PLEORA TECHNOLOGIES INC.

Video Server API

eBUS SDK Version 5.0

Quick Start Guide

Pleora Technologies

Document Number

EX001-017-0004, Version 8.0 2/2/17

Table of Contents

Chapter 1

About this Guide

This chapter describes the purpose and scope of this guide, and provides a list of complimentary guides.

The following topics are covered in this chapter:

What this Guide Provides

This guide provides you with the information you need to install the eBUS SDK (which lets you use the Video Server API) and an overview of the system requirements. It also provides examples of scenarios in which the Video Server API can be used.

When conducting sales demonstrations, you can review the recommended equipment, and see how easy it is to use the sample applications to demonstrate the capabilities of the Video Server API. Instructions for compiling and using the sample code are provided, along with an overview of the basic calls that can be used to build custom applications using the Video Server API.

For troubleshooting information, frequently asked questions, and technical support contact information for Pleora Technologies, see the last few chapters of this guide.

Related Documents

The *Video Server API Quick Start Guide* is complemented by the following guides:

- *eBUS Player Quick Start Guide*
- *eBUS Player User Guide*
- *vDisplay User Guide*
- *eBUS SDK Programmer's Guide*
- *eBUS SDK C++ API Help File*
- *eBUS SDK .NET API Help File*
- *eBUS SDK for Linux Quick Start Guide*
- *eBUS SDK Licensing Application Note*
- *Configuring Your Computer and Network Adapters for Best Performance Application Note*

Chapter 2

Introducing the Video Server API

This chapter describes the Video Server API, which is a feature of the eBUS SDK that transmits images over the network from a computer to one or more alternate destinations, in a GigE Vision® compliant manner.

The following topics are covered in this chapter:

About the Video Server API

The Video Server API transmits images over the network from a computer to one or more alternate destinations. Images can be captured (using operating system API functions or an SDK) and transferred to the Video Server API from several types of devices or applications. For example:

- From one or more GigE Vision compatible cameras
- From a file, such as a WMV file
- From one or more cameras using USB, Firewire, or other point-to-point technology
- From computer-generated video, representing raw data, such as performance counters

This API can be used with the Windows® and Linux operating systems. For system requirements, see "System Requirements" on page 8.

Programming Languages

The eBUS SDK supports C++, C#.NET, and VB.NET.

The sample code provided for the Video Server API is written using the C++ programming language, with the exception of the **TransmitTestPattern** and **TransmitTiledImages** samples, which are also available as C#.NET samples.

Comparing the Video Server API and GigE Vision Compatible Hardware Devices

While the Video Server API includes many of the same basic features as a traditional GigE Vision transmitter (such as a camera), additional features become available. The following table provides a comparison of these two installations.

Table 1: Comparison Between the Video Server API and GigE Vision Compatible Hardware Devices

Features	Traditional GigE Vision transmitter (such as a camera)	Video Server API
Data format	Accepts a single format (typically video).	Accepts multiple formats, including images and raw data.
Image acquisition	Uses the eBUS SDK to obtain images.	
Serial communication	Uses the eBUS SDK for serial communication. One or more physical ports are available.	Uses a custom protocol and out-of-band communications method. The number of ports depend on your computer's configuration.
GPIO	Uses the eBUS SDK for control of input and output signals. One or more pins are available.	Not typically available on a computer, and not available with the Video Server API.
PLC	Uses the eBUS SDK to configure and control signals.	Not typically available on a computer, and not available with the Video Server API.
Image stream configuration, including start and stop	Includes GenICam™ integration. Uses the eBUS SDK to configure the image stream.	Uses a custom protocol and out-of-band communications method to configure the image stream.
Packet resend	Uses packet resend to obtain lost or out-of-order packets.	Does not use packet resend.
Miscellaneous configuration	Includes GenICam integration. Uses the eBUS SDK to configure settings.	Uses a custom protocol and out-of-band communications method. Minimal GenICam implementation is available.

As the table above shows, the Video Server API is used primarily to transfer images from a computer to GigE Vision compliant receivers, such as a vDisplay HDI-Pro External Frame Grabber, eBUS Player, or any other GigE Vision compliant receiver. Control of the image stream and other parameters is not performed using GenICam, as is done with a typical camera.

However, an image acquisition application created using the eBUS SDK can control a video server application (created using the eBUS SDK's Video Server API) using a custom protocol and an out-of-band communications method. An out-of-band communications method is a custom protocol that is not provided, and not handled, by the eBUS SDK, which your application can use to interact with the video server. A good example is one that uses a Web service or a simple TCP/IP connection. This out-of-band communications method can still be used on the same cable and GigE network interface that is used to transmit images using the Video Server API.

eBUS SDK Licenses

While the Video Server API is a licensed product, you can trial the API without purchasing a license. However, the following limitations apply:

- Transmitted images have an embossed watermark
- Raw data cannot be transmitted using the Video Server API

Understanding Licensing

A single transmitter license is required for each output stream, regardless of whether the output stream is unicast or multicast. For example, if an application outputs a single multicast stream, it requires a single license, regardless of whether there are no multicast clients or up to 100 multicast clients.

Each license is associated to the MAC address of a network interface card (NIC).

When you purchase a transmit or receive license, your Pleora representative will request the MAC address of a NIC in the workstation. Pleora includes the MAC address in the license file that your representative provides you with, which allows the eBUS SDK to accept the license. The MAC address is used to identify the workstation, and is required regardless of the protocol (GigE Vision or USB3 Vision).

Activating an eBUS SDK License

For detailed information about licensing, including details on activating a license, see the *eBUS SDK Licensing Application Note* available on the Pleora Technologies Support Center.

Chapter 3

Installing the eBUS SDK with the Video Server API

The Video Server API is installed on your computer during the installation of the eBUS SDK.

The instructions in this chapter are based on the Windows 7 operating system. The steps may vary depending on your computer's operating system.

The following topics are covered in this chapter:

System Requirements

Ensure the computer on which you install the eBUS SDK meets the following recommended requirements:

- At least one Gigabit Ethernet NIC.
- An appropriate compiler or integrated development environment (IDE):
  - **For C++.** Visual Studio 2012 or Visual Studio 15.
  - **For C#.NET or VB.NET.** Version 4.0 of the .NET Framework, Visual Studio 2012, or Visual Studio 2015.

