APPLICATION NOTE

Trigger over Ethernet - Action Commands

V1.0.0

Scope of this document

This document covers the following topics:

- Benefits of Action Commands (also known as Trigger over Ethernet or ToE)
- Options for sending out Action Commands
- Using Action Commands with Vimba Viewer and Vimba APIs
- Advanced scenarios, Troubleshooting

Benefits of Action Commands

Single cable solution

Action Commands are also known as Trigger over Ethernet (ToE), since no additional trigger cable is required. You can send the trigger signal to one or multiple cameras.

High-precision trigger broadcast

In contrast to a software trigger, which is the established Trigger over Ethernet method, Action Commands enable broadcasting a trigger signal to multiple GigE cameras at almost the same time. Therefore, Action Commands are more precise than a software trigger.

Flexible multi-camera applications

To enhance the flexibility of your vision application, Allied Vision GigE cameras and our Vimba SDK support two different Action Commands: Action0 and Action1 are provided. Additionally, you can define camera groups and assign cameras to one or multiple sub-groups.

Fast reaction

Trigger signals sent out from the host PC always have some latency caused by the PC's operating system. If your vision application requires a particularly fast reaction to a trigger signal, you can send out Action Commands with special hardware.

Compatible with GigE Vision

Action Commands are covered by the GigE Vision Specification and thus are supported by a large selection of machine vision hardware and software.

Prerequisites

To use Allied Vision cameras and Action Commands, you need:

- Allied Vision GigE camera(s) with Action Commands support. To find out if your camera supports Action Commands, see the technical manual or the camera model webpage on https://www.alliedvision.com.
- Vimba 2.1 or special hardware capable of sending out Action Commands.

This document assumes you have already installed and configured the host adapter and applicable drivers according to the instructions in the GigE Installation Manual.

Additional information

We recommend downloading the following documents:

- The **GigE Installation Manual** contains installation and troubleshooting instructions for your camera and the host adapter.
- The **GigE Features Reference** provides a general overview of GigE camera and driver features. Our website may provide a later version than your Vimba installation.
- The **technical manual** for your camera contains camera-specific information and technical data.
- The **Vimba Viewer Guide** explains the basic camera setup. It is automatically installed in the Vimba program folder.

Download additional documentation

You can download additional documentation from:

https://www.alliedvision.com/en/support/technical-documentation.html

Options for sending out Action Commands

There are several options for sending out Action Commands.

- You can send out Action Commands from the host PC with:
  - Vimba Viewer
  - Vimba C, C++, or .NET API
  - A third-party software with Action Commands support
- You can send out Action Commands from special hardware, e.g:
  - Advantech Intelligent GigE Vision frame grabbers PCIE-1172 or PCIE-1174
  - IMAGO Technologies VisionBox embedded vision computer

All options require these basic steps:

1. Open the camera(s).
2. Configure the trigger:
   a. Set the trigger selector (e.g., FrameStart).
   b. Set the trigger source (Action0 or Action1).
   c. Set the trigger mode (On).

4. Configure the ActionControl parameters on each camera.
5. Configure the ActionControl parameters on the host PC or the special third-party hardware.
6. Start acquisition.
7. Send out an Action Command.

Configuring ActionControl parameters

The following ActionControl parameters must be configured on the camera(s) and then on the host PC.

- `ActionDeviceKey` must be equal on the camera and on the host PC. Before a camera accepts an Action Command, it verifies if the received key is identical with its configured key. Note that `Action-DeviceKey` must be set each time the camera is opened.
  Range (camera and host PC): 0 to 4294967295
- `ActionGroupKey` means that each camera can be assigned to exactly one group for Action0 and a different group for Action1. All grouped cameras perform an action at the same time. If this key is identical on the sender and the receiving camera, the camera performs the assigned action.
  Range (camera and host PC): 0 to 4294967295
- `ActionGroupMask` serves as a filter that specifies which cameras within a group react to an Action Command. It can be used to create sub-groups. To react to an Action Command, the value in the camera must have one bit in common with the received value from the sender. You can define different Group Masks for Action0 and Action1.
  Range (camera): 0 to 4294967295
  Range (host PC): 1 to 4294967295

Using Action Commands with Vimba Viewer

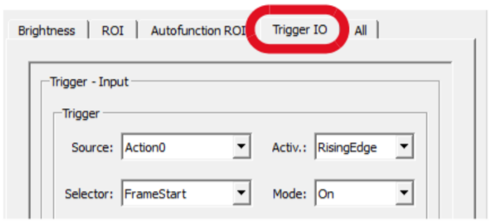
Using Action Commands requires configuring them first on the camera and then on the host PC.

