

PLEORA TECHNOLOGIES INC.



Linux Quick Start Guide



Table of Contents

1	About this Guide.....	3
1.1	About the eBUS SDK for Linux	3
1.2	What this Guide Provides	3
1.3	Documented Product Version.....	4
1.4	Related Documents	4
2	Installing the eBUS SDK for Linux.....	6
2.1	System Requirements	6
2.2	Installing the eBUS SDK for Linux.....	7
2.3	Installing the eBUS Runtime for Linux	24
2.4	Choosing Optimal Settings for the Jetson Modules	28
3	eBUS SDK Licenses.....	30
3.1	Activating a File-Based License.....	30
3.2	Activating a USB License Dongle.....	30
3.3	Activating a Cryptlex based Evaluation License	31
4	Optimizing Operation with GigE Vision Devices	33
4.1	Using the eBUS Universal Pro for Ethernet Filter Driver	33
4.2	Enabling Jumbo Ethernet Frames	34
4.3	Additional optimization steps	35
5	Providing Access to Third-Party USB3 Vision Transmitter Devices	36
6	Using eBUS Player to Configure Devices and Stream Images	37
7	Compiling and Running Sample Applications.....	40
8	Troubleshooting	42

Document Number and Revisions

Document ID: QSG-0023

Revision	Reason for Revision	Date
1.1	Update Install section	January 2026
1.0	Initial Release with eBUS 7.0.0	November 2025

1 About this Guide

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

1.1 About the eBUS SDK for Linux

eBUS SDK is built on a single API to receive video over GigE, 10 GigE, and USB 3.0 that is portable across Windows and Linux operating systems. With an eBUS SDK Seat License, designers can develop production-ready software applications in the same environment as their end-users, and quickly and easily modify applications for different media, while avoiding supporting multiple APIs from various vendors. Compared to camera vendor provided SDKs, eBUS frees developers from being tied to a specific camera, and instead they can choose the device that is best for the application.

eBUS Edge for Sensor Devices

eBUS Edge is a software implementation of a full device level GigE Vision transmitter, without requiring any additional hardware. Adding eBUS Edge to a CPU's software stack turns it into a fully compliant GigE Vision device that supports image transmission and enables the device to respond to control requests from a host controller. eBUS Edge is GigE Vision and GenICam compliant, meaning end-users can use any standards-compliant third-party image processing system. eBUS Edge currently supports the GigE Vision standard.

eBUS Receive for Host Applications

eBUS Receive manages high-speed reception of images or data into buffers for hand-off to the end application for further analysis. Developers can write applications that run on a host computer to seamlessly control and configure an unlimited number of GigE Vision or USB3 Vision and GenICam compliant sensors.

The eBUS Universal Pro driver reduces CPU usage when receiving images or data, leaving more processing power for analysis and inspection applications while helping meet latency and throughput requirements for real-time applications. The eBUS Universal Pro driver is easily integrated into third-party processing software to bring performance advantages to end-user applications.

1.2 What this Guide Provides

This guide provides you with the steps to use the eBUS SDK on the Linux operating system, on a Linux x86_64, or ARM platform. This guide is intended for novice Linux users, although advanced Linux users may be interested in some of the eBUS SDK-specific elements of this guide.

1.3 Documented Product Version

This guide covers Release 7.0 of the eBUS SDK. The features and functionality documented in this guide may vary if you are using an earlier or later version of the eBUS SDK.

1.4 Related Documents

eBUS SDK Related Documents

Guides are complemented by the following Pleora Technologies documents, which are available on the Pleora Technologies Support Center (supportcenter.pleora.com):

- [C++ API Quick Start Guide](#)
This guide provides you with the information you need to install the eBUS SDK (which lets you use the eBUS SDK C++ API) and an overview of the system requirements.
- [Docker Support for eBUS User Guide](#)
This guide provides instruction on how to get started with using eBUS SDK in a Docker environment. It also provides basic instruction on how to set up Docker on a Ubuntu-based system.
- [Getting Started with eBUS Edge](#)
The eBUS Edge API (introduced in eBUS SDK 7.0) allows developers to create a software-based GigE Vision transmitter device. eBUS Edge is fully compliant with GigE Vision and GenICam and will work with any GigE Vision and GenICam compliant third-party image processing systems or software.
- [eBUS SDK 7.0 \(and later\) Licensing](#)
This knowledge base article explains the eBUS SDK license structure, explains how to obtain a license, and provides activation instructions. If you are experiencing difficulty activating your license, please review the troubleshooting steps at the end of this publication.
- [Linux Quick Start Guide](#)
This guide provides you with the steps to use the eBUS SDK on the Linux operating system, on a Linux x86_64, or ARM platform. This guide is intended for novice Linux users, although advanced Linux users may be interested in some of the eBUS SDK-specific elements of this guide.
- [.NET API Quick Start Guide](#)
This guide provides you with 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 eBUS SDK .NET API.
- [eBUS Player User Guide](#)
This guide provides in-depth details about setting up and using the eBUS Player software application to control your GigE Vision or USB3 Vision compliant video transmitters (cameras) and receivers.
- [eBUS Player Quick Start Guide](#)
This quick start guide provides you with the information you need to start using the eBUS Player application, which lets you control the parameters of your GigE Vision or USB3 Vision compliant device and lets you view imaging video and data.
- [Python API Quick Start Guide](#)
This guide provides you with the information you need to install the eBUS SDK (which lets you use the eBUS SDK Python API) and an overview of the system requirements.
- [Raspberry PI 4/5 Quick Start Guide](#)
This guide provides you with the steps to use the eBUS SDK on a Raspberry PI 4/5. This guide is

intended for novice Raspberry PI 4/5 users, although advanced Raspberry PI 4/5 users may be interested in some of the eBUS SDK-specific elements of this guide.

- [Supported Software Ecosystem](#)

This support summary provides an overview of the supported protocols, operating systems, ARM platforms, development environments, and drivers. Pleora is currently supporting eBUS SDK 7.0 and later.

2 Installing the eBUS SDK for Linux

This chapter provides system requirements and steps that you will use to install the eBUS SDK for Linux.

2.1 System Requirements

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



You can access installation files from the Pleora Support Center at supportcenter.pleora.com¹.

At least one Gigabit Ethernet NIC (if you are using GigE Vision devices) or at least one USB 3.0 port (if you are using USB3 Vision devices).

💡 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 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.

One of the following operating systems:

- For x86_64 Linux platform:
 - Ubuntu 24.04 LTS 64-bit
 - Ubuntu 22.04 LTS 64-bit
 - Ubuntu 20.04 LTS 64-bit
 - RHEL 9.7, 64-bit
 - CentOS Stream 9, 64 bit
- For Linux-ARM based platform:
 - NVIDIA Jetson AGX Orin, Jetson Orin NX, Jetson Orin Nano running Jetpack 6.2. (Ubuntu 22.04)
 - NVIDIA Jetson AGX Xavier, Jetson Xavier NX, Jetson AGX Orin, Jetson Orin NX running Jetpack 5.1.4. (Ubuntu 20.04)
 - Raspberry Pi 4B, Raspberry Pi 5 running Raspberry Pi OS (64 bits) (Debian aarch64 GNU/Linux 13 Trixie)
 - ST Micro STM32 MP2 running OpenSTLinux 6.1 OS. Please refer to *eBUS Edge on OpenSTLinux (ST Micro MP2 Platform) Quick Start Guide* available on the Pleora Technologies Support Center at supportcenter.pleora.com²

1. <http://supportcenter.pleora.com>

2. <http://supportcenter.pleora.com/>

- Yocto 4.0 (Kirkstone) and Yocto 5.0 (Scarthgap) (on NXP i.MX8, Raspberry Pi 4B / 5, MediaTek Genio 510). Please refer to *Using eBUS SDK in Yocto Environments User Guide* available on the Pleora Technologies Support Center at supportcenter.pleora.com³
- ROS2 camera-driver support for GigE Vision Receivers, See *ROS2 Receiver Driver for GigE Vision Cameras* available on the Pleora Technologies Support Center at supportcenter.pleora.com⁴

 According to the x86_64 Linux or Linux-ARM based platform, we provide eBUS Python for the stock Python version of the OS:

- CentOS Stream 9 (64-bit), eBUS Python for Python 3.9 is installed with the SDK
- Ubuntu 24.04 Desktop (64-bit), eBUS Python for Python 3.12 is installed with the SDK
- Ubuntu 24.04 for ARM (64-bit), eBUS Python for Python 3.12 is installed with the SDK
- Ubuntu 22.04 Desktop (64-bit), eBUS Python for Python 3.10 is installed with the SDK
- Ubuntu 22.04 for ARM (64-bit), eBUS Python for Python 3.10 is installed with the SDK
- Ubuntu 20.04 Desktop (64-bit), eBUS Python for Python 3.8 is installed with the SDK
- Ubuntu 20.04 for ARM (64-bit), eBUS Python for Python 3.8 is installed with the SDK
- Raspberry Pi Desktop (64-bit), eBUS Python for Python 3.13 is installed with the SDK

 For non-default Python versions for different Linux ARM/x86_64 platforms, consult your Pleora support representative.