    Sample project files (.vcxproj) are compatible with Visual Studio 2012 and Visual Studio 2015.
  - **For the Linux operating system**.
    - gcc 4.4.7 (or later) to compile non-GUI samples
    - QMake version 2.01a, with Qt version 4.6.2 to compile GUI-based samples
    - Kernel-devel package for your specific kernel version to compile the eBUS Universal Pro for Ethernet filter driver
- One of the following operating systems:
  - Microsoft® Windows 10, 32-bit or 64-bit
  - Microsoft Windows 8.1, 32-bit or 64-bit
  - Microsoft Windows 7 with Service Pack 1 (or later), 32-bit or 64-bit
  - Windows 2008 Server with Service Pack 3 (or later), 32-bit or 64-bit
  - Windows Server 2012, 64-bit
  - Red Hat Enterprise Linux 7, 64-bit
  - CentOS 7, 64-bit
  - Ubuntu 14.04 and 16.04 LTS, 32-bit or 64-bit

Depending on the incoming and outgoing bandwidth requirements, as well as the performance of each NIC, you may require multiple NICs. For example, even though Gigabit Ethernet is full duplex (that is, it can simultaneously manage 1 Gbps incoming and 1 Gbps outgoing), the computer's PCIe bus may not have enough bandwidth to support this. This means that while your NIC can — in theory — accept four cameras at 200 Mbps each incoming, and output a 750 Mbps stream on a single NIC, the NIC you choose may not support this level of performance.

If you use the Linux operating system, you must install the SDK as superuser.

Installing the eBUS SDK

Because the Video Server API is part of the eBUS SDK, it is included in the eBUS SDK installation package.

To install the eBUS SDK

- Follow the standard installation instructions to install the eBUS SDK on your computer.

  If you do not have the CD or USB stick, you can access installation files from the Pleora Support Center at www.pleora.com.

  Please note that runtime packages are also available, for easy integration into applications developed with the eBUS SDK.

  The runtime packages install the items that are required to run an application created with the eBUS SDK (including GigE Vision and USB3 Vision drivers). They exclude the items that are used to create applications with the eBUS SDK (such as header files, libraries, sample code, and API Help files).

Recommended Demonstration Equipment

To demonstrate the Video Server API, we recommend you have the following equipment at the demonstration site:

- A computer running the eBUS SDK version 2.1 (or later), which will transmit images.
- A receiver, such as a vDisplay HDI-Pro External Frame Grabber, eBUS Player, or any other GigE Vision compliant receiver.
- One or more GigE Vision cameras, depending on the sample application you are using (for example, if the sample application is streaming images from multiple live cameras).

To use the **TransmitTiledImages** and **TransmitProcessedImage** sample applications, you must have a GigE Vision transmitter device.

Chapter 4

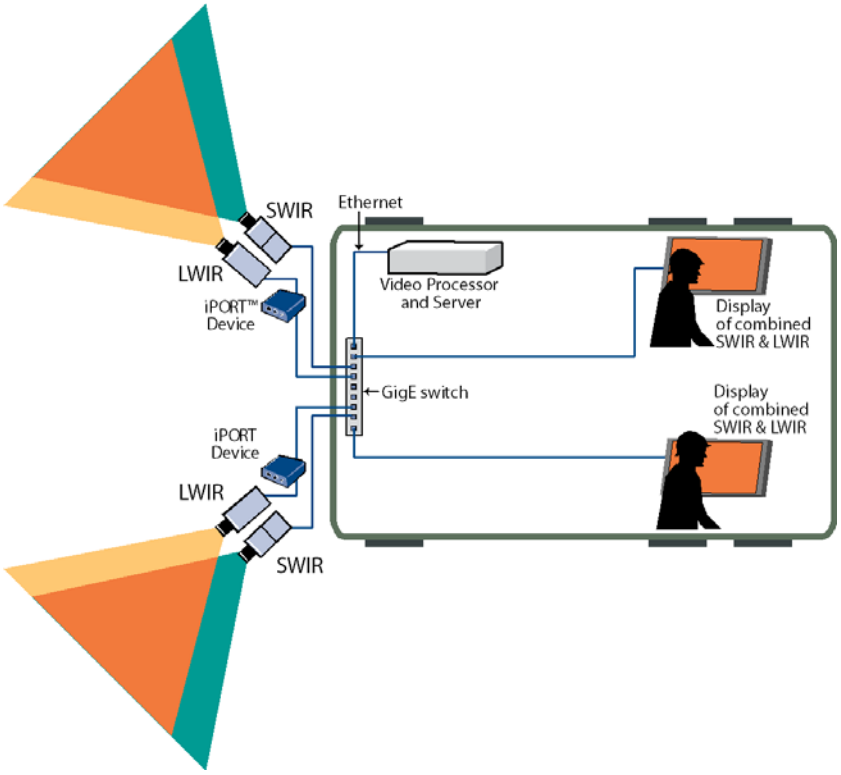
Example Scenarios

To understand how the Video Server API can be deployed, review the example scenarios provided in this chapter. It is important to note that these are example scenarios, and that there are a variety of environments in which the Video Server API can be used — this guide highlights some common scenarios that best illustrate the benefits of using the API.

The following topics are covered in this chapter:

Military

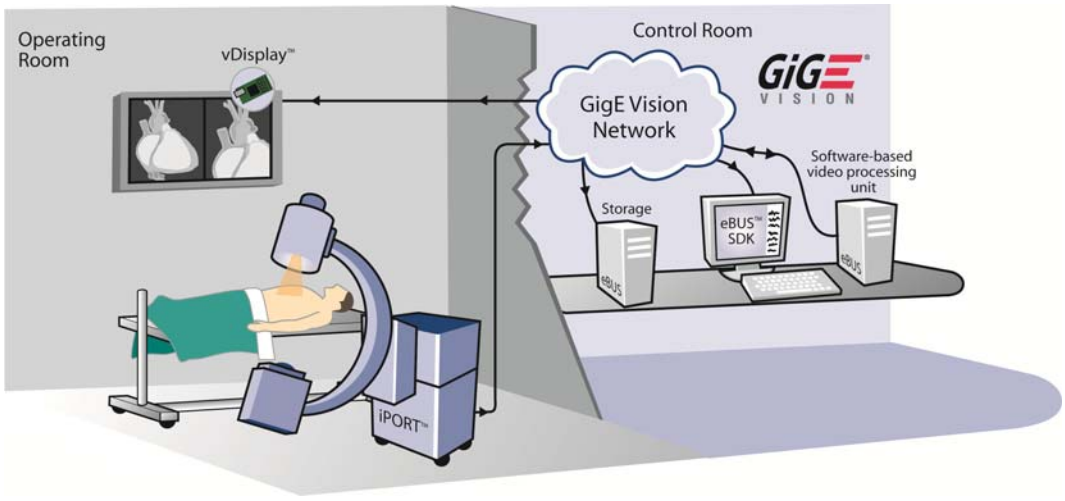
In a military implementation, images are captured by short-wavelength infrared (SWIR) and long-wavelength infrared (LWIR) cameras (in this example, two pairs of SWIR and LWIR cameras). The iPORT devices (one for each camera) transmit the images to the video processor/server, which merges the SWIR and LWIR images (which each highlight different information in low-light environments) into a single image. The video processor/server then distributes the resulting single image to one or more in-vehicle displays, providing a more complete view to military personnel on the vehicle.



This example shows one (of many) possible implementations in a military environment, and illustrates the benefits of using the API. The API can be implemented in a variety of ways, to suit your particular needs.