Configuring Action Commands with Vimba Viewer

- First set the Selector (e.g., FrameStart), then select the Source (`Action0` or `Action1`).
- Before sending out Action Commands, click the Freerun button.

Configuring Action Commands on the camera

1. On the **Trigger IO** tab, choose a Selector (e.g., FrameStart), select Source: *Action0* or *Action1*, and Mode: *On*.

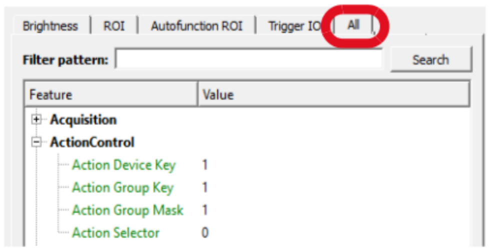


Source: Action0
Activ.: RisingEdge
Selector: FrameStart
Mode: On

Figure 1: Set the Selector, Source, and Mode

Adjust the other trigger parameters as required by your use case.

2. On the **All** tab, click *ActionControl* and enter values for the ActionControl parameters.



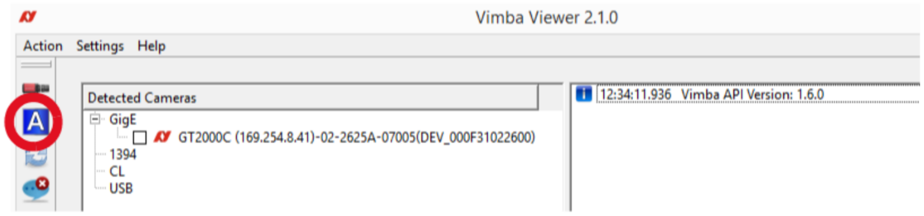
Feature	Value
Acquisition	
ActionControl	
Action Device Key	1
Action Group Key	1
Action Group Mask	1
Action Selector	0

Figure 2: Enter ActionControl values

3. Click the **Freerun** button.

Configuring Action Commands on the host PC

1. While the main window remains open, go to the Camera Selector window and click the **Trigger over Ethernet - Action Commands** icon.

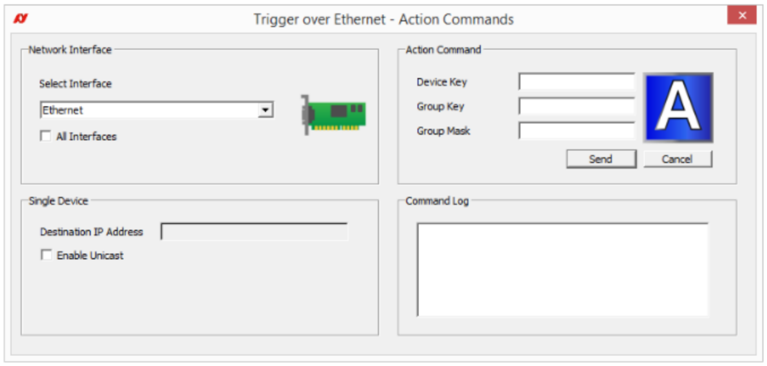


GT2000C (169.254.8.41)-02-2625A-07005(DEV_000F31022600)

12:34:11.936 Vimba API Version: 1.6.0

Figure 3: Action Commands icon

The Trigger over Ethernet - Action Commands window opens.



Select Interface: Ethernet

Figure 4: Trigger over Ethernet - Action Commands window

2. Select the host adapter your cameras are connected to or select *All interfaces* to broadcast the command via all host adapters. To send out an Action Command to a single device, check *Enable Unicast* and enter the device's IP address in the Destination IP Address field.