 Ensure that you uninstall any previous versions of eBUS SDK from your machine before installing eBUS SDK 7.x.

2.2 Installing the eBUS SDK for Linux

Use the installation package to install the eBUS SDK and eBUS Runtime for Linux.

Linux x86_64

CentOS Stream 9

Steps	Comments
As super user, • yum install python3-numpy	Without python3-numpy, eBUS SDK for CentOS Stream 9 cannot be installed

3. <http://supportcenter.pleora.com/>

4. <http://supportcenter.pleora.com/>

As super user, • yum install python3-pip	Without python3-pip, eBUS Python package cannot be installed.
As super user, • dnf install chkconfig	Without chkconfig, set_usbfs_memory_size.sh is not run and eBUS Universal Pro driver is not started when installing eBUS SDK.
As super user, rpm -iv eBUS_SDK_CentOS-RHEL-86_64-<version-build number>.rpm	eBUS SDK for CentOS Stream 9 is installed
As super user, • dnf install libglvnd-opengl	Without libglvnd-opengl, eBUSLicensingTool and eBUSPlayer fail to start.
<u>Install FFMPEG runtime:</u> As super user, 1. yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm ⁵ 2. yum install -y https://download1.rpmfusion.org/free/el/rpmfusion-free-release-9.noarch.rpm ⁶ https://download1.rpmfusion.org/nonfree/el/rpmfusion-nonfree-release-9.noarch.rpm ⁷ 3. yum config-manager --set-enabled crb 4. yum install -y ffmpeg	Without FFMPEG runtime package, • eBUS Player, Classic eBUS Player and DualSource cannot be started, because depends on libavformat.so.59 which is not installed by default. • C++ GUI samples cannot be compiled, because depends on libavcodec.so.59 which is not installed by default on OS. • DualSource, • eBUSPlayerClassic • All Python samples cannot be run because depends of libavutil.so.59 which is not installed by default.
As super user, • dnf install mpg123	Without mpg123, • eBUS Player, Classic eBUS Player and DualSource cannot be started, because mpg123_param2 is undefined. • eBUSPlayerClassic and DualSource samples cannot be compiled, because mpg123_param2 is undefined. • All Python samples cannot be run because mpg123_param2 is undefined.

5. <https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm>

6. <https://download1.rpmfusion.org/free/el/rpmfusion-free-release-8.noarch.rpm>

7. <https://download1.rpmfusion.org/nonfree/el/rpmfusion-nonfree-release-8.noarch.rpm>

As super user, • dnf install xcb-util-cursor	Without xcb-util-cursor, • eBUS Player cannot be started, because cannot load the Qt xcb platform plugin.
<u>Remove firewall:</u> As super user, 1. systemctl disable firewalld 2. log off and log on OR 1. systemctl stop firewalld 2. systemctl disable firewalld	With the firewall, eBUS Player can be started, but GEV devices are not detected.
<u>Optional step to use eBUS DotNet:</u> As super user, • dnf install dotnet-sdk-8.0	dotnet-sdk-8.0 is required to use eBUS DotNet on Linux.
<u>Optional step to get Python display:</u> 1. python3 -m pip install --user --upgrade pip 2. python3 -m pip install --user 'opencv-python<4' --prefer-binary	Without python3-opencv package, some Python samples cannot display the streaming in a separate window. • PvPipelineSample.py • PvStreamSample.py • ImageProcessing.py • ReceiveMultiPart.py

Optional step to get MP4 saving option with eBUSPlayerClassic sample:

1. As super user, yum install ffmpeg-devel
2. modify the Makefile in eBUSPlayerClassic folder.
 - a. Add these lines at the end of the Makefile

```
LDLFLAGS += -lswscale \
            -lavcodec \
            -lavformat\
            -lavutil

CPPFLAGS += -D
__STDC_CONSTANT_MACROS -D
PV_ENABLE_MP4 -I /usr/include/
ffmpeg
```

3. Clean the previous compilation with *make clean*.
4. Recompile eBUSPlayerClassic sample with the *make* command.

Without FFMPEG header files, eBUSPlayerClassic sample cannot included the MP4 saving option.

RHEL 9 (Red Hat Enterprise Linux 9)

Steps	Comments
As super user, • yum install python3-numpy	Without python3-numpy, eBUS SDK or eBUS Runtime for RHEL 9 cannot be not installed. But, eBUS_Base_Runtime_RHEL-CentOS-x86_64- <version-build number>.rpm can be installed without problem.
As super user, • yum install python3-pip	Without python3-pip, eBUS Python package cannot be installed.
As super user, • dnf install chkconfig	Without chkconfig, set_usbfs_memory_size.sh is not run and eBUS Universal Pro driver is not started when installing eBUS SDK.
As super user, rpm -iv eBUS_SDK_RHEL-CentOS-86_64-<version-build number>.rpm	eBUS SDK for RHEL 9 is installed
As super user, • dnf install libglvnd-opengl	Without libglvnd-opengl, eBUSLicensingTool and eBUSPlayer fail to start.

<u>Install FFMPEG runtime:</u> As super user, 1. yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm⁸ 2. yum install -y https://download1.rpmfusion.org/free/el/rpmfusion-free-release-9.noarch.rpm⁹ https://download1.rpmfusion.org/nonfree/el/rpmfusion-nonfree-release-9.noarch.rpm¹⁰ 3. yum config-manager --set-enabled crb 4. yum install -y ffmpeg	Without FFMPEG runtime package, • eBUS Player, Classic eBUS Player and DualSource cannot be started, because depends on libavformat.so.59 which is not installed by default. • C++ GUI samples cannot be compiled, because depends on libavcodec.so.59 which is not installed by default on OS. • DualSource, • eBUSPlayerClassic • All Python samples cannot be run because depends of libavutil.so.59 which is not installed by default.
As super user, • dnf install mpg123	Without mpg123, • eBUS Player, Classic eBUS Player and DualSource cannot be started, because mpg123_param2 is undefined. • eBUSPlayerClassic and DualSource samples cannot be compiled, because mpg123_param2 is undefined. • All Python samples cannot be run because mpg123_param2 is undefined.
As super user, • dnf install xcb-util-cursor	Without xcb-util-cursor, • eBUS Player cannot be started, because cannot load the Qt xcb platform plugin.
<u>Remove firewall:</u> 1. sudo systemctl disable firewalld 2. log off and log on OR 1. sudo systemctl stop firewalld 2. sudo systemctl disable firewalld	With the firewall, eBUS Player can be started, but GEV devices are not detected.
<u>Optional step to use eBUS DotNet:</u> As super user, • dnf install dotnet-sdk-8.0	dotnet-sdk-8.0 is required to use eBUS DotNet on Linux.

8. <https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm>

9. <https://download1.rpmfusion.org/free/el/rpmfusion-free-release-8.noarch.rpm>

10. <https://download1.rpmfusion.org/nonfree/el/rpmfusion-nonfree-release-8.noarch.rpm>

Optional step to get Python display: 1. <code>python3 -m pip install --user --upgrade pip</code> 2. <code>python3 -m pip install --user 'opencv-python<4' --prefer-binary</code>	Without python3-opencv package, some Python samples cannot display the streaming in a separate window. • PvPipelineSample.py • PvStreamSample.py • ImageProcessing.py • ReceiveMultiPart.py
Optional step to get MP4 saving option with eBUSPlayerClassic sample: 1. <code>sudo yum install ffmpeg-devel</code> 2. modify the Makefile in eBUSPlayerClassic folder. a. Add these lines at the end of the Makefile <pre>LDFLAGS += -lswscale \ -lavcodec \ -lavformat\ -lavutil CPPFLAGS += -D __STDC_CONSTANT_MACROS -D PV_ENABLE_MP4 -I /usr/include/ ffmpeg</pre> 3. Clean the previous compilation with <i>make clean</i>. 4. Recompile eBUSPlayerClassic sample with the <i>make</i> command.	

