Basler Vision Connector Documentation

Document number: AW001850
Version: 01 Language: 000 (English)
Release date: 26 January 2024

BASLER
the power of sight

Contacting Basler Support Worldwide

Europe, Middle East, Africa

Basler AG
An der Strusbek 60–62
22926 Ahrensburg
Germany
siemens-support@baslerweb.com

The Americas

Basler, Inc.
855 Springdale Drive, Suite 203
Exton, PA 19341
USA
siemens-support@baslerweb.com

www.baslerweb.com

Table of Contents

Introduction 4

Required Knowledge 4

Additional Information 5

Prerequisites 5

Getting Started 7

Installing the Basler Vision Connector on Local IE Management System 7

Enabling the Layer 2 Network 9

Installing the Basler Vision Connector on Local IE Device 11

Configuring the Camera's IP Address 16

Using the *.test.json Files 17

Next Steps 18

Messaging 19

Topic Structure 19

Device Identification 20

Image Format 20

Message Envelope 20

Metadata JSON 21

Image Detail Object 22

Metadata Example 23

Camera Operation 24

Discovering the Camera 24

Opening the Camera Connection 26

Closing the Camera Connection 29

Configuring the Camera Parameters 30

Retrieving Parameter List 30

Setting Camera Parameters 35

Setting Camera Quick Parameters 38

Defining Camera User Settings 41
Downloading the Camera Configuration 49
Uploading the Camera Configuration 50
Image Streaming 52
Starting Image Streaming 52
Stopping Image Streaming 54
Software Triggered Image Acquisition 56
Getting the Application Status 57
Message Envelope for ZMQ Message 58
JSON Payload 58
Response 58
Response Object 58
Camera Object 59
Application Feedback 60
Message Codes 60
Application Logging 61

Introduction

This documentation is intended to help you install the Basler Vision Connector from the Siemens Industrial Edge Hub to your local Industrial Edge Management and to your in-factory industrial edge devices. In addition this manual shows you how to configure and operate your Basler cameras via the Basler Vision Connector.

Required Knowledge

The Basler Vision Connector and Basler digital cameras for machine vision are designed for use by professionals only, concerned with the following tasks:

For installing and configuring your Industrial Edge devices and the Basler Vision Connector app:

- Knowledge about MQTT / ZeroMQ protocol communication
- Knowledge working with Docker and Docker container is helpful.

For installing the Basler camera or cameras and accessories:

- Camera integration and setup
- Everyday camera use

The Basler Vision Connector documentation assumes the skills of electrical engineers, systems engineers, mechatronics engineers, technicians, and information technology engineers.
Vision Connector and camera use isn't allowed for persons with insufficient skills or with impairments that prevent them from doing the following tasks:

- Using cameras correctly
- Adhering to the safety instructions
- Fully understanding all relevant camera documentation and other linked documentation

Additional Information

- For camera and accessory information, see the Basler Product Documentation
- For detailed information about Siemens IE devices and management, see the Siemens Industrial Edge Documentation
- The **ReleaseNotes.txt** file is delivered with the Basler Vision Connector.
  The **Terms and Conditions.pdf** file (pylon End User Agreement and Third-Party License Information) can be found in the Siemens Industrial Hub (**Application Provisioning** menu item).

Prerequisites

Make sure you have the following components available and installed:

- Working Internet connection
- Installed and working Siemens devices for your use case, i.e. devices for processing and managing tasks (e.g. Industrial Edge - SIMATIC IPC847E). For detailed information, see the Siemens documentation.
- Access to the Siemens Industrial Edge Hub (provided by Siemens).
  Contact Siemens to get your personal access to the hub. Access link to the Siemens Hub.
- Installed and configured Siemens IEM Instances and all IE devices you require for your use case. For information about installing the IEM instances and IE devices, see the Siemens documentation.
- Installed Basler camera
  For further information about Basler camera installation and configuration:
  Go to the Basler Product Documentation, select the camera model you installed in your factory via the camera filter (located to the left of the **Search** field).
  When the camera model is selected, the camera and its feature documentation are displayed. Read the safety instructions of your camera model.

- The following, currently supported, Basler Vision Connector applications are available in the Siemens marketplace:
  - Basler Vision Connector (Single Camera Connection).
    Use this app, if you want to use only one camera.
  - Basler Vision Connector (Multi Camera Connection).
    Use this app, if you want to use several cameras.

Info

You can configure different Basler camera models (GigE, USB) with the Basler Vision Connector app. However, at the moment the Basler Vision Connector app officially supports only GigE cameras as the Siemens Industrial Edge devices can currently only deal with GigE cameras, e.g. Basler ace 2.

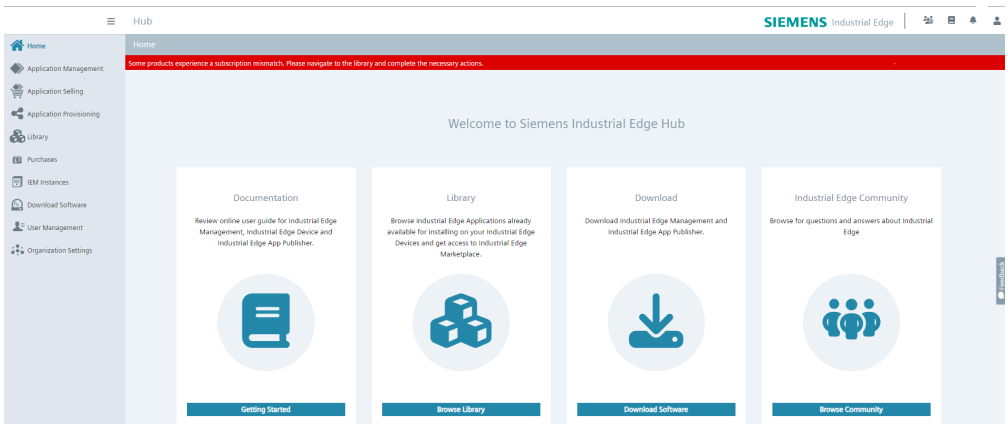
- Programming and test tools
  You can use the Siemens Industrial Edge Flow Creator application (IE Flow Creator).
  For information, see the Siemens documentation for IE Flow Creator.
  The IE Flow Creator uses the flow-based Node-RED development tool. For information about Node-RED, see the Node-RED website.
- Siemens Databus application for testing
  For information, see the Siemens Database documentation.
- Siemens Data Flow Monitoring application: For information, see the Data Flow Monitoring documentation.
- Siemens Databus application for testing: For information, see the Databus documentation.

Getting Started

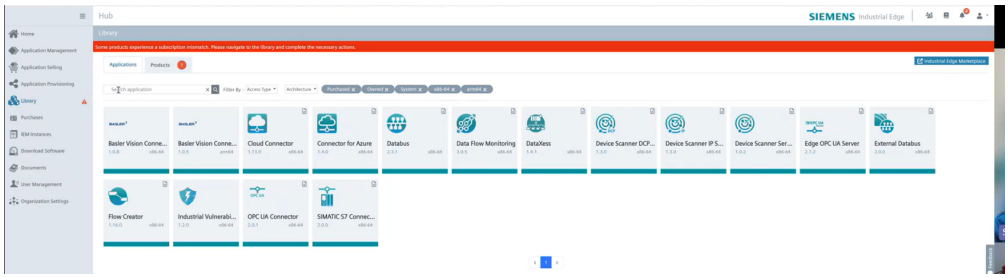
Installing the Basler Vision Connector on Local IE Management System

To install the Basler Vision Connector on the local IE Management system:

1. Open the Siemens Hub page and the Siemens Marketplace website.
2. In the Siemens Industrial Management Marketplace:
   Buy the Basler Vision Connector app type you require (Multi camera connection or Single camera connection: see information under Prerequisites).
   You get the access code for the purchased Basler Vision Connector app from your Siemens contact person.
3. Log in to the Siemens Industrial Management Hub with your access data (provided by Siemens).