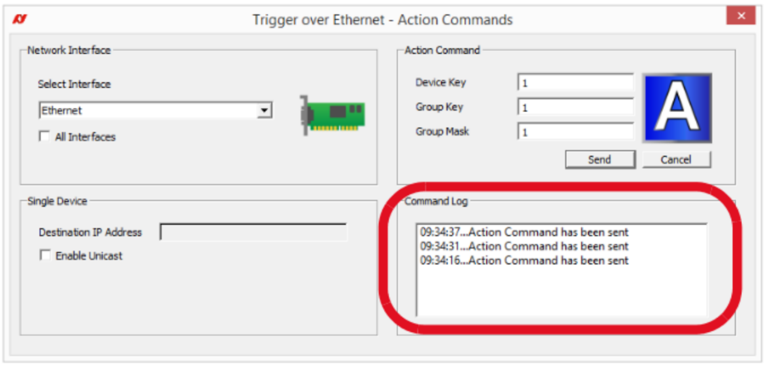
Listed network interfaces

All Gigabit Ethernet adapters with a connected device are shown, even if the device does not support Action Commands.

No reload

The Trigger over Ethernet - Action Commands window does not reload when you plug in or out a camera. If you plug in GigE cameras while this window is open, close the window, wait until the device is detected, an then reopen it.

3. Enter the values for Device Key, Group Key, and Group Mask from the camera settings into the empty fields to configure them on the host PC.
4. To execute an Action Command, click the *Send* button. The Command Log shows successfully sent Action Commands.



Device Key: 1
Group Key: 1
Group Mask: 1

```
09:34:37...Action Command has been sent
09:34:31...Action Command has been sent
09:34:16...Action Command has been sent
```

Figure 5: Command log

Programming Action Commands with Vimba

Loading camera settings with a Vimba API

If you have configured the camera settings for Action0 with Vimba Viewer, you can conveniently load them with a Vimba API. To do this, save the setting in Vimba Viewer and then see the description in the Vimba API manuals and Vimba's included LoadSaveSettings programming example.

The following C code snippet shows how to send out an Action Command to all Ethernet interfaces.

```c
VmbUint32_t count;
VmbCameraInfo_t* cameras;
VmbHandle_t* handles;
int deviceKey = 11, groupKey = 22, groupMask = 33;

// Start Vimba
VmbStartup();

// Discover GigE cameras
VmbFeatureBoolGet( gVimbaHandle , "GeVTLIsPresent", &isGigE );
if( VmbBoolTrue == isGigE )
{
    VmbFeatureIntSet( gVimbaHandle , "GeVDiscoveryAllDuration", 250 );
    VmbFeatureCommandRun( gVimbaHandle , "GeVDiscoveryAllOnce" );
}

// Get cameras
VmbCamerasList( NULL , 0, &count , sizeof (* cameras) );
cameras = (VmbCameraInfo_t *) malloc( count * sizeof (* cameras) );
VmbCamerasList( cameras , count , &count , sizeof (* cameras) );

// Allocate space for handles
handles = (VmbHandle_t *) malloc( count * sizeof(VmbHandle_t) );

for( int i=0; i<count; ++i )
{
    const char* cameraId = cameras[i]. cameraIdString;

    // Open camera
    VmbCameraOpen( cameraId , VmbAccessModeFull , &handles[i] );

    // Set device key, group key, and group mask
    // Configure trigger settings (see programming example)
    VmbFeatureIntSet( handles[i], "ActionDeviceKey", deviceKey );
    VmbFeatureIntSet( handles[i], "ActionGroupKey", groupKey );
    VmbFeatureIntSet( handles[i], "ActionGroupMask", groupMask );
}

// Set Action Command to API
// Allocate buffers and enable streaming (see programming example)
VmbFeatureIntSet( gVimbaHandle , "ActionDeviceKey", deviceKey );
VmbFeatureIntSet( gVimbaHandle , "ActionGroupKey", groupKey );
VmbFeatureIntSet( gVimbaHandle , "ActionGroupMask", groupMask );

// Send Action Command
 VmbFeatureCommandRun( gVimbaHandle , "ActionCommand" );

 // If no further Actions will be applied: close cameras, shutdown API, and
 // free allocated space as usual
```

Example 1: Action Commands C code snippet

Download Action Commands programming examples

Download Action Commands programming examples for Vimba APIs.

Please add the examples to the folders for the existing Vimba examples.

Advanced scenarios

Scenario A: Using Action0 and Action1

You can use `Action0` and `Action1` to trigger different kinds of actions such as starting and stopping image acquisition. If you send out Action Commands with Vimba Viewer, the camera automatically toggles between `AcquisitionStart` and `AcquisitionEnd` each time an Action Command is received.

