac endpoint where the PCoIP Client is installed.

To enable client side PCoIP Virtual Channel functionality on macOS:

- Copy the *.dylibs* files that implement the client side of the plug-in into the **vchan plugins** folder relative to the client executable. For example, /Applications/PCoIPClient.app/Contents/MacOS/vchan_plugins.

Installing PCoIP Virtual Channel Plug-Ins on an iOS or Android Endpoint

You must integrate the iOS or Android plug-ins with a PCoIP client. If you want to build plug-ins for iOS or Android platforms, contact the Teradici Support Team by submitting a support ticket.

Updating PCoIP Virtual Channel Plug-Ins

You update Teradici plug-ins differently than custom plug-ins.

To update Teradici plug-ins, updating or reinstalling the agent and/or client software automatically installs the Teradici plug-ins. For custom plug-ins, you need to first uninstall the agent and/or client software, and then reinstall the custom plug-ins.

To update a Teradici plug-in:

- Update or re-install the agent and/or client software. Updating or reinstalling the agent and/or client software automatically updates the Teradici plug-in.

To update a custom plug-in:

1. Update or reinstall the agent and/or client software.
2. Reinstall the custom plug-in.

Uninstalling PCoIP Virtual Channel Plug-Ins

You uninstall Teradici plug-ins differently than custom plug-ins.

For Teradici plug-ins, uninstalling the agent and/or client software automatically uninstalls the Teradici plug-ins and their associated registry keys. For custom plug-ins, you need to first uninstall the agent and/or client software, and then use your custom uninstaller to uninstall the plug-in (or manually delete the custom plug-in).

Creating a custom application installer

For developers of custom plug-ins, Teradici recommends that you create an installer that will automatically install and uninstall the plug-ins.

To uninstall Teradici plug-ins:

- Uninstall the agent and/or client software. Uninstalling the agent or client software automatically uninstalls the Teradici plug-ins and their associated registry keys.

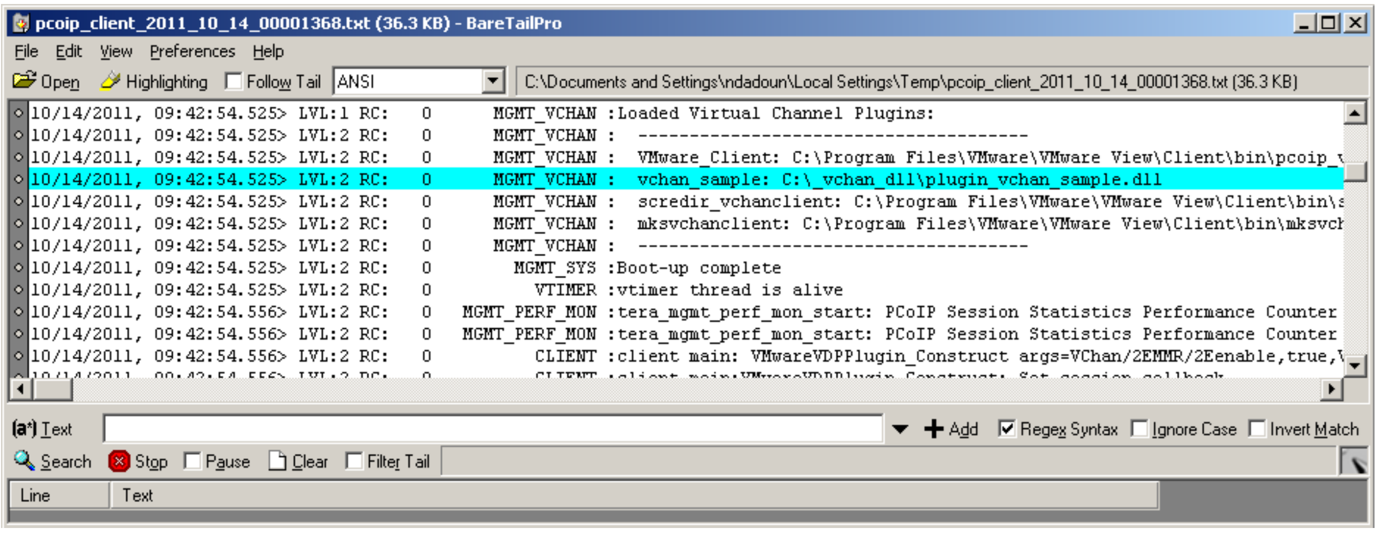
To uninstall custom plug-ins:

1. Uninstall the agent and/or client software.
2. Uninstall the plug-ins by doing one of the following:
    - If you have a custom uninstaller for the plug-ins, run the application uninstaller to uninstall the plug-ins.
    - If you don't have a custom uninstaller, manually delete the custom plug-ins from the *vchan_plugins* folder.