4. In the Industrial Edge Management Hub:
   In the **Library** menu, click the **Applications** tab.
   A window with all available/purchased apps appears.



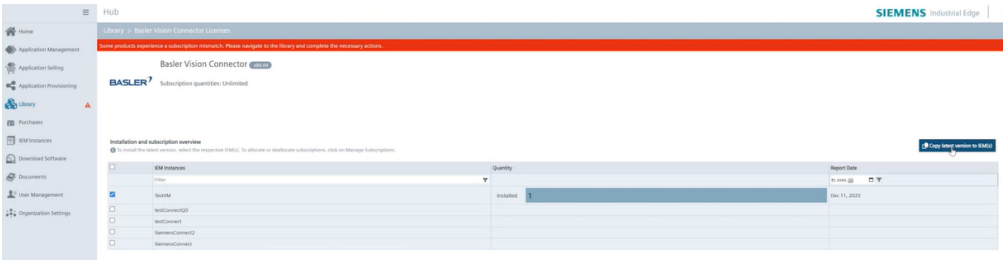
Depending on your use case, select the corresponding Basler Vision Connector app:

- If you want to use only one camera:
  Select the Basler Vision Connector (Single Camera Connection) app.
- If you want to use more than one camera:
  Select the Basler Vision Connector (Multi Camera Connection) app.

It's possible to use the Multi Camera Connection version of the Basler Vision Connector with a single camera. But if you use the Single Camera Connection version with more than one camera, an error message is displayed indicating that this isn't possible.

In the following sections the generic Basler Vision Connector term is used whenever it doesn't matter how many cameras are used. If it's important to differentiate, the specific Basler Vision Connector name is indicated, i.e. Basler Vision Connector (Multi Camera Connection) or Basler Vision Connector (Single Camera Connection).

5. Select the Basler Vision Connector app version you want to install by clicking the app name.
   The **Installation and subscription overview** list appears.



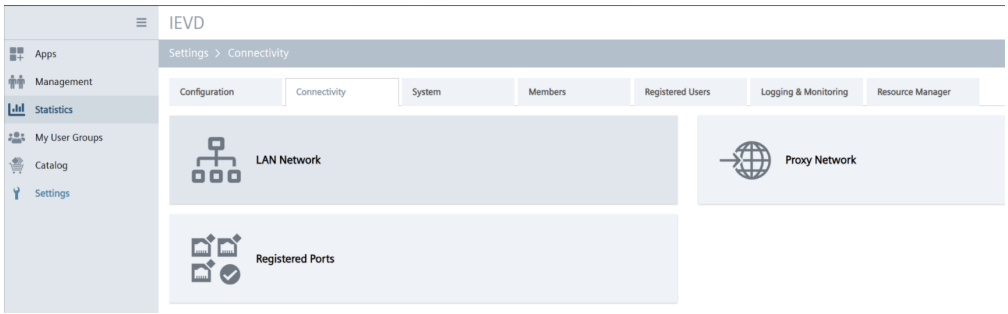
   The **Installation and subscription overview** list displays the installed IE Management instances.

6. Select the local repository (IEM) where you want to copy the Basler Vision Connector app to: Activate the corresponding check box.

7. Click **Copy latest version to IEM(s)**.
   The Basler Vision Connector app is now be copied to your local Industrial Edge Device management system.

Enabling the Layer 2 Network

1. On your Edge Device: Go to the **Settings** menu item.



2. Click the **Connectivity** tab > **LAN Network** > **Edit** (pencil symbol).

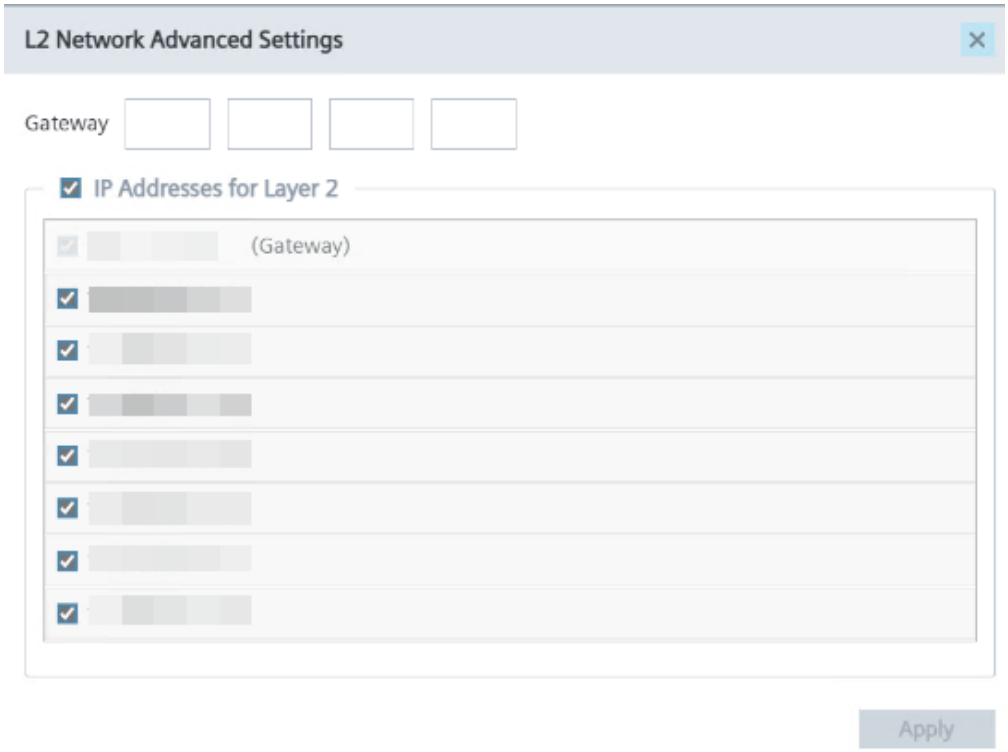
The **Edit Network Interface** dialog appears:

3. In the **Layer 2 (L2) for Apps (Optional)** group:
   Enter the number for the start address in the **Start IP Address** field.

4. Click **Update**.
   The Layer 2 network is now enabled.

5. For a later step, check the available IP addresses:
   Click **Advanced Settings** to see the available IP addresses.

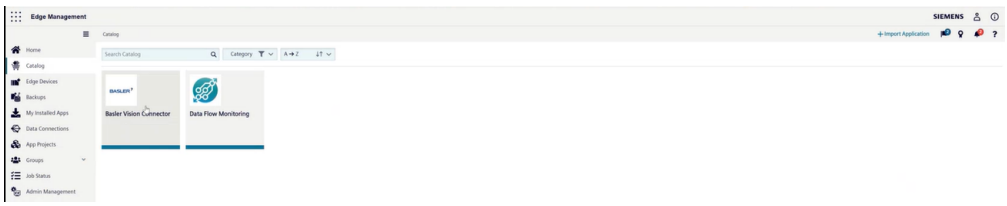
These numbers are important when configuring the IP address of the camera.
IP address numbers with green check boxes are available (see example screenshot below).