Ubuntu 20.04

Steps	Comments
<pre>sudo apt-get update sudo apt-get install build-essential</pre>	Without build-essential, eBUS SDK, eBUS Runtime and eBUS Runtime Base for Ubuntu 20.04 cannot be installed.
<pre>sudo apt-get install python3-pip</pre>	Without python3-pip, eBUS Python package cannot be installed.
<pre>sudo dpkg -i eBUS_SDK_Ubuntu-20.04-x86_64- <version-build number>.deb</pre>	eBUS SDK for Ubuntu 20.04 is installed

sudo apt-get install libxcb-cursor0	Without libxcb-cursor0, eBUS Player, eBUSLicensingTool, DualSource and Classic eBUS Player cannot started.
sudo apt-get install libavcodec58	Without libavcodec58, • Classic eBUS Player and DualSource cannot be started, because depends on libavcodec.so.58 which is not installed by default on OS. • C++ GUI samples cannot be compiled, because depends on libavcodec.so.58 which is not installed by default on OS: • DualSource • eBUSPlayerClassic • All others C++ samples can be compiled and executed. •  When python3-numpy is installed, all Python samples can be run without libavcodec58.
sudo apt-get install python3-numpy	Without python3-numpy, eBUS Python cannot used and all Python samples cannot be started, because depends on "numpy.core.multiarray" module which is not installed by default.  eBUS Python doesn't require libavcodec.
Optional step to get Python display: sudo apt-get install python3-opencv	Without python3-opencv, some Python samples cannot display the streaming in a separate window: • PvPipelineSample.py • PvStreamSample.py • ImageProcessing.py • ReceiveMultiPart.py
Optional steps to use eBUS DotNet: Add the Microsoft package repository: • wget https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.deb -O packages-microsoft-prod.deb • sudo dpkg -i packages-microsoft-prod.deb • rm packages-microsoft-prod.deb Install the SDK: • sudo apt-get update && \ • sudo apt-get install -y dotnet-sdk-8.0	dotnet-sdk-8.0 is required to use eBUS DotNet on Linux.

Optional step to get MP4 saving option with eBUSPlayerClassic sample:

1. `sudo apt-get install libswscale-dev libavcodec-dev libavformat-dev`
2. modify the Makefile in eBUSPlayerClassic folder.
 - a. Add these lines at the end of the Makefile

```
LDFLAGS += -lswscale \
           -lavcodec \
           -lavformat\
           -lavutil

CPPFLAGS += -DPV_ENABLE_MP4
```

3. Clean the previous compilation with *make clean*.
4. Recompile eBUSPlayerClassic sample with the *make* command.

Without FFMPEG header files, eBUSPlayerClassic sample cannot be compiled with MP4 saving option.

Ubuntu 22.04

Steps	Comments
<code>sudo apt-get update</code> <code>sudo apt-get install build-essential</code>	Without build-essential, eBUS SDK, eBUS Runtime and eBUS Base Runtime for Ubuntu 22.04 cannot be installed.
<code>sudo apt-get install gcc-12</code>	Without gcc-12, eBUS SDK for Ubuntu 22.04 is installed, but eBUS Universal Pro driver build fails and his installation is skipped.
<code>sudo apt-get install python3-pip</code>	Without python3-pip, eBUS Python package cannot be installed.
<code>sudo dpkg -i eBUS_SDK_Ubuntu-22.04-x86_64-<version-build number>.deb</code>	eBUS SDK for Ubuntu 22.04 is installed without error to build and install eBUS Universal Pro driver.
<code>sudo apt-get install libxcb-cursor0</code>	Without libxcb-cursor0, - eBUS Player cannot be started because libxcb-cursor0 is needed to load Qt xcb platform plugin.  eBUSLicensingTool can started without issue.

<pre>sudo apt-get install libavcodec58</pre>	Without libavcodec58, • Classic eBUS Player and DualSource cannot be started, because depends on libavcodec.so.58 which is not installed by default on OS. • C++ GUI samples cannot be compiled, because depends on libavcodec.so.58 which is not installed by default on OS. • DualSource • eBUSPlayerClassic. • All others C++ samples can be compiled and executed. •  When python3-numpy is installed, all Python samples can be run without libavcodec58. •  eBUSLicensingTool can started without issue.
<pre>sudo apt-get install python3-numpy</pre>	Without python3-numpy, eBUS Python cannot used and all Python samples cannot be started, because depends on "<i>numpy.core.multiarray</i>" module which is not installed by default.  eBUS Python doesn't require libavcodec.
<u>Optional step</u> to get Python display: <pre>sudo apt-get install python3-opencv</pre>	Without python3-opencv, some Python samples cannot display the streaming in a separate window. • PvPipelineSample.py • PvStreamSample.py • ImageProcessing.py • ReceiveMultiPart.py
<u>Optional step</u> to use eBUS DotNet: Install the SDK: • <code>sudo apt-get update && \</code> <code>sudo apt-get install -y dotnet-sdk-8.0</code>	dotnet-sdk-8.0 is required to use eBUS DotNet on Linux.

Optional step to get MP4 saving option with eBUSPlayerClassic sample: 1. sudo apt-get install libswscale-dev libavcodec-dev libavformat-dev 2. modify the Makefile in eBUSPlayerClassic folder. a. Add these lines at the end of the Makefile <pre>LDFLAGS += -lswscale \ -lavcodec \ -lavformat\ -lavutil CPPFLAGS += -DPV_ENABLE_MP4</pre> 3. Clean the previous compilation with <i>make clean</i>. 4. Recompile eBUSPlayerClassic sample with the <i>make</i> command.	Without FFMPEG header files, eBUSPlayerClassic sample cannot be compiled with MP4 saving option.
---	---

Ubuntu 24.04

Steps	Comments
sudo apt-get update sudo apt-get install build-essential	Without build-essential, eBUS SDK, eBUS Runtime and eBUS Base Runtime for Ubuntu 24.04 cannot be installed.
sudo dpkg -i eBUS_SDK_Ubuntu-24.04-x86_64-<version-build number>.deb	eBUS SDK for Ubuntu 24.04 is installed without error to build and install eBUS Universal Pro driver.  For this OS, eBUS Python installation is skipped when installing eBUS SDK package.
sudo apt-get install libxcb-cursor-dev	Without libxcb-cursor-dev, eBUS Player cannot be started because libxcb-cursor0 is needed to load Qt xcb platform plugin.

<pre>sudo apt-get install libavcodec60</pre>	Without libavcodec60, • Classic eBUS Player, DualSource cannot be started, because depends on libavcodec.so.60 which is not installed by default on OS. • C++ GUI samples cannot be compiled, because depends on libavcodec.so.60 which is not installed by default on OS. • DualSource • eBUSPlayerClassic • All others C++ samples can be compiled and executed.
<u>Create your own virtual environment to install eBUS Python:</u> 1. <code>sudo apt-get install -y python3.12-venv</code> 2. <code>python3 -m venv ~/ebus_env</code>	Without this virtual environment, the installation of eBUS Python will not be possible for this OS
<u>Steps to install eBUS Python (base,edge and receive):</u> 1. <code>source ~/ebus_env/bin/activate</code> 2. <code>python3 -m pip install /opt/pleora/ebus_sdk/Ubuntu-24.04-x86_64/lib/eBUSPython/ebus_base-7.x.y-<build number>-py312-none-manylinux2014_x86_64.whl</code> 3. <code>python3 -m pip install /opt/pleora/ebus_sdk/Ubuntu-24.04-x86_64/lib/eBUSPython/ebus_edge-7.x.y-<build number>-py312-none-manylinux2014_x86_64.whl</code> 4. <code>python3 -m pip install /opt/pleora/ebus_sdk/Ubuntu-24.04-x86_64/lib/eBUSPython/ebus_receive-7.x.y-<build number>-py312-none-manylinux2014_x86_64.whl</code> 5. <code>deactivate</code>	eBUS Python for Ubuntu 24.04 is installed.
1. <code>source ~/ebus_env/bin/activate</code> 2. <code>pip install 'numpy<2'</code> 3. <code>deactivate</code>	Without python3-numpy in this virtual environment, eBUS Python cannot be used and all Python samples cannot be started, because depends on "<i>numpy.core.multiarray</i>" module.
<u>Optional step to get Python display:</u> 1. <code>source ~/ebus_env/bin/activate</code> 2. <code>pip install 'opencv-python<4'</code> 3. <code>deactivate</code>	Without opencv-python in this virtual environment, some eBUS Python samples cannot display the streaming in a separate window: • <code>PvPipelineSample.py</code> • <code>PvStreamSample.py</code> • <code>ImageProcessing.py</code> • <code>ReceiveMultiPart.py</code>

Optional step to use eBUS DotNet: sudo apt-get install dotnet-sdk-8.0	dotnet-sdk-8.0 is required to use eBUS DotNet on Linux.
Optional step to get MP4 saving option with eBUSPlayerClassic sample: 1. sudo apt-get install libswscale-dev libavcodec-dev libavformat-dev 2. modify the Makefile in eBUSPlayerClassic folder. a. Add these lines at the end of the Makefile <pre>LDFLAGS += -lswscale \ -lavcodec \ -lavformat\ -lavutil CPPFLAGS += -DPV_ENABLE_MP4</pre> 3. Clean the previous compilation with <i>make clean</i>. 4. Recompile eBUSPlayerClassic sample with the <i>make</i> command.	Without FFMPEG header files, eBUSPlayerClassic sample cannot be compiled with MP4 saving option.

