SPECIM
A Konica Minolta Company

Lumo Sensor SDK User Manual 1.6

Contents

1 Legal Information....8

2 API Overview....9

3 Initial Setup....10
3.1 Installing Lumo Sensor SDK....10
3.2 Pleora eBUS Software Development Kit....16
3.3 Frame Grabber Installation....16
3.4 Configuring the FileReader....17
3.5 Installing USB Serial Devices....17

4 Walkthrough....22
4.1 Including Lumo Sensor SDK....22
4.2 Linking Lumo Sensor SDK....22
4.2.1 Static Linking....22
4.2.2 Dynamic Linking....22
4.3 Copying the Dependency Binaries....24
4.4 Loading and Unloading....25
4.5 Error Checking....25
4.5.1 Debug and Warning Message Logging....26
4.6 Listing Sensor Profiles....26
4.7 Opening and Closing a Sensor....27
4.8 Initializing a Sensor....27
4.9 Buffer Management....29

5 Specim Sensor Profile....32
5.1 SSP File Structure....32
5.1.1 name....32
5.1.2 description....32
5.1.3 type....32
5.1.4 modules....33
5.1.5 dependencies....33
5.1.6 setup....33
5.1.7 features....34
5.2 Setting up SSP for the First Time....36
5.3 Changing the Feature Values Using SSP and Functions....36

6 Feature Lists....37
6.1 SI_SYSTEM Features....37
6.1.1 DeviceName....37
6.1.2 DeviceDescription....37
6.1.3 DeviceCount....38
6.1.4 Sensor<insert index>.IsOpen....38
6.1.5 ProfilesDirectory....38
6.2 Sensor Features....39

6.2.1 Acquisition.Acquiring....39
6.2.2 Acquisition.CalculatedFrameRate....39
6.2.3 Acquisition.Error....39
6.2.4 Acquisition.FrameCounter....40
6.2.5 Acquisition.FrameCounter.Reset....40
6.2.6 Acquisition.PostStartCallback....40
6.2.7 Acquisition.PostStopCallback....41
6.2.8 Acquisition.PreStartCallback....41
6.2.9 Acquisition.PreStopCallback....42
6.2.10 Acquisition.RingBuffer.Lag....42
6.2.11 Acquisition.RingBuffer.Size....42
6.2.12 Acquisition.RingBuffer.Sync....43
6.2.13 Acquisition.SetupFilePath....43
6.2.14 Acquisition.Start....43
6.2.15 Acquisition.Stop....43
6.2.16 Acquisition.Timeout....44
6.2.17 AcquisitionWindow.Height....44
6.2.18 AcquisitionWindow.Left....45
6.2.19 AcquisitionWindow.Top....45
6.2.20 AcquisitionWindow.Width....46
6.2.21 Camera1.ExposureTime....46
6.2.22 Camera1.FrameRate....47
6.2.23 Camera2.ExposureTime....47
6.2.24 Camera2.FrameRate....47
6.2.25 Camera.AutoNUC....48
6.2.26 Camera.Binning.Average....48
6.2.27 Camera.Binning.Average.IsHardware....48
6.2.28 Camera.Binning.HardwareModes....49
6.2.29 Camera.Binning.Spatial....49
6.2.30 Camera.Binning.Spectral....50
6.2.31 Camera.BitDepth....50
6.2.32 Camera.BPR....50
6.2.33 Camera.BPR.BadPixelCount....51
6.2.34 Camera.BPR.BadPixelPercentage....51
6.2.35 Camera.BPR.CreateMap....51
6.2.36 Camera.BPR.CreateMap.InProgress....52
6.2.37 Camera.BPR.Gain.Max....52
6.2.38 Camera.BPR.Gain.Min....52
6.2.39 Camera.BPR.IsHardware....53
6.2.40 Camera.BPR.MapAvailable....53
6.2.41 Camera.BPR.Noise.Max....53
6.2.42 Camera.BPR.Offset.Max....53
6.2.43 Camera.BPR.Offset.Min....54
6.2.44 Camera.ByteDepth....54
6.2.45 Camera.CalibrationPack....54
6.2.46 Camera.CalibrationPack.IsLoaded....55
6.2.47 Camera.Channel....55
6.2.48 Camera.Channels....56
6.2.49 Camera.CheckTemperatures....56
6.2.50 Camera.CloseShutter....56
6.2.51 Camera.Count....56
6.2.52 Camera.Debug....57
6.2.53 Camera.Debug.FrameNumber....57
6.2.54 Camera.ExposureTime....57
6.2.55 Camera.ExposureTime.Auto....58

6.2.56 Camera.FrameRate....58
6.2.57 Camera.FWHM....58
6.2.58 Camera.Gain.Analog....59
6.2.59 Camera.Gain.Digital....59
6.2.60 Camera.Image.Embedding.Enabled....59
6.2.61 Camera.Image.Embedding.Featurenames....60
6.2.62 Camera.Image.Embedding.Row....60
6.2.63 Camera.Image.Height....60
6.2.64 Camera.Image.IsTestImage....61
6.2.65 Camera.Image.ReadoutMode....61
6.2.66 Camera.Image.ReadoutTime....62
6.2.67 Camera.Image.SizeBytes....62
6.2.68 Camera.Image.SizePixels....62
6.2.69 Camera.Image.Transformation....63
6.2.70 Camera.Image.Width....63
6.2.71 Camera.IsConnected....64
6.2.72 Camera.MaxDataNumber....64
6.2.73 Camera.Model....64
6.2.74 Camera.MROI.Clear....65
6.2.75 Camera.MROI.Enable....65
6.2.76 Camera.MROI.MultibandString....65
6.2.77 Camera.NUC....66
6.2.78 Camera.OpenShutter....66
6.2.79 Camera.Pixel.Height....66
6.2.80 Camera.Pixel.Width....66
6.2.81 Camera.Preprocessing.Enabled....67
6.2.82 Camera.Reset....67
6.2.83 Camera.Sensor.Height....68
6.2.84 Camera.Sensor.Width....68
6.2.85 Camera.Shutter.IsConnected....68
6.2.86 Camera.Shutter.IsOpen....68
6.2.87 Camera.Shutter.IsToggle....69
6.2.88 Camera.Stabilization.Enabled....69
6.2.89 Camera.Stabilization.IsStable....69
6.2.90 Camera.Temperature....70
6.2.91 Camera.Trigger.Mode....70
6.2.92 Camera.Wavelength.Start....70
6.2.93 Camera.Wavelength.Stop....71
6.2.94 Camera.WavelengthTable....71
6.2.95 Dalsa.CameraFile....72
6.2.96 Detector.Image.Height....72
6.2.97 Detector.Image.Width....72
6.2.98 DeviceCount....73
6.2.99 DeviceDescription....73
6.2.100 DeviceName....73
6.2.101 FeatureList....74
6.2.102 Grabber.Channel....74
6.2.103 Grabber.Channels....75
6.2.104 Grabber.PlaybackFile....75
6.2.105 Imperx.Tap1.AnalogOffset....75
6.2.106 Imperx.Tap2.AnalogGain....76
6.2.107 Imperx.Tap2.AnalogOffset....76
6.2.108 Imperx.Tap1.AnalogGain....76
6.2.109 Initialize....77
6.2.110 IsInitialized....77

6.2.111 ProfilesDirectory 77
6.2.112 Scb.Channel 78
6.2.113 Scb.Channels 78
6.2.114 Scb.IsOpen 78
6.2.115 Scb.Temperature.Channel1 79
6.2.116 Scb.Temperature.Channel2 79
6.2.117 Scb.Temperature.Channel3 79
6.2.118 Scb.Temperature.Channel4 80
6.2.119 Sensor.Detector.Type 80
6.2.120 Sensor<insert index>.IsOpen 80
6.2.121 Sensor.SerialNumber 81
6.2.122 Shutter.Channel 81
6.2.123 Shutter.Channels 82
6.2.124 VCECLB.CameraConfig 82
6.2.125 Xenics.CalibrationFile 82
6.2.126 Xenics.CalibrationFolder 83
6.2.127 Xenics.CalibrationOutputFolder 83
6.2.128 Xenics.CalibrationRange 84
6.2.129 Xenics.TargetTemperature 84
6.3 Features per Sensor Profile 85
6.3.1 Fenix Feature List 85
6.3.2 Fenix1k Feature List 87
6.3.3 Fenix1k with Epix Feature List 90
6.3.4 Filereader Feature List 93
6.3.5 FX10 and FX17 Feature List 95
6.3.6 FX50 Feature List 97
6.3.7 FX120 Feature List 100
6.3.8 GX17 Feature List 102
6.3.9 Kestrel10 with Epix Feature List 104
6.3.10 Kestrel16 with Epix Feature List 106
6.3.11 Kestrel16 with NI Feature List 109
6.3.12 LWIR with NI Feature List 111
6.3.13 MWIR Flir with GigE Feature List 113
6.3.14 MWIR Flir with NI Feature List 115
6.3.15 OWL3 with Dalsa Feature List 118
6.3.16 OWL3 with NI Feature List 120
6.3.17 RGB with NI Feature List 123
6.3.18 RHNIR SCB with Framelink Feature List 125
6.3.19 RHNIR SCB with NI Feature List 127
6.3.20 RHNIR with NI Feature List 130
6.3.21 SWIR3 with Framelink Feature List 132
6.3.22 SWIR3 with NI Feature List 135
6.3.23 SWIR with Framelink Feature List 138
6.3.24 SWIR with NI Feature List 140
6.3.25 Universal with NI Feature List 143
6.3.26 VLNIR SCB with NI Feature List 145
6.3.27 VLNIR with NI Feature List 147
6.3.28 VNIR Basler GigE with NI Feature List 150
6.3.29 VNIR HS SCSB with Framelink Feature List 152
6.3.30 VNIR HS SCB with NI Feature List 154
6.3.31 VNIR HS with Framelink Feature List 157
6.3.32 VNIR HS with NI Feature List 160
6.3.33 VNIR PFD 2080 Custom 1x1 with NI Feature List 162
6.3.34 VNIR PFD 2080 Custom 3x3 with NI Feature List 164
6.3.35 VNIR PFD 2080 SCB with NI Feature List 167

6.3.36 VNIR PFD 2080 with NI Feature List....169
6.3.37 VNIR PFD GigE Feature List....171
6.3.38 VNIR PFD SCB with Framelink Feature List....174
6.3.39 VNIR PFD SCB with NI Feature List....176
6.3.40 VNIR PFD with Framelink Feature List....179
6.3.41 VNIR PFD with NI Feature List....181
6.3.42 VNIR sCMOS with Bitflow Feature List....183
6.3.43 XLNIR Feature List....186

7 Functions....189
7.1 Function List....189
7.2 SI_Load....190
7.3 SI_Unload....190
7.4 SI_Open....190
7.5 SI_Close....190
7.6 SI_IsImplemented....190
7.7 SI_IsReadOnly....190
7.8 SI_IsImplemented....191
7.9 SI_IsWritable....191
7.10 SI_IsReadable....191
7.11 SI_UnregisterFeatureCallback....191
7.12 SI_SetInt....191
7.13 SI_GetInt....192
7.14 SI_GetIntMax....192
7.15 SI_GetIntMin....192
7.16 SI_SetFloat....192
7.17 SI_GetFloat....192
7.18 SI_GetFloatMax....193
7.19 SI_GetFloatMin....193
7.20 SI_SetBool....193
7.21 SI_GetBool....193
7.22 SI_Command....193
7.23 SI_SetString....193
7.24 SI_GetString....194
7.25 SI_GetStringMaxLength....194
7.26 SI_GetEnumStringMaxLength....194
7.27 SI_SetEnumIndex....194
7.28 SI_SetEnumIndexByString....195
7.29 SI_SetEnumString....195
7.30 SI_GetEnumIndex....195
7.31 SI_GetEnumCount....195
7.32 SI_GetEnumStringByIndex....195
7.33 SI_IsEnumIndexAvailable....195
7.34 SI_IsEnumIndexImplemented....196
7.35 SI_Wait....196
7.36 SI_CreateBuffer (Deprecated)....196
7.37 SI_DisposeBuffer (Deprecated)....197
7.38 SI_RegisterDataCallback....197
7.39 SI_UnregisterDataCallback....197
7.40 SI_GetFeatureType....197

8 Function Return Codes....199
8.1 Generic Error Codes....199

8.2 File Error Codes....200
8.3 XML Error Codes....200
8.4 Sensor Profile Error Codes....200
8.5 Feature Error Codes....201
8.6 Command Error Codes....202
8.7 Module Error Codes....202
8.8 Acquisition Error Codes....203
8.9 Header File Error Codes....203
8.10 Buffering Error Codes....204
8.11 Communication Error Codes....204
8.12 Camera Error Codes....204
8.13 SCB Error Codes....205
8.14 Licensing Error Codes (depracated)....205
8.15 Generic Warning Codes....206
8.16 Sensor Profile Warning Codes....206
8.17 Feature Warning Codes....206
8.18 Module Warning Codes....207
8.19 Buffering Warning Codes....207
8.20 Calibration Warning Codes....207

9 Code Examples....208
9.1 Enumerating Through Detected Devices....208
9.2 Basic Set and Get Functionality and Waiting for Frames....209
9.3 Feature Callbacks and Data Callback....211

10 Appendix: Multiple Regions of Interest....214
10.1 Example: Creating a Single Region of Interest....214
10.2 Example: Creating Multiple Regions of Interest....214

11 Change History....215

1 Legal Information

Certification

Specim, Spectral Imaging Ltd has developed Quality Management System for its use covering design, development, manufacturing, sales and support of optical measurement devices. QMS follows ISO9001:2015 and enables more efficient operation and product management in a systematic way utilizing metrics. Specim QMS has been audited by Bureau Veritas Certification Holding SAS – UK Branch and found to be in accordance with the requirements of the management system standards.

Disclaimer

All information provided in this guide and provided manuals is believed to be complete, accurate and reliable at the time of delivery. No responsibility is assumed by Specim, Spectral Imaging Ltd. for its use. Specim, Spectral Imaging Ltd reserves the right to make changes to this information without notice. Reproduction of this manual in whole or in part, by any means, is prohibited without prior permission having been obtained from Specim, Spectral Imaging Ltd.

Contacting Specim

Further information and technical support are available from **Specim, Spectral Imaging Ltd.** in Finland. Contact information:

- Community: www.specim.com/community
- WWW: www.specim.com

2 API Overview

The API is build with simplicity in mind. Several design options have been made to ensure backwards compatibility, scalability and the ease of use from both the end-user and the developer point of view.

The exported functions that the Lumo Sensor SDK provides are generic functions to handle generic types of data (integer, boolean, floating-point, string, enumeration, command) and the features of different sensors, frame rate for example, have been hidden as text parameters to each of these generic functions. In practice this means that instead of having a `SetFrameRate(float FrameRate)` function to change the frame rate we will have a `SI_SetFloat` function where we pass the feature name "Camera.FrameRate" and the floating-point value for the frame rate. That same function will work for any other floating-point value that we want to set, such as the exposure time.

All of these generic types also provide a `get` function which is used to get the values of the desired features. With this design model we are able to simplify the API dramatically and we do not have to change it whenever new features arrive – we just add those features to the feature list and you will be able to use them straight up without breaking the existing code you have written.

The feature-based system allows the addition of new features or providing a different subset of features for different sensors in the future while still using the same API calls. The design model was chosen also to hide any binary structures from the upper application levels and handle them inside the API.

The outcome for the developers using this design is that they do not have to care about how the cameras are individually controlled; they will simply use the available feature subsets to control the selected sensors.

All of the features have a name and a type.

The different feature types are:

- `string` for strings. String features are those features that can be represented using a text string of one or more characters. `SI_WC (wchar_t)` built-in type is used for those features.
- `int` for integers. Integer features are those features that can be represented by a single integer value. `SI_64 (long long)` built-in type is used for those features.
- `bool` for boolean values. Boolean features can only have two possible values; the logical states true and false. `SI_BOOL (bool)` built-in type is used for those features.
- `command`. This is a feature which does not hold any value but will run the command specified by its name. Command features represent a single action.
- `enum` for enumeration. Enumeration acts as a vector of strings. They are used to represent those features that can be assigned one value from a set of possible options.
- `signal`. A signal feature does not hold any value. Signal features act as events. When a signal gets triggered by the application, the feature callback will be triggered.

All features support callback functionality where any changes to values of the features can be caught by user-defined callback functions. It is possible to set one or multiple callbacks to a feature. It is the developer's responsibility to return from the feature's callback as soon as possible not to impact the timing of the Lumo Sensor SDK. From a callback one can query the features properties but it is crucial to keep in mind not to modify the feature from the callback itself or lockup situations will arise.

Incoming acquisition data can be caught by two different methods:

- Calling the `SI_Wait` function to wait a certain amount of time for a frame to arrive.
- Registering a data callback function to be called every time a new frame arrives.

If a data callback function is registered, SI_Wait function will be disabled.

Finally, string handling in Lumo Sensor SDK uses 16-bit wide characters defined as `SI_WC`.

3 Initial Setup

The Qt runtime library is required to use the Specim software. These libraries are installed on the operating system level. Since they are named using Specim-infix (e.g. Qt5CoreSpecim.dll), they don't interfere with other possible Qt runtime installations on the same computer.

The Qt runtime DLLs are copied to the Windows system folder and plugins to a Specim -folder below that:

- Qt runtime DLLs for 32-bit compilation: `C:\Windows\SysWOW64`
- Qt plugin DLLs for 32-bit compilation: `C:\Windows\SysWOW64\Specim`
- Qt runtime DLLs for 64-bit compilation: `C:\Windows\System32`
- Qt plugin DLLs for 64-bit compilation: `C:\Windows\System32\Specim`

Lumo Sensor SDK makes use of sensor profile files having an `*.ssp` extension (Specim Sensor Profile). Those profiles hold the sensor definitions: the supported features, default channels for different devices, regions of interest, and so on. These SSP files may need to be updated to add new features, change the default parameters, and so on. Contact Specim should you have any questions.

Lumo Sensor SDK will enumerate all the SSP files, and check for XML syntax errors when `SI_Load` is called. If no errors are found, the sensor defined in the SSP file will be added to the `DeviceName` and `DeviceDescription` features. The sensor handle can be opened using the index in the `DeviceName` feature. See example 1 in the SDK.

On Windows 10 and onwards, the SSPs are copied to `C:\Users\Public\Documents\Specim`.

3.1 Installing Lumo Sensor SDK

This section walks you through the Lumo Sensor SDK installation wizard.

Proceed as follows:

1. Open the Lumo Sensor SDK installation wizard.

Figure 1: Lumo Sensor SDK Installation Wizard

2. Press the **Next** button.
3. Accept the license agreement.

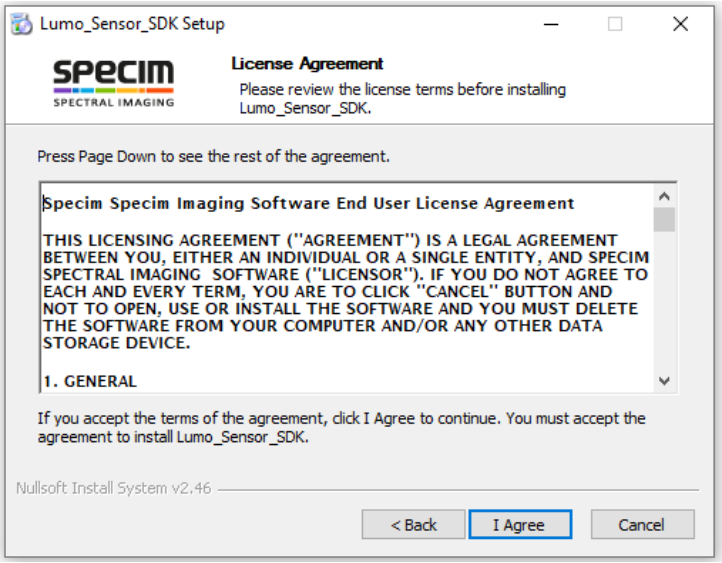

Figure 2: Accepting the License

4. Select the components to install.

Note: Select and install the components for the devices connected to the computer ONLY. Do not install unused camera profiles.