Installing the Basler Vision Connector on Local IE Device

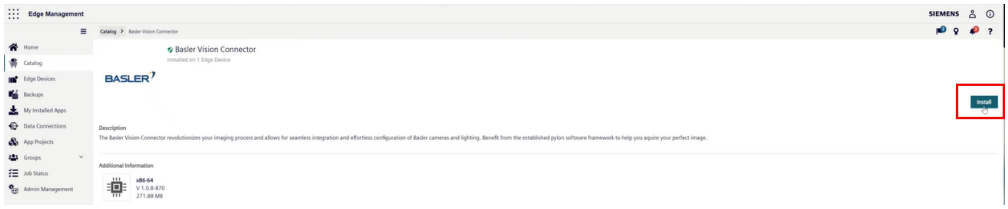
To install the Basler Vision Connector app to a local IE device in your factory:

1. Open the Siemens **Edge Management** platform.



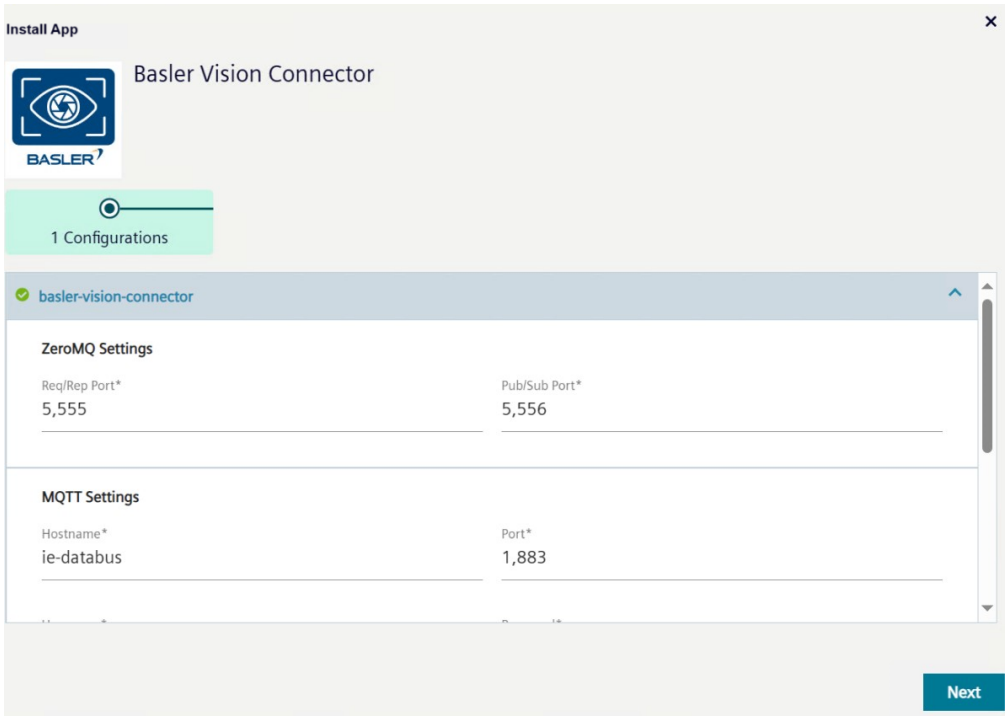
2. Make sure that your required Edge devices are online.
3. Go to the **Catalog** menu item. In this section the apps installed in the local edge management are displayed.

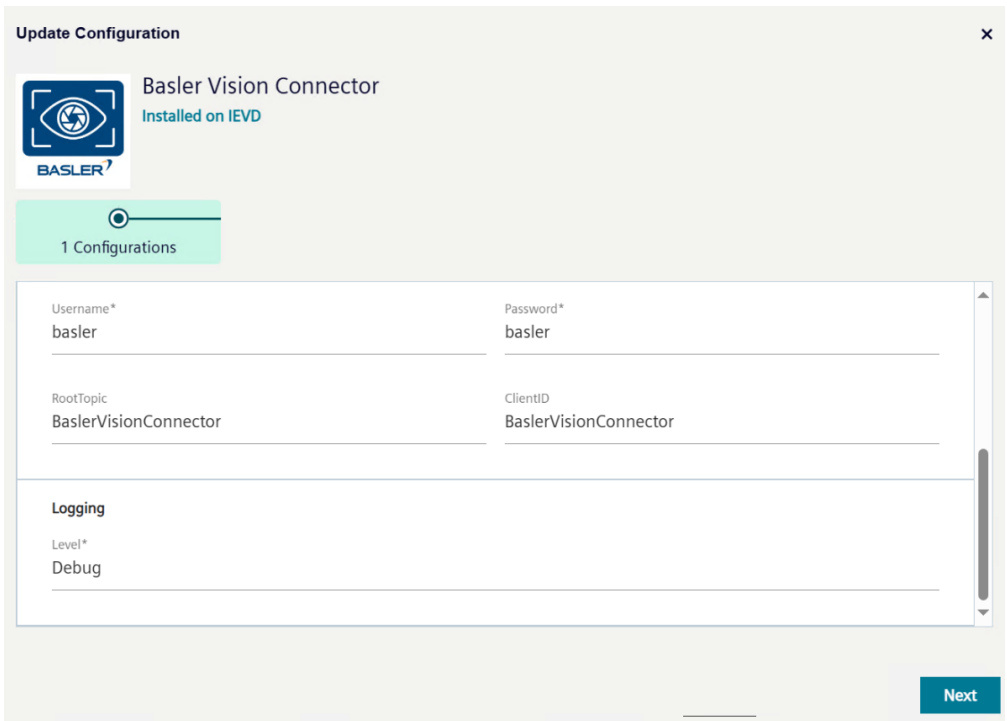
4. Click the required Basler Vision Connector app (if you installed different app types in your local Industrial Edge Management). See Prerequisites - Single/Multi camera connection. The **Catalog** window displays the selected Basler Vision Connector app.



5. Click **Install**.

   The **Install App (1 Configuration)** wizard appears.