Linux-ARM

NVIDIA JetPack 5.1.x (Ubuntu 20.04 Linux-ARM based)

Steps	Comments
sudo apt-get install python3-pip	Without python3-pip, eBUS Python libraries are not installed during the installation of eBUS SDK
sudo dpkg -i eBUS_SDK_Jetson_<Jetpack version>_linux-aarch64-arm-<version-build number>.deb	eBUS SDK for Jetson linux arm is installed.

<pre>sudo apt-get update sudo apt-get install qt5-default</pre>	Without qt5-default, • Classic eBUS Player and DualSource cannot be started because the shared library libQt5Widgets is missing and not installed by default on OS. • C++ GUI samples cannot be compiled because qmake command is missing and not installed by default on OS: • DualSource • eBUSPlayerClassic • All others C++ samples can be compiled and executed.
<pre>sudo apt-get install libyaml-cpp0.6</pre>	libyaml-cpp0.6 is required for eBUS Player, Python samples, eBUS Edges samples and Video API Server samples.

Optional step to use eBUS DotNet:

1. download dotnet-sdk-8.0 for Linux Arm64 Binary from <https://dotnet.microsoft.com/en-us/download/dotnet/thank-you/sdk-8.0.416-linux-arm64-binaries> (<https://builds.dotnet.microsoft.com/dotnet/Sdk/8.0.416/dotnet-sdk-8.0.416-linux-arm64.tar.gz>)
2. copy the downloaded package on you HOME folder
3. install .NET dependencies packages
 - a. `sudo apt-get install ca-certificates`
 - b. `sudo apt-get install libc6`
 - c. `sudo apt-get install libssl1.1`
 - d. `sudo apt-get install tzdata`
4. Open a terminal and go the Home directory
5. `DOTNET_ARCHIVE=dotnet-sdk-8.0.416-linux-arm64.tar.gz`
6. `export DOTNET_ROOT=$(pwd)/.dotnet`
7. `mkdir -p "$DOTNET_ROOT" && tar xzf "$DOTNET_ARCHIVE" -C "$DOTNET_ROOT"`
8. Open a terminal and edit Bash shell profile: `.bashrc`
 - a. `gedit ~/.bashrc`
 - b. at the end of the file, add these following lines
 - i. `export DOTNET_ROOT=$HOME/.dotnet`
 - ii. `export PATH=$PATH:$DOTNET_ROOT`
 - c. save the file
9. Close terminal and reopen terminal. So that, `.bashrc` get sourced.

dotnet-sdk-8.0 is required to use eBUS DotNet on Linux.

Optional step to get MP4 saving option with eBUSPlayerClassic sample:

1. `sudo apt-get install libswscale-dev libavcodec-dev libavformat-dev`
2. modify the Makefile in eBUSPlayerClassic folder.
 - a. Add these lines at the end of the Makefile

```
LDFLAGS += -lswscale \
          -lavcodec \
          -lavformat\
          -lavutil

CPPFLAGS += -DPV_ENABLE_MP4
```

3. Clean the previous compilation with *make clean*.
4. Recompile eBUSPlayerClassic sample with the *make* command.

Without FFMPEG header files, eBUSPlayerClassic sample cannot be compiled with MP4 saving option.

NVIDIA JetPack 6.2.x (Ubuntu 22.04 Linux-ARM based)

Steps	Comments
<code>sudo apt-get install python3-pip</code>	Without python3-pip, eBUS Python libraries are not installed during the installation of eBUS SDK
<code>sudo dpkg -i eBUS_SDK_Jetson_<Jetpack version>_linux-aarch64-arm-<version-build number>.deb</code>	eBUS SDK for Jetson linux arm is installed.

<pre>sudo apt-get update</pre> <pre>sudo apt-get install qtbase5-dev qt5-qmake</pre>	Without qtbase5-dev qt5-qmake, • Classic eBUS Player and DualSource cannot be started because the shared library libQt5Widgets is missing and not installed by default on OS. • C++ GUI samples cannot be compiled because qmake command is missing and not installed by default on OS: • DualSource • eBUSPlayerClassic • All others C++ samples can be compiled and executed.
<pre>sudo apt-get install libyaml-cpp0.6</pre>	libyaml-cpp0.6 is required for Classic eBUS Player, Python samples, eBUS Edges samples and Video API Server samples.
Optional step to use eBUS DotNet: <pre>sudo apt-get install dotnet-sdk-8.0</pre>	dotnet-sdk-8.0 is required to use eBUS DotNet on Linux.
Optional step to get MP4 saving option with eBUSPlayerClassic sample: 1. <code>sudo apt-get install libswscale-dev libavcodec-dev libavformat-dev</code> 2. modify the Makefile in eBUSPlayerClassic folder. a. Add these lines at the end of the Makefile <pre>LD_FLAGS += -lswscale \ -lavcodec \ -lavformat\ -lavutil CPPFLAGS += -DPV_ENABLE_MP4</pre> 3. Clean the previous compilation with <i>make clean</i>. 4. Recompile eBUSPlayerClassic sample with the <i>make</i> command.	Without FFMPEG header files, eBUSPlayerClassic sample cannot be compiled with MP4 saving option.

Raspberry Pi OS (Trixie)

Steps	Comments
-------	----------

<pre>sudo dpkg -i eBUS_SDK_Raspberry_Pi4_Pi5_linux- aarch64-arm-<version-build number>.deb</pre>	eBUS SDK for Raspberry Pi linux arm is installed.  For this OS, eBUS Python installation is skipped when installing eBUS SDK package.
<pre>sudo apt-get update sudo apt-get install qtbase5-dev qt5-qmake</pre>	Without qt5, • Classic eBUS Player and DualSource cannot be started because the shared library libQt5Widgets is missing and not installed by default on OS. • C++ GUI samples cannot be compiled because qmake command is missing and not installed by default on OS. • DualSource • eBUSPlayerClassic
<u>Create your own virtual environment to install eBUS Python:</u> <pre>python3 -m venv ~/ebus_env</pre>	Without this virtual environment, the installation of eBUS Python will not be possible for this OS
<u>Steps to install eBUS Python (base,edge and receive):</u> 1. source ~/ebus_env/bin/activate 2. python3 -m pip install /opt/pleora/ebus_sdk/linux-aarch64-arm/lib/eBUSPython/ebus_base-7.x.y-<build number>-py313-none-manylinux2014_aarch64.whl 3. python3 -m pip install /opt/pleora/ebus_sdk/linux-aarch64-arm/lib/eBUSPython/ebus_edge-7.x.y-<build number>-py313-none-manylinux2014_aarch64.whl 4. python3 -m pip install /opt/pleora/ebus_sdk/linux-aarch64-arm/lib/eBUSPython/ebus_receive-7.x.y-<build number>-py313-none-manylinux2014_aarch64.whl 5. deactivate	eBUS Python for Raspberry Pi linux arm is installed.
1. source ~/ebus_env/bin/activate 2. pip install 'numpy<2' 3. deactivate	Without python3-numpy, eBUS Python cannot be used and all Python samples cannot be started, because depends on "<i>numpy.core.multiarray</i>" module.