Medical

In a medical implementation, images are captured by an imager within a fluoroscope, for example during an angioplasty operation, and then transmitted to an iPORT device. The image is then multicast over a GigE Vision network to a storage computer and a computer that is used for display in the control room. Images are also sent to a software-based video processing unit that creates a single image stream — one that highlights areas of interest, includes pre-op images, and shows vital signs. The composite image stream can be sent to a video receiver, such as a vDisplay HDI-Pro External Frame Grabber, which in the diagram is attached to the monitor in the operating room.

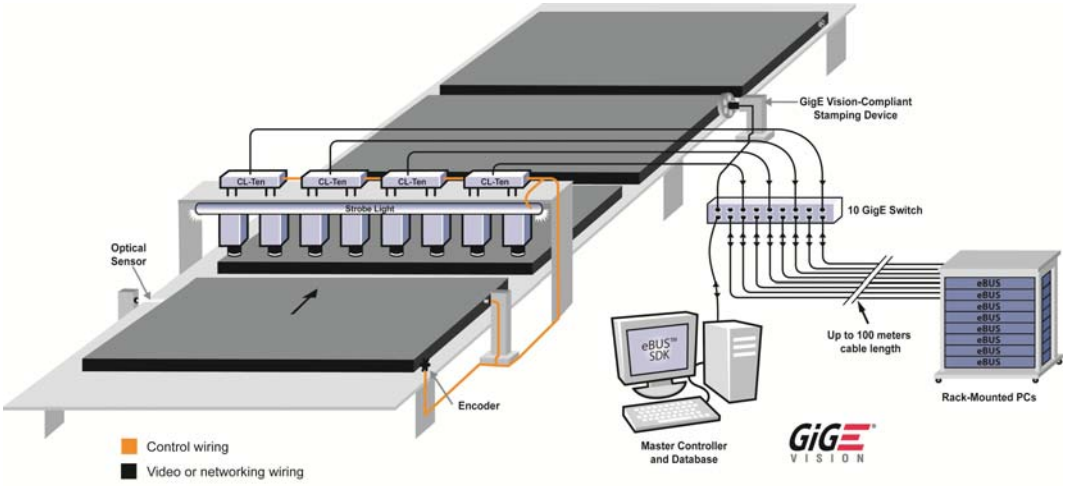


This example shows one (of many) possible implementations in a medical environment, and illustrates the benefits of using the API. The API can be implemented in a variety of ways, to suit your particular needs.

Quality Inspection

In a quality inspection system, images are received by a computer that pre-processes all images. After each image is pre-processed, it is then distributed to other computers that each analyze it for specific product defects. The results are then collected by a master computer that makes a decision on the quality of the product, for example whether the product passes or fails inspection.

After the system makes its quality decisions, an image is also transmitted from the main computer to a vDisplay HDI-Pro External Frame Grabber, and is displayed on a monitor. Any defects are highlighted and the words “PASSED” or “FAILED” appear on the screen. This type of setup is ideal, as it allows you to display results on a monitor that is located close to the analysis equipment — the computer can be located in another area of the facility, away from the potentially dusty environment. It also allows operators who are standing close to the analysis equipment to confirm that the system is working correctly.



This example shows one (of many) possible implementations in a quality inspection environment, and illustrates the benefits of using the API. The API can be implemented in a variety of ways, to suit your particular needs.

Chapter 5

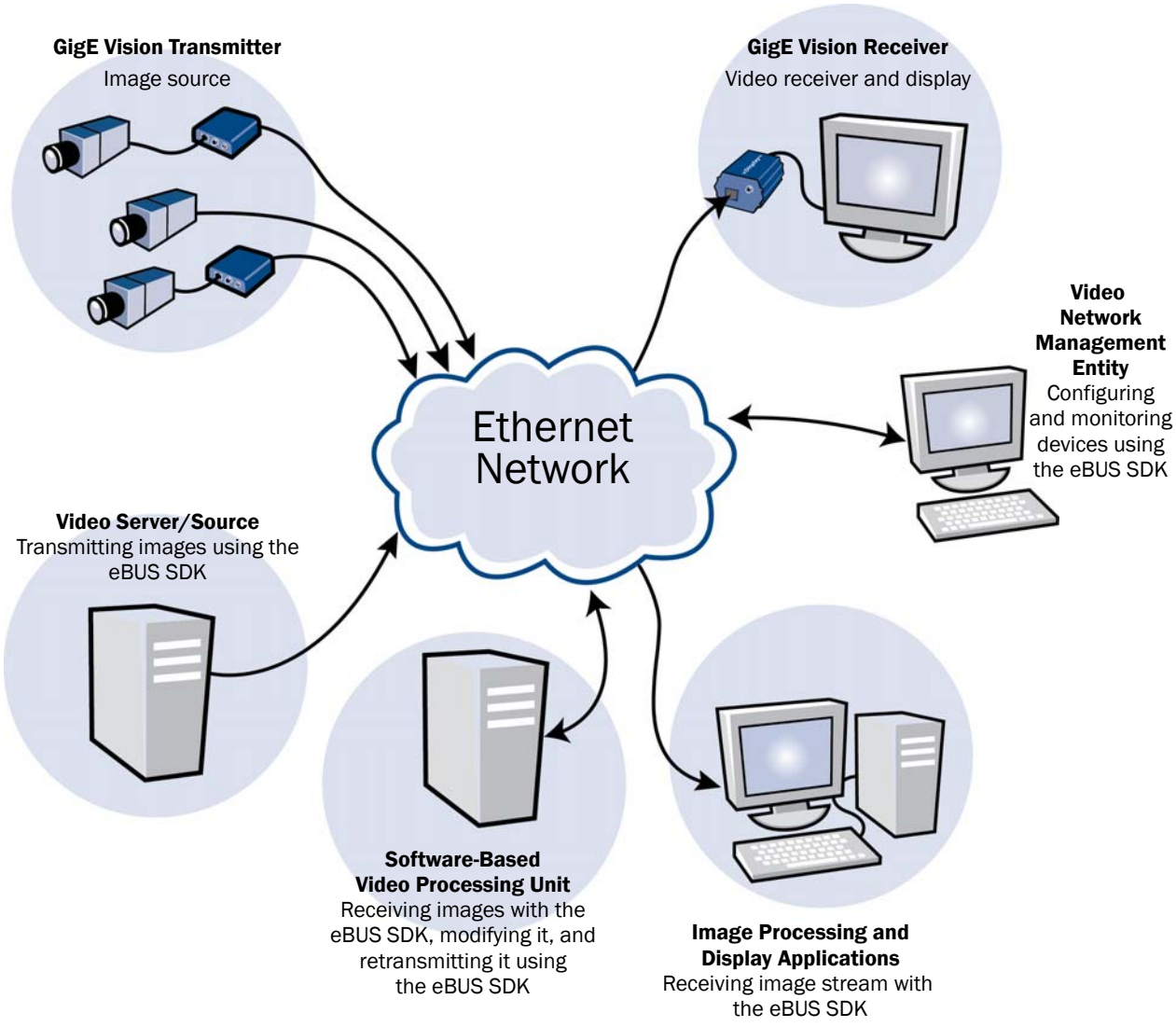
Using the Sample Code

To illustrate how you can use the Video Server API to transmit images, the SDK includes sample code that you can use. This chapter provides a description of the sample code and provides general information about accessing and compiling the code to create sample applications.