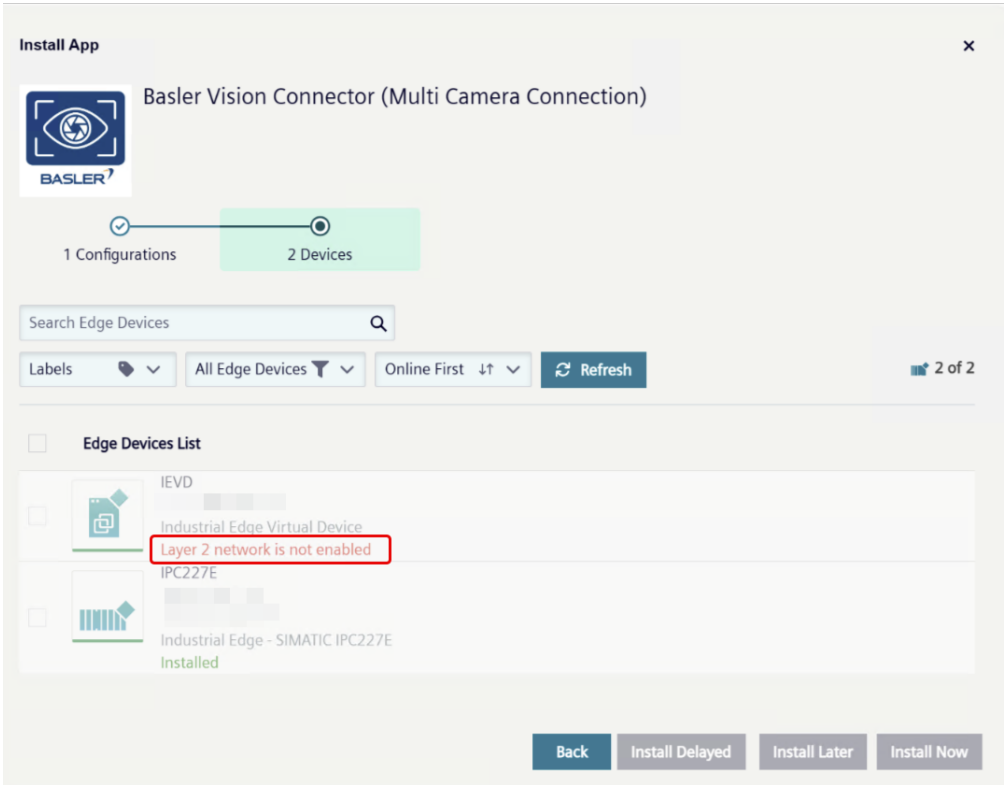


6. Check the entries displayed in the different fields:
   The default values are shown in the following table:

Default Values		
ZeroMQ Settings	Req/Rep Port: 5555	Pub/Sub Port: 5556
MQTT Settings	Hostname: ie-databus	Port: 1883
Username	basler	Password: basler
RootTopic	BaslerVisionConnector	Client ID: BaslerVisionConnector
Logging > Level	Debug (possible settings: Info > Warning > Error > Debug)	

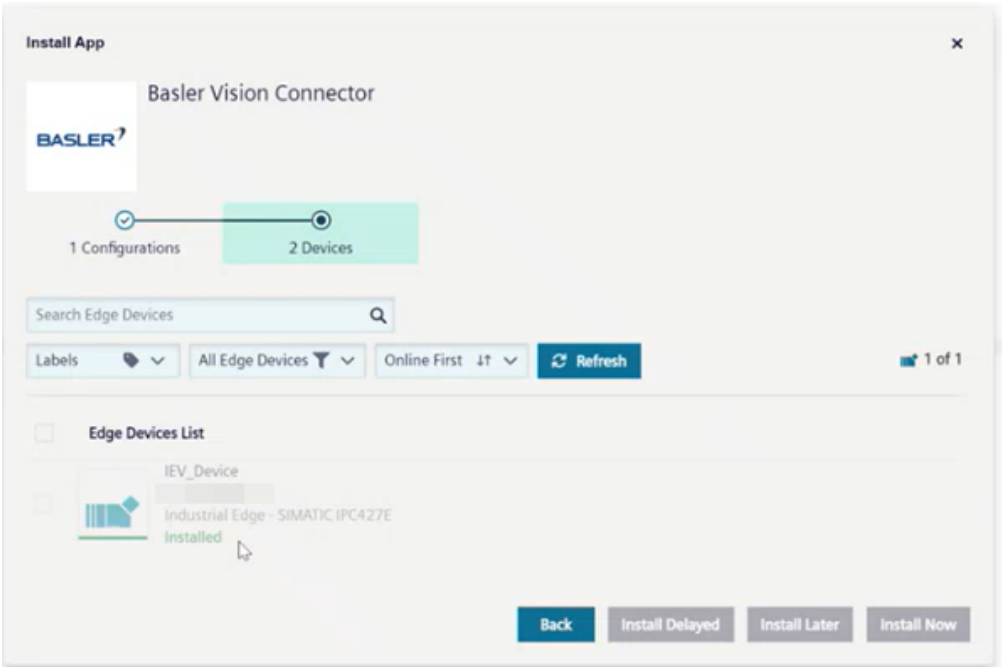
Layer 2 Network isn't Enabled Message

If the layer 2 network wasn't configured before (see Enabling the Layer 2 Network), an error message is displayed; see example screenshot:



If this message appears, do the following:

1. Click **Next**.
   The **Install App** > **2 Devices** wizard screen appears.



2. In the **Edge Devices Liste** section:
   Activate the checkbox of the local IE device(s) you want to install the Basler Vision Connector app onto.

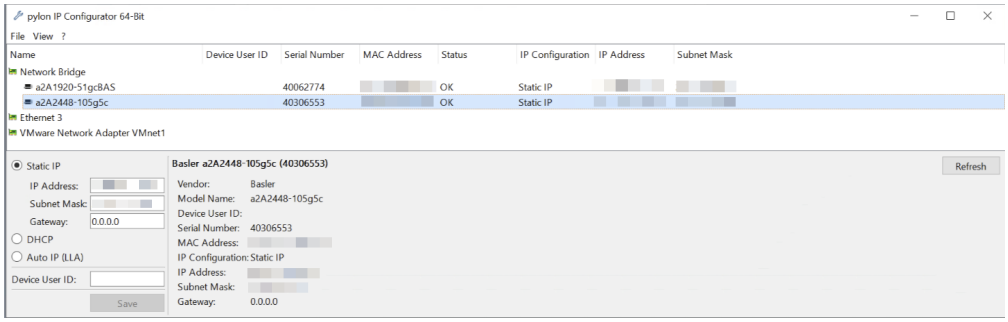
3. Click **Install Now**.
   The Basler Vision Connector app is now installed on your local Industrial Edge device or devices.
   In the **Edge Management**, under **Edge Devices** you can now see the apps installed on your local hardware device (e.g. Basler Vision Connector).

Configuring the Camera's IP Address

To configure the IP address of your camera:

1. Make sure the camera you want to configure is powered and online.

2. Make sure that the camera's IP address is in the range defined in the layer 2 network. Only then the camera is visible to the IE device and hence to the Basler Vision Connector app.
   You can use the pylon IP Configurator to check and if necessary to adapt the camera's IP address (see screenshot below).
   For information, see Assigning an IP Address to a Camera in the Basler Product Documentation.



Info

Observe the following:
The following camera parameter names are always taken directly from the camera firmware:
- `SetParameters`
- `GetParameters`
The `SetQuickParameters` changes all gain parameter names from the camera firmware to `Gain`.

Using the *.test.json Files

To use the ***.test.json** files:

1. For the next steps (Open camera, Setting parameters, etc.), download the following ***test.json** files to your computer:

   - mqtt_test.json
   - zmq_test.json

2. Make sure that the following apps are installed in addition to the Basler Vision Connector:

   For the MQTT test you need:
   - Industrial Edge Flow Creator app
   - Data Flow Monitoring app
   - Databus app

   For the ZMQ test you need:
   - Industrial Edge Flow Creator app
   - Data Flow Monitoring app

3. Import the ***.json** test files to the Siemene IE Flow Creator application.

4. Check the settings in Industrial Edge Flow Creator (see values below).
   If necessary, adapt accordingly.

Default Values		
ZeroMQ Settings	Req/Rep Port: 5555	Pub/Sub Port: 5556
MQTT Settings	Hostname: ie-databus	Port: 1883
Username	basler	Password: basler
RootTopic	BaslerVisionConnector	Client ID: BaslerVisionConnector
Logging > Level	Debug (possible settings: Info > Warning > Error > Debug)	

Next Steps

In the previous steps you've installed the Basler Vision Connector and configured the IP address of your camera. The next steps are:

- Setting up and configuring the accessories you use in your setup (e.g. lens type, illumination, etc.)
- Configuring the camera: For information about the camera features and how to use them, see the Feature documentation in the Basler Product Documentation
- Configuring the image processing
  For information about the different apps and possiblities, see the Siemens documentation "How to build your use case" (e.g. AI Interference Server that can be used for further image processing).