Optional step to get Python display: 1. source ~/ebus_env/bin/activate 2. pip install 'opencv-python<4' 3. deactivate	Without python3-opencv, some Python samples cannot display the streaming in a separate window. • PvPipelineSample.py • PvStreamSample.py • ImageProcessing.py • ReceiveMultiPart.py
Optional step to use eBUS DotNet: <pre>wget https://dot.net/v1/dotnet-install.sh -O dotnet-install.sh sudo chmod +x dotnet-install.sh ./dotnet-install.sh --channel 8.0 echo 'export DOTNET_ROOT=\$HOME/.dotnet' >> ~/.bashrc echo 'export PATH=\$PATH:\$HOME/.dotnet: \$HOME/.dotnet/tools' >> ~/.bashrc source ~/.bashrc</pre>	dotnet-sdk-8.0 is required to use eBUS DotNet on Linux.
Optional step to get MP4 saving option with eBUSPlayerClassic sample: 1. sudo apt-get install libswscale-dev libavcodec-dev libavformat-dev 2. modify the Makefile in eBUSPlayerClassic folder. a. Add these lines at the end of the Makefile <pre>LDFLAGS += -lswscale \ -lavcodec \ -lavformat\ -lavutil CPPFLAGS += -DPV_ENABLE_MP4</pre> 3. Clean the previous compilation with <i>make clean</i>. 4. Recompile eBUSPlayerClassic sample with the <i>make</i> command.	Without FFMPEG header files, eBUSPlayerClassic sample cannot be compiled with MP4 saving option.

2.3 Installing the eBUS Runtime for Linux

The installation of eBUS Runtime is very similar to the installation of eBUS SDK. According to eBUS Runtime type, some requirements, in the section "Installing the eBUS SDK for Linux" can be skipped.

After that you have installed the dependency packages for eBUS, you can install eBUS Runtime of your choice. From the terminal, execute the following command/s according to the eBUS Runtime Type.

eBUS Runtime Type	Packages	Commands
All Runtime Packages	eBUS Runtime	• <code>sudo dpkg -i eBUS_Runtime_<distribution targeted>-7.x.y-<build number>.deb</code> or • <code>rpm -iv eBUS_Runtime_<distribution targeted>-7.x.y-<build number>.rpm</code>
eBUS Edge (C++ and DotNet)	eBUS Base Runtime eBUS Edge Runtime	• <code>sudo dpkg -i eBUS_Base_Runtime_<distribution targeted>-7.x.y-<build number>.deb</code> • <code>sudo dpkg -i eBUS_Edge_Runtime_<distribution targeted>-7.x.y-<build number>.deb</code> or • <code>rpm -iv eBUS_Base_Runtime_<distribution targeted>-7.x.y-<build number>.rpm</code> • <code>rpm -iv eBUS_Edge_Runtime_<distribution targeted>-7.x.y-<build number>.rpm</code>
eBUS Edge for Python	eBUS Base Runtime eBUS Edge Runtime eBUS Base Python whl file eBUS Edge Python whl file	Same as eBUS Edge + the following commands: • <code>python3 -m pip install ebus_base-7.x.y-<build number>-py<python version>-none-manylinux2014_x86_64.whl</code> • <code>python3 -m pip install ebus_edge-7.x.y-<build number>-py<python version>-none-manylinux2014_x86_64.whl</code> i For Linux-ARM, replace x86_64 by aarch64
eBUS Receive (C++ and DotNet)	eBUS Base Runtime eBUS Receive Runtime	• <code>sudo dpkg -i eBUS_Base_Runtime_<distribution targeted>-7.x.y-<build number>.deb</code> • <code>sudo dpkg -i eBUS_Receive_Runtime_<distribution targeted>-7.x.y-<build number>.deb</code> or • <code>rpm -iv eBUS_Base_Runtime_<distribution targeted>-7.x.y-<build number>.rpm</code> • <code>rpm -iv eBUS_Receive_Runtime_<distribution targeted>-7.x.y-<build number>.rpm</code>

eBUS Receive for Python	eBUS Base Runtime eBUS Receive Runtime eBUS Base Python whl file eBUS Receive Python whl file	Same as eBUS Receive + the following commands: - python3 -m pip install ebus_base-7.x.y-<build number>-py<python version>-none-manylinux2014_x86_64.whl - python3 -m pip install ebus_receive-7.x.y-<build number>-py<python version>-none-manylinux2014_x86_64.whl i For Linux-ARM, replace x86_64 by aarch64
-------------------------	--	--

We recommend that you reboot the Linux to ensure that the correct environment variables are set.

💡 If there is a failure to compile and install the eBUS Universal Pro driver due to permissions or missing dependencies, you can manually compile and install the driver. For more information, see [“To manually install the eBUS Universal Pro for Ethernet Filter Driver”](#).

✔ If the eBUS SDK installation is not successful, your system may be missing the required Linux kernel headers.
For more information, see [“Error message: Cannot find the files to build kernel module in this PC”](#).

Uninstalling the eBUS SDK (and eBUS Runtime)

You can use the dpkg or rpm command to uninstall the eBUS SDK.

To uninstall the eBUS SDK

From the terminal, execute the following command.

- For Ubuntu based (x86_64 and ARM):
 - sudo dpkg -P ebus-sdk
- For RedHat/CentOS:
 - sudo rpm -e ebus-sdk

You can use the pip command to uninstall the eBUS Python. You can use the dpkg or rpm command to uninstall the eBUS Runtime.

To uninstall eBUS Runtime packages, Release 7.0 and higher

Execute the following commands according to the case:

- if eBUS Python Receive Runtime is installed:
 - python -m pip uninstall ebus-receive
- if eBUS Python Edge Runtime is installed:
 - python -m pip uninstall ebus-edge
- if eBUS Python Base Runtime is installed:
 - python -m pip uninstall ebus-base
- if eBUS_Receive_Runtime is installed:
 - For Ubuntu (x86_64 and ARM)

- `sudo dpkg -P ebus-receive-runtime`
- For RedHat/CentOS
 - `sudo rpm -e ebus-receive-runtime`
- if eBUS_Edge_Runtime is installed:
 - For Ubuntu based (x86_64 and ARM)
 - `sudo dpkg -P ebus-edge-runtime`
 - For RedHat/CentOS
 - `sudo rpm -e ebus-edge-runtime`
- if eBUS_Base_Runtime is installed:
 - For Ubuntu based (x86_64 and ARM)
 - `sudo dpkg -P ebus-base-runtime`
 - For RedHat/CentOS
 - `sudo rpm -e ebus-base-runtime`
- if eBUS_Runtime is installed:
 - For Ubuntu based (x86_64 and ARM)
 - `sudo dpkg -P ebus-runtime`
 - For RedHat/CentOS
 - `sudo rpm -e ebus-runtime`

Uninstalling an old eBUS SDK (and eBUS Runtime) for Linux

1. From the terminal, execute one of the following commands.
The command varies depending on the distribution you are using.
 - For eBUS SDK 6.3 (and higher):
 - For Ubuntu based (x86_64 and ARM)
 - `sudo dpkg -r ebus-sdk`
 - For RedHat/CentOS
 - `sudo rpm -e ebus-sdk`
 - For eBUS SDK 4.0 through 6.2:
 - On Ubuntu:
 - `sudo dpkg -r ebus_sdk_<distribution_name>`
 - Where *<distribution_name>* can be of the format below (dependent on the OS used and which version of eBUS SDK is installed, this list will vary, so only examples are provided):
 - Ubuntu-20.04-x86_64
 - Ubuntu-18.04-x86_64
 - Ubuntu-16.04-x86_64
 - For ARM Platforms:

- `sudo dpkg -r ebus_sdk_<distribution_name>`
 - linux-aarch-arm
- On RHEL/CentOS, as super user:
 - `rpm -e ebus_sdk_<distribution_name>`
 - rhel-centos-x86_64

✓ To see installed packages, you can execute one of the following commands:

- On Ubuntu:


```
dpkg -l | grep ebus
```
- On RHEL/CentOS:


```
rpm -qa | grep ebus
```

✓ To uninstall eBUS Runtime packages, for Release 6.4 and higher, execute the following commands:

```
sudo dpkg -r ebus-runtime
sudo rpm -e ebus-runtime
```

2.4 Choosing Optimal Settings for the Jetson Modules

We strongly recommend that you modify the Jetson AGX Xavier, Xavier NX, AGX Orin, Orin NX and Orin Nano configuration for power and performance management. If you do not perform these steps, you may encounter freezes or the eBUS SDK may stop responding.