The following topics are covered in this chapter:

Overview: System Components

The following illustration shows the components that are used, illustrating the relationship between the eBUS SDK, GigE Vision receivers, and GigE Vision transmitters.



Description of Samples

The following table provides a description of the sample code that is available for the Video Server API.

Table 2: Sample Code

Sample code	Function	Type of application that is created
TransmitProcessedImage	Receives images from a GigE Vision device, prints text on them, and transmits them to a given destination.	Command line. Only available for the C++ programming language.
TransmitTestPattern	Transmits a test pattern to a given destination.	UI-based for C#.NET. Command line for C++.
TransmitTiledImages	Receives image streams from up to four GigE Vision compatible transmitters (typically cameras), tiles them into a single image feed, and then transmits the tiled image stream to a given destination. A precompiled version of this sample is available in Program Files\Pleora Technologies Inc\eBUS SDK\Binaries.	UI-based

Required Equipment and Software

The following table lists the equipment and software that are required to compile the sample code and test that the sample is transmitting and receiving images.

All sample code requires the following general equipment and software:

- A computer running eBUS SDK version 4.1 (or later), which will transmit images.
- A receiver, such as another computer running the eBUS SDK version 4.1 (or later), a vDisplay HDI-Pro External Frame Grabber, eBUS Player, or any other GigE Vision compliant receiver.

Table 3: Required Equipment for Demonstrations

Sample code	Required equipment
TransmitTestPattern	• General equipment and software (listed above)
TransmitTiledImages	• General equipment and software (listed above) • One or more GigE Vision transmitter devices, depending on the sample application you are using (for example, if the sample application is streaming images from multiple live cameras) Note: This sample is only compatible with the Windows operating system.
TransmitProcessedImage	• General equipment and software (listed above)

You must have a GigE Vision transmitter device for the TransmitTiledImages and TransformProcessedImage applications.

Accessing the Sample Code

The sample code is installed on your computer in the Pleora Technologies Inc. folder, as part of the eBUS SDK.

To access the sample code (Windows operating system)

- **To access the C++sample code.** On the Windows **Start** menu, click **All Programs** > **Pleora Technologies Inc** > **eBUS SDK** > **C++ Code Samples Directory**.

  - Or -

- **To access the sample code for .NET based languages.** On the Windows **Start** menu, click **All Programs** > **Pleora Technologies Inc.** > **eBUS SDK** > **.NET Code Samples Directory**.

  Windows Explorer opens to the location of the sample code.

  Copy the sample code to a location on your computer, such as your C: drive.

You can also access the sample code by navigating to the following locations (on the computer running the eBUS SDK):

C:\Program Files\Pleora Technologies Inc\eBUS SDK\Samples

C:\Program Files\Pleora Technologies Inc\eBUS SDK\SamplesDotNet

To access the sample code (Linux operating system)

- Navigate to **/share/samples**

Compiling the Sample Code

All of the sample code, with the exception of TransmitTiledImages (which is already compiled and can be run as a GUI application), needs to be compiled before you can use it.

You can use a compiler, such as a compiler included in an Integrated Development Environment (IDE), to compile the sample code. For example, you can use Microsoft Visual Studio® to compile the sample code.

The compiled GUI application is located in one of the following locations, depending on your operating system:

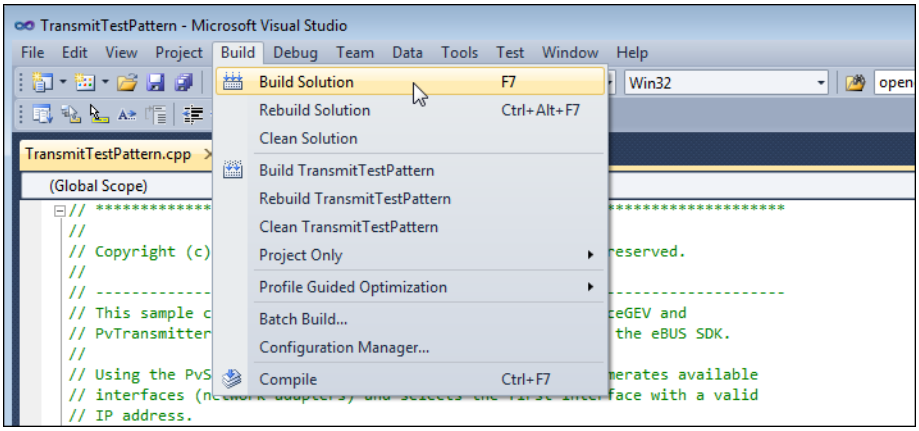
For 32-bit operating systems: C:\Program Files\Pleora Technologies Inc\eBUS SDK\Binaries

For 64-bit operating systems: C:\Program Files (x86)\Pleora Technologies Inc\eBUS SDK\Binaries

To compile the sample code (Windows operating system)

You must copy the sample code from the **Code Samples** folder to another folder on your computer, such as a folder on your computer's **C:** drive.

1. Start your IDE, such as Microsoft Visual Studio.
2. Open the project file for the sample code that you want to compile.
3. Click **Build** > **Build Solution**.



When the code is compiled (as indicated in the **Output** window at the bottom of Microsoft Visual Studio), the sample application is placed in a **Debug** folder within your project. For example, the sample application is stored in **C:\Users\my_user\Video Server API\TransmitTestPattern\Win32\Debug**.

4. You can now run the executable file that is stored in the **Debug** folder. If the application is a command line application, see "Running Command Line Samples" on page 22.

You can also run the executable using the IDE. For more information, see the documentation accompanying the IDE.

To compile the sample code (Linux operating system)

1. Set up the environment. In the installation folder, type **source ./set_puregev_env** directory, then press the **ENTER** key.
2. To build all of the samples at once, go to the directory that contains the Linux samples (**/opt/pleora/ebus_sdk/RHEL-6-x86/bin/share/samples** or **/opt/pleora/ebus_sdk/RHEL-6-x86_64/share/samples**), type **./build.sh**, and then press the **ENTER** key. To build a particular sample, go to the corresponding subfolder and type **make**.

Preparing Your System

Keep the following considerations in mind when preparing your system to use the sample applications:

- You must have at least one network interface card (NIC) installed in the computer that is running the Video Server API. To maximize the performance of your system, enable Jumbo Packets on your NIC and increase the number of Tx Descriptors (for the transmitting computer) and the Rx Descriptors (on the receiving computer) to meet your needs.
- You should have at least one GigE Vision device (such as an iPORT embedded video interface or iPORT external frame grabber) or camera connected to your computer for samples that retransmit received images. The device or camera should be on the same subnet as the NIC from which it will be transmitting.
- You must ensure each NIC is on a different subnet (when you are using multiple NICs in a single computer).
- You are aware of the amount of bandwidth that is available for transmitting and receiving images. For best results, use one NIC for transmitting and another for receiving. Ensure that your desired throughput is less than your theoretical bandwidth (typically less than 1 Gbps upstream or downstream per NIC).