Messaging

The message exchange is based on JSON messages. The following sections outline the minimum content for the message payload.

For ZMQ, the JSON string payload is encoded in a UTF-8 single-frame message.

In MQTT messages, the JSON payload serves as the content of the message to be sent.

> **Info**
>
> By default, response messages are not sent for message code **0x00000000 - Request was completed successfully.**

Topic Structure

ZMQ

When using ZMQ connections, use the expected structure outlined below:

- To send commands, use single-frame messages sent to the REQ/REP port.
- The image streaming topic on the PUB/SUB port: **{DeviceID}/stream**
- For receiving application feedback, subscribe to topic on the PUB/SUB port: **logging**

MQTT

For MQTT connections, the topic structure is as follows:

- To send commands, publish messages to: **{mqttRoot}/request**
- For receiving application command responses on: **{mqttRoot}/response**
- Subscribe to the topic for receiving application feedback: **{mqttRoot}/logging**

The message content should be formatted in JSON, employing the message field names as property names for each call, as described in the subsequent sections of this document.

Device Identification

To identify a device, you can specify the serial number (**SerialNumber** field), user identification (**UserDefinedName** field), IP address (for GigE devices, in the **IPAddress** field), and MAC address (for GigE devices, in the **mac** field) in the Basler Vision Connector. While these fields are commonly used, they are not mandatory. See Discovering the Camera for details about camera discovery.

In the following sections, the term **DeviceID** refers to the identifying string that was used for a camera device when opening it. This exact string is used for identifying the opened camera device while it's used and until it's closed again. Only after that the camera device can be opened and used again with a different **DeviceID**. The value of the **DeviceID** can be one of the fields returned during the device info. See more details in section Opening the Camera Connection.

Using a different **DeviceID** for a camera device while it's open, even if it refers to the same device, may lead to errors. Exceptions to this rule are allowed, if documented by the application.

Image Format

To standardize image transfer between camera streaming applications and image-driven applications, use the following format for transferring images. The ZMQ message consists of a multipart message with each frame/part described below.

This message format is prepared to support multiple stream cameras.

Message Envelope

Frame	Field	Description	Type
1	topic	Device identification	Binary string (UTF-8) to identify the device in the following syntax: {DeviceID}/stream

Frame	Field	Description	Type
2	metadata	Metadata	Binary metadata JSON (UTF-8).
3	image	Image 1	Binary
4 … n	image	Image n - For multi-stream camera	Binary

Metadata JSON

Field	Optional	Type	Default	Description
version	No	String	1	Metadata format version.
count	No	Integer	1	Number of images on message.
timestamp	No	String	-	Date and time string UTF-8 ISO8601. Application host datetime.
detail	No	Image Detail Object Array	-	Images details.

Image Detail Object

Field	Optional	Type	Default	Description
id	No	String	-	Unique image identifier. Suggested mnemonic format like {camid}_{camstream} _{datetime}.
seq	No	Integer	-	Sequential number for the camera image. Starting at 1, incremented by 1 on each frame grabbed.
height	No / Yes (encoded formats)	Integer	-	Image height.
width	No / Yes (encoded formats)	Integer	-	Image width.
formatns	Yes	String	Genicam	Image format namespace. For GenICam pixel format use **Genicam** and **Compressed** for compressed formats.[1]
format	No	String	-	Image format. [1] [2]

Field	Optional	Type	Default	Description
linepadding	Yes	Integer	0	Number of bytes added on the end of byte stream to reach a multiple of 4 bytes. E.g. an image with 61 pixels, mono8, will have a **linepadding** of 3, filling the stream to reach 64 bytes.
timestamp	Yes	String[3]	-	Timestamp sent by the camera.[4]

[1] The Basler Vision Connector only supports GenICam, no compressed formats.

[2] For cameras using a GenICam standardized pixel format, it's recommended to use the GenICam naming convention as described in section 4.35 GenICam*PFNC*2_4.pdf.

[3] A string might be used since, in the PTP standard, the tick count is expected to be an 80-bit number. However, the current GenICam protocol returns a 64-bit integer.

[4] Timestamp provided by the cameras are ticks count. This can be referencing the Unix epoch or the number of ticks since the camera is on or count of ticks informed/in sync with the PTP server.

Metadata Example


```
{
  "version": "1",
  "count": 1,
  "timestamp": "2014-01-09T13:35:34.000000000+0100",
  "detail":[
    {
      "id":"mycamera_1_2014-01-09T13:35:34.000000000+0100",
      "seq": 3,
      "height": 852,
      "width": 1280,
```

```
        "formatns": "Genicam",
        "format": "BGR8",
        "linepadding": 0,
        "timestamp": "123456"
      }
    ]
}
```

Camera Operation

Discovering the Camera

As initial step to establish communication, the Basler Vision Connector must discover and list all available cameras.

Message Envelope for ZMQ Message

Frame	Field	Description	Type
1	payload	Method payload	Binary string (UTF-8)

JSON Payload

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
Action	No	String	Method called. Fixed value: *EnumerateDevices*

```
{
    "TransactionID": "123456",
    "Action": "EnumerateDevices"
}
```

Response

Frame	Field	Description	Type
1	payload	Response object	Binary string (UTF-8)

Response Object

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
ReturnCode	No	Integer	Message code following Message Codes.
Message	Yes	String	Response message.
DeviceList	No	Array of camera devices	Array of camera devices.

Camera Object

Field	Optional	Type	Description
SerialNumber[1]	Yes	string	Camera identification
UserDefinedName[1]	Yes	String	Camera user-defined name
ModelName	No	String	Camera model

Field	Optional	Type	Description
VendorName	No	String	Camera vendor
Interface	No	String	Camera interface *U3V* (USB3 Vision), *CXP* (CoaXPress), *GEV* (GigEVision), *CamEmu* (Emulation)
IPAddress	Yes	String	Camera IP address in IP v4 format
{additional identification fields}[1]	Yes	String	Additional device identification offered by the Basler Vision Connector.

[1] Used for device identification.

```
{
    "TransactionID": "123456",
    "ReturnCode": 0,
    "Message": "Cameras found.",
    "DeviceList": [
        {
            "SerialNumber": "548451887",
            "UserDefinedName": "MyCam1",
            "ModelName": "Camera Model 1234",
            "VendorName": "The camera Factory",
            "Interface": "GEV",
            "IPAddress": "192.168.0.1"
        }
    ]
}
```

Opening the Camera Connection

GenICam cameras demand the connection to be open before you can start to set parameters and image streaming. This connection is an exclusive access, meaning that no other App can access the camera and change parameters or stream images.

You can open a camera by specifying an identification string in the **DeviceID** field. The Basler Vision Connector then tries to find a matching camera by searching for this identification string amongst the infos of the available cameras. The order in which it searches is the following:

1. SerialNumber
2. IPAddress
3. UserDefinedName
4. ModelName
5. VendorName
6. Interface

Example:

An identification string is provided in the **DeviceID** field. Then the Basler Vision Connector first tries to find a camera with a matching serial number. If no matching serial number was found, the Basler Vision Connector tries to find a camera with a matching IP address. The Basler Vision Connector continues until a matching camera is found and tries to open this camera.

If several cameras are found matching the same info, then the Basler Vision Connector tries to open the first one (it's up to the Basler Vision Connector which camera is the first one).

