The World's Sixth Sense®

Spinnaker Nodes

Revised 2/9/2018

Applicable products

- Blackfly®S
- Blackfly®
- Chameleon®3
- Flea®3
- Grasshopper®3
- Oryx®

Application note description

This application note describes Spinnaker nodes, how they are categorized and identified, and how to access them in the Spinnaker API.

Preparing for use

Before you use your camera, we recommend that you are aware of the following resources available from our downloads page:

- ***Getting Started Manual* for the camera**—provides information on installing components and software needed to run the camera.
- ***Technical Reference* for the camera**—provides information on the camera's specifications, features and operations, as well as imaging and acquisition controls.
- **Firmware updates**—ensure you are using the most up-to-date firmware for the camera to take advantage of improvements and fixes.
- **Tech Insights**—Subscribe to our monthly email updates containing information on new knowledge base articles, new firmware and software releases, and Product Change Notices (PCN).

What is the Spinnaker API?

Spinnaker API is built around the GenICam standard, which offers a generic programming interface for various cameras and interfaces. Spinnaker is an extension of GenAPI. Spinnaker provides quick and easy access to your camera.

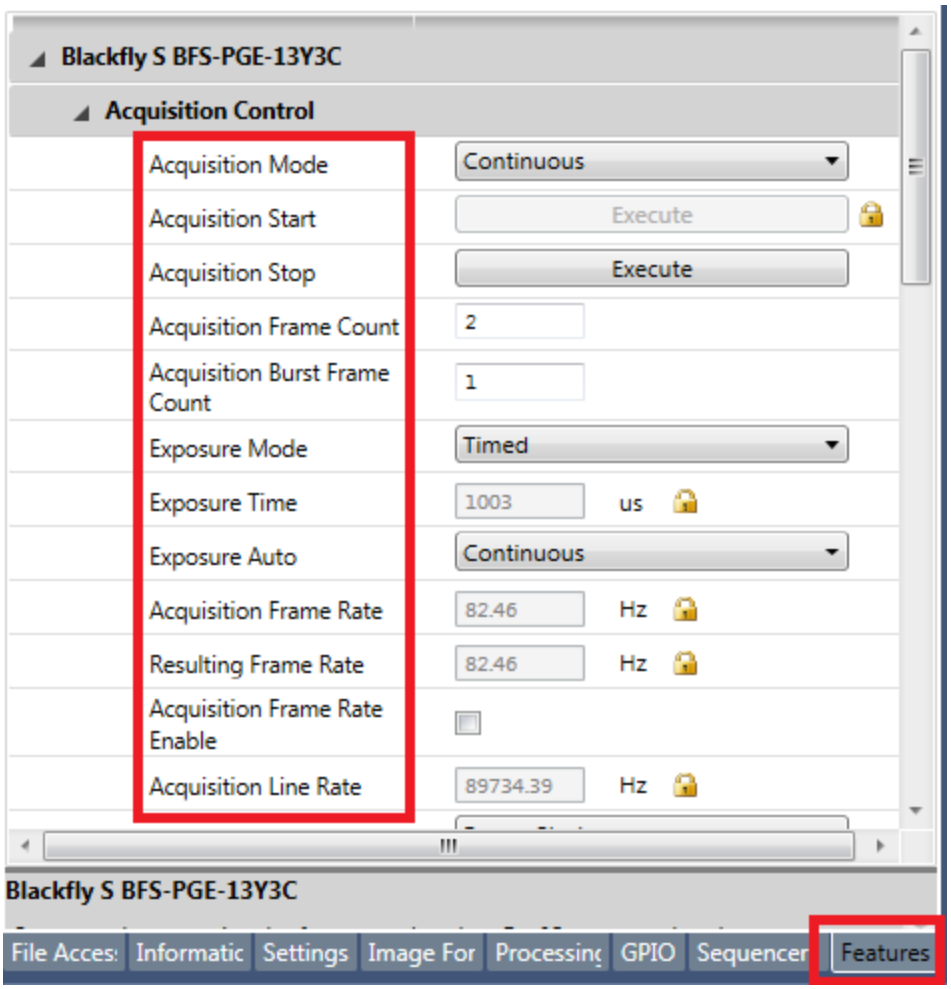
Spinnaker API includes two major components:

Image Acquisition—This is the acquisition engine that is responsible for setting up image buffers and image grabbing.

Camera Configuration—This is the configuration engine that is responsible for controlling your camera. This component consists of the QuickSpin API, which is a wrapper that makes GenAPI easy to use.