✓ On Jetson Xavier you can select the unconstrained mode 0, MAXN, or less performing mode 3, MODE_30W_ALL

To choose the optimal nvpmode mode

✓ Before changing mode, it is recommended to run the following command and take note of the current mode:

```
nvpmode -q
```

- Execute the following command:
(**Note:** The mode is preserved on reboot.)
`sudo nvpmode -m 0 # Xavier`

To list all modes available on your Jetson module

- `nvpmodel -p --verbose`

To see the current clock settings

- Execute the following command:
`sudo jetson_clocks --show`

To increase the CPU clock to the maximum value

⚠ The increased CPU clock speed is not preserved on reboot.

1. Back up the default value to a file by executing the following command:
`sudo jetson_clocks.sh --store <filename_to_store>`
2. Increase the CPU clock speed by executing the following command:
`sudo jetson_clock`

To enable jumbo frames

- The use of jumbo frames reduces the amount of Ethernet, IP, UDP, and GVSP packet overhead when transmitting images, which reduces the CPU load and memory requirements. Using the script that is detailed in [“To enable jumbo packets”](#), set jumbo frames to **9000**.

3 eBUS SDK Licenses

Some components of the eBUS SDK require a Pleora license to remove transmit and receive limitations.

For more information on licenses refer to "eBUS SDK 7.0 (and later) Licensing: Document onPleora Support Center at <https://supportcenter.pleora.com>¹¹.

3.1 Activating a File-Based License

This section explains how to activate a standalone license file on your computer or embedded computer. If you purchased a pre-programmed USB dongle, see "Using a Pre-Programmed USB Dongle".

Please take note of the following important points:

- DO NOT rename the license file provided by your Pleora representative.
- DO NOT disable or remove the NIC (or WiFi adapter) that is associated with the license (Windows and Linux).
- Ensure the Pleora eBUS Player Toolkit or eBUS SDK is installed on your computer or embedded computer.

Activating eBUS File Based Licenses on Linux (x86 / ARM)

1. Copy the license file to:
`/opt/pleora/ebus_sdk/<Distribution_Targeted_by_eBUS_Installer>/licenses`
2. For USB3 Vision devices that use third-party non-Pleora transmitter technology: You must add the device's vendor ID to the eBUS SDK.
Tip: If you are not sure if your device uses Pleora transmitter technology, observe the USB GUID that appears on the device's label or in your software application. If it begins with the Pleora vendor ID (28b7), it uses Pleora transmitter technology.
To add the vendor ID to the eBUS SDK:
`cd /opt/pleora/ebus_sdk/<Distribution_Targeted_by_eBUS_Installer>/bin sudo ./set_udev_rules.sh.`
3. Close and reopen any applications that use the eBUS SDK, such as eBUS Player.

3.2 Activating a USB License Dongle

Connect the pre-programmed USB dongle to the workstation. When you use eBUS Player, a Pleora sample application, or an application created with the eBUS SDK, the license is detected and the restrictions are removed.

The pre-programmed USB dongle is compatible with:

¹¹. <https://supportcenter.pleora.com/>

- On the Windows operating system, Version 0 (or later) of the eBUS Player and Pleora sample applications, and applications created with Version 5.0 (or later) of the eBUS SDK.
- On the Linux operating system for x86 platforms, Version 2 (or later) of the eBUS Player and Pleora sample applications, and applications created with Version 6.2 (or later) of the eBUS SDK.
- Pleora eBUS Receive licenses (not eBUS Edge or eBUS SDK Seat licenses).
- The Windows operating system and the Linux operating systems for x86

The USB dongle must remain connected to the workstation for the license to be detected. The Pleora applications periodically check that the USB dongle is connected.

If you are using a Linux ARM operating system, you should purchase the standalone license file from Pleora.

3.3 Activating a Cryptlex based Evaluation License

All eBUS package installs the eBUS Licensing Tool, which is used to activate and manage Cryptlex-based licenses. The tool is available as both a GUI application and a command-line interface (CLI), except on Linux ARM platforms, where only the CLI is provided.

3.3.1 Accessing the eBUS Licensing Tool



On platforms where the GUI is available, the Licensing Tool can also be launched from **eBUS Player** → **File** → **License**, if eBUS Player is installed.

Linux (x86_64)

- **GUI:**
The GUI can be launched from the system's application launcher or by running: eBUSLicensingTool
- **CLI:**
Accessible through terminal: eBUSLicensingTool

Linux (ARM)

- **CLI Only:**
Accessible through terminal: eBUSLicensingTool

3.3.2 Activating a Cryptlex license Using the GUI

1. Open the eBUS Licensing Tool.
2. After obtaining your license key, enter it into the Enter License Key field.
3. Click Activate.
4. When activation is successful, the application will confirm that your license is active.

3.3.3 Activating a Cryptlex license Using the Command-Line Interface (CLI)

You can manage Cryptlex-based activation and deactivation through the eBUSLicensingTool CLI. Run the following command to view available options:

```
eBUSLicensingTool --help
```

All available commands are shown in the reference block below:

```
Online Activation:
<eBUSLicensingTool> activate --key=<product_key>

Offline Activation (Step 1: Generate challenge file):
<eBUSLicensingTool> activate --key=<product_key> offline --save-
path=<challenge_file_save_location>

Offline Activation (Step 2: Apply response file):
<eBUSLicensingTool> activate --key=<product_key> offline --response-
path=<path_to_response.dat_file>

Online Deactivation:
'<eBUSLicensingTool> deactivate'

Offline Deactivation (Generate file):
'<eBUSLicensingTool> deactivate offline --save-path=<challenge_file_save_location>'
```

4 Optimizing Operation with GigE Vision Devices

This chapter provides some steps you can take to optimize operation when using GigE Vision devices with the eBUS SDK.

4.1 Using the eBUS Universal Pro for Ethernet Filter Driver

The eBUS Universal Pro for Ethernet Filter driver is automatically installed and loaded on your workstation or embedded computer during the installation of the eBUS SDK. This driver optimizes operation with GigE Vision devices. It also enhances the performance of your system by allowing GigE Vision Stream Protocol (GVSP) data to bypass some (or all) of the operating system's network stack, delivering the data directly to the eBUS SDK.

When installing the eBUS SDK, if the eBUS Universal Pro for Ethernet Filter Driver fails to compile and install the eBUS Universal Pro driver due to permissions or missing dependencies, you can manually compile and install the driver.

4.1.1 Rebuild driver

To manually build the eBUS Universal Pro for Ethernet Filter Driver

1. Navigate to the following directory:
`/opt/pleora/ebus_sdk/<distribution_targeted>/module`
2. Execute the following command:
`sudo ./build.sh --kernel=/usr/src/<linux_headers source>`
 - Where `<linux_headers>` is the corresponding headers source for the host system.
Examples for Jetson Linux ARM platform and example for any x86_64 Ubuntu Desktop OS:

If you are using JetPack 5.1.4, navigate to the following directory:

`/opt/pleora/ebus_sdk/linux-aarch64-arm/module/`

Execute the following command:

```
sudo ./build.sh --kernel=/usr/src/linux-headers-5.10.216-tegra-ubuntu20.04_aarch64/
kernel-5.10
```

If you are using JetPack 6.2, navigate to the following directory:

`/opt/pleora/ebus_sdk/linux-aarch64-arm/module/`

Execute the following command:

```
sudo ./build.sh --kernel=/usr/src/linux-headers-5.15.148-tegra-ubuntu22.04_aarch64/3rdparty/canonical/linux-jammy/kernel-source
```

If you are using Ubuntu 22.04 LTS x64, navigate to the following directory:

`/opt/pleora/ebus_sdk/Ubuntu-22.04-x86_64/module/`

Execute the following command:

```
sudo ./build.sh --kernel=/usr/src/linux-headers-$(uname -r)
```

Error message: Cannot find the files to build kernel module in this PC

This error, which can occur when you are installing the eBUS SDK or eBUS Universal Pro for Ethernet Filter Driver, indicates that the kernel headers are not present on your system. The kernel headers are required to compile one of the Linux modules and they must match the system kernel version.

To install the Linux kernel headers, execute one of the following commands:

- On Ubuntu based:
sudo apt-get install linux-headers-\$(uname -r)
- On RHEL/CentOS, as super user:
yum install kernel-devel

After, perform the above steps 1 and 2 to manually build the eBUS Universal Pro For Ethernet driver

After that the eBUS Universal Pro For Ethernet driver is built, you can install eBUS driver on your host system.