If a camera was opened using a certain identification string, then this identification string must be used in all **DeviceID** fields of all requests regarding this camera until it's closed again. A camera can't be opened with multiple identification strings at the same time.

Message Envelope for ZMQ Message

Frame	Field	Description	Type
1	payload	Method payload	Binary string (UTF-8)

JSON Payload

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
Action	No	String	Method called. Fixed value: *OpenDevice*
DeviceID	No	String	Device identification.

```
{
    "TransactionID": "1232156",
    "Action": "OpenDevice",
    "DeviceID": "548451887"
}
```

Response

Frame	Field	Description	Type
1	payload	Response object	Binary string (UTF-8)

Response Object

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
ReturnCode	No	Integer	Message code following Message Codes.
Message	Yes	String	Response message.

```
{
    "TransactionID": "123456",
    "ReturnCode": 0,
    "Message": "Camera open."
}
```

Closing the Camera Connection

Use the following messages to close the camera connection. This releases the exclusive access and stops any running image stream, and releases the camera.

Message Envelope for ZMQ Message

Frame	Field	Description	Type
1	payload	Method payload	Binary string (UTF-8)

JSON Payload

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
Action	No	String	Method called. Fixed value: *CloseDevice*
DeviceID	No	String	Device identification.

```
{
    "TransactionID": "1232116",
    "Action": "CloseDevice",
    "DeviceID": "548451887"
}
```

Response

Frame	Field	Description	Type
1	payload	Response object	Binary string (UTF-8)

Response Object

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
ReturnCode	No	Integer	Message code following Message Codes.
Message	Yes	String	Response message.

```
{
    "TransactionID": "1232116",
    "ReturnCode": 0,
    "Message": "Camera closed."
}
```

Configuring the Camera Parameters

Retrieving Parameter List

With the following message, you can request a list of available parameters.

Message Envelope for ZMQ Message

Frame	Field	Description	Type
1	payload	Method payload	Binary string (UTF-8)

JSON Payload

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
Action	No	String	Method called. Fixed value: *GetParameters*
DeviceID	No	String	Device identification.
ParameterList	No	Array of string	The list of parameters that shall be read.

```
{
    "TransactionID": "2345645",
    "Action": "GetParameters",
    "DeviceID": "548451887",
    "ParameterList": [
        "Width",
        "Height"
    ]
}
```

Response

Frame	Field	Description	Type
1	payload	Response object	Binary string (UTF-8)

Response Object

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
ReturnCode	No	Integer	Message code following Message Codes.
Message	Yes	String	Response message.
ParameterList	No	Parameter Object Array	Parameter object array containing the parameter details.

Parameter Object

Field	Optional	Type	Description
Name	No	String	Unique parameter name.
DisplayName	Yes	String	Display name of the parameter for the user interface.

Field	Optional	Type	Description
Type	No	String	Parameter value type. Possible values: *String*, *Integer*, *Float*, *Enumeration*, *Boolean*
Value	Yes	String/ Int/ Float/ Bool	Current defined value. On enumerations, it's the string defined.
IntValue	Yes	Int	On enumerations, it's the integer defined.
Readable	No	Boolean	Flag indicating whether the parameter can be read. The readable flag depends on the current camera state.
Writable	No	Boolean	Flag indicating whether the parameter can be written. The writable flag depends on the current camera state.
Minimum	Yes	Int/ Float	Minimum allowed value for numeric type parameters.
Maximum	Yes	Int/ Float	Maximum allowed value for numeric type parameters.
Increment	Yes	Int/ Float	Allowed increment for the parameter.
EnumEntries	Yes	Enum Entries Array	Array containing the entries available for the parameter.

Enum Entries

Field	Optional	Type	Description
DisplayName	Yes	String	Display name of the enum entry for the user interface.
Value	No	String	Parameter value that needs to be set.
IntValue	No	Int	Integer value of the enum entry.
Description	Yes	String	Enum entry description.


```
{
  "TransactionID": "2345645",
  "ReturnCode": 0,
  "Message": "Parameters read successfully",
  "ParameterList": [
    {
      "Name": "Width",
      "DisplayName": "Width",
      "Type": "Integer",
      "Value": 1024,
      "Writable": true,
      "Minimum": 1,
      "Maximum": 4096,
      "Increment": 1
    },
    {
      "Name": "PixelFormat",
      "DisplayName": "Pixel Format",
      "Type": "Enumeration",
      "Value": "Mono8",
      "IntValue": 17301505,
      "Writable": true,
      "EnumEntries": [
        {
          "DisplayName": "Mono 8",
          "Value": "Mono8",
          "IntValue": 17301505,
          "Description": "This enumeration value sets the pixel format to
```

```
Mono 8."
        },
        {
          "DisplayName": "Mono 10",
          "Value": "Mono10",
          "IntValue": 17825795,
          "Description": "This enumeration value sets the pixel format to
Mono 10."
        },
        {
          "DisplayName": "Mono 12",
          "Value": "Mono12",
          "IntValue": 17825797,
          "Description": "This enumeration value sets the pixel format to
Mono 12."
        },
        {
          "DisplayName": "Mono 16",
          "Value": "Mono16",
          "IntValue": 17825799,
          "Description": "This enumeration value sets the pixel format to
Mono 16."
        }
      ]
    }
  ]
}
```


Setting Camera Parameters

To set the camera parameters, you can send the message described in this section. It's possible to update a range of parameters; however, any error that may occur resets the camera to the original state without applying any modification.

You can define enumeration parameter types using the human-readable value. For example, you can set the pixel format using the value as **Mono8**, instead of the enumeration integer value.

For string and integer parameters, the Basler Vision Connector expects an exact match for the set value. For float parameters, a difference of no more than 1% is allowed due to floating-point comparison. If the provided value falls outside the expected range, the setting call returns an error.

The list of set values is returned in the response message.

Message Envelope for ZMQ Message

Frame	Field	Description	Type
1	payload	Method payload	Binary string (UTF-8)

JSON Payload

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
Action	No	String	Method called. Fixed value: *SetParameters*
DeviceID	No	String	Device identification.
ParameterList	No	Parameter Object Array	Object to encapsulate the parameters to update.

Parameter Object

Field	Optional	Type	Description
Name	No	String	Unique parameter name.
Value[1]	No	String/Int/Float/Bool	New value.

[1] On enumerations, it can be the integer value or the symbolic name. See Retrieving Parameter List for further information.

```
{
    "TransactionID": "123567",
    "Action": "SetParameters",
    "DeviceID": "548451887",
    "ParameterList":[
        {"Name": "Width", "Value": 800},
        {"Name": "Height", "Value": 600},
        {"Name": "PixelFormat", "Value": "Mono8"} //Setting using the human
readable string. This could be also the correspondent enum integer.
    ]
}
```

Response

Frame	Field	Description	Type
1	payload	Response object	Binary string (UTF-8)

Response Object

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
ReturnCode	No	Integer	Message code following Message Codes.
Message	Yes	String	Response message.
ParameterList	No	Parameter Object Array	Object to encapsulate the parameters to update.

Parameter Object

Field	Optional	Type	Description
Name	No	String	Unique parameter name.[2]
Value	No	String/Int/Float/Bool	New value.

```
{
    "TransactionID": "123567",
    "ReturnCode": 0,
    "Message": "Paramters updated.",
    "ParameterList":[
        {"Name": "Width", "Value": 800},
        {"Name": "Height", "Value": 600},
        {"Name": "PixelFormat", "Value": "Mono8"}
    ]
}
```

Setting Camera Quick Parameters

Additionally to the parameter settings, the **Quick Parameters** feature provides a way to quickly define the most commonly used parameters. These parameters use common known names, and the Basler Vision Connector maps them to the corresponding camera parameter.