Nodes

Every GenICam compliant camera has an XML description file. The XML describes camera registers, their interdependencies, and all other information needed to access high-level features by means of low level register read and write operations. These features include Gain, Exposure Time, Image Format, and others. The elements of a camera description file are represented as software objects called nodes. A nodes map is a list of nodes created dynamically at run time.



All the features and functionality described on the Features tab in SpinView represent nodes.

Node types

The majority of nodes fall within 7 types. They are:

Type	Description	Example
Enumeration	Any feature that has a selection of text entries available	Auto Gain or Acquisition Mode
EnumEntry	The individual entry within an enumeration feature	Continuous, Once, Off or Continuous, Multi frame, Single frame
Command	Any feature that requires a command to execute, normally denoted by an execute button in SpinView	Timestamp Latch or Trigger Software
Float	Any feature that has a number entry that may include a decimal point	Gain or Exposure Time
Boolean	Any feature that acts as an on or off switch for that feature normally denoted by a checkbox in SpinView	ReverseX or Acquisition Frame Rate Enable
Integer	Any feature that has a number entry without a decimal point	Width or Height
String	Any feature that has a user-defined or static text entry	Vendor Name (static) or Device User ID (user-defined)

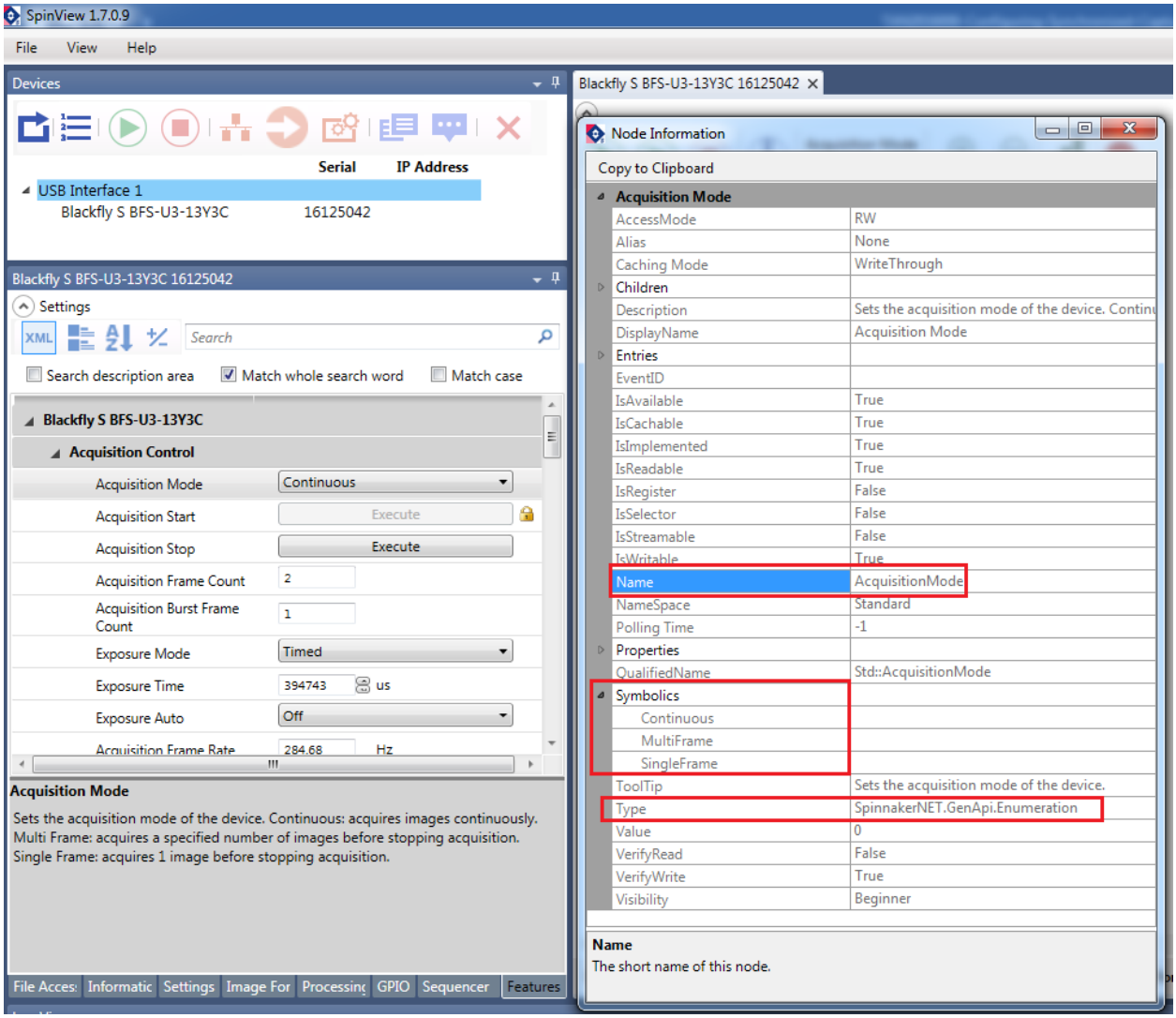
Methods to determine node types and names