Note: Components called **Spectral Camera FX10e** and **Spectral Camera FX17e** are unlicensed beta-versions.

Note: FX Camera profiles with Pleora eBUS require a separate Pleora license.

Keep the **Qt runtime library** component checked, if the Qt runtime libraries have not yet been installed.

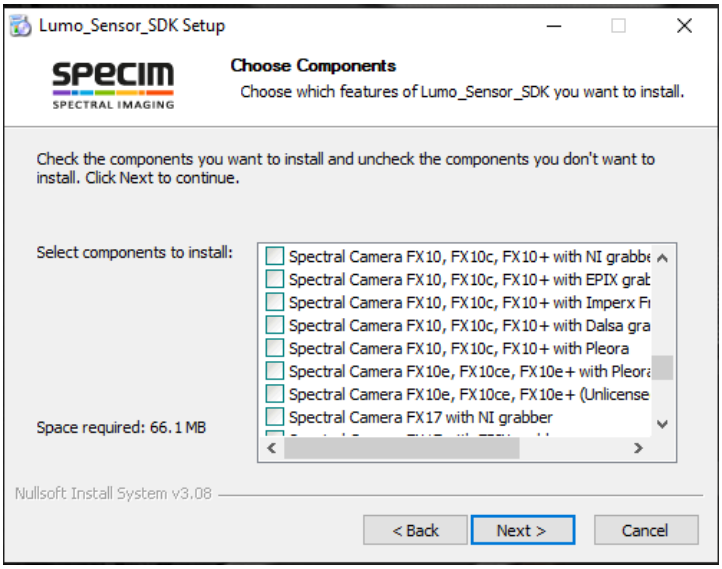

Figure 3: Selecting the Components

5. Press the **Next** button.
6. Choose the start menu folder.

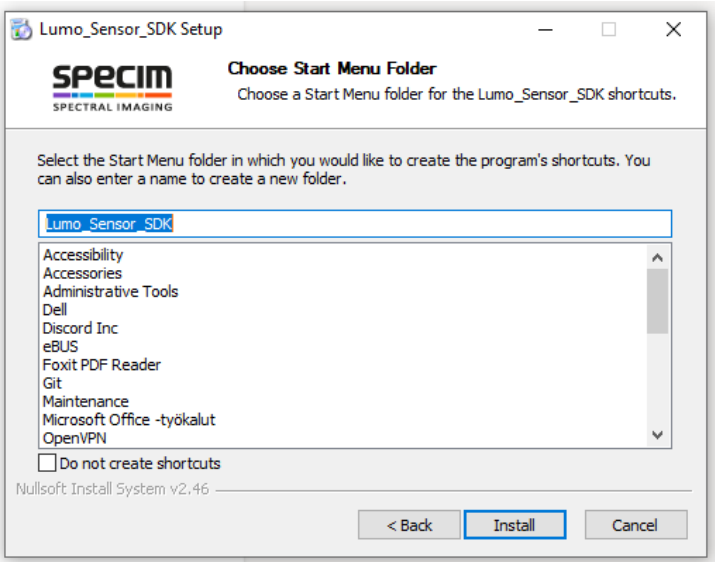

Figure 4: Choosing the Start Menu Folder

7. Wait while Lumo Sensor SDK is being installed.

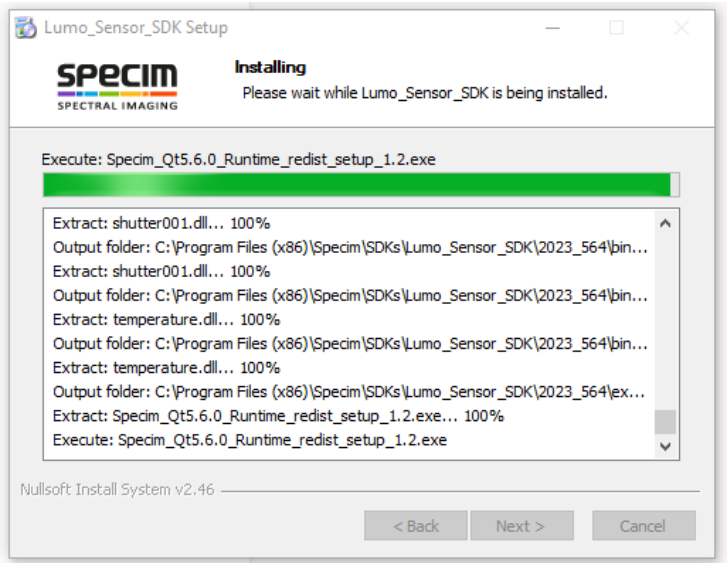

Figure 5: Installing Software Components

8. The Lumo Sensor SDK contains an embedded Qt runtime installer, which is launched at this phase.

   The Qt installation wizard is opened.

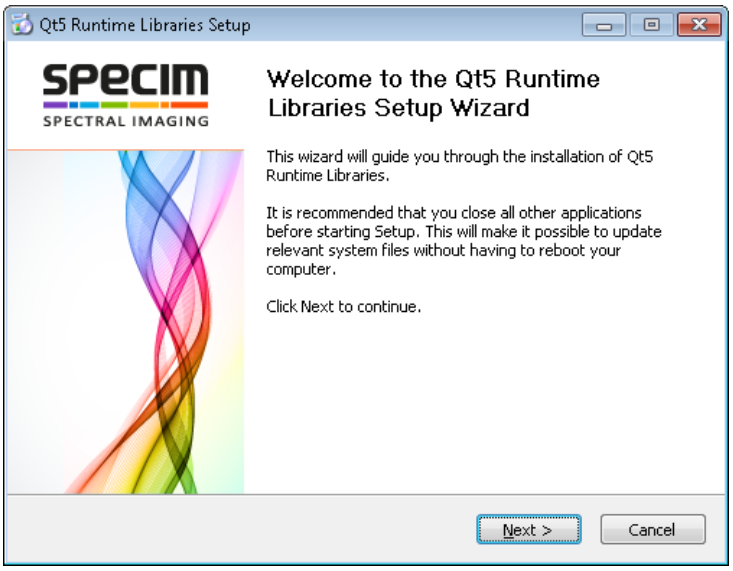

Figure 6: Qt Installation Wizard

9. Select **Next**.

   The license screen is opened.

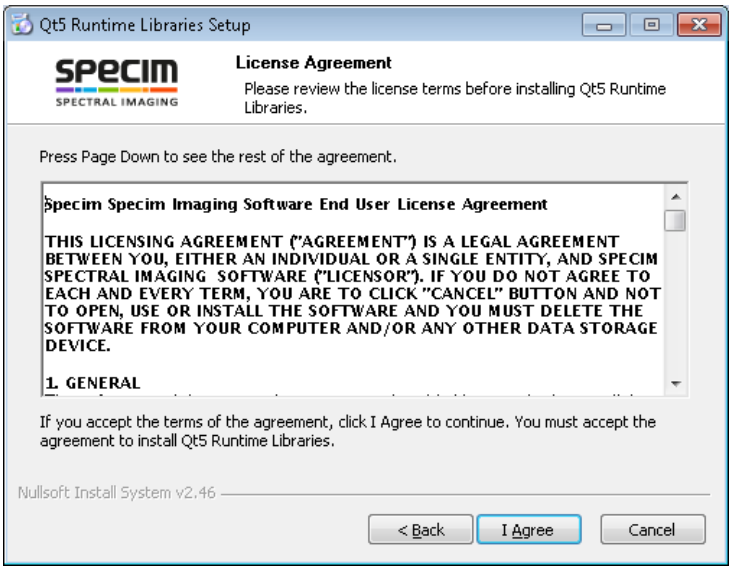

Figure 7: Qt License Screen

10. Accept the license by selecting **I Agree**.

    The component selection screen is opened.

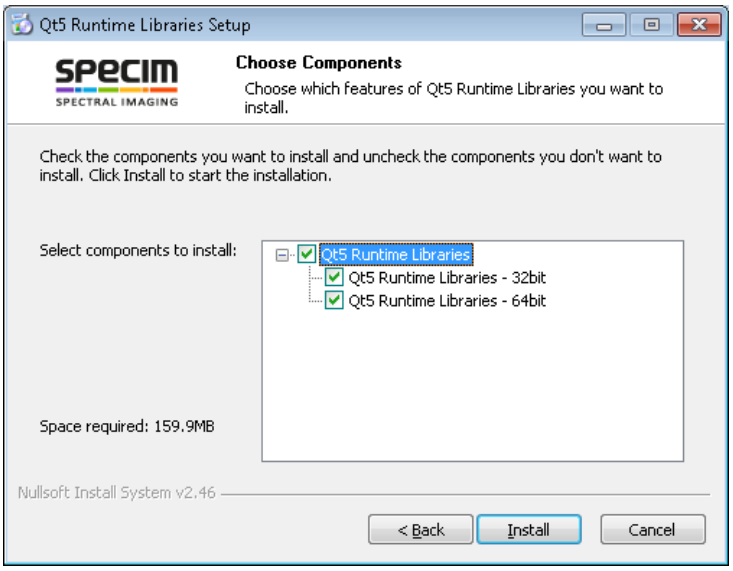

Figure 8: Qt Component Selection Screen

Choose the components to be installed. We recommend using the default settings.

11. Select **Install**.

The installation progress screen is opened.

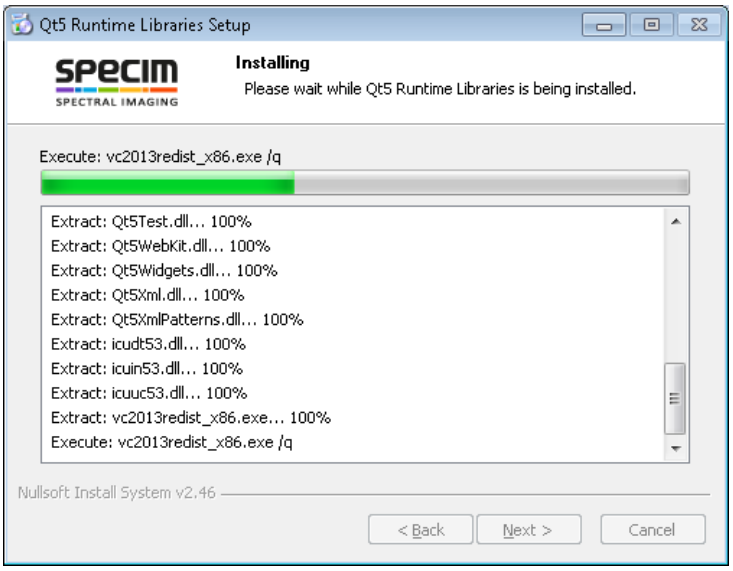

Figure 9: Qt Installation Progress Screen

12. After a successful installation, the wizard prompts you to reboot the computer.

The installer declares some environment variables at the system scope, thus we recommend rebooting the computer to insure the new settings are properly propagated.

Reboot the computer by selecting the **Reboot now** radio button and selecting **Finish**.

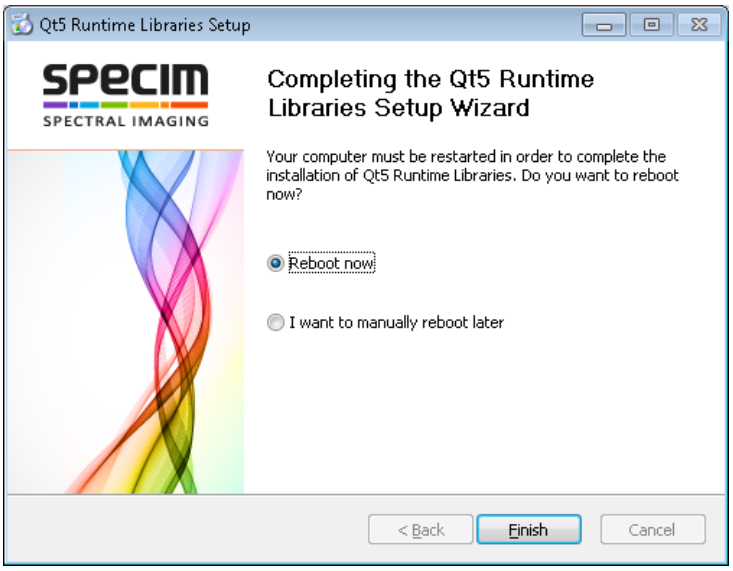

Figure 10: Completing the Qt Installation

13. The wizard comes to its end.

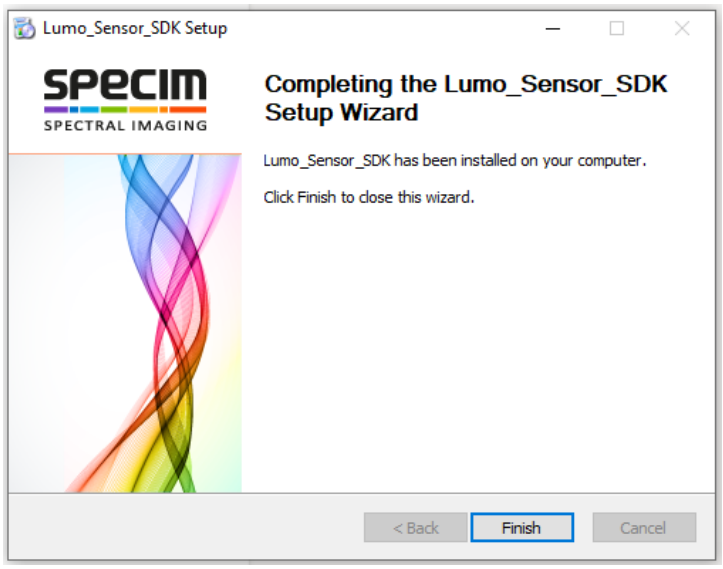

Figure 11: Wizard Exits

14. Press the **Finish** button.
15. The installation consists of the folders below:

- The `bin` folder contains the binaries for a 32-bit (Win32 subfolder) platform, and 64-bit platform (x64 subfolder) if the installer detected a 64-bit operating system.
- The `doc` folder contains the Lumo Sensor SDK documentation files.
- The `examples` folder contains code examples on how to use the Lumo Sensor SDK.
- The `external` folder contains the external hardware configuration files.
- The `include` folder contains the header files required for the SDK.
- The `profiles` folder contains all the sensor profile files that were selected to be installed.

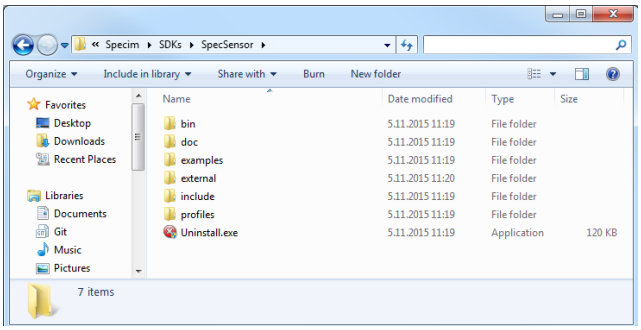

Figure 12: Installation Folder

3.2 Pleora eBUS Software Development Kit

When you use a GigE camera together with Lumo, the system requires a Pleora eBUS SDK installed on the data acquisition PC.

Note: You need to acquire a license for the Pleora eBUS SDK directly from Pleora Technologies.

See the table below for compatibility details.

Table 1: Lumo and Pleora eBUS SDK compatibility

Lumo	Pleora eBUS SDK version
Lumo 2019 or older	4.1.7.3988
Lumo 2020 or newer	5.1.10.4642

Note: Pleora eBUS version 6 is not supported.

3.3 Frame Grabber Installation

To use Lumo software with a frame grabber, you must install the correct version of the software package related to your grabber. If you purchased the frame grabber through Specim, the required drivers are included in the delivery.

These grabbers are currently compatible with Lumo software:

- NI-1427 EOL (SWIR3, FX10, FX17)
- NI-1429 EOL (FX10, FX17)
- NI-1430 EOL (FENIX, FX10, FX17, PFD4K/2080)
- NI-1433 (SWIR3, FX10, FX17, PFD4K/2080)
- NI-1437 (FX10, FX17)
- EPIX E4 (SWIR3, FX10, FX17)
- EPIX EB1 (KESTREL10, KESTREL16)
- Bitflow (sCMOS)
- DALSA X64 Xcelera (OWL)
- Dalsa Xtium CL MX4 EOL (FX10, FX17)
- Dalsa Xtium2-CL MX4 (FX10, FX17, GX17)

Visit the frame grabber manufacturer's website for instructions.

3.4 Configuring the FileReader

This section describes how to configure the FileReader sensor mockup for playing back a previously captured dataset.

Proceed as follows:

1. Open the FileReader SSP in a text editor.
2. Search for the Grabber.PlaybackFile feature configuration within the sensor profile.

   Tip: Use the *Find/Search* feature in the editor to look for `Grabber.PlaybackFile`.

   The feature configuration will look like this:

```
<feature>
    <name>Grabber.PlaybackFile</name>
    <module>grabber</module>
    <type>string</type>
    <value></value>
</feature>
```

3. Insert the absolute file path of a valid `*.raw` file inside the `<value>` tags.

   An example configuration might look like this:

```
<feature>
    <name>Grabber.PlaybackFile</name>
    <module>grabber</module>
    <type>string</type>
    <value>D:\data\example_dataset\capture\example.raw</value>
</feature>
```

4. Save changes to the SSP.

   Note: If the sensor profile is already loaded by an application, restart the application for the changes to take effect.

3.5 Installing USB Serial Devices

This section describes how to install any of the Specim USB serial devices, such as SCB.

Proceed as follows:

1. Connect the USB serial device to the computer.

   A Windows tray notification message, signalling driver installation, appears.

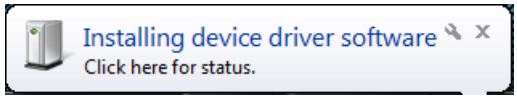

Figure 13: Installation Message

2. Wait for the installation to proceed.

   If Windows Update is enabled, a driver will be automatically installed, and the tray notification message below will appear.

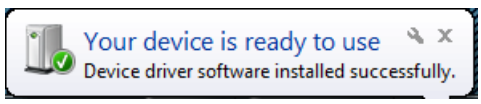

Figure 14: Device Ready Message

If the driver cannot be installed automatically, proceed to step 4.

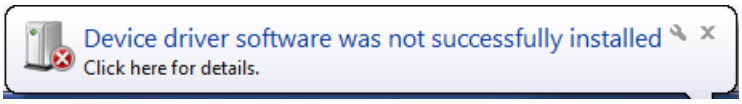

Figure 15: Installation Failed Message

3. Make sure the provided FTDI driver package (*CDM 2.04.16 WHQL Certified* or similar) is available.

   Extract the driver files on the target computer file system.

4. Open the Windows Device Manager, and locate the devices to be installed.

   Tip: Type *device manager* in the search field in **Start Menu** to locate the Device Manager shortcut.

   Tip: The devices to be installed will be marked with a yellow warning sign, and usually appear under **Other devices**.

   Right click the first device, to open the drop-down menu.