Running the PCoIP Virtual Channel Plug-In within a PCoIP Session

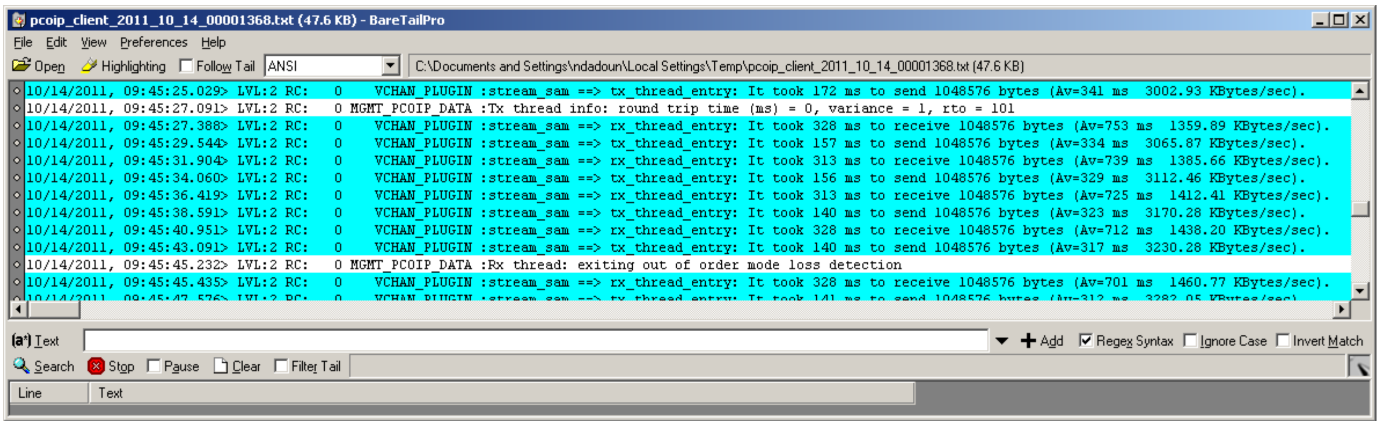
After installing the plug-ins on both the agent and client, open a PCoIP session between the endpoints. The PCoIP Virtual Channel plug-ins specified in the registry are loaded when the session starts.

Confirm that the sample plug-in is running by checking the loaded plug-in list in the PCoIP server/client log files, as shown next:


pcoip_client_2011_10_14_00001368.txt (36.3 KB) - BareTailPro
File Edit View Preferences Help
Open Highlighting Follow Tail ANSI
C:\Documents and Settings\ndadoun\Local Settings\Temp\pcoip_client_2011_10_14_00001368.txt (36.3 KB)
(a*) Text
Add Regex Syntax Ignore Case Invert Match
Search Stop Pause Clear Filter Tail
Line Text

```
10/14/2011, 09:42:54.525> LVL:1 RC:   0      MGMT_VCHAN :Loaded Virtual Channel Plugins:
10/14/2011, 09:42:54.525> LVL:2 RC:   0      MGMT_VCHAN :  ---------------------------------------
10/14/2011, 09:42:54.525> LVL:2 RC:   0      MGMT_VCHAN :  VMware_Client: C:\Program Files\VMware\VMware View\Client\bin\pcoip_v
10/14/2011, 09:42:54.525> LVL:2 RC:   0      MGMT_VCHAN :  vchan_sample: C:\_vchan_dll\plugin_vchan_sample.dll
10/14/2011, 09:42:54.525> LVL:2 RC:   0      MGMT_VCHAN :  scredir_vchanclient: C:\Program Files\VMware\VMware View\Client\bin\s
10/14/2011, 09:42:54.525> LVL:2 RC:   0      MGMT_VCHAN :  mksvchanclient: C:\Program Files\VMware\VMware View\Client\bin\mksvch
10/14/2011, 09:42:54.525> LVL:2 RC:   0      MGMT_VCHAN :  ---------------------------------------
10/14/2011, 09:42:54.525> LVL:2 RC:   0        MGMT_SYS :Boot-up complete
10/14/2011, 09:42:54.525> LVL:2 RC:   0          VTIMER :vtimer thread is alive
10/14/2011, 09:42:54.556> LVL:2 RC:   0   MGMT_PERF_MON :tera_mgmt_perf_mon_start: PCoIP Session Statistics Performance Counter
10/14/2011, 09:42:54.556> LVL:2 RC:   0   MGMT_PERF_MON :tera_mgmt_perf_mon_start: PCoIP Session Statistics Performance Counter
10/14/2011, 09:42:54.556> LVL:2 RC:   0          CLIENT :client main: VMwareVDPPlugin_Construct args=VChan/2EMMR/2Eenable,true,\
```