4.1.2 Manual installation / uninstallation and check status**To manually install the eBUS Universal Pro for Ethernet Filter Driver**

- Execute the following script to install manually the eBUS Universal Pro driver:
sudo /opt/pleora/ebus_sdk/<distribution_targeted>/module/install_driver.sh --install

If you would like to uninstall the filter driver, you can do so. Keep in mind that you can still work with GigE Vision devices after you uninstall the eBUS Universal Pro for Ethernet Filter driver. However, the operation of these devices is not optimized.

To uninstall the eBUS Universal Pro for Ethernet Filter Driver

- Execute the following script and follow the prompts:
sudo /opt/pleora/ebus_sdk/<distribution_targeted>/module/install_driver.sh --uninstall

To start, stop, or check the status of the eBUS Universal Pro for Ethernet Filter Driver

- Execute the following command:
sudo /opt/pleora/ebus_sdk/<distribution_targeted>/module/ebdriverlauncher.sh <command> (where <command> can be start, stop, or status)

 You can also check the status of the driver by executing the following command:
lsmod | grep ebUniversalProForEthernet

4.2 Enabling Jumbo Ethernet Frames

If supported by your network card, GigE Vision device, and switch (if applicable), we recommend that you enable jumbo Ethernet frames when using GigE Vision devices. The use of jumbo frames reduces the amount of Ethernet, IP, UDP, and GVSP packet overhead when transmitting images, which reduces the CPU load and memory requirements.

To enable jumbo packets

- Execute the following command:
`sudo ifconfig <network_interface_ID> mtu [SIZE]`
where, `[network_interface_ID]` is the name of the network interface card (NIC) `[SIZE]` = desired frame size
For example:
`sudo ifconfig eth0 mtu 9000`

💡 To see the name of the NIC in your system, execute the following command:

```
sudo lshw -c network -businfo
```

To see the current maximum transmission unit (MTU) value, execute the following command:

```
ifconfig | grep MTU
```

4.3 Additional optimization steps

For GigE Vision devices, when a driver is installed, further optimization can be achieved by running the `set_socket_buffer_size.sh` script located in `/opt/pleora/ebus_sdk/ <distribution_targeted>/bin`, as `sudo` with the following command:

```
sudo ./set_socket_buffer_size.sh.
```

For further information, see the *eBUS SDK C++ API Quick Start Guide* or the *eBUS SDK Python API Quick Start Guide* for `PvStreamGEV::SetUserModeSocketRxBufferSize` details for Linux virtual NIC support.

5 Providing Access to Third-Party USB3 Vision Transmitter Devices

To access third-party, non-Pleora USB3 Vision transmitter devices, you must add the device's vendor ID to the eBUS SDK.

💡 If you are not sure if your device is a Pleora transmitter, observe the USB GUID that appears on the device's label or in your software application. If it begins with the Pleora vendor ID (28b7), it uses Pleora's transmitter technology.

To set up access for USB3 Vision devices that are not enabled with Pleora's technology

1. Execute the following command (this command works on any Linux platform):
`./opt/pleora/ebus_sdk/<distribution_targeted>/bin/set_udev_rules.sh`
2. When prompted, enter the vendor ID assigned to the device's This number often appears on the device's label.

6 Using eBUS Player to Configure Devices and Stream Images

You can use eBUS Player (which is precompiled and installed with the release in the bin folder) to connect to, configure, and stream images from GigE Vision and USB3 Vision devices.

You must disable the firewall before using GigE Vision devices. Some optimization may be desirable before using GigE Vision or USB3 Vision devices with the eBUS SDK in a production environment, as was discussed in [“Optimizing Operation with GigE Vision Devices”](#).

✔ Qt is required to run eBUS Player. Relevant Qt version must be installed for your particular OS prior to launching/running eBUS Player.

✔ When starting eBUS Player on Ubuntu 22.04, Ubuntu 24.04, CentOS Stream 9 or RedHat platforms, the following warning or similar message is displayed in the terminal:

Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland anyway.

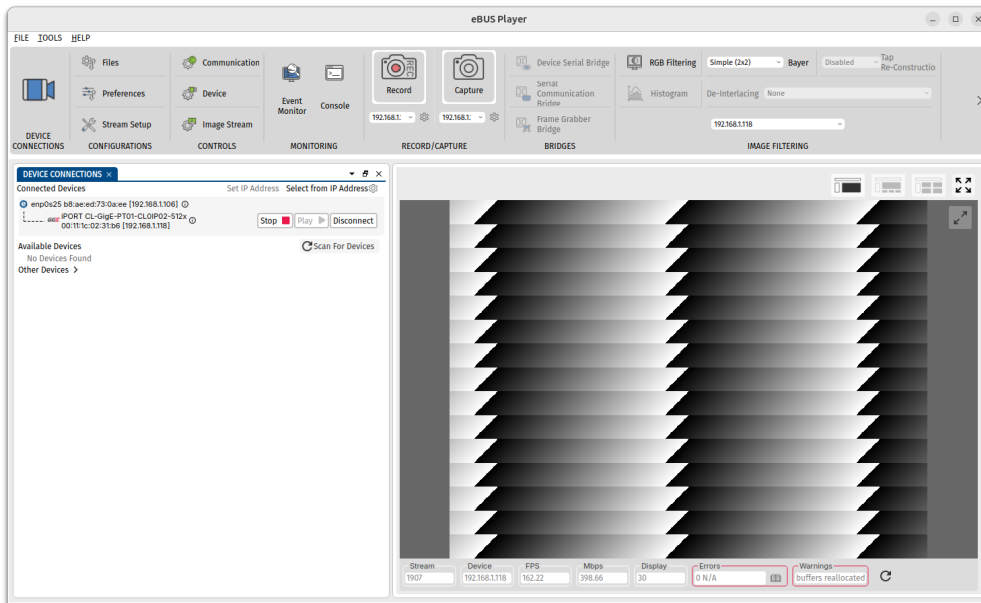
Workaround for **Ubuntu 22.04** and **Ubuntu 24.04**, at the login screen, before entering your password, locate the cog wheel in the lower right corner of the screen, and click on it. Select **Ubuntu on xorg** and proceed with entering your password.

Workaround for **CentOS Stream 9** and **RedHat**, at the login screen, before entering your password, locate the cog wheel in the lower right corner of the screen, and select a session with **x11 display server** and proceed with entering your password.

✔ In the `/opt/pleora/<distribution_targeted>/ebus_sdk/bin` folder, you will find both an eBUSPlayer script and an eBUSPlayer.bin executable. We strongly recommend executing the eBUSPlayer script, which sets the proper environment variables, and then automatically starts the eBUSPlayer.bin executable. The eBUS environment variables take permanent effect only after the next system reboot or after logging out and logging back.

With a terminal from `/opt/pleora/<distribution_targeted>/ebus_sdk/bin` directory, run this command to start eBUS Player:

```
./eBUSPlayer
```

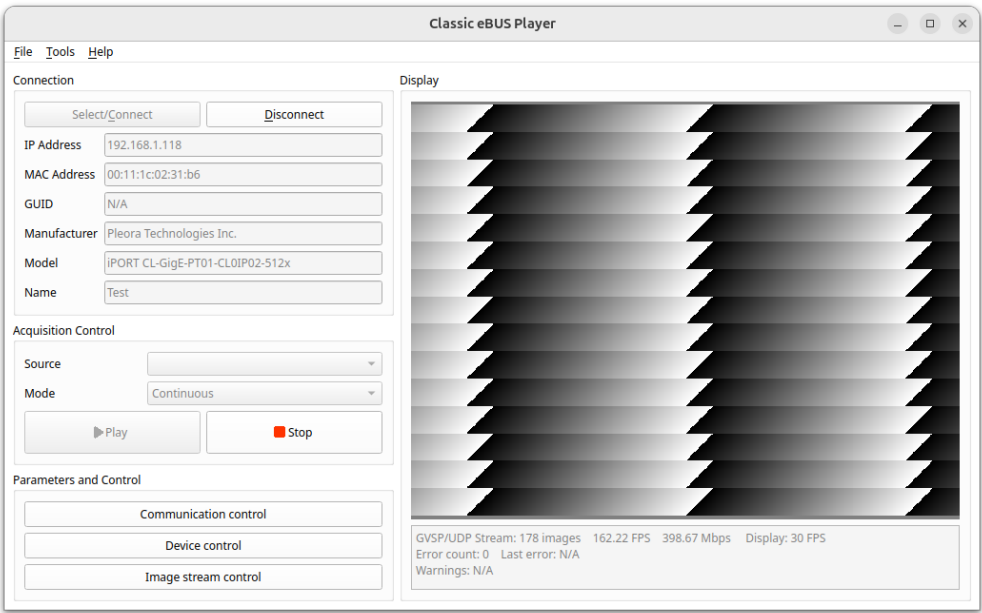


i For Linux-ARM based platforms, eBUS SDK doesn't install eBUS Player, but only Classic eBUS Player, because these Linux-ARM platforms don't support Qt6 which is required for the new eBUS Player.

i Classic eBUS Player is available for all supported OS (Linux, Windows) by eBUS SDK.