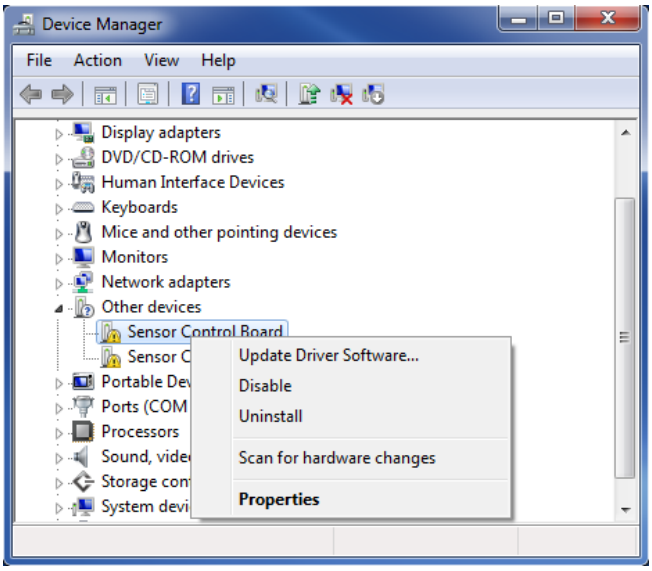

Figure 16: Device Manager

5. Select **Update Driver Software...**.

   The driver search selection dialog is opened.

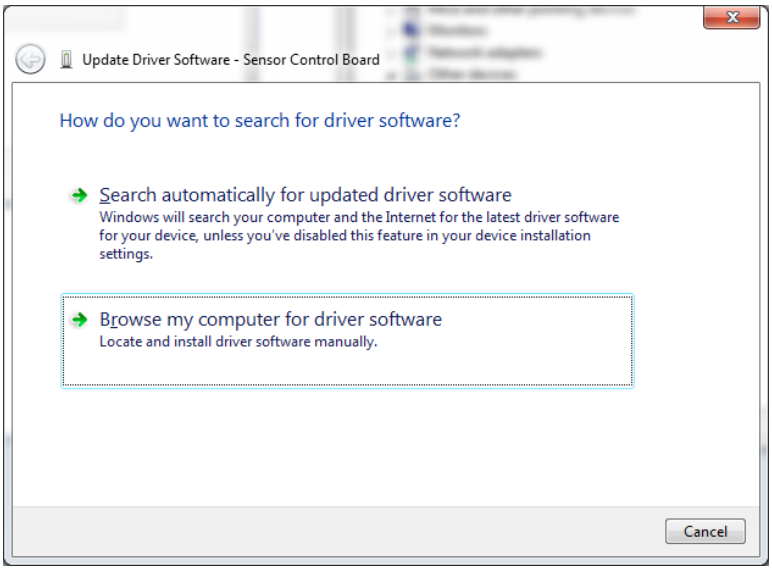

Figure 17: Driver Search Selection

6. Select **Browse my computer for driver software**.

   The driver software browse screen is opened.

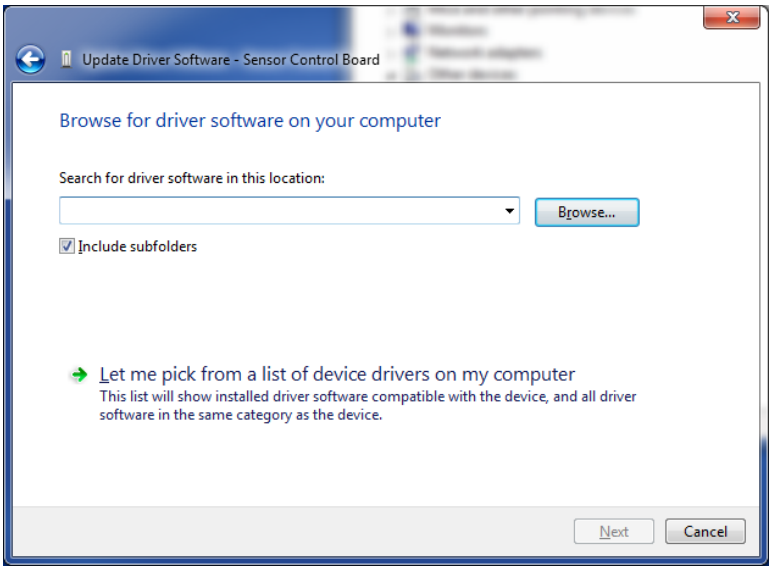

Figure 18: Driver Browse Screen

7. Select **Browse...**.

   A file browser dialog is opened.

   Locate and select the folder containing the provided drivers.

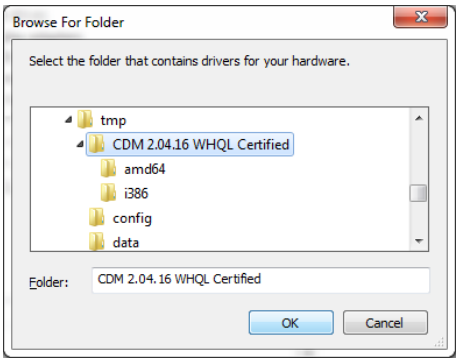

Figure 19: File Browser Dialog

8. Select **OK**.

The driver browse screen is shown.

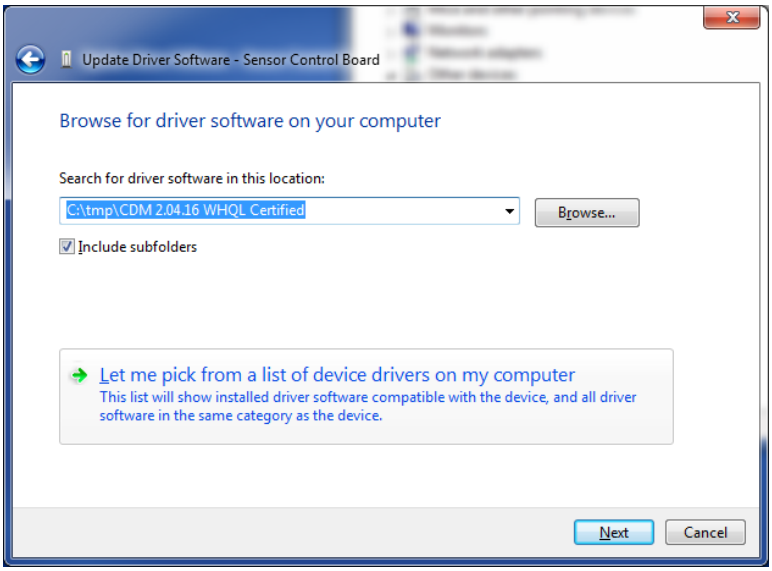

Figure 20: Driver Browse Screen with Selected Location

9. Select **Next**.

The device driver is installed, and the installation complete screen is opened.

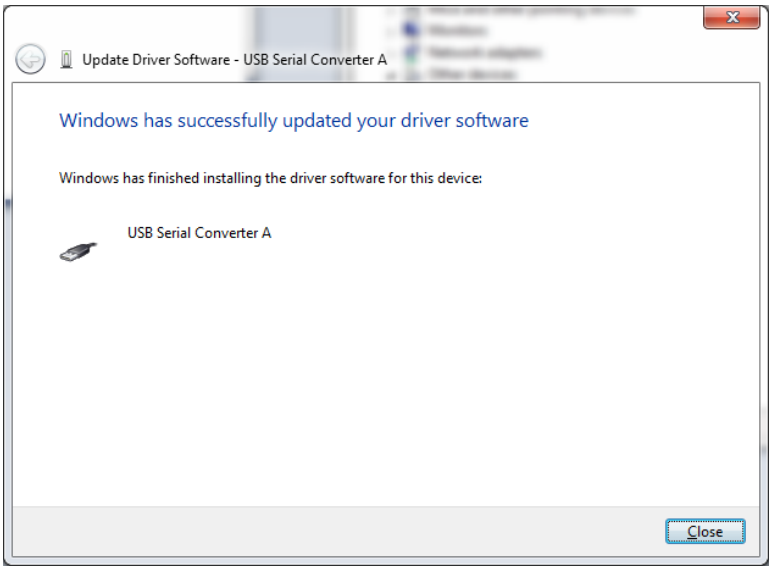

Figure 21: Installation Complete

10. Repeat Steps 4, 5, 6, 7, 8, 9 for all devices with a warning sign.

USB serial ports are visible, and no devices with warning signs remain in Device Manager.

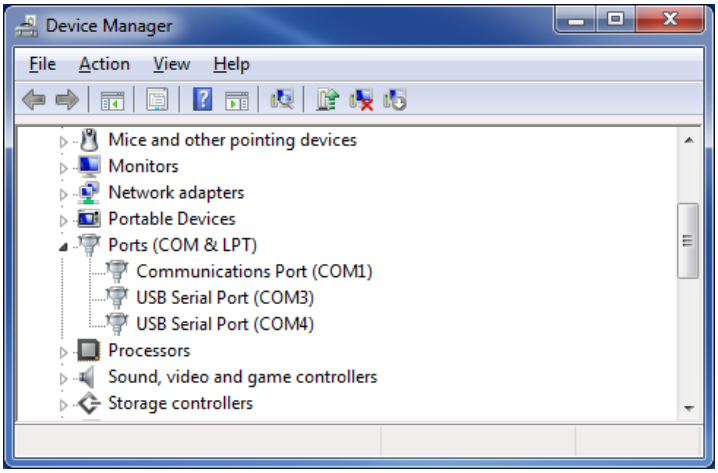

Figure 22: All Devices Installed

4 Walkthrough

4.1 Including Lumo Sensor SDK

You have to include the header files below in your project:

- `SI_sensor.h` - contains the all the Lumo Sensor SDK exported functions.
- `SI_types.h` - defines all used data types and callback function signatures.
- `SI_errors.h` - contains all the Lumo Sensor SDK error codes.

The headers are located in the `include` directory within the Lumo Sensor SDK installation folder.

Tip: We recommend adding the Lumo Sensor SDK include directory in your project's include path.

4.2 Linking Lumo Sensor SDK

You must link Lumo Sensor SDK into your application to make the functions available.

Link Lumo Sensor SDK to the project. There are two different ways to link it:

- Using the provided import library (`SpecSensor.lib`). This is the easiest and the recommended way.
- Manually resolving the functions, using the function signatures in the header file (`SI_sensor.h`).

4.2.1 Static Linking

This section describes how to statically link the Lumo Sensor SDK library.

There are two ways to statically link the Lumo Sensor SDK library:

- From the IDE's project properties.
- Inline from the code.

Proceed as follows:

1. Ensure that the folder where `SpecSensor.lib` is located, is added into the library directories list, or the linker cannot find the library.
2. From IDE's project properties
   a. Open your IDE project properties.
   b. Find the linker section.
   c. Find the **Additional Depencies**.
   d. Add `SpecSensor.lib` to these dependencies.
3. Inline from the code.

   Add the following line.

```
#pragma comment( lib, "SpecSensor" )
```

4.2.2 Dynamic Linking

This section describes how to dynamically link Lumo Sensor SDK library.

Dynamically linking and resolving the library symbols provide more controls to the application on how and when to load and map the symbols. It also ensures complete binary isolation, as the linking library is not used.

The following example shows how to load Lumo Sensor SDK and map some functions from C++, using the Qt libraries. Other frameworks use different functions, but the principle remains the same.

```
#include "SI_errors.h"
#include <QLibrary>

// Define signature types for symbols
typedef int (SI_IMPEXP_CONV* LOAD_LICENSE_FUNCT)(const SI_WC*);
typedef int (SI_IMPEXP_CONV* OPEN_DEVICE_FUNCT)(const int, SI_H*);
typedef int (SI_IMPEXP_CONV* CLOSE_DEVICE_FUNCT)(SI_H);
typedef int (SI_IMPEXP_CONV* UNLOAD_FUNCT)();

// Create function pointers
LOAD_LICENSE_FUNCT  pLoad;
OPEN_DEVICE_FUNCT   pOpen;
VOID_FUNCT          pClose;
UNLOAD_FUNCT  pUnload;

// Initialize pointers to 0
unmapSymbols()

// Create a QLibrary instance for library module
QLibrary library = QLibrary("SDK");

// Load library module and map the symbols
bool loadSDK()
{
    if (library.load())
    {
        mapSymbols();
        return true;
    }

 return false;
}

// Resolve each symbols and assign them to the function pointers
void mapSymbols()
{
    m_pLoad = (LOAD_LICENSE_FUNCT) library.resolve("SI_Load");
    m_pOpen = (LOAD_LICENSE_FUNCT) library.resolve("SI_Open");
    m_pClose = (LOAD_LICENSE_FUNCT) library.resolve("SI_Close");
    m_pUnload = (LOAD_LICENSE_FUNCT) library.resolve("SI_Unload");
}

// Wrapper function for load taking care of checking is SI_Load was resolved
int load(const SI_WC* _szPath)
{
    if(pLoad)
    {
        return pLoad(_szPath);
    }

    return siModuleUnresolvedSymbol;
}


// Wrapper function for load taking care of checking is SI_Load was SI_Open
int open(const int _nDeviceIndex, SI_H* _pHandle)
{
 if(pOpen)
```

```
 {
  return pOpen(_nDeviceIndex, _pHandle);
 }

 return siModuleUnresolvedSymbol;
}

// Wrapper function for load taking care of checking is SI_Close was SI_Open
int close(const int _nDeviceIndex, SI_H _pHandle)
{
 if(pClose)
 {
  return pClose(_pHandle);
 }

 return siModuleUnresolvedSymbol;
}

// Wrapper function for load taking care of checking is SI_Close was SI_Unload
void unload()
{
 if(m_pUnload)
 {
  m_pUnload();
 }
}

// Unload library module and unmap the symbols
bool unloadSDK()
{
    library.unload();
    unmapSymbols();
}

// Reset pointers to 0
void unmapSymbols()
{
 m_pLoad = 0;
 m_pOpen = 0;
 m_pClose = 0;
    pUnload = 0;
}
```

Proceed as follows:

1. Include `SI_errors.h`.
2. Include **QLibrary**.
3. Define signature types for symbols.
4. Create function pointers.
5. Initialize pointers to 0.
6. Create a **QLibrary** instance for the library module.
7. Load the library module and map the symbols.
8. Resolve each symbol and assign them to the function pointers.
9. Use the wrapper functions: `load`, `open`, `close`.
10. Unload the library module and unmap the symbols.

4.3 Copying the Dependency Binaries

The Lumo Sensor SDK binaries must be visible in the search path of your application, or the module loading will fail.

The binaries are Qt and Lumo Sensor SDK DLLs. They are installed in:

- Qt runtime DLLs for 32-bit: `C:\Windows\SysWOW64`
- Qt plugin DLLs for 32-bit: `C:\Windows\SysWOW64\Specim`
- Qt runtime DLLs for 64-bit: `C:\Windows\System32`
- Qt plugin DLLs for 64-bit: `C:\Windows\System32\Specim`
- Lumo Sensor SDK for 32-bit: `C:\Program Files (x86)\Specim\SDKs\Lumo_Sensor_SDK\<year_version>\bin\Win32`.
- Lumo Sensor SDK for 64-bit: `C:\Program Files (x86)\Specim\SDKs\Lumo_Sensor_SDK\<year_version>\bin\x64`.

We recommend copying the Lumo Sensor SDK binaries into your application folder. The easiest way to distribute the Qt runtime DLLs (Qt5*Specim.dll) and the plugin DLLs is to run the Specim Qt5 runtime redistributable installer in your delivery to install the binaries in the correct places.

If you want to distribute the Qt dependency binaries in another way, it's also possible to copy the Qt runtime DLLs to other locations that are searched by your application to load binaries, such as the application folder. The Qt plugins are searched by default from the Specim-folder below the Windows system folder. In case you want to deliver those binaries in your application to some other location, Lumo Sensor SDK can be configured to load the plugins from any path you specify. Note that this call has to be made before calling SI_Load:

```
SI_CHK(SI_SetString(Handle, L"QtLibraryPaths", L"C:\path\to\the\folder"));
```

Tip: In Visual Studio, and other IDEs, you can execute a post build events. Add an `xcopy` command to copy the binaries automatically to the output folder.

4.4 Loading and Unloading

The first function call to Lumo Sensor SDK must be SI_Load:

```
SI_CHK(SI_Load(L""));
```

`SI_Load` enumerates all the available sensor profiles SSP. If the function call succeeds, you can start using Lumo Sensor SDK.

The sensor profiles are located in:

- Windows 10 and later: `C:\Users\Public\Documents\Specim`

The last function call to Lumo Sensor SDK must be SI_Unload:

```
SI_CHK(SI_Unload());
```

`SI_Unload` will release and cleanup all the resources used by Lumo Sensor SDK .

Note: Not calling `SI_Unload` will result in memory leaks and hardware handles not being freed.

4.5 Error Checking

Every function, except `SI_GetErrorString`, will return an integer error code.

Tip: We recommend you to check each return code before executing the next statement.

Lumo Sensor SDK provides four macros to ease the error checking:

- `SI_CHK(SI_Function)` - sets `nError` to be the return value of the `SI_Function`. If the return value is negative, the code will then jump to, and continue executing, the code below the `Error:` label.
- `SI_FAILED(Value)` - returns true if `Value` is negative, otherwise false.
- `SI_SUCCEEDED(Value)` - returns true if `Value` is positive or 0, otherwise false.
- `SI_GOTO(Value)` - sets `nError` to `Value` and jumps to the `Error:` label.

This code example demonstrates the use of these macros:

```
int nError = siNoError;
SI_H Handle = 0;
SI_CHK(SI_Load(L""));
SI_CHK(SI_Open(0, &Handle));

if(!SI_SUCCEEDED(SI_SetFloat(Handle, L"Camera.FrameRate", 100000.0)))
{
  SI_GOTO(siOutOfRange);
}
Error:
if (SI_FAILED(nError))
{
  SI_WC ErrorString[100];
  ErrorString = SI_GetErrorString(nError);
  //Print the ErrorString.
}

return nError;
```

`nError` must always be declared on the top of the function. This variable must always be set to the return value of the `SI_*` functions. It is initialized to `siNoError`.

If the `SI_Load` function call fails, the code will jump to the `Error:` label. Any code between `SI_Load` and `Error` will be skipped.

If all the functions return successfully without any errors, the return value will remain `siNoError`.

If an error occurs, `SI_GetErrorString` gives you a text representation of the error code.

4.5.1 Debug and Warning Message Logging

By design, Lumo Sensor SDK reports debug and warning messages through the standard output and also stores the same information to a time stamped log file. Log files are located in `C:\ProgramData\Specim`.

4.6 Listing Sensor Profiles

The code below illustrates how to enumerate all the device names and descriptions. This is the right way of detecting the device index you need to connect to. This index is then later used for Opening and Closing a Sensor.

```
...
SI_64 nDeviceCount = 0;
SI_WC szDeviceName[4096];
SI_WC szDeviceDescription[4096];

SI_CHK(SI_Load(L""));
```

```
SI_CHK(SI_GetInt(SI_SYSTEM, L"DeviceCount", &nDeviceCount));

for (int n = 0; n < nDeviceCount; n++)
{
 SI_CHK(SI_GetEnumStringByIndex(SI_SYSTEM, L"DeviceName", n, szDeviceName,
 4096));
 SI_CHK(SI_GetEnumStringByIndex(SI_SYSTEM, L"DeviceDescription", n,
 szDeviceDescription, 4096));
 //Store the names and descriptions to and display them
}
...
```

DeviceCount, DeviceName and DeviceDescription features are accessed using the `SI_SYSTEM` handle, since these features do not belong to a specific device.

Remember: `SI_Load` must be the first API call, otherwise all the feature calls will fail.

4.7 Opening and Closing a Sensor

Sensor-specific features become available after a successful call to SI_Open:

```
SI_H Handle = 0;
SI_CHK(SI_Open(0, &Handle));
```

Note: `SI_Open` function dynamically loads the sensor-specific modules. You will get errors if some modules cannot be found, or if the license is invalid.

Tip: For debugging module loading issues, download http://download.sysinternals.com/files/ProcessMonitor.zip.

After opening a sensor with `SI_Open`, it can be closed with a call to `SI_Close`.

All opened sensors should be closed with SI_Close before calling SI_Unload.

```
SI_CHK(SI_Close(Handle));
SI_CHK(SI_Unload());
```

4.8 Initializing a Sensor

You must initialize your sensor to gain access to it and acquire data:

```
SI_H Handle = 0;
SI_CHK(SI_Load(L""));
SI_CHK(SI_Open(0, Handle));
SI_CHK(SI_Command(Handle, L"Initialize"));
```

The Initialize command feature is used together with SI_Command.