Message Envelope for ZMQ Message

Frame	Field	Description	Type
1	payload	Method payload	Binary string (UTF-8)

JSON Payload

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
Action	No	String	Method called. Fixed value: *SetQuickParameters*
DeviceID	No	String	Device identification.
Width	Yes	Integer/ String	Image width.[1]
PixelFormat	Yes	Integer/ String	Pixel format.[1,2]
Height	Yes	Integer/ String	Image height.[1]
Gain	Yes	Float/ Int/String	Gain.[1] To activate automatic gain, use the string *Auto*.
ExposureTime	Yes	Float/ Int/String	Exposure time in microseconds[1]. To activate automatic exposure, use the string *Auto*.

[1] Camera-dependent values, invalid values are listed in the returning error message.

[2] GenICam standardized pixel format, complying with the **GenICam Naming Convention** as described in section 4.35 GenICam*PFNC*2_4.pdf.

```
{
    "TransactionID": "23456435",
    "Action": "SetQuickParameters",
    "DeviceID": "548451887",
    "Width": 800,
    "Height": 600,
    "Gain": "Auto",
    "PixelFormat": "Mono8",
    "ExposureTime": 234.5
}
```

Response

Frame	Field	Description	Type
1	payload	Response object	Binary string (UTF-8)

Response Object

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
ReturnCode	No	Integer	Message code following Message Codes.
Message	Yes	String	Response message.

```
{
    "TransactionID": "23456435",
    "ReturnCode": 0,
    "Message": "Parameters updated."
}
```

Defining Camera User Settings

With the GenICam protocol you can store parameter sets on the camera, and with the Basler Vision Connector you can save, load, and define the default user set to be used on camera restart.

Getting Available User Sets

With the following message, you can retrieve the available user sets on the camera.

Message Envelope for ZMQ Message

Frame	Field	Description	Type
1	payload	Method payload	Binary string (UTF-8)

JSON Payload

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
Action	No	String	Method called. Fixed value: *GetUserSets*
DeviceID	No	String	Device identification.

```
{
    "TransactionID": "1232116",
    "Action": "GetUserSets",
    "DeviceID": "12323454754"
}
```

Response

Frame	Field	Description	Type
1	payload	Response object	Binary string (UTF-8)

Response Object

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
ReturnCode	No	Integer	Message code following Message Codes.
Message	Yes	String	Response message.
UserSetList	No	User Set Object Array	Object containing the available parameter settings.

User Set Object

Field	Optional	Type	Description
DisplayName	Yes	String	This represents the name of the user set.
Value	No	String	Parameter value that needs to be used to set the user set value.
IntValue	No	Int	Integer value of the user set.

Field	Optional	Type	Description
Description	Yes	String	User set description.


```
{
    "TransactionID": "12323454754",
    "ReturnCode": 0,
    "Message": "Userset retrieved.",
    "UserSetList": [
      {
        "DisplayName": "Default User Set",
        "Value": "Default",
        "IntValue": 0,
        "Description": "The default factory set can be loaded."
      },
      {
        "DisplayName": "User Set 1",
        "Value": "UserSet1",
        "IntValue": 1,
        "Description": "User set 1 can be saved, loaded, or configured."
      },
      {
        "DisplayName": "User Set 2",
        "Value": "UserSet2",
        "IntValue": 2,
        "Description": "User set 2 can be saved, loaded, or configured."
      },
      {
        "DisplayName": "User Set 3",
        "Value": "UserSet3",
        "IntValue": 3,
        "Description": "User set 3 can be saved, loaded, or configured."
      }
    ]
}
```


Loading the UserSet Configuration

To load a configured **UserSet** on the camera, you can send the *LoadUserSet* message.

Message Envelope for ZMQ Message

Frame	Field	Description	Type
1	payload	Method payload	Binary string (UTF-8)

JSON Payload

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
Action	No	String	Method called. Fixed value: *LoadUserSet*.
DeviceID	No	String	Device identification.
UserSet	No	Int/ String	{Value} or {IntValue} of the User Set Object as defined in Getting Available User Sets.

```
{
  "TransactionID": "23423423563456",
  "Action": "LoadUserSet",
  "DeviceID": "548451887",
  "UserSet": "UserSet2"
}
```

Response

Frame	Field	Description	Type
1	payload	Response object	Binary string (UTF-8)

Response Object

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
ReturnCode	No	Integer	Message code following Message Codes.
Message	Yes	String	Response message.

```
{
  "TransactionID": "1232116",
  "ReturnCode": 0,
  "Message": "Userset loaded."
}
```

Saving the Configuration

After the camera is parametrized, you can save the current parameters in a user set. To save the configuration, send the *SaveUserSet* message.

Message Envelope for ZMQ Message

Frame	Field	Description	Type
1	payload	Method payload	Binary string (UTF-8)

JSON Payload

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.

Field	Optional	Type	Description
Action	No	String	Method called. Fixed value: *SaveUserSet*
DeviceID	No	String	Device identification.
UserSet	No	Int/ String	{Value} or {IntValue} of the User Set Object as defined in Getting Available User Sets

```
{
  "TransactionID": "489494",
  "Action": "SaveUserSet",
  "DeviceID": "548451842342387",
  "UserSet": "UserSet2"
}
```

Response

Frame	Field	Description	Type
1	payload	Response object	Binary string (UTF-8)

Response Object

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
ReturnCode	No	Integer	Message code following Message Codes.
Message	Yes	String	Response message.

```
{
  "TransactionID": "489494",
  "ReturnCode": 0,
  "Message": "User set updated."
}
```

Defining the Default Configuration

To define the default camera parameter set, which is loaded to the camera when the camera is restarted, you can send the *SetDefaultUserSet* message.

Message Envelope for ZMQ Message

Frame	Field	Description	Type
1	payload	Method payload	Binary string (UTF-8)

JSON Payload

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
Action	No	String	Method called. Fixed value: *SetDefaultUserSet* .
DeviceID	No	String	Device identification.
UserSet	No	Int/ String	{Value} or {IntValue} of the User Set Object as defined in Getting Available User Sets

```
{
  "TransactionID": "1232116",
  "Action": "SetDefaultUserSet",
  "DeviceID": "548451887",
  "UserSet": "UserSet2"
}
```

Response

Frame	Field	Description	Type
1	payload	Response object	Binary string (UTF-8)

Response Object

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
ReturnCode	No	Integer	Message code following Message Codes.
Message	Yes	String	Response message.

```
{
  "TransactionID": "1232116",
  "ReturnCode": 0,
  "Message": "Default userset defined."
}
```

Downloading the Camera Configuration

With the following message, you can download the current camera configuration, e.g. to make back-ups.

> **Note**
>
> Don't edit the configuration file manually in a text editor. Editing the file manually leads to conflicts when you upload the camera configuration.