✓ In the `/opt/pleora/<distribution_targeted>/ebus_sdk/bin` folder, you will find both an `eBUSPlayerClassic` script and an `eBUSPlayerClassic.bin` executable. We strongly recommend executing the `eBUSPlayerClassic` script, which sets the proper environment variables, and then automatically starts the `eBUSPlayerClassic.bin` executable. The eBUS environment variables take permanent effect only after the next system reboot or after logging out and logging back. With a terminal from `/opt/pleora/<distribution_targeted>/ebus_sdk/bin` directory, run this command to start Classic eBUS Player:

```
./eBUSPlayerClassic
```



7 Compiling and Running Sample Applications

We strongly recommend that you copy the sample applications from the `/opt/pleora/ebus_sdk/<distribution_targeted>/share/` directory to your personal development workspace before modifying or compiling them.

A list of C++ samples, with a description of each, can be found in the `index.html` file in the `/opt/pleora/ebus_sdk/<distribution_targeted>/share/samples` directory.

✔ Compilation will fail for GUI-based samples if the Qt development package is not installed. For Qt version information, see ["System Requirements"](#). Qt5 is required for linux-arm platforms. For x86_64 desktop platforms, Qt6 is included and installed with eBUS SDK 7.x.

A list of DotNet samples can be found in the `/opt/pleora/ebus_sdk/<distribution_targeted>/share/samples/dotnet_samples` directory.

A list of Python samples can be found in the `/opt/pleora/ebus_sdk/<distribution_targeted>/share/samples/python_samples` directory.

For information about building the sample applications and creating your own applications, see the *eBUS SDK C++ API Quick Start Guide*, *eBUS SDK DotNet API Quick Start Guide* and the *eBUS SDK Python API Quick Start Guide*.

To compile the eBUS SDK C++ sample applications

1. If you have not restarted your computer since installing the eBUS SDK, we recommend that you do so. This ensures that the correct environment variables are set at startup.
2. Make a copy of the following directory, as a backup for the original source files:
`/opt/pleora/ebus_sdk/<distribution_targeted>/share/samples`
3. Navigate to the directory that contains the copy of the sample code (that was created in Step 2 above).
4. Do one of the following:
 - **To compile all sample applications at one time:** Execute the sh script by typing

```
./build.sh
```

- **To compile a specific sample application:** Navigate to the directory of the sample (for example, `MulticastMaster`) and execute the `make` command.

```
make
```

To compile the eBUS SDK DotNet sample applications

1. If you have not restarted your computer since installing the eBUS SDK, we recommend that you do so. This ensures that the correct environment variables are set at startup.
2. Make a copy of the following directory, as a backup for the original source files:
`/opt/pleora/ebus_sdk/<distribution_targeted>/share/samples/dotnet_samples`
3. Navigate to the directory that contains the copy of the sample code (that was created in Step 2 above).
4. Do one of the following:
 - **To compile all sample applications at one time:** Execute the sh script by typing
`./build.sh`
 - **To compile and launch a specific sample application:** Navigate to the directory of the sample (for example, `PvStreamSample`) and execute the `dotnet run --configuration Release` command.

To use the eBUS SDK Python sample applications

1. If you have not restarted your computer since installing the eBUS SDK, we recommend that you do so. This ensures that the correct environment variables are set at startup.
2. Make a copy of the following directory, as a backup for the original source files:
`/opt/pleora/ebus_sdk/<distribution_targeted>/share/samples/python/ebus`
3. Navigate to the directory that contains the copy of the sample code (that was created in Step 2).
4. Launch the Python application by executing the following command:
 - **python3 ./<python_sample>.py.**

✔ If you encounter issues with running the samples due to the environment variables not being set, you can restart your computer or set the environment variables manually, as outlined in [“The sample applications compiled successfully, but they will not run.”](#).

8 Troubleshooting

Cannot detect or connect to GigE Vision devices.

In the **Device Selection** dialog box, select the **Show unreachable Network Devices** check box. If the device still does not appear, do one of the following:

- Disable the firewall.
- And/or -
- Execute the following script and then choose option **0** (to disable strict Reverse Path Forwarding filtering):

```
sudo /opt/pleora/ebus_sdk/<distribution_targeted>/bin/set_rp_filter.sh
```

Cannot detect or connect to USB3 Vision devices.

If the device does not use Pleora's transmitter technology and the device's vendor ID has not been added to the eBUS SDK, you will be unable to detect or connect to it. To see if your device uses Pleora transmitter technology, observe the USB GUID that appears on the device's label or in your software application. If it begins with the Pleora vendor ID (28b7), it uses Pleora transmitter technology. For information about accessing the camera, see [?Providing Access to Third-Party USB3 Vision Transmitter Devices?](#).

A high number of GTK errors or warnings appear when running an application.

It is likely that you are running the sample application as superuser but are logged on as a standard user.

The sample applications compiled successfully, but they will not run.

As superuser, execute the `/opt/pleora/ebus_sdk/<distribution_targeted>/bin/install_libraries.sh` script to ensure that the required libraries have been added to the system. Also, ensure that you source the `bin/set_puregev_env.sh` script to set the environment variables (`source set_puregev_env.sh`).

💡 You can also launch the sample applications using the `RunFromEnv.sh` script. For example: `source ./RunFromEnv.sh && ~/samples/PvStreamSample/PvStreamSample`

✅ In this example, it is assumed that a copy of the sample applications is available in your Home directory and that you have "write" access.

The following error appears: `GENICAM_ROOT_V3_4 is not set`.

Restart the computer. This issue can occur if you did not restart the computer after installing the eBUS SDK. If the issue persists, follow the steps to set the environment variables, as outlined above in ["The sample applications compiled successfully, but they will not run."](#)

A watermark appears on received images.

You require a receive license. For more information, see ["Activating eBUS SDK Licenses"](#) or the *eBUS SDK Licensing Overview Knowledge Base Article*, available on the Pleora Support Center (supportcenter.pleora.com).

Error message: Cannot find the files to build kernel module in this PC

This error, which can occur when you are installing the eBUS SDK or eBUS Universal Pro for Ethernet Filter Driver, indicates that the kernel headers are not present on your system. The kernel headers are required to compile one of the Linux modules and they must match the system kernel version.

To install the Linux kernel headers, execute one of the following commands:

- On Ubuntu:
sudo apt-get install linux-headers-\$(uname -r)
- On RHEL/CentOS, as super user:
yum install kernel-devel

The eBUS SDK applications freeze or stop responding on the Jetson module

Ensure that you have chosen the nvpmodel mode, increased the CPU clock the maximum value, and configured jumbo packets for 9000 bytes. For more information, see [“Choosing Optimal Settings for the Jetson Modules”](#).

Technical Support

On the Pleora Support Center, you can:

- Download the latest software and firmware.
- Log a support issue.
- View documentation for current and past releases.
- Browse for solutions to problems other customers have encountered.
- Read knowledge base articles for information about common tasks.

To visit the Pleora Support Center:

- Go to supportcenter.pleora.com.
- Most material is available without logging in to a Support Center account.
- To access software and firmware downloads, in addition to other content, log in to the Support Center.
- If you do not have an account, click Request Account.
- Accounts are usually validated within one business day.

Copyright Information

Copyright © 2026 Pleora Technologies Inc.

These products are not intended for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Pleora Technologies Inc. (Pleora) customers using or selling these products for use in such applications do so at their own risk and agree to indemnify Pleora for any damages resulting from such improper use or sale.

Trademarks

VIVOEPlayer, PureGEV, eBUS, iPORT, vDisplay, AutoGEV, AutoGen, AI Gateway, eBUS Studio, Vaira and all product logos are trademarks of Pleora Technologies. Third party copyrights and trademarks are the property of their respective owners.

Notice of Rights

All information provided in this manual is believed to be accurate and reliable. No responsibility is assumed by Pleora for its use. Pleora reserves the right to make changes to this information without notice. Redistribution of this manual in whole or in part, by any means, is prohibited without obtaining prior permission from Pleora.

Document ID: QSG-0023