Warning:

The initialization phase requires certain features to be set to correct values, or it will fail.

The features below must be set to correct values, to successfully intitialize your sensor:

- All hardware channel features available for your sensor:

  Note:

  These features must only be set, if the `modules` section in the SSP defines that specific hardware must be used in the sensor. If not, the `channel` feature can be ignored. See section Specim Sensor Profile for more information on the `modules` section.

  - `Grabber.Channel` (for all physical grabbers)
  - `Grabber.PlaybackFile` (for file replay)
  - `Camera.Channel`
  - `Shutter.Channel`
  - `GPS.Channel`
  - `Scb.Channel`
  - `Temperature.Channel`
- Windowing features:
  - If the calibration pack is provided:
    - `Camera.CalibrationPack`
  - Otherwise
    - `AcquisitionWindow.Left`
    - `AcquisitionWindow.Top`
    - `AcquisitionWindow.Width`
    - `AcquisitionWindow.Height`
- `Camera.Shutter.IsToggle`

The most important features to be set before calling `Initialize` are the channel strings (`Grabber.Channel`, `Camera.Channel`, `Shutter.Channel`, and so on), for each device in use. Normally, the devices used for a sensor include a grabber, a camera and a shutter, but these can differ between sensors. The devices used for a sensor are defined in the SSP `modules` section. See the SSP configuration section for more information.

These features can either be set straight from the SSP file pre-hand, or by calling Lumo Sensor SDK functions in between opening and initializing a sensor.

Important:

It is highly recommended to set these features from the SSP.

If the values have been set pre-hand from the SSP, the only thing you need to do after opening the device is to call the command feature `Initialize`.

If our SSP looks like:

```
<modules>
    <module type="grabber">grab001</module>
    <module type="camera">cam001</module>
    <module type="shutter">shutter001</module>
</modules>

...
...


<feature>
      <name>Grabber.Channel</name>
      <module>grabber</module>
      <type>string</type>
```

```
      <value>img0</value>
</feature>

<feature>
      <name>Camera.Channel</name>
      <type>string</type>
      <module>camera</module>
      <value>port0</value>
</feature>

<feature>
      <name>Shutter.Channel</name>
      <type>string</type>
      <module>shutter</module>
      <value>SPVI2H2UA</value>
</feature>
```

Now that our SSP has been set properly, we can save it and call our code:

```
SI_H Handle = 0;
SI_CHK(SI_Load(L""));
SI_CHK(SI_Open(0, &Handle));
SI_CHK(SI_Command(Handle, L"Initialize"));
```

If the values have not been set in the SSP, you can set them via function calls:

```
SI_H Handle = 0;
SI_CHK(SI_Load(L""));
SI_CHK(SI_Open(0, Handle));
SI_CHK(SI_SetString(Handle, L"Grabber.Channel", L"img0"));
SI_CHK(SI_SetString(Handle, L"Camera.Channel", L"port0"));
SI_CHK(SI_SetString(Handle, L"Shutter.Channel", L"SP350000A"));
SI_CHK(SI_Command(Handle, L"Initialize"));
```

Note:

For more information on the mandatory and optional features that are needed before initializing a sensor, see section Specim Sensor Profile and the feature list chapter.

4.9 Buffer Management

Buffer management is handled by Lumo Sensor SDK automatically. Under the hood it uses a ring buffer of certain size which is specified by the value of `Acquisition.RingBuffer.Size` and is applied at every time the acquisition is started. Right before the acquisition actually starts, Lumo Sensor SDK creates same number of frame buffers to store incoming frames in FIFO fashion. The producer (grabber device) of the ring buffer will always store the next incoming frame to the next buffer in line and it will not wait for the consumers to read the frame, so it is the consumer's responsibility to make sure it reads fast enough.

The data is aligned in a BIL (Band-interleave) format which means that data is presented row-by-row (band-by-band). For a frame of 640 spatial pixels and 224 bands the frame looks like this (S=sample, B=band):

```
[B1_S1] [B1_S2] ... [B1_S639] [B1_S640]
[B2_S1] [B2_S2] ... [B2_S639] [B2_S640]
... ... ...
```

```
[B223_S1] [B223_S1] ... [B223_S640]
[B224_S1] [B224_S1] ... [B224_S640]
```

In memory, the data is laid out as follows:

```
[B1_S1] [B1_S2] ... [B1_S640] [B2_S1][B2 S2] ... [B2_S640] [B3_S1] ...
```

Both `SI_Wait` function and the data callback provide the frame number of the incoming frame which should be used to determine whether any frames have been dropped.

There are a couple of extra features that help detecting any problems with the buffering:

- `Acquisition.RingBuffer.Lag`. This feature defines the delay (in terms of frames) between the producer and the consumer (the `SI_Wait` function or data callback). Whenever this delay value reaches the size of the ring buffer, we have dropped a whole buffer of frames.
- `Acquisition.DroppedFrames`. This feature automatically compares the incoming frame numbers to the previous ones and reports the amount of dropped frames within one acquisition sequence.

Both of these features are reset to 0 when acquisition is stopped.

Because of the internal buffering, it is more than enough to only use one buffer to store the incoming data. When using the `SI_Wait` function to capture frames, it is recommended to use the `SI_CreateBuffer` for allocation and `SI_DisposeBuffer` for de-allocation of buffers to ensure correct alignment. When using the data callback function one can use any kind of allocation and de-allocation method for the buffers to store the incoming data.

See SI_Wait and data callback function descriptions for more information.

An example of buffer management using the `SI_Wait` functionality:

```
SI_64 nFrameNumber = 0;
SI_64 nFrameSize = 0;
void* pBuffer = 0;
int nTimeout = 5000;
SI_CHK(SI_GetInt(Handle, L"Camera.Image.SizeBytes", &nFrameSize));
SI_CHK(SI_CreateBuffer(Handle, nFrameSize, &pBuffer));
SI_CHK(SI_Wait(Handle, (SI_U8*)pBuffer, &nFrameSize, &nFrameNumber,
 nTimeout));

//Do something with the data...

SI_CHK(SI_DisposeBuffer(Handle, pBuffer));
```

An example of buffer management using the `DataCallback` functionality:

```
void MyClass::RegisterCallbacks()
{
// ...
SI_CHK(SI_RegisterDataCallback(Handle, onDataCallback, this));
// ...
}

int MyClass::onDataCallback(SI_U8* _pBuffer, SI_64 _nFrameSize, SI_64
 _nFrameNumber, void* _pContext)
{
 MyClass* pMyClass = (MyClass*)_pContext;
 // Our buffer is preallocated on 'Acquisition.PreStartCallback' callback.
 // using 'Camera.Image.SizeBytes' feature.
 memcpy((void*)pMyClass->m_pBuffer, (void*)_pBuffer, _nFrameSize);
```

```
 pMyClass->setFrameNumber(_nFrameNumber);
}
```

5 Specim Sensor Profile

Specim Sensor Profiles SSP are XML sensor configuration files for Lumo Sensor SDK.These profiles define the available features, which modules are used, which modules the features are assigned to, the windowing parameters, and so on.

These files are prepared by Specim, and supplied at delivery time. If your sensor is planned to be used with different grabber types, Specim will supply an SSP for each sensor + grabber combination.

A profile is unique for each sensor, but you need to update the features related to hardware configuration during the sensor installation, such as:

- Grabber channel.
- Camera channel.
- Shutter channel.
- SCB channel.
- GPS channel.
- and so on.

5.1 SSP File Structure

The SSP XML structure includes the following sections:

- name
- description
- type
- modules
- dependencies
- setup
- features

5.1.1 name

The `name` tag sets the name, which is listed in DeviceName for the sensor. If the tag is empty, the SSP file name, without the extension, will be used instead.

Warning: Two or more SSP files cannot have the same `name`. The second one will be ignored, and you will see a warning in the logs.

```
<name>VNIR PFD with NI</name>
```

5.1.2 description

The `description` is an optional field to add a sensor description into DeviceDescription.

```
<description>Spectral Camera VNIR PFD with National Instrument grabber.</
description>
```

5.1.3 type

The `type` defines to Lumo Sensor SDK, if the profile is for an HSI sensor or for a GPS sensor. Valid values are:

- Sensor
- GPS

```
<type>Sensor</type>
```

5.1.4 modules

The `modules` tag contains the list of modules required for the sensor:

```
<modules>
  <module type="grabber">grab001</module>
  <module type="camera">swir</module>
  <module type="shutter">shutter001</module>
</modules>
```

Each `module` has a `type` and a `value`. The different module types are:

- grabber
- camera (camera1, camera2, and so one, in multi sensor setup)
- shutter
- scb
- gps
- temperature

The `value` is the name of the module DLL without the extension.

Tip:

If you ever need to disable the usf certain hardware in an SSP, comment out the line referring to that hardware module, within the modules section. This example shows how to only leave the shutter controls in an SSP :

```
<modules>
    <!--<module type="grabber">grab001</module>-->
    <!--<module type="camera">swir</module>-->
    <module type="shutter">shutter001</module>
</modules>
```

5.1.5 dependencies

Reserved for later use.

```
<dependencies>
</dependencies>
```

5.1.6 setup

The `setup` section holds the `cameracount` tag, which defines the number of detectors in the sensor.

Currently, FENIX and FENIX1K are the only sensors with `cameracount` 2; the others have `cameracount` 1.

```
<setup>
  <cameracount>1</cameracount>
```

```
</setup>
```

5.1.7 features

The `features` section defines the available features for your sensor.

This section is prepared by Specim, and is different for every sensor + grabber setup. Unless absolutely necessary, you do not need to change its content. Contact Specim for assistance.

```
...
  <!-- Defines the camera feature -->
  <features>
    <feature>
      <name>Model</name>
      <type>string</type>
      <value>VNIRPFD</value>
      <readonly>true</readonly>
    </feature>

    <!-- Channel features -->
    <feature>
      <name>Grabber.Channel</name>
      <module>grabber</module>
      <type>string</type>
      <value>img0::0</value>
    </feature>

    <feature>
      <name>Camera.Channel</name>
      <type>string</type>
      <module>camera</module>
      <value>port0</value>
    </feature>

    <feature>
      <name>Shutter.Channel</name>
      <type>string</type>
      <module>shutter</module>
      <value>DEMOUNITA</value>
    </feature>
...
```

Each `feature` has the following properties:

- `name` - The `name` tag defines the name of the feature.
- `type` - The `type` tag has one of the following common types as `value` :
  - `integer` - Used for features that can be represented by numbers without a decimal component.
  - `float` - Used for features that can be represented by numbers with a decimal component (double precision).
  - `string` - Used for features that can be represented by text.
  - `boolean` - Used for features that can be represented by either true or false.
  - `command` - Used for features that can be executed and have no actual value.
  - `enumeration` - Used for features that can have value out of a predefined set of possible values.
  - `signal` - Used for features that can be considered as events triggered by Lumo Sensor SDK.
- `module` - The `module` tag defines the module type name where the actual feature is implemented at. The valid values for the `module` tag are the same ones as defined in the modules section.

Note: There can be multiple `module` tags for cases where the feature is implemented in multiple modules:

```
<feature>
  <name>AcquisitionWindow.Width</name>
  <module>grabber</module>
  <module>camera</module>
  <type>integer</type>
  <value>1312</value>
  <stopacquisition>true</stopacquisition>
</feature>
```

The order, in which the `module` tags are declared, defines the order, in which the feature is accessed for each module. In the example above, the calls would go to the `grabber` module first, and then the `camera` module.

A feature can have the following optional properties:

- `value` - Defines the original value or the read only value for features of type:
  - `integer`
  - `float`
  - `string`
  - `boolean`
- `values` - Defines the sets of `value` for features of type `enumeration`:

```
 <feature>
  <name>Camera.Binning.Spatial</name>
  <type>enumeration</type>
  <module>grabber</module>
  <values>
  <default>1</default>
    <value>1</value>
    <value>2</value>
    <value>4</value>
    <value>8</value>
  </values>
  <stopacquisition>true</stopacquisition>
</feature>
```

Each `value` is treated as a `string`.

Enumeration features have an index, which represents the order in which the values have been added into the enumeration. In the example above, **Band** is at index 0 and **Point** at index 1.

Enumeration features also have a current index used to select a particual value. The `default` tag sets the default current index. In the example above, the default current index is 1.

- `stopacquisition` - The `stopacquisition` tag allowed values are `true` or `false`. When `true`, the following actions happen:
  - The acquisition is stopped.
  - The value is changed.
  - The acquistion is started again.
- `min` - The `min` tag defines the minimum allowed value for features of type `integer` and `float`.
- `max` - The `max` tag defines the maximum allowed value for features of type `integer` and `float`.
- `signalallupdates` - The `signalallupdates` tag allowed values are `true` or `false`, and the default value is `true`.

  - `true` - The feature callback is triggered when registering a feature callback.
  - `false` - The feature callback is not triggered when registering a feature callback.
- `readonly` - The `readonly` tag allowed values are `true` or `false`. The tag specifies whether the feature can be read.
- `writable` - The `writable` tag allowed values are `true` or `false`. The tag specifies whether the feature can be written.

5.2 Setting up SSP for the First Time

This section describes how to set up SSP for the first time.

Proceed as follows:

1. Check that the modules section has all the modules required for the sensor system. If you don't have a shutter, then comment out the shutter line from the modules section.
2. Change all the channel features for these modules to match the correct interfaces and ports. See full feature description for more details on different channel values for different devices.
3. Set a proper calibration pack path, if available.
4. Set the correct acquisition window parameters (to match the spectral calibration) if calibration pack is not used. If calibration pack is in use, the acquisition window parameters are ignored.
5. After opening and initializing the sensor once, check the polarity of the USB shutter by opening the shutter and seeing whether it opens or closes.

   If `Camera.OpenShutter` closes it, set `Camera.Shutter.IsToggle` to true permanently.

5.3 Changing the Feature Values Using SSP and Functions

One important thing to understand about the values set in the SSP is that they are just the default starting values when opening a sensor in Lumo Sensor SDK - nothing more. There still are valid reasons to change values straight from the SSP:

- Some values rarely change so might as well set them from the SSP.
  - The device channels hardly ever change, so it easier to change them from the SSP, so you do not have to call those functions every time you open a sensor.
  - The acquisition window parameters do not change unless the sensor is recalibrated.
- Sometimes there is a need to enable or disable different functionalities
  - Disabling VNIR PFD hardware binning modes to force software binning in case of long camera link cables. See Camera.Binning.HardwareModes for more information.
  - Disabling Bad Pixel Replacement when doing tests on a sensor.
- When modifying sensor specific values
  - Changing the exposure times for the NUC tables using the Camera.NUC enumeration feature values.
  - Changing the sensor serial number.

In most cases, the only things that need to be changed in the SSP are the device channels, the calibration pack path and the possible inverting of the USB shutter polarity.

The SSPs are not just used for setting up the default values. They serve a feature generation purpose for Lumo Sensor SDK as well. This means that any feature you want access to, has be listed in the SSP, and therefore most of the SSP features are ones that you will never have to touch.

6 Feature Lists

6.1 SI_SYSTEM Features

6.1.1 DeviceName

Description

This feature specifies the names of all the SSPs available, as an enumerated list. This feature is only accessible by using the `SI_SYSTEM` handle. This feature becomes available after a successful SI_Open call.

Type

Enumeration

Access

Read

Example

```
int nDeviceCount = 0;
SI_CHK(SI_GetInt(SI_SYSTEM, L"DeviceCount", &nDeviceCount));

// Iterate through each devices
for (int n = 0; n < nDeviceCount; n++)
{
  SI_WC sName[255] = {0};
  SI_CHK(SI_GetEnumStringByIndex(SI_SYSTEM, L"DeviceName", n,
 sName, 255));
}
```

6.1.2 DeviceDescription

Description

This feature specifies the descriptions of all the SSPs available, as an enumerated list. This feature is only accessible by using the `SI_SYSTEM` handle. This feature becomes available after a successful SI_Open call.

Type

Enumeration

Access

Read

Example

```
int nDeviceCount = 0;
SI_CHK(SI_GetInt(SI_SYSTEM, L"DeviceCount", &nDeviceCount));

// Iterate through each devices
for (int n = 0; n < nDeviceCount; n++)
{
  SI_WC sDescription[255] = {0};
  SI_CHK(SI_GetEnumStringByIndex(SI_SYSTEM,
 L"DeviceDescription", n, sDescription, 255));
}
```

6.1.3 DeviceCount

Description

This feature specifies the number of SSPs available. This feature is only accessible by using the `SI_SYSTEM` handle. This feature becomes available after a successful SI_Open call.

Type

Integer

Access

Read

Example

```
SI_64 nDeviceCount = 0;
SI_CHK(SI_GetInt(SI_SYSTEM, L"DeviceCount", &nDeviceCount));
```

6.1.4 Sensor<insert index>.IsOpen

Description

This feature specifies whether a sensor at a certain index has been opened successfully using SI_Open call. This feature is only accessible by using the `SI_SYSTEM` handle.

Note: The `<insert index>` in the name refers to the same index used with SI_Open. The maximum value of this index is always DeviceCount – 1.

Type

Boolean

Access

Read

Example

```
SI_CHK(SI_Load(L""));
SI_CHK(SI_RegisterFeatureCallback(SI_SYSTEM,
 L"Sensor3.IsOpen", onFeatureCallback, this));

//When SI_Open for index 3 finishes successfully,
 onFeatureCallback is called automatically.
SI_H Handle = 0;
SI_CHK(SI_Open(3, &Handle));
```

6.1.5 ProfilesDirectory

Description

This feature specifies the folder path where Lumo Sensor SDK looks for SSP files. If you need to change this feature, you must do it before calling SI_Load.

If you want to load the SSP files from a different folder than the default, change this feature to a valid path.

The default path for Windows 10 (and later) it is `C:/Users/Public/Documents/Specim`.

Type

String

Access

Read / Write

Example

```
SI_CHK(SI_SetString(SI_SYSTEM, L"ProfilesDirectory", L"C:/
ssp"));
SI_CHK(SI_Load(L""));
```

6.2 Sensor Features

6.2.1 Acquisition.Acquiring

Description

This feature specifies whether the acquisition is currently ongoing.

Type

Boolean

Access

Read

Example

```
bool bIsAcquiring = false;
SI_CHK(SI_GetBool(Handle, L"Acquisition.Acquiring",
 &bIsAcquiring));
```

6.2.2 Acquisition.CalculatedFrameRate

Description

This feature specifies the approximation of the current real frame rate. It is calculated in Lumo Sensor SDK for every second whenever the acquisition is ongoing. This value is useful when compared with the value of the `Camera.FrameRate` feature, to determine whether the frame rate actually is what it was set to.

Type

Float

Access

Read

Example

```
SI_CHK(SI_RegisterFeatureCallback(Handle,
 L"Acquisition.CalculatedFrameRate", MyCallback, 0));
```

6.2.3 Acquisition.Error

Description

This signal feature is signaled every time an acquisition-related error occurs. Whenever this feature is signaled, the acquisition is also automatically stopped.

Type

Signal

Access

-

Example

```
SI_CHK(SI_RegisterFeatureCallback(Handle,
 L"Acquisition.Error", MyCallback, pMyClass));
```

6.2.4 Acquisition.FrameCounter

Description

This feature specifies the current frame number. Call the `Acquisition.FrameCounter.Reset` command feature to reset the value. Calling `Acquisition.Stop` will also reset the frame number counter.

Note:

Both `SI_Wait` and the data callback also give the frame number when called.

Type

Integer

Access

Read

Example

```
SI_64 nFrameCounter = 0;
SI_CHK(SI_GetInt(Handle, L"Acquisition.FrameCounter",
 &nFrameCounter));
```

6.2.5 Acquisition.FrameCounter.Reset

Description

This feature resets the `Acquisition.FrameCounter` feature value. Calling `Acquisition.Stop` will also reset the frame number counter.