Message Envelope for ZMQ Message

Frame	Field	Description	Type
1	payload	Method payload	Binary string (UTF-8)

JSON Payload

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
Action	No	String	Method called. Fixed value: *GetDeviceConfig*
DeviceID	No	String	Device identification.

```
{
    "TransactionID": "23423456",
    "Action": "GetDeviceConfig",
    "DeviceID": "548451887"
}
```

Response

Frame	Field	Description	Type
1	payload	Response object	Binary string (UTF-8)

Response Object

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
ReturnCode	No	Integer	Message code following Message Codes.
Message	Yes	String	Response message.
FileContent	No	String	Base64 string of the configuration file, which contains the mime type prefix.

```
{
    "TransactionID": "23423456",
    "ReturnCode": 0,
    "Message": "Success",
    "FileContent": "data:text/plain;base64,IyB7MDVEOEMyOTQtRjI5NS00Z[...]"

}
```

Uploading the Camera Configuration

With the following message you can provide the camera configuration file, which you have downloaded as described in section Downloading the Camera Configuration, to the camera to restore the camera setup.

Message Envelope for ZMQ Message

Frame	Field	Description	Type
1	payload	Method payload	Binary string (UTF-8)

JSON Payload

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
Action	No	String	Method called. Fixed value: *SetDeviceConfig*.
DeviceID	No	String	Device identification.
FileContent	No	String	Base64 string of the configuration file, which contains the mime type prefix.

```
{
    "TransactionID": "234423465465",
    "Action": "SetDeviceConfig",
    "DeviceID": "548451887",
    "FileContent": "data:text/plain;base64,IyB7MDVEOEMyOTQtRjI5NS00Z[...]"
}
```

Response

Frame	Field	Description	Type
1	payload	Response object	Binary string (UTF-8)

Response Object

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
ReturnCode	No	Integer	Message code following Message Codes.
Message	Yes	String	Response message.

```
{
  "TransactionID": "234423465465",
  "ReturnCode": 0,
  "Message": "Configuration loaded."
}
```

Image Streaming

With the Basler Vision Connector you can control the image streaming by starting and stopping the image stream. The control doesn't affect any camera property, including the trigger definitions. If you want to change these camera properties, see the messages described in Setting Camera Quick Parameters.

Starting Image Streaming

With this command you can initiate image streaming for a specific camera. When you activate it, continuous streaming publishes images to the streaming topic. For triggered image acquisition, the Basler Vision Connector sends images based on trigger activations configured in the parameters section.

Message Envelope for ZMQ Message

Frame	Field	Description	Type
1	payload	Method payload	Binary string (UTF-8)

JSON Payload

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
Action	No	String	Method called. Fixed value: *StartStreaming*
DeviceID	No	String	Device identification.

```
{
    "TransactionID": "12341263412635465",
    "Action": "StartStreaming",
    "DeviceID": "548451887"
}
```

Response

Frame	Field	Description	Type
1	payload	Response object	Binary string (UTF-8)

Response Object

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
ReturnCode	No	Integer	Message code following Message Codes.
Message	Yes	String	Response message.

```
{
    "TransactionID": "12341263412635465",
    "ReturnCode": 0,
    "Message": "Stream started."
}
```

Stopping Image Streaming

This command terminates camera image streaming.

Message Envelope for ZMQ Message

Frame	Field	Description	Type
1	payload	Method payload	Binary string (UTF-8)

JSON Payload

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.

Field	Optional	Type	Description
Action	No	String	Method called. Fixed value: *StopStreaming*
DeviceID	No	String	Device identification.

```
{
    "TransactionID": "32143242356465",
    "Action": "StopStreaming",
    "DeviceID": "548451887"
}
```

Response

Frame	Field	Description	Type
1	payload	Response object	Binary string (UTF-8)

Response Object

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
ReturnCode	No	Integer	Message code following Message Codes.
Message	Yes	String	Response message.

```
{
    "TransactionID": "32143242356465",
    "ReturnCode": 0,
    "Message": "Stream stopped."
}
```

Software Triggered Image Acquisition

Use this command when the camera is in software trigger operation. When you use this command, image acquisition is triggered as configured on the camera. Images are published on the image streaming topic.

Message Envelope for ZMQ Message

Frame	Field	Description	Type
1	payload	Method payload	Binary string (UTF-8)

JSON Payload

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
Action	No	String	Method called. Fixed value: *SoftwareTrigger*
DeviceID	No	String	Device identification.

```
{
    "TransactionID": "2342546345",
    "Action": "SoftwareTrigger",
    "DeviceID": "548451887"
}
```

Response

Frame	Field	Description	Type
1	payload	Response object	Binary string (UTF-8)

Response Object

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
ReturnCode	No	Integer	Message code following Message Codes.
Message	Yes	String	Response message.

```
{
    "TransactionID": "2342546345",
    "ReturnCode": 0,
    "Message": "Trigger sent."
}
```

Getting the Application Status

With this message you can retrieve the connected camera list with the status, streaming topic, application version, and implemented specification version.

Message Envelope for ZMQ Message

Frame	Field	Description	Type
1	payload	Method payload	Binary string (UTF-8)

JSON Payload

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.
Action	No	String	Method called. Fixed value: *GetStatus*.

```
{
    "TransactionID": "46543214635244",
    "Action": "GetStatus"
}
```

Response

Frame	Field	Description	Type
1	payload	Response object	Binary string (UTF-8)

Response Object

Field	Optional	Type	Description
TransactionID	Yes	String	Transaction identification.

Field	Optional	Type	Description
ReturnCode	No	Integer	Message code following Message Codes.
Message	Yes	String	Response message.
ApplicationVersion	No	String	Connector application version.
SpecificationVersion	No	String	Specification version of the Basler Vision Connector.
DeviceList	No	Array of Camera Object	Array of the camera objects.

Camera Object

Field	Optional	Type	Description
DeviceID	No	String	Current camera identification.
StreamingTopic	No	String	Topic where the camera streams images.
Status	No	String	Camera status. Possible values: *Connected*, *Streaming*, *Error*.

```
{
    "TransactionID": "46543214635244",
    "ReturnCode": 0,
    "Message": "Success.",
    "ApplicationVersion": "1.0.1",
    "SpecificationVersion": "1.0.0",
    "DeviceList":
    [
        {
            "DeviceID": "548451887",
            "StreamingTopic": "548451887/stream",
            "Status": "Streaming"
        }
    ]
}
```


Application Feedback

Message Codes

The following message code are sent as responses.

Info Message Codes

Value range for info messages: 0x00000000 … 0x00FFFFFF

Message Codes	Meaning
0x00000000	Request was completed successfully.
0x00000001	Application is online.
0x00000002	Application is offline.

Warning Message Codes

Value range for warning messages: 0x01000000 … 0x01FFFFFF

Message Codes	Meaning
0x01000000	A request was only partially successful.

Debug Message Codes

Value range for debug messages: 0x02000000 … 0x02FFFFFF

Error Message Codes

Value range for error messages: 0x03000000 … 0x03FFFFFF

Message Codes	Meaning
0x03000000	Error while parsing a request.
0x03000001	Error while handling a request.
0x03000002	An opened device was removed.

Application Logging

The application log is published in the logging topic. In this log, you find any general application log that is no direct response to any given command, such as application unhandled errors, camera error messages and camera connectivity issues.

Response

Frame	Field	Description	Type
1	payload	Response object	Binary string (UTF-8)

Response Object

Field	Optional	Type	Description
Timestamp	Yes	String	Application host date and time on ISO8601 format, when the message was created.
Code	Yes	Integer	Message code following Message Codes.
Level	No	String	Log level - *Debug*, *Info*, *Warning*, *Error*.
Message	No	String	Response message.

```
{
    "Timestamp":"2014-01-09T13:35:34.000000000+0100",
    "Code": 50331650,
    "Level":"Error",
    "Message":"Lost connection with myCam1"
}
```

Revision History

Document Number	Date	Changes
AW00185001000	26 January 2024	Initial version of this document.