Example:

For Action0 and Action1, set a common `ActionDeviceKey` (for example, `ActionDeviceKey = 1`)

For `Action0`, use Selector `AcquisitionStart` and, for example:

```
ActionGroupKey  = 1
ActionGroupMask = 1
```

For `Action1`, use Selector `AcquisitionEnd` and, for example:

```
ActionGroupKey  = 1
ActionGroupMask = 1
```

Scenario B: Assigning cameras to different groups

You can use `ActionGroupKey` to assign each camera to one group within Action0 and optionally a different group within Action1. Assigning one camera to multiple groups within the same Action is not possible.

Example:

Selector `FrameStart` and `Action0` are used for all cameras. As soon as an Action Command is sent, all assigned cameras acquire one frame.

Camera1 and Camera2 are members of ActionGroup 1:

```
ActionDeviceKey = 1
ActionGroupKey  = 1
ActionGroupMask = 1
```

Camera3 and Camera4 are members of ActionGroup 2:

```
ActionDeviceKey = 1
ActionGroupKey  = 2
ActionGroupMask = 1
```

Scenario C: Assigning cameras to sub-groups

You can use `ActionGroupMask` as a filter for sub-groups. A camera can be assigned to one or multiple sub-groups or it can be excluded from all of them. If the Group Mask on the host PC and on the camera have one or more bits with value 1 in common, then the camera reacts to an Action Command.

Examples of binary Group Mask values (for simplification, the example shows four digits):

Sent Group Mask value	Camera Group Mask value	Camera reaction
1111	0001	Yes
1101	0010	No
1111	0011	Yes
1110	0011	Yes

Table 1: Examples of binary Group Mask values

Cameras can be assigned to one, multiple, or no sub-groups:

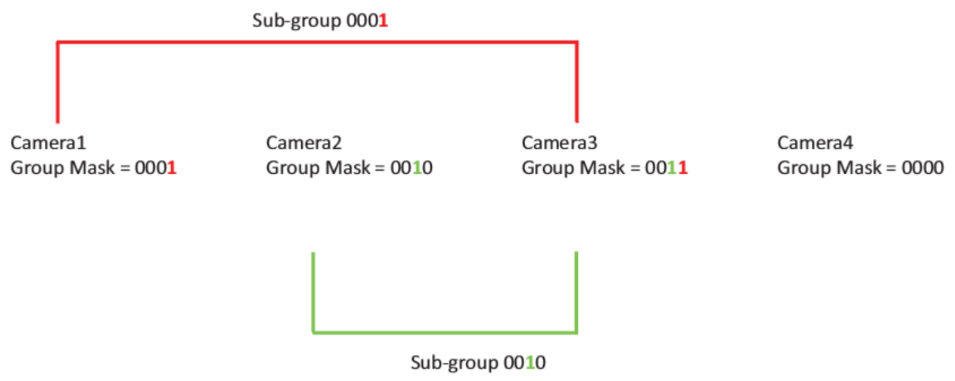


Figure 6: Group Mask principle: A common bit 1 = a common sub-group

While Group Mask value 0 is valid on a camera (this particular camera is currently excluded from all groups and sub-groups and ignores Action Commands), sending out an Action Command with value 0 is not allowed per the GigE Vision Specification: It causes an error because no camera will react to it. Therefore, the Group Mask range on the PC starts with 1, while the range on the camera starts with 0.

Combining scenarios

You can combine the scenarios as required. For example, you can use `Action0` and `Action1`, assign the cameras to different groups, and use Group Mask to create additional sub-groups.

Tips and troubleshooting

- If Action Commands are unavailable although your camera should support them, download the latest firmware from: https://www.alliedvision.com/en/support/firmware.html
- `ActionDeviceKey` must be set each time the camera is opened. Do not close the camera while working with Action Commands.
- Vimba Viewer: Click the Freerun button before sending out Action Commands.
- If you program with Vimba APIs: Sending out Action Commands via Interface only sends them to the devices connected to the host adapter. To reach all GigE devices, send out Action Commands via System.
- If you use an Ethernet router, make sure all cameras are in the same subnet. Using a switch normally does not affect Action Commands.
- Be aware that the camera verifies if the received key matches its configured key in the following order:
  1. ActionDeviceKey
  2. ActionGroupKey
  3. ActionGroupMask
- Per the GigE Vision Specification, Action signals can only be asserted if the device has an open primary control channel or when unconditional action mode is enabled. (Note that unconditional action mode is not supported by all devices.)
- For troubleshooting, use EventAction0 and EventAction1 (see GigE Features Reference).