Running Command Line Samples

After you compile any of the command line samples, you can run them and then add command line arguments to change the default behavior of the applications. For example, you can change the destination IP address to which video is transmitted or customize the image by changing the width and height.

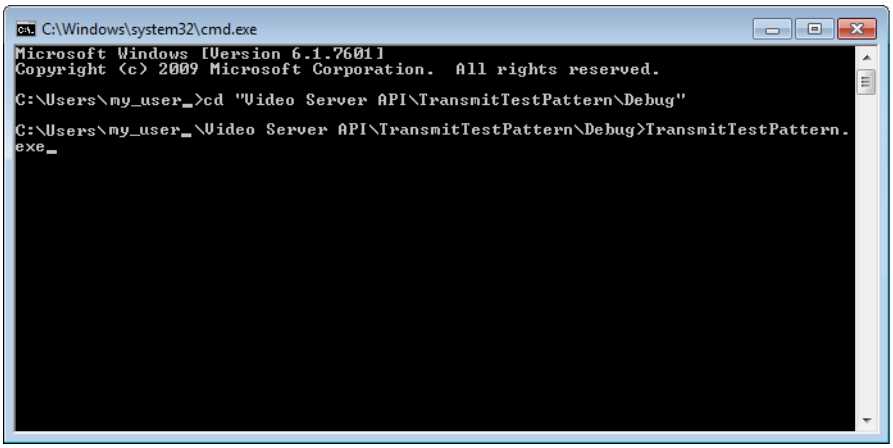
To run a command line sample (Windows operating system)

1. On the Windows **Start** menu, type **cmd** in the search box.
2. In the **Command Prompt** window, navigate to the folder in which you created the application. For example:

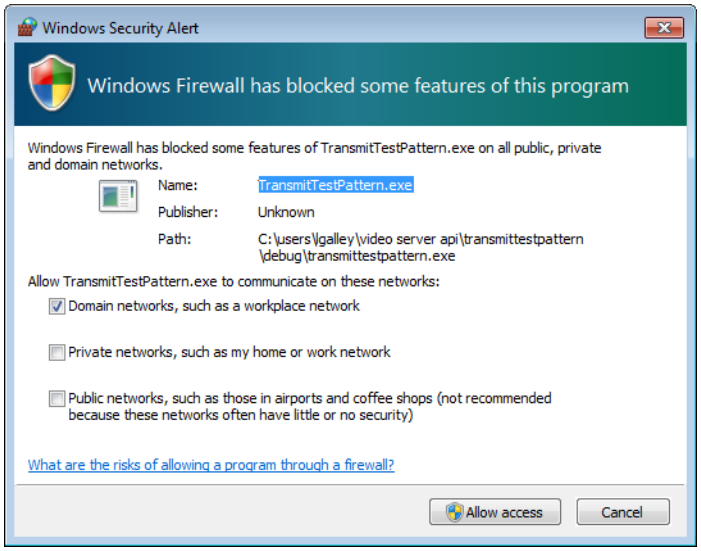
   cd Users\my_user\Video Server API\TransmitTestPattern\Win32\Debug
3. At the command prompt, type the name of the executable and press the **ENTER** key. For example:

   TransmitTestPattern.exe

To help you quickly type information at the command line prompt, you can use the **TAB** key to automatically fill in a partially-typed file name. For example, to automatically fill in the executable name (instead of typing it) you can type the first letter of the executable (for example, **T**) and then press the **TAB** key until the executable file appears, such as **TransmitTestPattern.exe**. Press **ENTER** to accept the file name.



A **Windows Security Alert** may appear, indicating that the Windows Firewall has blocked some features of the sample. Select the check boxes that suit your system and then click **Allow access**.



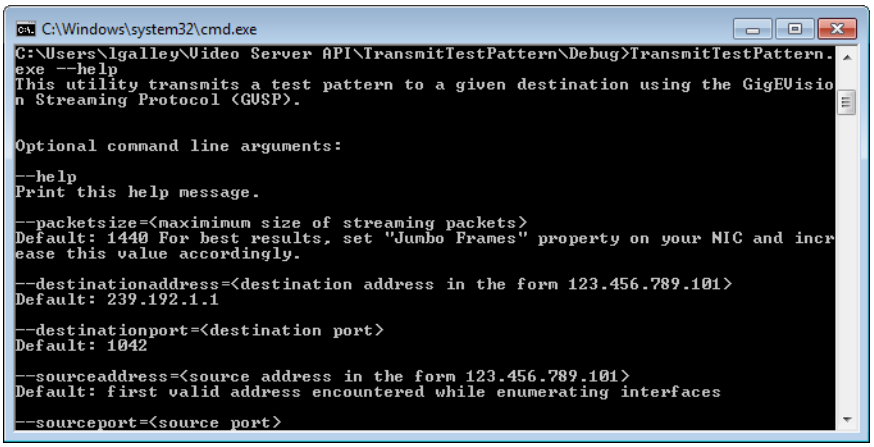
To run a command line sample (Linux operating system)

1. In the directory that contains the sample, type ***./name of sample***, and then press the **ENTER** key. For example, **./TransmitTestPattern**.

To view the list of command line options that you can change (all operating systems)

- At the command line prompt, type the name of the executable followed by -- and **help** (ensure there is a space between the **.exe** and the two dashes). For example:

  TransmitTestPattern.exe --help

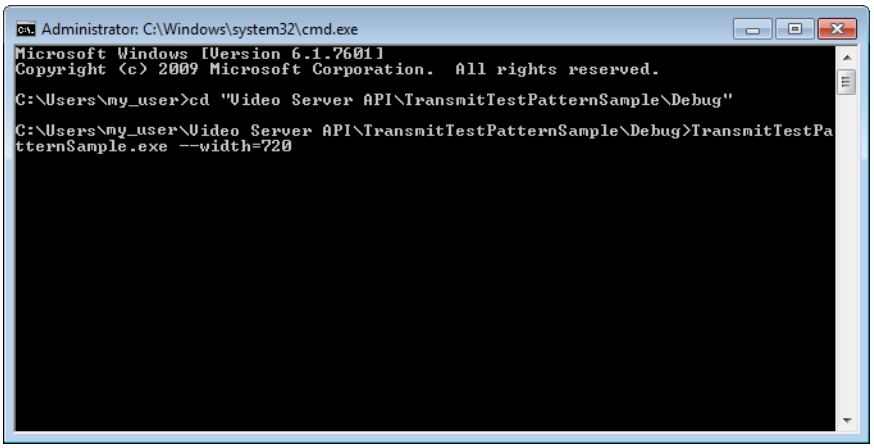


  A list of command line options appears, with guidance on how to change each option. For more information about changing a command line option, see "To change a command line option (all operating systems)" on page 24.