Type

Command

Access

-

Example

```
SI_CHK(SI_Command(Handle, L"Acquisition.FrameCounter.Reset"));
```

6.2.6 Acquisition.PostStartCallback

Description

This signal feature is signaled every time the acquisition is started - whether it is by calling Acquisition.Start or internally by the *SDK*.

Type

Signal

Access

-

Example

```
SI_CHK(SI_RegisterFeatureCallback(Handle,
 L"Acquisition.PostStartCallback", Callback, this));
```

6.2.7 Acquisition.PostStopCallback

Description

This signal feature is signaled every time the acquisition is stopped - whether it is by calling `Acquisition.Stop` or internally by Lumo Sensor SDK.

Type

Signal

Access

-

Example

```
SI_CHK(SI_RegisterFeatureCallback(Handle,
 L"Acquisition.PostStopCallback", Callback, this));
```

6.2.8 Acquisition.PreStartCallback

Description

This signal feature is signaled every time the acquisition is about to be started - whether it is by calling `Acquisition.Start` or internally by Lumo Sensor SDK. At this point, we know what will be the final size of the image. Therefore, this feature is a useful one to register a feature callback, and query features like `Camera.Image.Width`, `Camera.Image.Height`, `Camera.Image.SizeBytes` and `Camera.Image.ByteDepth` within that callback, to create the required buffers and initialize GUI elements for displaying the data.

Type

Signal

Access

-

Example

```
SI_CHK(SI_RegisterFeatureCallback(handle,
 L"Acquisition.PreStartCallback", FeatureCallback, this));

int FeatureCallback(SI_H Handle, SI_WC* Feature, void*
 Context)
{
 MyClass* pMyClass = (MyClass*)Context;

 if (wcscmp(Feature, L"Acquisition.PreStartCallback") == 0)
 {
  SI_64 nWidth = 0;
  SI_64 nHeight = 0;
  SI_64 nSizeBytes = 0;

  SI_GetInt(Handle, L"Camera.Image.SizeBytes", &nSizeBytes);
  SI_GetInt(Handle, L"Camera.Image.Width", &nWidth);
  SI_GetInt(Handle, L"Camera.Image.Height", &nHeight);

  //Use nSizeBytes to create data buffers.
  //Use nWidth and nHeight to initialize GUI elements.
```

```
  }
}
```

6.2.9 Acquisition.PreStopCallback

Description

This signal feature is signaled every time the acquisition is about to be stopped - whether it is by calling `Acquisition.Stop` or internally by Lumo Sensor SDK.

Type

Signal

Access

-

Example

```
SI_CHK(SI_RegisterFeatureCallback(Handle,
 L"Acquisition.PreStopCallback", onCallback, this));
```

6.2.10 Acquisition.RingBuffer.Lag

Description

This feature specifies how many frames Lumo Sensor SDK is currently buffering. Whenever `Acquisition.RingBuffer.Lag` is greater than 0, Lumo Sensor SDK receives more frames than the application is able to handle, and buffers them.

The number of frames Lumo Sensor SDK can buffer is specified in Acquisition.RingBuffer.Size. If `Acquisition.RingBuffer.Lag` reaches the same value as Acquisition.RingBuffer.Size, all the buffered frames are lost, and `Acquisition.RingBuffer.Lag` is set back to 0.

Type

Integer

Access

Read

Example

```
SI_CHK(SI_RegisterFeatureCallback(handle,
 L"Acquisition.RingBuffer.Lag", onFeatureCallback, 0));
```

6.2.11 Acquisition.RingBuffer.Size

Description

This feature specifies how many frames Lumo Sensor SDK is able to buffer. The number is limited by the amount of RAM available on your computer. The use of buffering helps preventing dropped frames during performance fluctuaction.

Type

Integer

Access

Read / Write

Example

```
SI_CHK(SI_SetInt(Handle, L"Acquisition.RingBuffer.Size",
 200));
```

```
SI_CHK(SI_Command(Handle, L"Acquisition.Start"));
```

6.2.12 Acquisition.RingBuffer.Sync

Description

This feature synchronizes the internal ringbuffers in a way that the next frames coming from the sensor will be the next frames returned by SI_Wait or *data callback*. Any frames in the ringbuffer before synchronizing will be lost.

Note:

This function is useful, for example, when acquiring dark frame data. It ensures that Acquisition.RingBuffer.Lag is 0, and that no frames acquired prior to closing the shutter will be sent to the application.

Type

Command

Access

-

Example

```
SI_CHK(SI_Command(handle, L"Acquisition.RingBuffer.Sync"));
```

6.2.13 Acquisition.SetupFilePath

Description

This feature specifies the full path of a grabber-specific configuration file. This feature should be hard-coded through the SSP to point to the correct configuration file.

Type

String

Access

Read/Write

6.2.14 Acquisition.Start

Description

This command starts the acquisition.

Type

Command

Access

-

Example

```
SI_CHK(SI_RegisterDataCallback(Handle, onDataCallback, this));
SI_CHK(SI_Command(handle, L"Acquisition.Start"));
```

6.2.15 Acquisition.Stop

Description

This command stops the acquisition.

Type

Command

Access

-

Example

```
SI_CHK(SI_RegisterDataCallback(Handle, onDataCallback, this));
SI_CHK(SI_Command(handle, L"Acquisition.Start"));
// ...
SI_CHK(SI_Command(handle, L"Acquisition.Stop"));
```

6.2.16 Acquisition.Timeout

Description

This feature specifies the acquisition timeout in milliseconds. The acquisition is automatically aborted if the time between two adjacent frames is greater than Acquisition.Timeout.

Type

Float

Access

Read/Write

Example

```
SI_CHK(SI_SetFloat(handle, L"Acquisition.Timeout", 5000));
SI_CHK(SI_Command(handle, L"Acquisition.Start"));
```

6.2.17 AcquisitionWindow.Height

Description

This feature specifies the height of the acquisition window. This feature should always be set to the same width where the sensor calibration was done.

This feature is ignored if a proper calibration pack is set in the `Camera.CalibrationPack` feature. If this feature is set to 0, and no proper calibration pack is in use, the current values from the sensor will be read, instead.

On the other hand, when you set a valid calibration pack, the SSP value is ignored, and the value from the calibration pack is used, instead.

If you set the value to 0 in the SSP, and do not set a Camera.CalibrationPack, the current values from the sensor electronic will be used.

Sensors with multiple detectors have one set of `AcquisitionWindow.Height` for each detector, and `AcquisitionWindow` is suffixed with the detector index, such as:

- `AcquisitionWindow1.Height`
- `AcquisitionWindow2.Height`

`AcquisitionWindow.Height` is not the final height of the image, but the height before binning, transformation, or MROI.

Note: Changing this value will restart the acquisition automatically.

Type

Integer

Access

Read / Write

Example

```
SI_64 nHeight = 0;
SI_CHK(SI_GetInt(handle, L"AcquisitionWindow.Height",
 &nHeight));
```

6.2.18 AcquisitionWindow.Left

Description

This feature specifies the 0-based left offset of the acquisition window. When you set a valid calibration pack, the SSP value is ignored, and the value from the calibration pack is used, instead.

Sensors with multiple detectors have one set of `AcquisitionWindow.Left` for each detector, and `AcquisitionWindow` is suffixed with the detector index, such as:

- `AcquisitionWindow1.Left`
- `AcquisitionWindow2.Left`

Note: Changing this value automatically restarts the acquisition.

Type

Integer

Access

Read / Write

Example

```
SI_64 nLeft = 0;
SI_CHK(SI_GetInt(handle, L"AcquisitionWindow.Left", &nLeft));
```

6.2.19 AcquisitionWindow.Top

Description

This feature specifies the 0-based top offset of the acquisition window. When you set a valid calibration pack, the SSP value is ignored, and the value from the calibration pack is used, instead.

Sensors with multiple detectors have one set of `AcquisitionWindow.Top` for each detector, and `AcquisitionWindow` is suffixed with the detector index, such as:

- `AcquisitionWindow1.Top`
- `AcquisitionWindow2.Top`

Note: Changing this value automatically restarts the acquisition.

Type

Integer

Access

Read/Write

Example

```
SI_64 nTop = 0;
```

```
SI_CHK(SI_GetInt(handle, L"AcquisitionWindow.Top", &nTop));
```

6.2.20 AcquisitionWindow.Width

Description

This feature specifies the width of the acquisition window.

When you set a valid calibration pack, the SSP value is ignored, and the value from the calibration pack is used, instead.

If you set the value to 0 in the SSP, and do not set a Camera.CalibrationPack, the current values from the sensor electronic will be used.

Sensors with multiple detectors have one set of `AcquisitionWindow.Width` for each detector, and `AcquisitionWindow` is suffixed with the detector index, such as:

- `AcquisitionWindow1.Width`
- `AcquisitionWindow2.Width`

`AcquisitionWindow.Width` is not the final width of the image, but the width before binning, transformation, or MROI.

Note: Changing this value automatically restarts the acquisition.

Type

Integer

Access

Read / Write

Example

```
SI_64 nWidth = 0;
SI_CHK(SI_GetInt(handle, L"AcquisitionWindow.Width",
 &nWidth));
```

6.2.21 Camera1.ExposureTime

Description

A feature of the *Sensor Control Board*, this feature controls the high and low times of trigger1 TTL pulse. It is only used when Camera.Trigger.Mode is set to *External*. The high and low times are adjusted to get the requested frame rate.

The same feature name is also used when referring to the Camera.ExposureTime of the first detector in a multi-detector sensor, such as Fenix.

Type

Float

Access

Read / Write

Example

```
double dExposureTime = 0.0;
SI_CHK(SI_GetFloat(Handle, L"Camera1.ExposureTime",
 &dExposureTime");

SI_CHK(SI_SetFloat(Handle, L"Camera1.ExposureTime", 5.0));
```

6.2.22 Camera1.FrameRate

Description

A feature of the *Sensor Control Board*, this feature controls the low time of trigger1 TTL pulse. It is only used when Camera.Trigger.Mode is set to *External*. The low time adjusted to keep the current high time for the resulting frame time. If the current high time of trigger 1 is larger than the requested frame time, the high time is first adjusted to fit within the requested frame rate.

The same feature name is also used when referring to the Camera.FrameRate of the first detector in a multi-detector sensor, such as Fenix.

Type

Float

Access

Read / Write

Example

```
double dFrameRate = 0.0;
SI_CHK(SI_GetFloat(Handle, L"Camera1.FrameRate",
 &dFrameRate");

SI_CHK(SI_SetFloat(Handle, L"Camera1.FrameRate", 50.0));
```

6.2.23 Camera2.ExposureTime

Description

A feature of the *Sensor Control Board*, this feature controls the high and low times of trigger2 TTL pulse. It is only used when Camera.Trigger.Mode is set to *External*. The high and low times are adjusted to get the requested frame rate.

The same feature name is also used when referring to the Camera.ExposureTime of the second detector in a multi-detector sensor, suc as Fenix.

Type

Float

Access

Read / Write

Example

```
double dExposureTime = 0.0;
SI_CHK(SI_GetFloat(Handle, L"Camera2.ExposureTime",
 &dExposureTime");

SI_CHK(SI_SetFloat(Handle, L"Camera2.ExposureTime", 5.0));
```

6.2.24 Camera2.FrameRate

Description

A feature of the *Sensor Control Board*, this feature controls the low time of trigger2 TTL pulse. It is only used when Camera.Trigger.Mode is set to *External*. The low time adjusted to keep the current high time for the resulting frame time. If the current high time of trigger2 is larger than the requested frame time, the high time is first adjusted to fit within the requested frame rate.

The same feature name is also used when referring to the Camera.FrameRate of the second detector in a multi-detector sensor, suc as Fenix.

Type

Float

Access

Read / Write

Example

```
double dFrameRate = 0.0;
SI_CHK(SI_GetFloat(Handle, L"Camera2.FrameRate",
 &dFrameRate");

SI_CHK(SI_SetFloat(Handle, L"Camera2.FrameRate", 50.0));
```

6.2.25 Camera.AutoNUC

Description

This feature specifies whether the automatic NUC selection is enabled or disabled. When enabled, every time the exposure time is changed, it is compared with the values in `Camera.NUC`, and the closest NUC is selected. By default, this valus is set to true.

Type

Boolean

Access

Read / Write

Example

```
SI_CHK(SI_SetBool(Handle, L"Camera.AutoNUC", true));
```

6.2.26 Camera.Binning.Average

Description

This feature specifies whether to average the binned values or not.

Type

Boolean

Access

Read / Write

Example

```
SI_CHK(SI_SetBool(Handle, L"Camera.Binning.Average", true));
```

6.2.27 Camera.Binning.Average.IsHardware

Description

This feature specifies whether the underlying hardware supports averaged hardware binning. If not, software is used to average the hardware binning portion as well. These are set to their default values by Specim and should not be changed.

Type

Boolean

Access

Read / Write

Example

```
bool bIsHardware = false;
SI_CHK(SI_GetBool(Handle,
 L"Camera.Binning.Average.IsHardware", &bIsHardware));
```

6.2.28 Camera.Binning.HardwareModes

Description

This feature specifies all the binning modes supported by hardware. Supported hardware binning modes differ between sensors. The rest of the modes that are not supported by hardware will be carried out by using software.

Note:

Set this feature to only hold the '1x1' value to force software binning only.

Type

Enumeration

Access

Read / Write

Values

- AxB, where A,B = {1, 2, 4, 8}

6.2.29 Camera.Binning.Spatial

Description

This feature specifies the current spatial binning mode. The acquisition is automatically stopped if you change this feature, and restarted afterwards.

Note:

As an enumeration feature, you need to set index 0 to get 1x binning, index 1 to get 2x binning, and so on.

Type

Enumeration

Access

Read / Write

Values

- 1
- 2
- 4
- 8

Example

```
SI_SetEnumIndex(Handle, L"Camera.Binning.Spatial", 0);  //Set
 1x Spatial Binning
SI_SetEnumIndex(Handle, L"Camera.Binning.Spatial", 1);  //Set
 2x Spatial Binning
SI_SetEnumIndex(Handle, L"Camera.Binning.Spatial", 2);  //Set
 4x Spatial Binning
SI_SetEnumIndex(Handle, L"Camera.Binning.Spatial", 3);  //Set
 8x Spatial Binning
```

6.2.30 Camera.Binning.Spectral

Description

This feature specifies the current spectral binning mode. The acquisition is automatically stopped if you change this feature, and restarted afterwards.

Note:

As an enumeration feature, you need to set index 0 to get 1x binning, index 1 to get 2x binning, and so on.

Type

Enumeration

Access

Read / Write

Values

- 1
- 2
- 4
- 8

Example

```
SI_SetEnumIndex(Handle, L"Camera.Binning.Spectral", 0);  //Set
 1x Spectral Binning
SI_SetEnumIndex(Handle, L"Camera.Binning.Spectral", 1);  //Set
 2x Spectral Binning
SI_SetEnumIndex(Handle, L"Camera.Binning.Spectral", 2);  //Set
 4x Spectral Binning
SI_SetEnumIndex(Handle, L"Camera.Binning.Spectral", 3);  //Set
 8x Spectral Binning
```

6.2.31 Camera.BitDepth

Description

This feature specifies the bit depth of each pixel on the sensor.

Type

Integer

Access

Read

Example

```
SI_64 nBitDepth = 0;
SI_CHK(SI_GetBool(Handle, L"Camera.BitDepth", &nBitDepth));
```

6.2.32 Camera.BPR

Description

This feature specifies whether the bad pixel replacement is enabled or disabled.

Type

Boolean

Access

Read/Write

Example

```
SI_CHK(SI_SetBool(Handle, L"Camera.BPR", true));
```

6.2.33 Camera.BPR.BadPixelCount

Description

This feature returns the absolute amount of bad pixels from the current *Bad Pixel Map* either created by Camera.BPR.CreateMap, or loaded from *Calibration pack*.

Type

Integer

Access

Read

Example

```
SI_64 nBadPixelCount = 0;
SI_CHK(SI_GetInt(Handle, L"Camera.BPR.BadPixelCount",
 &nBadPixelCount));
```

6.2.34 Camera.BPR.BadPixelPercentage

Description

This feature returns the relative amount of bad pixels in percentage from the current *Bad Pixel Map* either created by Camera.BPR.CreateMap, or loaded from *Calibration pack*.

Type

Float

Access

Read

Example

```
double dBadPixelPercentage = 0.0;
SI_CHK(SI_GetFloat(Handle, L"Camera.BPR.BadPixelPercentage",
 &dBadPixelPercentage));
```

6.2.35 Camera.BPR.CreateMap

Description

This command feature runs the *Bad Pixel Replacement* workflow and tries to find bad pixels from two different sets of dark current.

The workflow is as follows:

- Call Camera.CloseShutter
- Set Camera.FrameRate to *50* Hz.
- Set Camera.ExposureTime to *1.0* ms.
- Capture *50* frames.
- Set Camera.ExposureTime to *10.0* ms.
- Capture *300* frames.
- Use the two captured datasets to determine which pixels are bad.

Type

Command

Access

-

Example

```
SI_CHK(SI_Command(Handle, L"Camera.BPR.CreateMap"));
```

6.2.36 Camera.BPR.CreateMap.InProgress

Description

This feature returns whether the Camera.BPR.CreateMap is currently running.

Type

Boolean

Access

Read

Example

```
bool bInProgress = false;
SI_CHK(SI_GetBool(Handle, L"Camera.BPR.CreateMap.InProgress",
 &bInProgress));
```

6.2.37 Camera.BPR.Gain.Max

Description

This feature configures the maximum threshold used for finding bad pixels with uneven gain, when calling the Camera.BPR.CreateMap feature. The *default* value for this feature is *1.5*.

Type

Float

Access

Read/Write

Example

```
SI_CHK(SI_SetFloat(Handle, L"Camera.BPR.Gain.Max", 2.0));
SI_CHK(SI_Command(Handle, L"Camera.BPR.CreateMap"));
```

6.2.38 Camera.BPR.Gain.Min

Description

This feature configures the minimum threshold used for finding bad pixels with uneven gain, when calling the Camera.BPR.CreateMap feature. The *default* value for this feature is *0.3*.

Type

Float

Access

Read/Write

Example

```
SI_CHK(SI_SetFloat(Handle, L"Camera.BPR.Gain.Min", 0.5));
SI_CHK(SI_Command(Handle, L"Camera.BPR.CreateMap"));
```

6.2.39 Camera.BPR.IsHardware

Description

This feature configures whether the *Bad Pixel Replacement* method is available on the detector, or whether it should be carried out through *software*. This feature is hardware dependent and is automatically used by the grabbing pipeline when Acquisition.Start has been called.

Type

Boolean

Access

Read

Example

```
bool bIsHardware = false;
SI_CHK(SI_GetBool(Handle, L"Camera.BPR.IsHardware",
 &bIsHardware));
```

6.2.40 Camera.BPR.MapAvailable

Description

This feature returns whether a valid *Bad Pixel Map* is available for the sensor.
If this feature is *false*, Camera.BPR is set to read-only mode. Either calling Camera.BPR.CreateMap or loading a valid *Calibration pack* with a valid *Bad Pixel Map* will automatically set this feature to *true*.

Type

Boolean

Access

Read

Example

```
bool bIsMapAvailable = false;
SI_CHK(SI_GetBool(Handle, L"Camera.BPR.MapAvailable",
 &bIsMapAvailable));
```

6.2.41 Camera.BPR.Noise.Max

Description

This feature configures the maximum threshold used for finding noisy bad pixels, when calling the Camera.BPR.CreateMap feature. The *default* value for this feature is *4.0*.

Type

Float

Access

Read/Write

Example

```
SI_CHK(SI_SetFloat(Handle, L"Camera.BPR.Noise.Max", 5.0));
SI_CHK(SI_Command(Handle, L"Camera.BPR.CreateMap"));
```

6.2.42 Camera.BPR.Offset.Max

Description

This feature configures the maximum threshold used for finding bad pixels with uneven offsets, when calling the Camera.BPR.CreateMap feature. The *default* value for this feature is *1.5*.