There are several ways to determine what type of node a feature is and what the exact node name within the API. Reviewing the Spinnaker API documentation that comes included with the Spinnaker SDK is one method, as it contains a complete list of the nodes currently available for all Spinnaker compatible cameras. However, it may include some nodes/entries that are not supported by your particular camera.

Another method would be to use SpinView. It lists only the nodes that the camera supports, and provides a "node information window" which contains detailed information on each individual node.

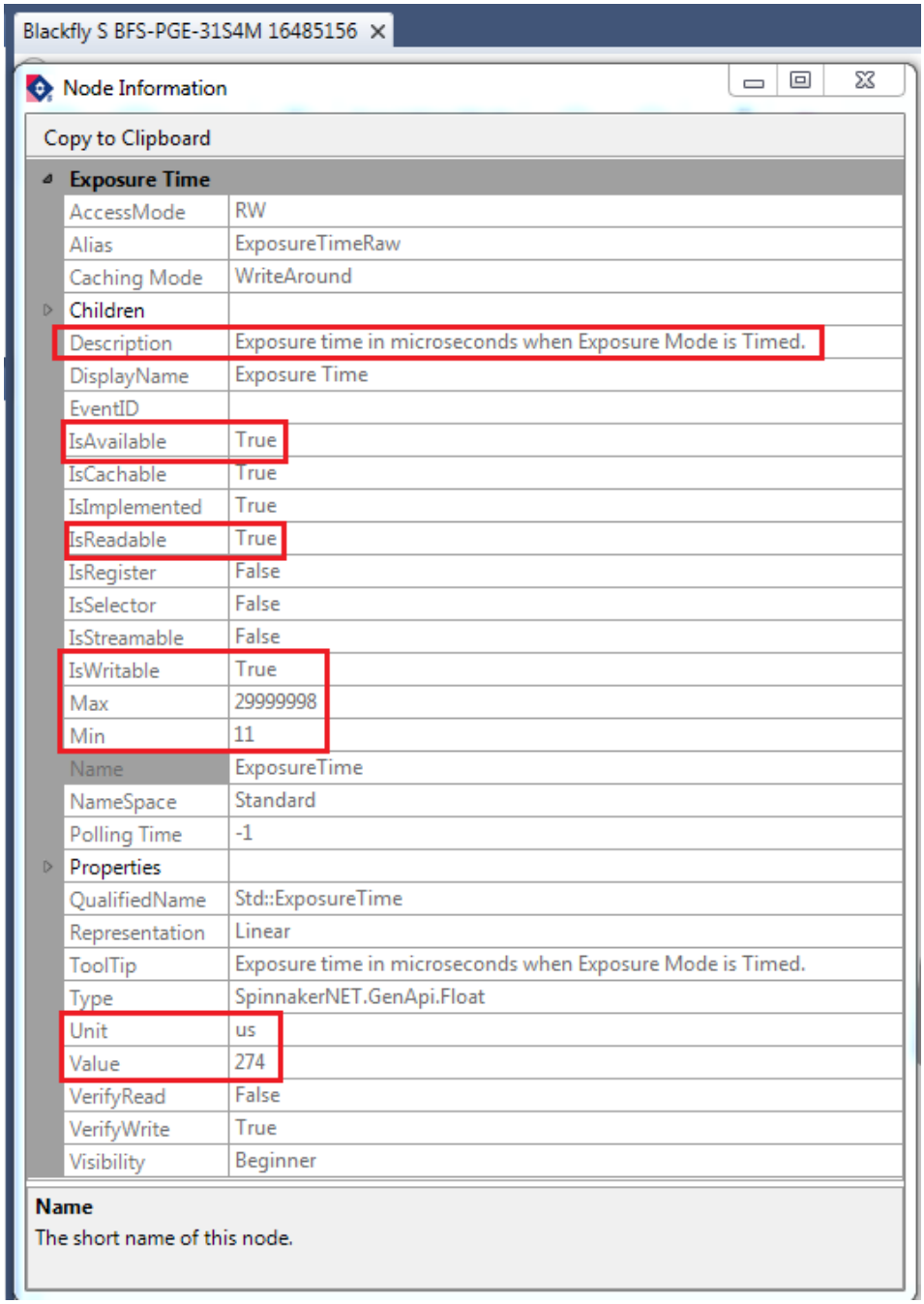
To access the node information window:

1. Open SpinView and select your camera.
2. On the Features tab, double-click on any node. In the example below, the Acquisition Mode has been selected.