To change a command line option (all operating systems)

- At the command line prompt, type the name of the executable, followed by -- , the name of the command line option, and the option you want to set (ensure there is a space between the .exe and the two dashes). For example:

  TransmitTestPattern.exe --width=720



Transmitting and Receiving a Test Pattern or Images

The TransmitTestPattern and TransmitProcessedImage sample applications allow you to transmit a test pattern and images. Images can be received by a GigE Vision receiver, such as a computer running eBUS Player, to ensure the application is working properly.

This section only applies to GigE Vision devices.

To multicast a test pattern or images to a GigE Vision receiver

Ensure you have a GigE Vision receiver that can receive and display the test pattern, images, video file, or screen contents. The receiver should be on the same subnet as the NIC that is transmitting. For example the transmitter is at IP address **192.168.128.100** with the subnet mask set to **255.255.255.0** and the receiver is at IP address **192.168.128.101** with the subnet mask set to **255.255.255.0**.

1. Ensure that the GigE Vision receiver is on the same subnet as the NIC that is transmitting.
2. Locate and start the **TransmitTestPattern** sample file.

   For more information, see "Running Command Line Samples" on page 22.

   If the transmitting computer only has one NIC, start the sample with the default options (no command line arguments). Otherwise, note the IP address of the NIC from which you want to transmit and start the sample with command line argument **--sourceaddress=<your IP address>**.

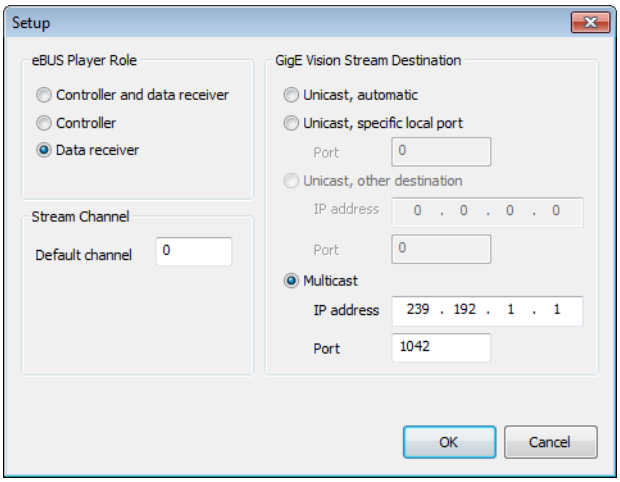
   Note: By default, samples transmit from a valid NIC to multicast address 239.192.1.1:1042. For information about how to specify the destination IP address, see "To view the list of command line options that you can change (all operating systems)" on page 24 and "To change a command line option (all operating systems)" on page 24.
3. Start the receiver. Depending on the version of the sample you are using, do one of the following:
   - **C++ version of the sample.** Press any key when you are ready to begin transmitting (and the receiver is ready to receive the image stream).
   - **C#.NET version of the sample.** Click **Play** when you are ready to begin transmitting (and the receiver is ready to receive the image stream).

   By default, a vDisplay HDI-Pro External Frame Grabber can receive the stream using the default settings, since it subscribes to the same multicast address and port by default.

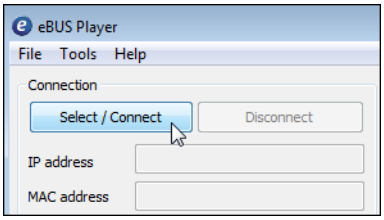
To receive the test pattern or images with eBUS Player

The computer that receives the test pattern or images should be on a different computer (or on the same computer, but using a different NIC) than the computer that is transmitting.

1. Start eBUS Player.
2. Click **Tools** > **Setup**.
3. Under **eBUS Player Role**, select **Data receiver**.
4. Under **GigE Vision Stream Destination**, select **Multicast**.
5. Click **OK**.

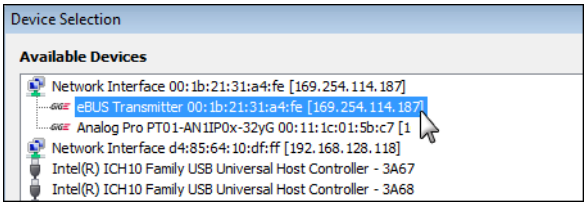


6. Click **Select/Connect**.



7. Select the device in the list that matches the IP address and MAC address of the NIC from which you are transmitting, and then click **OK**. By default, it is labeled eBUS Transmitter.

   Note: The eBUS Transmitter is discoverable because PvVirtualDeviceGEV is instantiated in the Video Server API samples.



   You will see the test pattern or images when the sample starts transmitting.

To unicast a test pattern or images to a GigE Vision receiver, and receive it

Ensure you have a GigE Vision receiver that can receive and display the test pattern, such as eBUS Player. The receiver should be on the same subnet as the NIC that is transmitting. For example the transmitter is at IP address **192.168.128.100** with the subnet mask set to **255.255.255.0** and the receiver is at IP address **192.168.128.101** with the subnet mask set to **255.255.255.0**.

1. Ensure that the GigE Vision receiver is on the same subnet as the NIC that is transmitting, and is reachable from the computer that is running the sample.
2. Start the sample with the following command line arguments:

   ```
   --sourceaddress=<ip address of the interface you're transmitting from> --
   destinationport=0 --destinationaddress=<ip address of the interface you're
   transmitting to>
   ```

   Since a destination port of 0 was provided, the sample will begin listening for device discovery requests and prompt for a destination port.
3. On a different computer, start eBUS Player.

   Note: You can start eBUS Player using the Windows **Start** menu (**Start** > **All Programs** > **Pleora Technologies Inc** > **eBUS SDK** > **eBUS Player**).
4. Click **Tools** > **Setup**.
5. Under **eBUS Player Role**, select **Data receiver**.
6. Under **GigE Vision Stream Destination**, select **Unicast, automatic**.
7. Click **OK**.
8. Click **Select/Connect**.
9. Select the device in the list that matches the IP address and MAC address of the NIC from which you are transmitting. By default, it is labeled **eBUS Transmitter**.

   At this point, you can see which port eBUS Player is listening on.
   - Click **Image stream control**.
   - Under **Connection**, observe the **DataPort** property.
10. Enter the destination port when prompted by the sample.

    You will see an image in the receiver when the sample starts transmitting.

Transmitting Tiled Images

The TransmitTiledImages sample receives images from up to four GigE Vision transmitters (typically cameras), tiles them into a single image feed, and then transmits the image feed to a location you specify.

To avoid dropped packets, ensure that the sum of the incoming data rate for each GigE Vision device does not exceed the maximum throughput of the NIC. To reduce the data rate, you can adjust the frame rate, crop the image, or take other steps to reduce the size of transmitted images. For additional recommendations, see the note included in "System Requirements" on page 8.

This section only applies to GigE Vision devices.

This sample is only compatible with the Windows operating system.