Type

Float

Access

Read/Write

Example

```
SI_CHK(SI_SetFloat(Handle, L"Camera.BPR.Offset.Max", 2.0));
SI_CHK(SI_Command(Handle, L"Camera.BPR.CreateMap"));
```

6.2.43 Camera.BPR.Offset.Min

Description

This feature configures the minimum threshold used for finding bad pixels with uneven offsets, when calling the Camera.BPR.CreateMap feature. The *default* value for this feature is *0.3*.

Type

Float

Access

Read/Write

Example

```
SI_CHK(SI_SetFloat(Handle, L"Camera.BPR.Offset.Min", 0.5));
SI_CHK(SI_Command(Handle, L"Camera.BPR.CreateMap"));
```

6.2.44 Camera.ByteDepth

Description

This feature specifies the byte depth of each pixel on the sensor.

Type

Integer

Access

Read

Example

```
SI_64 nByteDepth = 0;
SI_CHK(SI_GetInt(Handle, L"Camera.ByteDepth", &nByteDepth));
```

6.2.45 Camera.CalibrationPack

Description

This feature specifies the full path for the sensor calibration pack (SCP). Calibration packs are prepared by Specim and supplied at the delivery. They must be loaded before initializing the camera.

Note: It is possible to use the Lumo Sensor SDK without a valid calibration pack, but all calibration information will default to the generated estimated values, which are incorrect for your sensor. The spectral information being missing, the data will be unusable.

Type

String

Access

Read / Write

Example

```
SI_WC sPath[] = L"C:/Users/Public/Documents/Specim/calibration
 packs/Fenix1K_12345.scp";
SI_CHK(SI_SetString(handle, L"Camera.CalibrationPack",
 sPath));
```

6.2.46 Camera.CalibrationPack.IsLoaded

Description

This feature returns whether a valid *Calibration Pack* has been loaded.

Type

Boolean

Access

Read

Example

```
bool bIsLoaded = false;
SI_CHK(SI_GetBool(Handle, L"Camera.CalibrationPack.IsLoaded",
 &bIsLoaded));
```

6.2.47 Camera.Channel

Description

This feature specifies the channel to be used for the camera device.

The `Camera.Channel` feature is empty by default. Most of the time, the channel is automatically detected. However, if you have multiple cameras of the same type, you need to tell the SDK to which device it should connect.

You can set `Camera.Channel` either through the SSP, or by using SI_SetString.

Warning: An incorrect parameter for `Camera.Channel` will cause the Initialize command to fail.

Note: The list of available channels is printed in the log file after Initialization sequence has been completed.

Type

String

Access

Read / Write

Example

```
SI_CHK(SI_SetString(handle, L"Camera.Channel", L"port1"));
SI_CHK(SI_Command(handle, L"Initialize"));
```

6.2.48 Camera.Channels

This feature specifies all the available camera channels found for the specific sensor.

The feature parameters are:

Type

Enumeration

Access

Read

6.2.49 Camera.CheckTemperatures

Description

This feature updates all temperature features related to the sensor. In the case of *SWIR3*, it will update Camera.Temperature, Scb.Temperature.Channel1, Scb.Temperature.Channel2, Scb.Temperature.Channel3 and Scb.Temperature.Channel4.

Type

Command

Access

-

Example

```
SI_CHK(SI_RegisterFeatureCallback(Handle,
 L"Camera.Temperature", onTemperatureCallback, this));
SI_CHK(SI_RegisterFeatureCallback(Handle,
 L"Scb.Temperature.Channel1", onTemperatureCallback, this));
SI_CHK(SI_Command(Handle, L"Camera.CheckTemperatures"));
```

6.2.50 Camera.CloseShutter

Description

This feature closes the shutter.

Note:

If the shutter opens when calling this function, set `Camera.Shutter.IsToggle` to true in the SSP and, after that, your shutter will work as it should. This is related to polarity differences between different individual shutters.

Type

Command

Access

-

Example

```
SI_CHK(SI_Command(Handle, L"Camera.CloseShutter"));
```

6.2.51 Camera.Count

Description

This feature specifies the amount of detectors in the sensor. For single-detector sensors, the value is always 1. For sensors such as Fenix and Fenix1K, the value is 2.

Type

Integer

Access

Read

Example

```
SI_64 nCount = 0;
SI_CHK(SI_GetInt(handle, L"Camera.Count", &nCount));
```

6.2.52 Camera.Debug

Description

This feature controls whether additional debug messages are added to the *standard output* related to camera communication. The information added depends on the sensor in use. By *default*, the feature is set to *false*.

Type

Boolean

Access

Read / Write

Example

```
SI_CHK(SI_SetBool(Handle, L"Camera.Debug", true));
```

6.2.53 Camera.Debug.FrameNumber

Description

This feature controls whether a running number between 0 and 65535 is inserted to the first pixel of each frame. By *default*, this feature is set to *false*.

Type

Boolean

Access

Read / Write

Example

```
SI_CHK(SI_SetBool(Handle, L"Camera.Debug.FrameNumber", true));
SI_CHK(SI_Command(Handle, L"Acquisition.Start"));
```

6.2.54 Camera.ExposureTime

Description

This feature specifies the current exposure time in milliseconds of the detector.

Type

Float

Access

Read / Write

Example

```
double dExposureTime = 5.5;
SI_CHK(SI_SetFloat(Handle, L"Camera.ExposureTime",
 dExposureTime));
```

```
SI_CHK(SI_GetFloat(Handle, L"Camera.ExposureTime",
 &dExposureTime));
```

6.2.55 Camera.ExposureTime.Auto

Description

This feature controls the automatic exposure time mode. If set to *true*, everytime Camera.FrameRate is changed, the Camera.ExposureTime is maximed taking into account the value of Camera.Image.ReadoutTime.

By *default*, this feature is *disabled*, but it can be helpful especially in an *Airborne* system where Camera.ExposureTime is always maximized.

Type

Boolean

Access

Read/Write

Example

```
double dFrameRate = 100;
SI_CHK(SI_SetBool(Handle, L"Camera.ExposureTime.Auto", true));
SI_CHK(SI_SetFloat(Handle, L"Camera.FrameRate", dFrameRate));

double dReadoutTime = 0;
SI_CHK(SI_GetFloat(Handle, L"Camera.Image.ReadoutTime",
 &dReadoutTime));

double dExposureTime = 0;
SI_CHK(SI_GetFloat(Handle, L"Camera.ExposureTime",
 &dExposureTime));

// dReadoutTime + dExposureTime == 1000.0 / dFrameRate
```

6.2.56 Camera.FrameRate

Description

This feature specifies the current frame rate in *Hz* of the detector.

Type

Float

Access

Read/Write

Example

```
double dFrameRate = 25.0;
SI_CHK(SI_SetFloat(Handle, L"Camera.FrameRate", dFrameRate));
SI_CHK(SI_GetFloat(Handle, L"Camera.FrameRate", &dFrameRate));
```

6.2.57 Camera.FWHM

Description

This feature specifies the FWHM (Full width half maximum) table for the sensor. Camera.FWHM is applied at Acquisition.PreStart.Callback. If a valid calibration pack exists and has proper FWHM information, it will be used to populate this feature.

Type

Enumeration

Access

Read

Example

```
int nCount = 0;
SI_GetEnumCount(Handle, L"Camera.FWHM", &nCount));
```

6.2.58 Camera.Gain.Analog

Description

This feature specifies the analog gain mode, if supported by hardware. This feature is either an integer or an enumeration feature depending on the sensor in use.

Type

Integer / Enumeration

Access

Read / Write

Example

```
SI_WC sType[20] = {0};
SI_GetFeatureType(Handle, L"Camera.Gain.Analog", sType, 20));

if (wcscmp(sType, L"integer") == 0)
{
  SI_CHK(SI_SetInt(Handle, L"Camera.Gain.Analog", 50));
}
else if (wcscmp(sType, L"enumeration") == 0)
{
  SI_CHK(SI_SetEnumIndex(Handle, L"Camera.Gain.Analog", 1));
}
```

6.2.59 Camera.Gain.Digital

Description

This feature specifies the digital gain mode, if supported by hardware.

Type

Enumeration

Access

Read/Write

Example

```
SI_CHK(SI_SetEnumIndex(Handle, L"Camera.Gain.Digital", 0));
```

6.2.60 Camera.Image.Embedding.Enabled

Description

Enabling this feature will start the procedure of embedding feature information to the image data. Embedded features are selected with the Camera.Image.Embedding.FeatureNames feature described in this document. Defaults to false after sensor reboot.

Type

Boolean

Access

Read/Write

Example

```
SI_CHK(SI_SetBool(Handle,
 L"Camera.Image.Embedding.Enabled", true));
```

6.2.61 Camera.Image.Embedding.Featurenames

Description

Feature to select list of embeddable features. Names of the features are given as CSV (comma-separated values). Supports the following features:

- Camera.FrameRate
- Camera.Exposure.Time
- Camera.Temperature.FPA
- Camera.Temperature.Board
- Camera.Temperature.Scb1
- Camera.Temperature.Scb2
- Camera.Temperature.Scb3

Type

String

Access

Read/Write

Example

```
SI_CHK(SI_SetString(Handle,
 L"Camera.Image.Embedding.FeatureNames",
L"Camera.Temperature.FPA,Camera.Temperature.Scb2,
Camera.Temperature.Scb3"));
```

6.2.62 Camera.Image.Embedding.Row

Description

Feature determines the 0-based row index where the data is embedded within the image. Defaults to 0 after sensor reboot.

Type

Integer

Access

Read/Write

Values

Min: 0

Max: Image height – 1

Example

```
SI_CHK(SI_SetInt(Handle, L"Camera.Image.Embedding.Row", 0));
```

6.2.63 Camera.Image.Height

Description

This feature specifies the final image height (in pixels) coming out of Lumo Sensor SDK with `SI_Wait` or the data callback. It allows for factors such as windowing, binning and transformation. Camera.Image.Height feature is applied at Acquisition.PreStart.Callback.

Type

Integer

Access

Read

Example

```
SI_64 nHeight = 0;
SI_CHK(SI_GetInt(Handle, L"Camera.Image.Height", &nHeight));
```

6.2.64 Camera.Image.IsTestImage

Description

This feature is a readonly boolean feature available for FX50 informing if the detector test pattern is detected. The FX50 detector sends a test pattern during the startup. During that time, `Camera.Image.IsTestImage` is set to true. As soon as the pixel values change, the value is set to false.

Type

Boolean

Access

Read

Example

```
bool bIsTestImage = false;
SI_CHK(SI_GetBool(Handle, L"Camera.Image.IsTestImage",
 &bIsTestImage));
```

6.2.65 Camera.Image.ReadoutMode

Description

Readout mode of the sensor. Use this feature to change the ReadoutMode of FX50 (MY2024).

Type

Enumeration

Access

Read/Write

Values

- 0 (IWR)
- 1 (ITR)

Example

```
SI_SetEnumIndex(Handle, L"Camera.Image.ReadoutMode", 0);  //
Set IWR ReadoutMode
SI_SetEnumIndex(Handle, L"Camera.Image.ReadoutMode", 1);  //
Set ITR ReadoutMode
```

6.2.66 Camera.Image.ReadoutTime

Description

This feature specifies a readout time in milliseconds that it takes for the detector to read the pixel values, before being able to expose further ones. This value is set at Specim and it affects the maximum exposure time in the following way:

$$exposuretime_{max} = \frac{1}{framerate} - time_{readout}$$

Figure 23: Exposure Time

Setting `Camera.ExposureTime` feature will also take this value into consideration.

Type

Float

Access

Read

Example

```
double dReadoutTime = 0.0;
SI_CHK(SI_GetFloat(Handle, L"Camera.Image.ReadoutTime",
 &dReadoutTime));
```

6.2.67 Camera.Image.SizeBytes

Description

This feature specifies the size of the image in bytes coming from Lumo Sensor SDK, when using `SI_Wait` or the data callback. This should be used as size when creating buffers to store acquisition data. Camera.Image.SizeBytes feature is applied at Acquisition.PreStart.Callback.

Type

Integer

Access

Read

Example

```
SI_64 nBytes = 0;
SI_CHK(SI_GetInt(Handle, L"Camera.Image.SizeBytes", &nBytes));

SI_U8* pBuffer = new SI_U8[nBytes];
```

6.2.68 Camera.Image.SizePixels

Description

This feature specifies the size of the image in pixels coming from Lumo Sensor SDK, when using `SI_Wait` or the data callback.Camera.Image.SizePixels feature is applied at Acquisition.PreStart.Callback.

Type

Integer

Access

Read

Example

```
SI_64 nPixels = 0;
SI_CHK(SI_GetInt(Handle, L"Camera.Image.SizePixels",
 &nPixels));

unsigned short* pBuffer = new unsigned short[nPixels];
```

6.2.69 Camera.Image.Transformation

Description

This feature specifies the different image transformations that can be done to the image coming from the sensor. All eight possible transformations are supported either by hardware or by software.

Note:

Some of these transformations will swap the width and the height of the image.

Type

Enumeration

Access

Read/Write

Values

- No transformation
- Transpose
- Rotate 90 CW
- Rotate 90 CCW
- Rotate 180
- Rotate 90CW and Flip Vertical
- Flip Horizontal
- Flip Vertical

Example

```
SI_WC sTransformation[] = L"Flip Vertical";
SI_CHK(SI_SetEnumIndexByString(Handle,
 L"Camera.Image.Transformation", sTransformation));
SI_CHK(SI_SetEnumIndex(Handle, L"Camera.Image.Transformation",
 0)); // No transformation
```

6.2.70 Camera.Image.Width

Description

This feature specifies the final image width (in pixels) coming out of Lumo Sensor SDK, with `SI_Wait` or the data callback. It allows for factors such as windowing, binning and transformation. Camera.Image.Width feature is applied at Acquisition.PreStart.Callback.

Type

Integer

Access

Read

Example

```
SI_64 nWidth = 0;
SI_CHK(SI_GetInt(Handle, L"Camera.Image.Width", &nWidth));
```

6.2.71 Camera.IsConnected

Description

This feature returns whether the sensor is currently connected or not.

Type

Boolean

Access

Read

Example

```
bool bIsConnected = false;
SI_CHK(SI_GetBool(Handle, L"Camera.IsConnected",
 &bIsConnected));
```

6.2.72 Camera.MaxDataNumber

Description

This feature specifies the maximum pixel data number. For certain sensors, the maximum data number is determined by the bit depth. For other sensors, the value is dynamic, and depends on the binning mode used. See Camera.Binning.Spectral and Camera.Binning.Spatial.

This feature automatically changes its value depending on the current binning.

Sensors with multiple detectors have one set of `Camera.MaxDataNumber` for each detector, and `Camera` is suffixed with the detector index, such as:

- `Camera1.MaxDataNumber`
- `Camera2.MaxDataNumber`

Type

Integer

Access

Read

Example

```
SI_64 nMaxDataNumber = 0;
SI_CHK(SI_GetInt(handle, L"Camera.MaxDataNumber",
 &nMaxDataNumber));
```

6.2.73 Camera.Model

Description

This feature specifies the model of the detector.

Type

String

Access

Read

Example

```
SI_WC sModel[50] = {0};
SI_CHK(SI_GetString(Handle, L"Camera.Model", sModel, 50));
```

6.2.74 Camera.MROI.Clear

Description

This feature clears all the MROIs at the current mode.

Type

Command

Access

-

Example

```
SI_CHK(SI_Command(Handle, L"Camera.MROI.Clear"));
```

6.2.75 Camera.MROI.Enable

Description

This feature enables or disables the use of MROI. MROI is turned off by default. Setting this to false will not clear any existing MROIs.

Type

Boolean

Access

Read/Write

Example

```
SI_CHK(SI_SetBool(Handle, L"Camera.MROI.Enable", true));
```

6.2.76 Camera.MROI.MultibandString

This feature specifies a string that sets all multiband MROIs. Setting this feature will clear all the previous MROIs for the multiband mode.

The string syntax is as follows: `T1 H1;T2 H2;T3 H3`, where every new MROI region is separated by a semicolon ';' and the values (in order) within one region are top (T) and height (H), separated by space characters.

The feature parameters are:

Type

String

Access

Read/Write

Example

```
// The example sets two different MROIs.
// One between indices 20-39 and another one between indices
 60-89
SI_SetString(Handle, L"Camera.MROI.MultibandString", L"20
 20;60 30");
```

6.2.77 Camera.NUC

Description

This feature specifies the different NUC modes available. Enumeration values can differ between sensors. Normally, there is no need to change this feature, as `Camera.AutoNUC` will handle automatic NUC table selection when changing exposure times.

Type

Enumeration

Access

Read/Write

Example

```
SI_CHK(SI_SetBool(Handle, L"Camera.AutoNUC", false));
SI_CHK(SI_SetEnumIndex(Handle, L"Camera.NUC", 0));
```

6.2.78 Camera.OpenShutter

Description

This feature will open the shutter.

Note:

If the shutter closes when calling this function, set `Camera.Shutter.IsToggle` to true in the SSP and, after that, your shutter will work as it should. This is related to polarity differences between different individual shutters.

Type

Command

Access

-

Example

```
SI_CHK(SI_Command(Handle, L"Camera.OpenShutter"));
```

6.2.79 Camera.Pixel.Height

Description

This feature specifies the pixel height in nanometers.

Type

Float

Access

Read

Example

```
double dHeight = 0.0;
SI_CHK(SI_GetFloat(Handle, L"Camera.Pixel.Height", &dHeight));
```

6.2.80 Camera.Pixel.Width

Description

This feature specifies the pixel width in nanometers.

Type

Float

Access

Read

Example

```
double dWidth = 0.0;
SI_CHK(SI_GetFloat(Handle, L"Camera.Pixel.Width", &dWidth));
```

6.2.81 Camera.Preprocessing.Enabled

Description

This feature is a simpl on/off feature that specifies whether the camera preprocessing is on or off.

Caution: Turning preprocessing off is not recommended because it will disable also AIE corrections on the camera.

Type

Boolean

Access

Read/Write

Example

```
bool bEnabled = false;
SI_CHK(SI_GetBool(Handle, L"Camera.Preprocessing.Enabled",
 &bEnabled));
```

6.2.82 Camera.Reset

Description

This feature does a software reset on the camera. After resetting the camera, the user needs to close the current camera handle and re-open it. The user needs to start calling the Initialize feature until it is successful. This feature is only available for FX50 (MY2024) and FX120 cameras.

Type

Command

Access

-

Example

```
SI_Command(Handle, L"Camera.Reset");
SI_Close(Handle);
SI_Open(nIndex, &Handle);
// NOTE: Below is just an example of implementation as pseudo-
code
int error = -1;
startTimer() // User-created function to create a timer to not
 loop infinitely
while (error != 0 & & hasTimedOut() == false) // Check for
 success or timeout
 {
       error = SI_Command(Handle, L"Initialize");
       if (error != 0)
       {
```

```
            Sleep(5); // Sleep for 5 seconds
        }
    }
```

6.2.83 Camera.Sensor.Height

Description

This feature specifies the maximum detector height in pixels for the sensor.

Type

Integer

Access

Read

Example

```
SI_64 nHeight = 0;
SI_CHK(SI_GetInt(Handle, L"Camera.Sensor.Height", &nHeight));
```

6.2.84 Camera.Sensor.Width

Description

This feature specifies the maximum detector width in pixels for the sensor.

Type

Integer

Access

Read

Example

```
SI_64 nWidth = 0;
SI_CHK(SI_GetInt(Handle, L"Camera.Sensor.Width", &nWidth));
```

6.2.85 Camera.Shutter.IsConnected

Description

This feature specifies whether the shutter is connected or not. Shutter devices will be automatically reconnected if they get disconnected.

With GX17, the shutter is always disconnected.