PCoIP Virtual Channel sample plug-ins periodically log messages (by calling the *pcoip_vchan_log_msg()* API) and report the receive rate. These messages appear in the PCoIP server/client log file under the *VCHAN_PLUGIN* category (depending on your logging level configuration), as shown next:


pcoip_client_2011_10_14_00001368.txt (47.6 KB) - BareTailPro
File Edit View Preferences Help
Open Highlighting Follow Tail ANSI
C:\Documents and Settings\ndadoun\Local Settings\Temp\pcoip_client_2011_10_14_00001368.txt (47.6 KB)
(a*) Text
Add Regex Syntax Ignore Case Invert Match
Search Stop Pause Clear Filter Tail
Line Text

```
10/14/2011, 09:45:25.029> LVL:2 RC:   0    VCHAN_PLUGIN :stream_sam ==> tx_thread_entry: It took 172 ms to send 1048576 bytes (Av=341 ms  3002.93 KBytes/sec).
10/14/2011, 09:45:27.091> LVL:2 RC:   0 MGMT_PCOIP_DATA :Tx thread info: round trip time (ms) = 0, variance = 1, rto = 101
10/14/2011, 09:45:27.388> LVL:2 RC:   0    VCHAN_PLUGIN :stream_sam ==> rx_thread_entry: It took 328 ms to receive 1048576 bytes (Av=753 ms  1359.89 KBytes/sec).
10/14/2011, 09:45:29.544> LVL:2 RC:   0    VCHAN_PLUGIN :stream_sam ==> tx_thread_entry: It took 157 ms to send 1048576 bytes (Av=334 ms  3065.87 KBytes/sec).
10/14/2011, 09:45:31.904> LVL:2 RC:   0    VCHAN_PLUGIN :stream_sam ==> rx_thread_entry: It took 313 ms to receive 1048576 bytes (Av=739 ms  1385.66 KBytes/sec).
10/14/2011, 09:45:34.060> LVL:2 RC:   0    VCHAN_PLUGIN :stream_sam ==> tx_thread_entry: It took 156 ms to send 1048576 bytes (Av=329 ms  3112.46 KBytes/sec).
10/14/2011, 09:45:36.419> LVL:2 RC:   0    VCHAN_PLUGIN :stream_sam ==> rx_thread_entry: It took 313 ms to receive 1048576 bytes (Av=725 ms  1412.41 KBytes/sec).
10/14/2011, 09:45:38.591> LVL:2 RC:   0    VCHAN_PLUGIN :stream_sam ==> tx_thread_entry: It took 140 ms to send 1048576 bytes (Av=323 ms  3170.28 KBytes/sec).
10/14/2011, 09:45:40.951> LVL:2 RC:   0    VCHAN_PLUGIN :stream_sam ==> rx_thread_entry: It took 328 ms to receive 1048576 bytes (Av=712 ms  1438.20 KBytes/sec).
10/14/2011, 09:45:43.091> LVL:2 RC:   0    VCHAN_PLUGIN :stream_sam ==> tx_thread_entry: It took 140 ms to send 1048576 bytes (Av=317 ms  3230.28 KBytes/sec).
10/14/2011, 09:45:45.232> LVL:2 RC:   0 MGMT_PCOIP_DATA :Rx thread: exiting out of order mode loss detection
10/14/2011, 09:45:45.435> LVL:2 RC:   0    VCHAN_PLUGIN :stream_sam ==> rx_thread_entry: It took 328 ms to receive 1048576 bytes (Av=701 ms  1460.77 KBytes/sec).
```

Troubleshooting

For additional information and known issues, see the Teradici PCoIP Community Forum and the Teradici Support Knowledge Base.

If you need further assistance, contact the Teradici Support Team by submitting a support ticket.