To transmit tiled images

1. Locate and start the **TransmitTiledImages** sample file.
2. Under **Video Source**, click **Select/Connect**.

   You can add up to four video sources.
3. Select a GigE Vision transmitter from the list of available devices (devices must be on the same subnet as the NIC on which you want to receive images) and then click **OK**.
4. In the **Setup** dialog box, do not change the default role (**Controller and data receiver**).
5. Select a unicast or multicast option.
6. Click **OK**.
7. Repeat steps 2-6 to add to up to three additional devices (optional).
8. If required, adjust the communication, device, or image stream parameters by clicking the button that appears beside **Select/Connect** and selecting an option.
9. If required, you can change the **Width**, **Height**, and **Tiling Mode** options that appear under **Video Output**.
10. Under **Transmission**, update the connection options, as required:
    - **Destination Address** and **Destination Port**. The IP address and port of the GigE Vision receiver.
    - **Interface Address.** The NIC (in your computer) that you want to use to transmit. Also referred to as the source address.
    - **Desired Frame Rate (fps).** The rate at which you want to transmit images.
    - **Maximum Packet Size.** The maximum size of transmitted packets. Before you set this option, ensure that the NIC that is used to transmit, the NIC that is used to receive, and the network switch support this packet size.
11. Start your receiver and listen for data from the transmitter.

    If you are using eBUS Player as the receiver, perform the steps in "To receive the test pattern or images with eBUS Player" on page 26 to set up eBUS Player to receive the image stream.

12. Click **Play** under **Acquisition Controls**.

    A tiled input will appear in the application and the receiver that you started in step 11.

You can configure each device by right-clicking the device and clicking **Properties**.

Receiving, Processing, and Transmitting Images

The TransmitProcessedImage sample receives images from a GigE Vision transmitter, prints statistics on them, and multicasts them using the PvTransmitterRaw class to a given destination.

The TransmitProcessedImage sample is only available for the C++ programming language.

To receive, process, and transmit images

1. Ensure that the GigE Vision transmitter and GigE Vision receiver are on the same subnet, and is reachable from the computer that is running the sample.

   Typically, two different NICs are used, so that you can receive with one NIC and transmit with the other. However if the incoming and outgoing throughput is low enough, it is possible to transmit and receive on the same NIC.

   Note: If the transmitting computer has more than one NIC, we recommend that the NICs be on different subnets.

   Note: If your computer only has one NIC with a valid IP address, start the sample with the default options (no command line arguments). Otherwise, note the IP address of the NIC from which you want to transmit. Run the TransmitProcessedImage sample and then add the following command line argument:

   ```
   --sourceaddress=<the IP address of the NIC you wish to transmit from>
   ```

2. When prompted, select the GigE Vision transmitter from which you want to receive video.
3. Press any key when you are ready to begin transmitting (and the receiver is ready to receive the image stream).

   By default, a vDisplay HDI-Pro External Frame Grabber can receive the stream using the default settings, since it subscribes to the same multicast address and port by default.
4. Start your receiver and listen for data from the transmitter.

   If you are using eBUS Player as the receiver, perform the steps in "To receive the test pattern or images with eBUS Player" on page 26 to set up eBUS Player to receive the image stream.

Using the GigE Vision Packet Resend Feature

Version 5.0 of the eBUS SDK adds support for the GigE Vision packet resend feature, which allows the eBUS SDK to respond to packet resend requests from GigE Vision receivers, such as a vDisplay HDI-Pro External Frame Grabber, eBUS Player, or any other GigE Vision compliant receiver. This functionality allows the Video Server to retransmit packets that have been lost.

The TransmitTestPattern sample in version 5.0 (and later) of the eBUS SDK illustrates how the packet resend feature is used. For additional information, see the PvTransmitterGEV and PvVirtualDeviceGEV classes in the *eBUS SDK C++ API Help File* or the *eBUS SDK .NET API Help File*.

Chapter 6

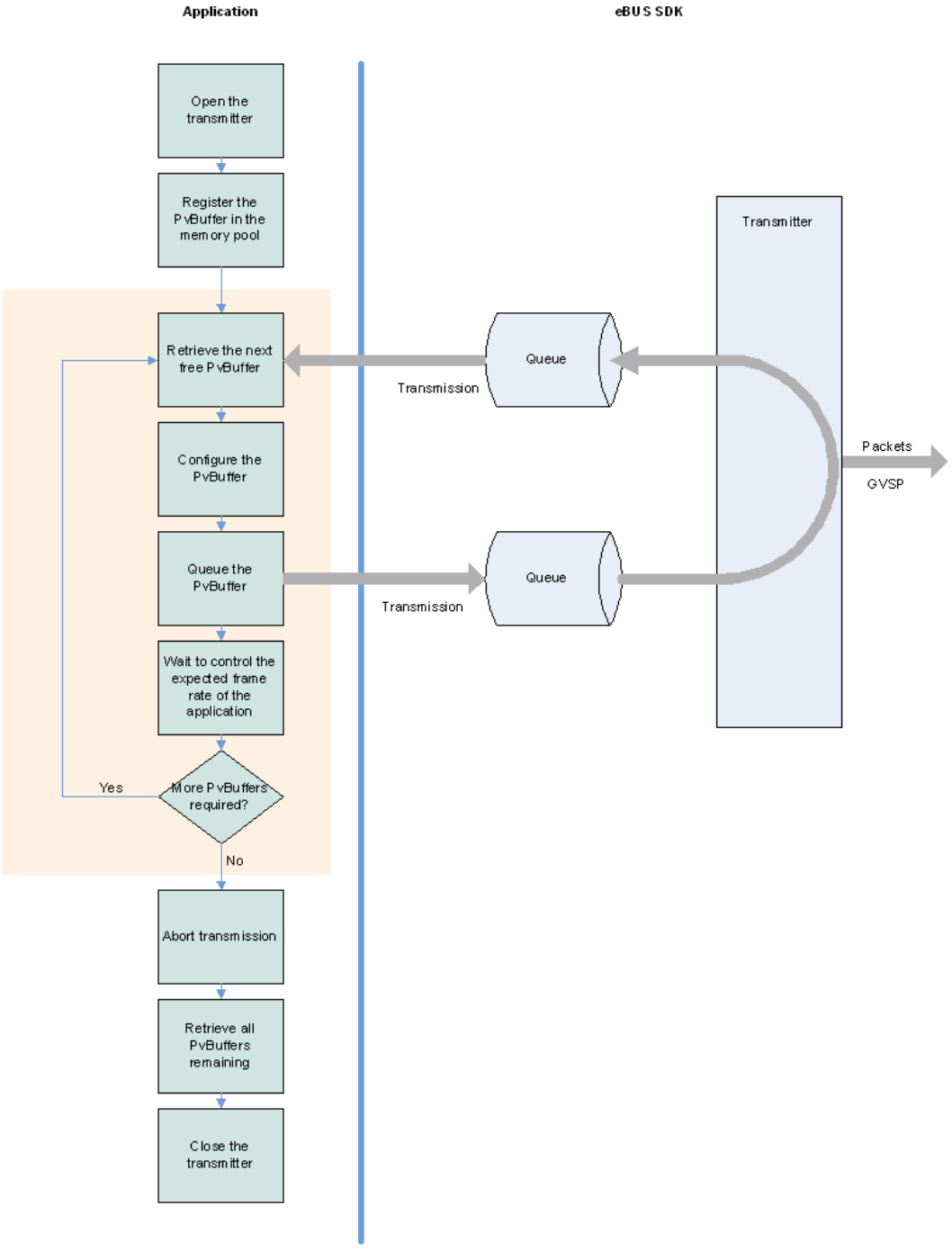
Basic API Calls to Create the Video Server Application

To create a video server application, you can use these basic API calls:

1. Create buffers by doing the following:
   - Allocate memory and specify the type (image), height, width, and pixel format using **PvBuffer::GetImage**() and **PvImage::Alloc**().