Type

Boolean

Access

Read

Example

```
bool bIsConnected = false;
SI_CHK(SI_GetBool(Handle, L"Camera.Shutter.IsConnected",
 &bIsConnected));
```

6.2.86 Camera.Shutter.IsOpen

Description

This feature specifies whether the shutter is open or closed.

Type

Boolean

Access

Read

Example

```
bool bIsOpen = false;
SI_CHK(SI_GetBool(Handle, L"Camera.Shutter.IsOpen",
 &bIsOpen));
```

6.2.87 Camera.Shutter.IsToggle

Description

This feature specifies whether the polarity of the shutter needs to be swapped (true) or not (false).

Type

Boolean

Access

Read/Write

Example

```
SI_CHK(SI_SetBool(Handle, L"Camera.Shutter.IsToggle", true));
```

6.2.88 Camera.Stabilization.Enabled

Description

Enabling the feature will start the stabilization routine which will prevent other functionality (like frame rate, exposure time and shutter) to be used while routine is enabled. It will automatically disable itself after finished. It can also be manually disabled at any time.

Type

Boolean

Access

Read/Write

Example

```
SI_CHK(SI_SetBool(Handle, L"Camera.Stabilization.Enabled",
 true));
```

6.2.89 Camera.Stabilization.IsStable

Description

Feature will be true if the stabilization routine has finished successfully and the image is stabilized. Otherwise this feature will be false. The signal is considered stable if the signal levels don’t change more than 0.2DN/sec for 3 minutes. Default value is false after sensor reboot. Feature is optional and not mandatory to use.

Type

Boolean

Access

Read

Example

```
bool bIsEnabled = false;
SI_CHK(SI_GetBool(Handle, L"Camera.Stabilization.IsEnabled",
 &bIsEnabled));
```

6.2.90 Camera.Temperature

Description

This feature specifies the detector temperature (usually in Kelvin). Use the following equation to convert Kelvin to Celcius:

$T_C = T_K - 273.15$

Figure 24: Kelvin to Celcius

Type

Float

Access

Read

Example

```
double dTemperature = 0.0;
SI_CHK(SI_GetFloat(Handle, L"Camera.Temperature",
 &dTemperature));
```

6.2.91 Camera.Trigger.Mode

Description

This feature specifies the triggering mode used.

- When set to `Internal`, the camera is doing the acquisition triggering.
- When set to `External` the camera waits for an external source to give the trigger signal for acquisition.
- `External Exposure` mode is not always supported, but it means that the high time of the trigger signal defines the exposure time.

Type

Enumeration

Access

Read / Write

Values

- Internal (default)
- External

Example

```
SI_CHK(SI_SetEnumIndexByString(Handle, L"Camera.Trigger.Mode",
 L"Internal"));
```

6.2.92 Camera.Wavelength.Start

Description

This feature specifies the starting wavelength for the automatic pseudo-wavelength generation.

Type

Float

Access

Read/write

Example

```
double dStart = 0.0;
SI_GetFloat(Handle, L"Camera.WavelengthTable.Start",
 &dStart));
```

6.2.93 Camera.Wavelength.Stop

Description

This feature specifies the stopping wavelength for the automatic pseudo-wavelength generation.

Type

Float

Access

Read / Write

Example

```
double dStop = 0.0;
SI_CHK(SI_GetFloat(Handle, L"Camera.Wavelength.Stop",
 &dStop));
```

6.2.94 Camera.WavelengthTable

Description

This feature specifies the wavelength table for the sensor. If a valid calibration pack exists and has proper wavelength information, it will be used to populate this feature. Otherwise, `Camera.Wavelength.Start` and `Camera.Wavelength.Stop` features will be used to create pseudo-wavelengths.

Type

Enumeration

Access

Read

Example

```
int nCount = 0;
SI_CHK(SI_GetEnumCount(Handle, L"Camera.WavelengthTable",
 &nCount));

SI_WC wcWavelength[100];
double* pWavelengths = new double[nCount];

for (int n = 0; n < nCount; n++)
{
 SI_CHK(SI_GetEnumStringByIndex(Handle,
 L"Camera.WavelengthTable", n, wcWavelength, 100));
 pWavelengths[n] = wcstod(wcWavelength, 0);
}
```

```
// ...

delete[] pWavelengths;
```

6.2.95 Dalsa.CameraFile

Description

This feature controls the path of the Dalsa camera configuration file (`.ccf`) to be used with the sensor. The feature value is used when the Initialize feature is called. The feature must be hardcoded in the SSP and there should be no reason to change it with SI_SetString.

Type

String

Access

Read / Write

Example

```
SI_WC wcFilePath[255] = {0};
SI_CHK(SI_GetString(Handle, L"Dalsa.CameraFile", wcFilePath,
 255));
```

6.2.96 Detector.Image.Height

Description

This internal feature returns the height of the detector frame. In single-detector setups, it always equals to Camera.Image.Height. In multi-detector setups, the value of the feature may differ between detectors - for example, *Detector1.Image.Height* differs from *Detector2.Image.Height*.

This feature is only needed in multi-detector setups when there is a need to differiante the different detectors from the combined frame data.

Type

Integer

Access

Read

Example

```
SI_64 nVNIRHeight = 0;
SI_64 nSWIRHeight = 0;
SI_CHK(SI_GetInt(Handle, L"Detector1.Image.Height",
 &nVNIRHeight));
SI_CHK(SI_GetInt(Handle, L"Detector2.Image.Height",
 &nSWIRHeight));
```

6.2.97 Detector.Image.Width

Description

This internal feature returns the width of the detector frame and it always equals to Camera.Image.Width.

Type

Integer

Access

Read

Example

```
SI_64 nWidth = 0;
SI_CHK(SI_GetInt(Handle, L"Detector.Image.Width", &nWidth));
```

6.2.98 DeviceCount

Description

This feature specifies the number of SSPs available. This feature is only accessible by using the `SI_SYSTEM` handle. This feature becomes available after a successful SI_Open call.

Type

Integer

Access

Read

Example

```
SI_64 nDeviceCount = 0;
SI_CHK(SI_GetInt(SI_SYSTEM, L"DeviceCount", &nDeviceCount));
```

6.2.99 DeviceDescription

Description

This feature specifies the descriptions of all the SSPs available, as an enumerated list. This feature is only accessible by using the `SI_SYSTEM` handle. This feature becomes available after a successful SI_Open call.

Type

Enumeration

Access

Read

Example

```
int nDeviceCount = 0;
SI_CHK(SI_GetInt(SI_SYSTEM, L"DeviceCount", &nDeviceCount));

// Iterate through each devices
for (int n = 0; n < nDeviceCount; n++)
{
  SI_WC sDescription[255] = {0};
  SI_CHK(SI_GetEnumStringByIndex(SI_SYSTEM,
 L"DeviceDescription", n, sDescription, 255));
}
```

6.2.100 DeviceName

Description

This feature specifies the names of all the SSPs available, as an enumerated list. This feature is only accessible by using the `SI_SYSTEM` handle. This feature becomes available after a successful SI_Open call.

Type

Enumeration

Access

Read

Example

```
int nDeviceCount = 0;
SI_CHK(SI_GetInt(SI_SYSTEM, L"DeviceCount", &nDeviceCount));

// Iterate through each devices
for (int n = 0; n < nDeviceCount; n++)
{
  SI_WC sName[255] = {0};
  SI_CHK(SI_GetEnumStringByIndex(SI_SYSTEM, L"DeviceName", n,
 sName, 255));
}
```

6.2.101 FeatureList

Description

This feature provides an enumerated list of all available features.

Type

Enumeration

Access

Read

Example

```
int nFeatureCount = 0;
SI_CHK(SI_GetGetEnumCount(hHandle, L"FeatureList",
 &nFeatureCount));

// Iterate through each feature
for (int n = 0; n < nFeatureCount; n++)
{
  SI_WC sFeature[255] = {0};
  SI_CHK(SI_GetEnumStringByIndex(hHandle, L"FeatureList", n,
 sFeature, 255));
}
```

6.2.102 Grabber.Channel

Description

Specifies the channel to be used for the grabber device. The value differs for each sensor, depending on the grabber used.

The Grabber.Channel feature is empty by default. Most of the time, the channel is automatically detected. However, if you have multiple grabbers of the same type, you need to tell the SDK to which device it should connect.

You can set `Grabber.Channel` either from the SSP, or using SI_SetString.

Warning: A wrong parameter for `Grabber.Channel` will cause the Initialize command to fail.

Note: The list of available channels is enumerated in the log file after Initialization sequence has been completed.

Type

String

Access

Read / Write

Example

```
SI_H Handle = 0;
SI_CHK(SI_Load(L""));
SI_CHK(SI_Open(0, &Handle));
SI_CHK(SI_SetString(Handle, L"Grabber.Channel", L"img0::0"));
SI_CHK(SI_Command(Handle, L"Initialize"));
```

6.2.103 Grabber.Channels

Description

This feature will be automatically populated by all the found grabber devices. If Grabber.Channel is empty, the first index of Grabber.Channels will be used as the channel to connect to.

This feature is normally populated after SI_Open is called.

Type

Enumeration

Access

Read

6.2.104 Grabber.PlaybackFile

Description

This file-reader specific feature specifies the full path of either the header or the raw file to be opened by the file reader. Both files have to exist on the same path with the same base name (`xxx.raw` and `xxx.hdr`). If this feature is left empty, a 2-frame gradient test image is created.

The header file must have the following properties:

- samples
- bands
- data type
- hroi
- vroi
- binning

Type

String

Access

Read/Write

Example

```
SI_CHK(SI_SetString(Handle, L"Grabber.PlaybackFile", "C:/data/
test.raw"));
SI_CHK(SI_Command(Handle, L"Initialize"));
```

6.2.105 Imperx.Tap1.AnalogOffset

Description

This feature specifies the analog offset value used for the first tap on the sensor. This feature differs depending on the grabbers and computer, and must be separately set for different hardware configurations to get an even signal in both taps.

Type

Integer

Access

Read / Write

Example

```
SI_CHK(SI_SetInt(Handle, L"Imperx.Tap1.AnalogOffset", 0));

SI_64 nOffset = 0;
SI_CHK(SI_GetInt(Handle, L"Imperx.Tap1.AnalogOffset",
 &nOffset));
```

6.2.106 Imperx.Tap2.AnalogGain

Description

This feature specifies the analog gain value used for the second tap on the sensor. This feature differs depending on the grabbers and computer, and must be separately set for different hardware configurations to get an even signal in both taps.

Type

Integer

Access

Read / Write

Example

```
SI_CHK(SI_SetInt(Handle, L"Imperx.Tap2.AnalogGain", 0));

SI_64 nGain = 0;
SI_CHK(SI_GetInt(Handle, L"Imperx.Tap2.AnalogGain", &nGain));
```

6.2.107 Imperx.Tap2.AnalogOffset

Description

This feature specifies the analog offset value used for the second tap on the sensor. This feature differs depending on the grabbers and computer, and must be separately set for different hardware configurations to get an even signal in both taps.

Type

Integer

Access

Read / Write

Example

```
SI_CHK(SI_SetInt(Handle, L"Imperx.Tap2.AnalogOffset", 0));

SI_64 nOffset = 0;
SI_CHK(SI_GetInt(Handle, L"Imperx.Tap2.AnalogOffset",
 &nOffset));
```

6.2.108 Imperx.Tap1.AnalogGain

Description

This feature specifies the analog gain value used for the first tap on the sensor. This feature differs depending on the grabbers and computer, and must be separately set for different hardware configurations to get an even signal in both taps.

Type

Integer

Access

Read / Write

Example

```
SI_CHK(SI_SetInt(Handle, L"Imperx.Tap1.AnalogGain", 0));

SI_64 nOffset = 0;
SI_CHK(SI_GetInt(Handle, L"Imperx.Tap1.AnalogGain",
 &nOffset));
```

6.2.109 Initialize

Description

This feature is called after SI_Open, and will initialize the sensor. This command fails if the sensor SSP is incorrectly configured. See Setting up SSP for the First Time.

Type

Command

Access

-

Example

```
SI_H Handle = 0;
SI_CHK(SI_Open(0, &Handle));
SI_CHK(SI_Command(Handle, L"Initialize"));
```

6.2.110 IsInitialized

Description

This feature specifies whether the sensor has been initialized successfully.

Type

Boolean

Access

Read

Example

```
SI_H Handle = 0;
SI_BOOL bInitialized = false;
SI_CHK(SI_Load(L""));
SI_CHK(SI_Open(3, &Handle));
SI_CHK(SI_Command(Handle, L"Initialize"));
SI_CHK(SI_GetBool(Handle, L"IsInitialized", &bInitialized));
```

6.2.111 ProfilesDirectory

Description

This feature specifies the folder path where Lumo Sensor SDK looks for SSP files. If you need to change this feature, you must do it before calling SI_Load.

If you want to load the SSP files from a different folder than the default, change this feature to a valid path.

The default path for Windows 10 (and later) it is `C:/Users/Public/Documents/Specim`.

Type

String

Access

Read / Write

Example

```
SI_CHK(SI_SetString(SI_SYSTEM, L"ProfilesDirectory", L"C:/ssp"));
SI_CHK(SI_Load(L""));
```

6.2.112 Scb.Channel

Description

This feature configures the *COM* port to be used for the *Sensor Control Board*. If left empty, the *Sensor Control Board* auto-connect tries to find the correct *COM* port to be used. Auto-connect happens when calling Initialize

Changing this feature needs to happen before the call to Initialize.

Type

String

Access

Read / Write

Example

```
SI_CHK(SI_SetString(Handle, L"Scb.Channel", "COM5"));
```

6.2.113 Scb.Channels

Description

This feature contains all the available *COM* ports where the *Sensor Control Board* can be connected to.

Type

Enumeration

Access

Read

Example

```
int nCount = 0;
SI_CHK(SI_GetEnumCount(Handle, L"Scb.Channels", &nCount));

for (int n = 0; n < nCount; n++)
{
  SI_WC sChannel[10] = {0}
  SI_CHK(SI_GetEnumStringByIndex(Handle, L"Scb.Channels", n,
 sChannel, 10));
}
```

6.2.114 Scb.IsOpen

Description

This feature returns whether the *Sensor Control Board* has been opened successfully or not.

Type

Boolean

Access

Read

Example

```
bool bIsOpen = false;
SI_CHK(SI_GetBool(Handle, L"Scb.IsOpen", &bIsOpen));
```

6.2.115 Scb.Temperature.Channel1

Description

This feature returns the temperature of Channel1. The temperature unit type is controlled by Scb.Temperature.Unit.

Type

Float

Access

Read

Example

```
double dTemperature = 0.0;
SI_CHK(SI_GetFloat(Handle, L"Scb.Temperature.Channel1",
 &dTemperature));
```

6.2.116 Scb.Temperature.Channel2

Description

This feature returns the temperature of Channel2. The temperature unit type is controlled by Scb.Temperature.Unit.

Type

Float

Access

Read

Example

```
double dTemperature = 0.0;
SI_CHK(SI_GetFloat(Handle, L"Scb.Temperature.Channel2",
 &dTemperature));
```

6.2.117 Scb.Temperature.Channel3

Description

This feature returns the temperature of Channel3. The temperature unit type is controlled by Scb.Temperature.Unit.

Type

Float

Access

Read

Example

```
double dTemperature = 0.0;
SI_CHK(SI_GetFloat(Handle, L"Scb.Temperature.Channel3",
 &dTemperature));
```

6.2.118 Scb.Temperature.Channel4

Description

This feature returns the temperature of Channel4. The temperature unit type is controlled by Scb.Temperature.Unit.

Type

Float

Access

Read

Example

```
double dTemperature = 0.0;
SI_CHK(SI_GetFloat(Handle, L"Scb.Temperature.Channel4",
 &dTemperature));
```

6.2.119 Sensor.Detector.Type

Description

This feature contains the type of the detector, such as *VNIR*. The valus should be hardcoded in the *SSP*.

Type

String

Access

Read

Example

```
SI_WC sType[20] = {0};
SI_CHK(SI_GetString(Handle, L"Sensor.Detector.Type", sType,
 20));
```

6.2.120 Sensor<insert index>.IsOpen

Description

This feature specifies whether a sensor at a certain index has been opened successfully using SI_Open call. This feature is only accessible by using the `SI_SYSTEM` handle.

Note: The `<insert index>` in the name refers to the same index used with SI_Open. The maximum value of this index is always DeviceCount – 1.

Type

Boolean

Access

Read

Example

```
SI_CHK(SI_Load(L""));
```

```
SI_CHK(SI_RegisterFeatureCallback(SI_SYSTEM,
 L"Sensor3.IsOpen", onFeatureCallback, this));

//When SI_Open for index 3 finishes successfully,
 onFeatureCallback is called automatically.
SI_H Handle = 0;
SI_CHK(SI_Open(3, &Handle));
```

6.2.121 Sensor.SerialNumber

Description

This feature holds your sensor serial number.

Type

String

Access

Read

Example

```
SI_WC sSerialNumber[255] = {0};
SI_CHK(SI_GetString(Handle, L"Sensor.SerialNumber",
 sSerialNumber, 255));
```

6.2.122 Shutter.Channel

Description

This feature specifies the channel to be used for the shutter. If you leave this value empty, the first detected shutter will be selected.

You can set `Shutter.Channel` either from the SSP or using SI_SetString.

Warning: A wrong parameter for `Shutter.Channel` will cause the Initialize command to fail.

The shutter is detected by the operating system as an FTDI device. The virtual COM has a serial number, such as **SP350000A**.

The **shutter001** module will list all the FTDI devices with serial number starting with **SP** and ending with **A**, and add them to Shutter.Channels. The number between **SP** and **A** is the serial number of your sensor.

If the shutter device is installed, and detected in Windows **Device Manager**, but detection fails, you most likely have an older sensor with the FTDI device not re-programmed with the serial number information. In this case, proceed as follows to find the correct FTDI serial number for your shutter:

1. Disconnect the shutter USB cable.
2. Open Windows **Device Manager**.
3. Expand Ports (COM & LPT).
4. Connect the shutter USB, and pay attention to the new COM detected.
5. Right click the new COM port, and select **Properties**.
6. Select the **Details** tab.
7. Locate the **Property** drop down menu.
8. Select **Device Instance Path**.
9. Check the **Value**. You will see a string such as `FTDIBUF \VID_0403+PID_6001+`**`DEMOUNITB`**`\0000`.

10. The part of the string ending with B, **DEMOUNITB** in the example above, is the camera auxiliary COM port for the scanner or the GPS.
11. The shutter is on port A. With the example above, the shutter port would be **DEMOUNITA**.
12. Set the channel from the SSP or using SI_SetString.

Type

String

Access

Read / Write

Example

```
SI_CHK(SI_SetString(Handle, L"Shutter.Channel",
 L"SP350000A"));
SI_CHK(SI_Command(Handle, L"Initialize"));
```

6.2.123 Shutter.Channels

Description

This feature specifies the serial numbers of all the available shutter devices.

Note:

This will only detect shutter devices whose serial number starts with `SP`.

Type

Enumeration

Access

Read

Example

```
SI_CHK(SI_RegisterFeatureCallback(Handle, L"Shutter.Channels",
 MyCallback, this));
```

6.2.124 VCECLB.CameraConfig

Description

This *Imperx FrameLink ExpressCard* grabber-specific feature specifies the full path of the camera configuration file (`.cxf`) required for the specific camera in use.

Type

String

Access

Read/Write

6.2.125 Xenics.CalibrationFile

Description

This feature lists the full paths of all available Xenics calibration files. If a calibration pack is used with Xenics calibration files, this feature is automatically populated. Otherwise, the order of the files listed here should match the order of the corresponding exposure times in Xenics.CalibrationRange. Do not modify this feature when using a calibration pack. Otherwise, you can modify it through the SSP.