Name shows the actual name of the node used within the Spinnaker API. **Type** shows the node type. As this is an enumeration type, the **Symbolics** section shows the EnumEntry names for this node.

The Node Information window contains additional information which can be helpful when accessing or modifying node values.



Description: provides the details of the purpose of the node.

IsAvailable, IsReadable, IsWriteable: can be used to check whether a node is available, or readable, or writeable before accessing it.

Min/Max: these define the minimum and maximum values for numerical based nodes, either integer or float.

Unit: displays the real-world units for the feature.

Value: displays the current value.

Accessing nodes in the API

Spinnaker supports multiple programming languages. Regardless of the programming language used, the same node setup applies. The code snippets below show how to access the different node types within three of the supported programming languages: C++, C#, and C.

> **Note:** The following code snippets do not implement availability, readability, or writability checks. Review the example code included in the Spinnaker SDK for details on how to do so.

Enumeration and EnumEntry nodes

The following code snippets show how to access Enumeration and EnumEntry nodes.

Spinnaker C++ API

```
// Enumeration node
CEnumerationPtr ptrGainAuto = pCam->GetNodeMap().GetNode("GainAuto");

//EnumEntry node (always associated with an Enumeration node)
CEnumEntryPtr ptrGainAutoOff = ptrGainAuto->GetEntryByName("Off");

//Turn off Auto Gain
ptrGainAuto->SetIntValue(ptrGainAutoOff->GetValue());
```

Spinnaker C# API

```
// Enumeration node
IEnum iGainAuto = nodeMap.GetNode<IEnum>"GainAuto");

//EnumEntry node (always associated with an Enumeration node)
IEnumEntry iGainAutoOff = iGainAuto.GetEntryByName("Off");

//Turn off Auto Gain
iGainAuto.Value = iGainAutoOff.Symbolic;
```

Spinnaker C API

```
spinNodeHandle hGainAuto = NULL;
spinNodeHandle hGainAutoOff = NULL;
int64_t GainAutoOff = 0;

// Retrieve node (Enumeration node in this case)
spinNodeMapGetNode(hNodeMap,"GainAuto", &hGainAuto);

//EnumEntry node (always associated with an Enumeration node)
spinEnumerationGetEntryByName(hGainAuto, "Off", &hGainAutoOff);

//Turn off Auto Gain
spinEnumerationEntryGetIntValue(hGainAutoOff, &GainAutoOff);
spinEnumerationSetIntValue(hGainAuto, GainAutoOff);
```

Command nodes

The following code snippets show how to access Command nodes.

Spinnaker C++ API

```
// Command node
CCommandPtr ptrTimestampLatch = pCam->GetNodeMap().GetNode("TimestampLatch");

//Execute command
ptrTimestampLatch->Execute();
```

Spinnaker C# API

```
// Command node
ICommand iTimestampLatch = nodeMap.GetNode<ICommand>"TimestampLatch");

//Execute command
iTimestampLatch.Execute();
```

Spinnaker C API

```
// Retrieve node (command)
spinNodeMapGetNode(hNodeMap,"TimestampLatch", &hTimestampLatch);

//Execute command
spinCommandExecute(hTimestampLatch);
```

Float nodes

The following code snippets show how to access Float nodes.

Spinnaker C++ API

```
// Float node
CFloatPtr ptrGain = pCam->GetNodeMap().GetNode("Gain");

//Set gain to 0.6 dB
ptrGain->SetValue(0.6);
```

Spinnaker C# API

```
// Float node
IFloat iGain = nodeMap.GetNode<IFloat>"Gain");

//Set gain to 0.6 dB
iGain.Value = 0.6;
```

Spinnaker C API

```
spinNodeHandle hGain = NULL;

// Retrieve node (float)
spinNodeMapGetNode(hNodeMap,"Gain", &hGain);

//Set gain to 0.6 dB
spinFloatSetValue(hGain, 0.6);
```

Boolean nodes

The following code snippets show how to access Boolean nodes.