     Note: The eBUS SDK .NET classes expose properties wherever possible, that replace the **Get** and **Set** methods that are used in the C++ version. In the call above, **PvBuffer::GetImage**() is replaced by **PvBuffer.Image**.
   - Initialize the transmitter with the PvBuffer just created by calling **PvTransmitterGEV::LoadBufferPool**().
2. Start listening for connection requests.
   - It is necessary for receiver applications using the eBUS SDK to connect to a server application, even if they are unicast or multicast receivers only.
   - Use **PvVirtualDeviceGEV::StartListening**().
3. Specify the stream source and destination.
   - Choose the source IP address (especially when there are multiple NICs on each computer) and port (if required).
   - Choose the destination IP address (either unicast or multicast) and port.
   - Call **PvTransmitterGEV::Open**().
4. Transmit images in a loop until the server process is complete.
   - Call **PvTransmitter::RetrieveFreeBuffer**() to get an available buffer.
   - Fill the buffer with image data.
   - Call **PvTransmitterGEV::QueueBuffer**() to have the SDK asynchronously send the buffer.
5. Shut down the video server.
   - Call **PvTransmitterGEV::AbortQueuedBuffers**() to stop transmitting.
   - Loop through the buffers using **PvTransmitter::RetrieveFreeBuffer**() and free the memory.
   - Call **PvVirtualDeviceGEV::StopListening**()
   - Call**PvTransmitterGEV::Close**()

Figure 1: Basic API Calls Workflow



Chapter 7

Troubleshooting

This chapter provides you with troubleshooting tips and recommended solutions for issues that can occur when using the Video Server API.

Not all scenarios and solutions are listed here. You can refer to the Pleora Technologies Support Center at www.pleora.com for additional support and assistance. Details for creating a customer account are available on the Pleora Technologies Support Center.

Refer to the product release notes that are available on the Pleora Technologies Support Center for known issues and other product features.

Troubleshooting Tips

The scenarios and known issues listed in this chapter are those that you might encounter during the setup and operation of your device. Not all possible scenarios and errors are presented. The symptoms, possible causes, and resolutions depend upon your particular setup and operation.

If you perform the resolution for your issue and the issue is not corrected, we recommend you review the other resolutions listed in this table. Some symptoms may be interrelated.

Table 4: Troubleshooting Tips

Symptom	Possible cause	Resolution
SDK cannot detect or connect to the Pleora device	Power not supplied to the device, or inadequate power supplied	Both the detection and connection to the device will fail if adequate power is not supplied to the device. Verify that the Network LED is active. For information about the LEDs, see the documentation accompanying the device. Re-try the connection to the device with your application.
	The device is not connected to the network	Verify that the network LED is active. If this LED is illuminated, check the LEDs on your network switch to ensure the switch is functioning properly. If the problem continues, connect the device directly to the computer to verify its operation. For information about the LEDs, see the documentation accompanying the device.
	The device and computer are not on the same subnet	Images might not appear in your application if the device and the computer running your application are not on the same subnet. Ensure that these devices are on the same subnet. In addition, ensure that these devices are connected using valid gateway and subnet mask information. You can view the IP address information in the **Available Devices** list in your application. A red icon appears beside the device if there is an invalid IP configuration.
SDK cannot detect the API or transmitter	NIC that is receiving and NIC that is transmitting are on different subnets	Ensure the transmitting and receiving NICs are on the same subnet.
Errors appear	The drivers for your NIC may not be the latest version	Ensure you have installed the latest drivers from the manufacturer of your NIC.

Table 4: Troubleshooting Tips (Continued)

Symptom	Possible cause	Resolution
SDK is able to connect, but no images appear in your application. In a multicast GigE Vision configuration, images appear on a display monitor connected to a vDisplay HDI-Pro External Frame Grabber but do not appear in your application.	In a multicast configuration, the device may not be configured correctly	Images only appear on the display if you have configured the device for a multicast network configuration. The device and all multicast receivers must have identical values for both the **GevSCDA** and **GevSCPHostPort** features in the **TransportLayerControl** section. For more information, see the documentation accompanying the device.
	In a multicast configuration, your computer's firewall may be blocking your application	Ensure that your application is allowed to communicate through the firewall.
	Anti-virus software or firewalls blocking transmission	Images might not appear in your application because of anti-virus software or firewalls on your network. Disable all virus scanning software and firewalls, and re-attempt a connection to the device with your application.
	Ensure jumbo packets are properly configured for the NIC	Enable jumbo packet support for the NIC and network switch (as required). If the NIC or network switch does not support jumbo packets, disable jumbo packets for the transmitter.

Table 4: Troubleshooting Tips (Continued)

Symptom	Possible cause	Resolution
Dropped packets: eBUS Player, NetCommand, or applications created using the eBUS SDK	Insufficient computer performance	The computer being used to receive images from the device may not perform well enough to handle the data rate of the image stream. The GigE Vision driver reduces the amount of computer resources required to receive images and is recommended for applications that require high throughput. Should the application continue to drop packets even after the installation of the GigE Vision driver, a computer with better performance may be required.
	Insufficient NIC performance	The NIC being used to receive images from the GigE Vision device may not perform well enough to handle the data rate of the image stream. For example, the bus connecting the NIC to the CPU may not be fast enough, or certain default settings on the NIC may not be appropriate for reception of a high-throughput image stream. Examples of NIC settings that may need to be reconfigured include the number of Rx Descriptors and the maximum size of Ethernet packets (jumbo packets). Additionally, some NICs are known to not work well in high-throughput applications. For information about maximizing the performance of your system, see the *Configuring Your Computer and Network Adapters for Best Performance Application Note*, available on the Pleora Support Center.
	RequestMissingPacket is set to **True** on the receiver	The Video Server API does not support packet resend. On the receiver, set **RequestMissingPacket** to **False** to stop the receiver from sending resend requests, which may affect the performance of the system.

Chapter 8

Technical Support

On the Pleora Support Center, you can:

- Download the latest software.
- Log a support issue.
- View documentation for current and past releases.
- Browse for solutions to problems other customers have encountered.
- Get presentations and application notes.
- Get the latest news and information about our products.
- Decide which of Pleora's products work best for you.

To visit the Pleora Support Center

- Go to www.pleora.com and click **Support Center**.

  If you have not registered yet, you are prompted to register.

  Accounts are usually validated within one business day.