Spinnaker C++ API

```
// Boolean node
CBooleanPtr ptrGain = pCam->GetNodeMap().GetNode("ReverseX");

//Set value to true (mirror the image)
ptrReverseX->SetValue(true);
```

Spinnaker C# API

```
// Boolean node
IFloat iGain = nodeMap.GetNode<IBool>"ReverseX");

//Set value to true (mirror the image)
iReverseX.Value = true;
```

Spinnaker C API

```
spinNodeHandle hReverseX = NULL;

// Retrieve node (boolean)
spinNodeMapGetNode(hNodeMap,"ReverseX", &hReverseX);

//Set value to true (mirror the image)
spinBooleanSetValue(hReverseX, True);
```

Integer nodes

The following code snippets show how to access Integer nodes.

Spinnaker C++ API

```
// Integer node
CIntegerPtr ptrWidth = pCam->GetNodeMap().GetNode("Width");

//Set width of 640 pixels
ptrWidth->SetValue(640);
```

Spinnaker C# API

```
// Integer node
IInteger iWidth = nodeMap.GetNode<IInteger>"Width");

//Set width of 640 pixels
iWidth.Value = 640;
```

Spinnaker C API

```
// Retrieve node (integer)
spinNodeMapGetNode(hNodeMap,"Width", &hWidth);

//Set width of 640 pixels
spinIntegerSetValue(hWidth, 640);
```

String nodes

The following code snippets show how to access String nodes.

Spinnaker C++ API

```
// String node
CStringPtr ptrVendorName = pCam->GetNodeMap().GetNode("DeviceVendorName");

//Display vendor name
cout << "VendorName: " << ptrVendorName->GetValue() << endl;
```

Spinnaker C# API

```
// String node
IString iVendorName = nodeMap.GetNode<IString>"DeviceVendorName");

//Display vendor name
Console.WriteLine("VendorName: {0}", iVendorName.Value);
```

Spinnaker C API

```
spinNodeHandle hVendorName = NULL;
char vendorName[MAX_BUFF_LEN];
size_t lenVendorName = MAX_BUFF_LEN;

// Retrieve node (string)
spinNodeMapGetNode(hNodeMap,"DeviceVendorName", &hVendorName);

//Display vendor name
spinStringGetValue(hVendorName, vendorName, &lenVendorName);
printf("Vendor name: %s \n", vendorName);
```

Troubleshooting

Unsure of how certain features or nodes work, or how to interact with them

The example code that comes with the Spinnaker SDK covers a large variety of nodes and their functionality.

Unable to make changes or to access a node

You may be trying to access a feature that is currently locked. For details on feature locking, see Technical Application Note Feature Locking in the Spinnaker API.

Questions not covered in the resources above

Contact our technical support team.

Downloads and support

FLIR endeavors to provide the highest level of technical support possible to our customers. Most support resources can be accessed through the Support section of our website.

The first step in accessing our technical support resources is to obtain a customer login account. This requires a valid name and email address. To apply for a customer login account go to our downloads page.

Customers with a customer login account can access the latest **software** and **firmware** for their cameras from our website. We encourage our customers to keep their software and firmware up-to-date by downloading and installing the latest versions.

Finding information

Spinnaker SDK—The Spinnaker SDK provides API examples and the SpinView camera evaluation application. Available from our Downloads page.

API Documentation—The installation of the Spinnaker SDK comes with API references for C++, C#, and C code. A Programmer's Guide is included in each of the references. Available from:

- Start Menu→All Programs→Point Grey Spinnaker SDK→Documentation
- The SpinView application Help menu

Getting Started with SpinView—A quick guide to using the SpinView camera evaluation application provided in the Spinnaker SDK. Available from:

- Start Menu→All Programs→Point Grey Spinnaker SDK→Documentation
- The SpinView application Help menu
- Camera Reference zip package

Camera Reference—A zip package containing PDF and HTML copies of the camera's references, including: Installation Guide, Technical Reference, and Getting Started. Available from our downloads page.

Knowledge Base—A database of articles and application notes with answers to common questions as well as articles and tutorials about hardware and software systems. Available from our knowledge base.

Learning Center—Our Learning Center contains links to many resources including videos, case studies, popular topics, other application notes, and information on sensor technology.

Contacting technical support

Before contacting Technical Support, have you:

1. Read the product documentation?
2. Searched the knowledge base?
3. Downloaded and installed the latest version of software and/or firmware?

If you have done all the above and still can't find an answer to your question, contact our technical support team.

Additional resources

GenICam—A programming interface for cameras and devices. More information available on the EMVA.org website.

USB3 Vision—A vision standard for the USB 3.1 interface that uses GenICam. More information available on the AIA Vision Online website.