Type

Enumeration

Access

Read

Example

```
<feature>
  <name>Xenics.CalibrationRange</name>
  <module>grabber</module>
  <type>enumeration</type>
  <values>
    <value>1</value>
    <value>5</value>
    <value>10</value>
  </values>
</feature>

<feature>
  <name>Xenics.CalibrationFile</name>
  <module>grabber</module>
  <type>enumeration</type>
  <values>
    <value>d:/calibrations/Xeneth_LG_1ms.xca</value>
    <value>d:/calibrations/Xeneth_LG_5ms.xca</value>
    <value>d:/calibrations/Xeneth_LG_10ms.xca</value>
  </values>
</feature>
```

6.2.126 Xenics.CalibrationFolder

Description

This feature specifies the full path of a folder that is used to extract calibration files from the calibration pack, before loading them to the sensor. The folder is created if it does not exist. If left empty, the current directory is used. This feature should be changed through the SSP.

Type

String

Access

Read / Write

6.2.127 Xenics.CalibrationOutputFolder

Description

This feature specifies the full path to a folder that it used to extract calibration files from the *Calibration Pack*, before loading them to the sensor. The folder is created if it does not exist. If left empty, the current directory is used. This feature should be changed from the SSP.

Type

String

Access

Read / Write

Example

```
SI_CHK(SI_SetString(Handle, L"Xenics.CalibrationOutputFolder",
 L"d:/calibrations"));
```

6.2.128 Xenics.CalibrationRange

Description

This feature specifies the exposure time range used with the currently available Xenics calibration files in either Xenics.CalibrationFile enumeration or within the calibration pack specified by Camera.CalibrationPack. This feature should be updated through the SSP.

Type

Enumeration

Access

Read

Example

```
<!--
  Let's assume that:
  - CalibrationPack is loaded with the following Xenics
 calibration files.
  - Xeneth_LG_1ms.xca
  - Xeneth_LG_5ms.xca
  - Xeneth_LG_10ms.xca

  Then the Xenics.CalibrationRange feature in the SSP should
 look like this.
-->

<feature>
  <name>Xenics.CalibrationRange</name>
  <module>grabber</module>
  <type>enumeration</type>
  <values>
    <value>1</value>
    <value>5</value>
    <value>10</value>
  </values>
</feature>
```

6.2.129 Xenics.TargetTemperature

Description

This feature specifies the target temperature that the sensor cooling is trying to achieve, if Xenics.TEC is enabled.

Type

Float

Access

Read / Write

Example

```
SI_CHK(SI_SetFloat(Handle, L"Xenics.TargetTemperature",
 25.0));

double dTemperature = 0.0;
SI_CHK(SI_GetFloat(Handle, L"Xenics.TargetTemperature",
 &dTemperature));
```

6.3 Features per Sensor Profile

6.3.1 Fenix Feature List

Below is a list of all available features for Fenix.

Table 2: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera1.ExposureTime
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.BPR

Feature name
Camera.BPR.BadPixelCount
Camera.BPR.BadPixelPercentage
Camera.BPR.CreateMap
Camera.BPR.CreateMap.InProgress
Camera.BPR.Gain.Max
Camera.BPR.Gain.Min
Camera.BPR.IsHardware
Camera.BPR.MapAvailable
Camera.BPR.Noise.Max
Camera.BPR.Offset.Max
Camera.BPR.Offset.Min
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Analog
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear

Feature name
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1
Scb.Temperature.Channel2
Scb.Temperature.Channel3
Scb.Temperature.Channel4

6.3.2 Fenix1k Feature List

Below is a list of all available features for Fenix1k.

Table 3: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera1.ExposureTime
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.BPR
Camera.BPR.BadPixelCount
Camera.BPR.BadPixelPercentage
Camera.BPR.CreateMap
Camera.BPR.CreateMap.InProgress

Feature name
Camera.BPR.Gain.Max
Camera.BPR.Gain.Min
Camera.BPR.IsHardware
Camera.BPR.MapAvailable
Camera.BPR.Noise.Max
Camera.BPR.Offset.Max
Camera.BPR.Offset.Min
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Analog
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter

Feature name
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1
Scb.Temperature.Channel2
Scb.Temperature.Channel3
Scb.Temperature.Channel4

6.3.3 Fenix1k with Epix Feature List

Below is a list of all available features for Fenix1k with Epix.

Table 4: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error

Feature name
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera1.ExposureTime
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.BPR
Camera.BPR.BadPixelCount
Camera.BPR.BadPixelPercentage
Camera.BPR.CreateMap
Camera.BPR.CreateMap.InProgress
Camera.BPR.Gain.Max
Camera.BPR.Gain.Min
Camera.BPR.IsHardware
Camera.BPR.MapAvailable
Camera.BPR.Noise.Max

Feature name
Camera.BPR.Offset.Max
Camera.BPR.Offset.Min
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Analog
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected

Feature name
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1
Scb.Temperature.Channel2
Scb.Temperature.Channel3
Scb.Temperature.Channel4

6.3.4 Filereader Feature List

Below is a list of all available features for Filereader.

Table 5: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback

Feature name
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model

Feature name
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.SerialNumber
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.PlaybackFile
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber

6.3.5 FX10 and FX17 Feature List

Below is a list of all available features for FX10 and FX17.

Table 6: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback

Feature name
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Digital
Camera.Image.AberrationCorrection.Enabled
Camera.Image.AberrationCorrection.LoadFromFlash
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels

Feature name
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.SerialNumber
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Grabber.Channel
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber

6.3.6 FX50 Feature List

Below is a list of all available features for FX50.

Table 7: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error

Feature name
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average
Camera.Binning.Average.IsHardware
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Image.Embedding.Enabled
Camera.Image.Embedding.Featurenames
Camera.Image.Embedding.Row

Feature name
Camera.Image.Height
Camera.Image.IsTestImage
Camera.Image.ReadoutMode
Camera.Image.ReadoutTime
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Preprocessing.Enabled
Camera.Reset
Camera.Sensor.Height
Camera.Sensor.Width
Camera.SerialNumber
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Stabilization.Enabled
Camera.Stabilization.IsStable
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Grabber.Channel
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber

6.3.7 FX120 Feature List

Below is a list of all available features for FX120.

Table 8: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average
Camera.Binning.Average.IsHardware
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count

Feature name
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Image.Embedding.Enabled
Camera.Image.Embedding.Featurenames
Camera.Image.Embedding.Row
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.ReadoutTime
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Preprocessing.Enabled
Camera.Reset
Camera.Sensor.Height
Camera.Sensor.Width
Camera.SerialNumber
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Stabilization.Enabled
Camera.Stabilization.IsStable
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable

Feature name
Grabber.Channel
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber

6.3.8 GX17 Feature List

Below is a list of all available features for GX17.

Table 9: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth

Feature name
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.CheckTemperatures
Camera.Count
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Digital
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.MaxDataNumber
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.MultibandString
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Grabber.Channel
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber
Xenics.CalibrationFile
Xenics.CalibrationFolder

Feature name
Xenics.CalibrationRange

6.3.9 Kestrel10 with Epix Feature List

Below is a list of all available features for Kestrel10 with Epix.

Table 10: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.SetupFilePath
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera1.ExposureTime
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral

Feature name
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Digital
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.SerialNumber
Camera.Shutter.IsConnected

Feature name
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1
Scb.Temperature.Channel2
Scb.Temperature.Channel3
Scb.Temperature.Channel4
Sensor.Detector.Type
Sensor.SerialNumber

6.3.10 Kestrel16 with Epix Feature List

Below is a list of all available features for Kestrel16 with Epix.

Table 11: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback

Feature name
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.SetupFilePath
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera1.ExposureTime
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM

Feature name
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.SerialNumber
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1

Feature name
Scb.Temperature.Channel2
Scb.Temperature.Channel3
Scb.Temperature.Channel4
Sensor.Detector.Type
Sensor.SerialNumber

6.3.11 Kestrel16 with NI Feature List

Below is a list of all available features for Kestrel16 with NI.

Table 12: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera1.ExposureTime
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Camera.Binning.Average

Feature name
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Image.Height
Camera.Image.ReadoutTime
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height

Feature name
Camera.Sensor.Width
Camera.SerialNumber
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
NiImaq.CameraFile
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1
Scb.Temperature.Channel2
Scb.Temperature.Channel3
Scb.Temperature.Channel4
Sensor.Detector.Type
Sensor.SerialNumber

6.3.12 LWIR with NI Feature List

Below is a list of all available features for *LWIR with NI*.

Table 13: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter

Feature name
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.FrameRate
Camera.FWHM
Camera.Image.Height
Camera.Image.ReadoutTime
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected

Feature name
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.SerialNumber
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
NiImaq.CameraFile
NiImaq.Mode
Sensor.Detector.Type
Sensor.SerialNumber

6.3.13 MWIR Flir with GigE Feature List

Below is a list of all available features for *MWIR Flir with GigE*.

Table 14: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset

Feature name
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.AutoNUC
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.BPR
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.CheckTemperatures
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels

Feature name
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.NUC
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber

6.3.14 MWIR Flir with NI Feature List

Below is a list of all available features for *MWIR Flir with NI*.

Table 15: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate

Feature name
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.AutoNUC
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.BPR
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto

Feature name
Camera.FrameRate
Camera.FWHM
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.NUC
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.SerialNumber
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber

6.3.15 OWL3 with Dalsa Feature List

Below is a list of all available features for *OWL3 with Dalsa*.

Table 16: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.SetupFilePath
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures

Feature name
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Dalsa.CameraFile
Detector.Image.Height
Detector.Image.Width

Feature name
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1
Scb.Temperature.Channel2
Scb.Temperature.Channel3
Scb.Temperature.Channel4
Sensor.Detector.Type

6.3.16 OWL3 with NI Feature List

Below is a list of all available features for *OWL3 with NI*.

Table 17: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.SetupFilePath
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height

Feature name
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera1.ExposureTime
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear

Feature name
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Camera1.ExposureTime
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1
Scb.Temperature.Channel2
Scb.Temperature.Channel3
Scb.Temperature.Channel4
Sensor.Detector.Type
Sensor.SerialNumber

6.3.17 RGB with NI Feature List

Below is a list of all available features for *RGB with NI*.

Table 18: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CloseShutter
Camera.Count
Camera.Debug

Feature name
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.SerialNumber
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize

Feature name
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber

6.3.18 RHNIR SCB with Framelink Feature List

Below is a list of all available features for *RHNIR with Framelink*.

Table 19: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.SetupFilePath
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera1.ExposureTime
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Camera.AutoNUC
Camera.Binning.Average

Feature name
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Analog
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.NUC
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width

Feature name
Camera.Sensor.Height
Camera.Sensor.Width
Camera.SerialNumber
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1
Scb.Temperature.Channel2
Scb.Temperature.Channel3
Scb.Temperature.Channel4
Sensor.Detector.Type
Sensor.SerialNumber
VCECLB.CameraConfig

6.3.19 RHNIR SCB with NI Feature List

Below is a list of all available features for *RHNIR SCB with NI*.

Table 20: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.FrameCounter

Feature name
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.SetupFilePath
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera1.ExposureTime
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Camera.AutoNUC
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug

Feature name
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FirmwareVersion
Camera.FrameRate
Camera.FWHM
Camera.Gain.Analog
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.NUC
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel

Feature name
Grabber.Channels
Initialize
IsInitialized
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1
Scb.Temperature.Channel2
Scb.Temperature.Channel3
Scb.Temperature.Channel4
Sensor.Detector.Type
Sensor.SerialNumber

6.3.20 RHNIR with NI Feature List

Below is a list of all available features for *RHNIR with NI*.

Table 21: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left

Feature name
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.AutoNUC
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Analog
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString

Feature name
Camera.NUC
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Shutter.IsToggle
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber
Shutter.Channel
Shutter.Channels

6.3.21 SWIR3 with Framelink Feature List

Below is a list of all available features for *SWIR3 with Framelink*.

Table 22: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset

Feature name
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.SetupFilePath
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera1.ExposureTime
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.BPR
Camera.BPR.BadPixelCount
Camera.BPR.BadPixelPercentage
Camera.BPR.CreateMap
Camera.BPR.CreateMap.InProgress
Camera.BPR.Gain.Max
Camera.BPR.Gain.Min
Camera.BPR.IsHardware
Camera.BPR.MapAvailable
Camera.BPR.Noise.Max
Camera.BPR.Offset.Max

Feature name
Camera.BPR.Offset.Min
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Temperature

Feature name
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1
Scb.Temperature.Channel2
Scb.Temperature.Channel3
Scb.Temperature.Channel4
Sensor.Detector.Type
Sensor.SerialNumber
VCECLB.CameraConfig

6.3.22 SWIR3 with NI Feature List

Below is a list of all available features for *SWIR3 with NI*.

Table 23: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback

Feature name
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.SetupFilePath
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera1.ExposureTime
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.BPR
Camera.BPR.BadPixelCount
Camera.BPR.BadPixelPercentage
Camera.BPR.CreateMap
Camera.BPR.CreateMap.InProgress
Camera.BPR.Gain.Max
Camera.BPR.Gain.Min
Camera.BPR.IsHardware
Camera.BPR.MapAvailable
Camera.BPR.Noise.Max
Camera.BPR.Offset.Max
Camera.BPR.Offset.Min
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded

Feature name
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Analog
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.NUC
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start

Feature name
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1
Scb.Temperature.Channel2
Scb.Temperature.Channel3
Scb.Temperature.Channel4
Sensor.Detector.Type
Sensor.SerialNumber

6.3.23 SWIR with Framelink Feature List

Below is a list of all available features for *SWIR with Framelink*.

Table 24: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.DroppedFrames
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size

Feature name
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.AutoNUC
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.BPR
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Analog
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width

Feature name
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.NUC
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Shutter.IsToggle
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber
Shutter.Channel
Shutter.Channels
VCECLB.CameraConfig

6.3.24 SWIR with NI Feature List

Below is a list of all available features for *SWIR with NI*.

Table 25: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.AutoNUC
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.BPR
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter

Feature name
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Analog
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.NUC
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Shutter.IsToggle
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height

Feature name
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber
Shutter.Channel
Shutter.Channels

6.3.25 Universal with NI Feature List

Below is a list of all available features for *Universal with NI*.

Table 26: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average

Feature name
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.FWHM
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize

Feature name
IsInitialized
NiImaq.CameraFile
Sensor.Detector.Type

6.3.26 VLNIR SCB with NI Feature List

Below is a list of all available features for *VLNIR SCB with NI*.

Table 27: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera1.ExposureTime
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Camera.AutoNUC
Camera.Binning.Average
Camera.Binning.HardwareModes

Feature name
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.BPR
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Digital
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.NUC
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width

Feature name
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1
Scb.Temperature.Channel2
Scb.Temperature.Channel3
Scb.Temperature.Channel4
Sensor.Detector.Type

6.3.27 VLNIR with NI Feature List

Below is a list of all available features for *VLNIR with NI*.

Table 28: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback

Feature name
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.AutoNUC
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.BPR
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Digital

Feature name
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.NUC
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Shutter.IsToggle
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber
Shutter.Channel

Feature name
Shutter.Channels

6.3.28 VNIR Basler GigE with NI Feature List

Below is a list of all available features for *VNIR Basler GigE*.

Table 29: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.CheckTemperatures

Feature name
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Analog
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Shutter.IsToggle
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height

Feature name
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber
Shutter.Channel
Shutter.Channels

6.3.29 VNIR HS SCSB with Framelink Feature List

Below is a list of all available features for *VNIR HS SCB with Framelink*.

Table 30: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera1.ExposureTime

Feature name
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter

Feature name
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Imperx.Tap1.AnalogGain
Imperx.Tap1.AnalogOffset
Imperx.Tap2.AnalogGain
Imperx.Tap2.AnalogOffset
Initialize
IsInitialized
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1
Scb.Temperature.Channel2
Scb.Temperature.Channel3
Scb.Temperature.Channel4
Sensor.Detector.Type
Sensor.SerialNumber
VCECLB.CameraConfig

6.3.30 VNIR HS SCB with NI Feature List

Below is a list of all available features for *VNIR HS SCB with NI*.

Table 31: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera1.ExposureTime
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels

Feature name
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.SerialNumber
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height

Feature name
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Imperx.Tap1.AnalogGain
Imperx.Tap1.AnalogOffset
Imperx.Tap2.AnalogGain
Imperx.Tap2.AnalogOffset
Initialize
IsInitialized
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1
Scb.Temperature.Channel2
Scb.Temperature.Channel3
Scb.Temperature.Channel4
Sensor.Detector.Type
Sensor.SerialNumber

6.3.31 VNIR HS with Framelink Feature List

Below is a list of all available features for *VNIR HS with Framelink*.

Table 32: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size

Feature name
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model

Feature name
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.SerialNumber
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Shutter.IsToggle
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Imperx.Tap1.AnalogGain
Imperx.Tap1.AnalogOffset
Imperx.Tap2.AnalogGain
Imperx.Tap2.AnalogOffset
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber
Shutter.Channel
Shutter.Channels
VCECLB.CameraConfig

6.3.32 VNIR HS with NI Feature List

Below is a list of all available features for *VNIR HS with NI*.

Table 33: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter

Feature name
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.SerialNumber
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Shutter.IsToggle
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width

Feature name
Grabber.Channel
Grabber.Channels
Imperx.Tap1.AnalogGain
Imperx.Tap1.AnalogOffset
Imperx.Tap2.AnalogGain
Imperx.Tap2.AnalogOffset
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber
Shutter.Channel
Shutter.Channels

6.3.33 VNIR PFD 2080 Custom 1x1 with NI Feature List

Below is a list of all available features for *VNIR PFD 2080 Custom 1x1 with NI*.

Table 34: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.SetupFilePath
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height

Feature name
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Digital
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString

Feature name
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Shutter.IsToggle
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber
Shutter.Channel
Shutter.Channels

6.3.34 VNIR PFD 2080 Custom 3x3 with NI Feature List

Below is a list of all available features for *VNIR PFD 2080 Custom 3x3 with NI*.

Table 35: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback

Feature name
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Digital
Camera.Image.Height
Camera.Image.SizeBytes

Feature name
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Shutter.IsToggle
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber
Shutter.Channel
Shutter.Channels

6.3.35 VNIR PFD 2080 SCB with NI Feature List

Below is a list of all available features for *VNIR PFD 2080 SCB with NI*.

Table 36: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera1.ExposureTime
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded

Feature name
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Digital
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop

Feature name
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1
Scb.Temperature.Channel2
Scb.Temperature.Channel3
Scb.Temperature.Channel4
Sensor.Detector.Type
Sensor.SerialNumber

6.3.36 VNIR PFD 2080 with NI Feature List

Below is a list of all available features for *VNIR PFD 2080 with NI*.

Table 37: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start

Feature name
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Digital
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear

Feature name
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Shutter.IsToggle
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber
Shutter.Channel
Shutter.Channels

6.3.37 VNIR PFD GigE Feature List

Below is a list of all available features for *VNIR PFD GigE*.

Table 38: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error

Feature name
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Image.Height
Camera.Image.SizeBytes

Feature name
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Shutter.IsToggle
Camera.Temperature
Camera.ToggleShutter
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber
Shutter.Channel
Shutter.Channels

6.3.38 VNIR PFD SCB with Framelink Feature List

Below is a list of all available features for *VNIR PFD SCB with Framelink*.

Table 39: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera1.ExposureTime
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded

Feature name
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Digital
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop

Feature name
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1
Scb.Temperature.Channel2
Scb.Temperature.Channel3
Scb.Temperature.Channel4
Sensor.Detector.Type
Sensor.SerialNumber
VCECLB.CameraConfig

6.3.39 VNIR PFD SCB with NI Feature List

Below is a list of all available features for *VNIR PFD SCB with NI*.

Table 40: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync

Feature name
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera1.ExposureTime
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Digital
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation

Feature name
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1
Scb.Temperature.Channel2
Scb.Temperature.Channel3
Scb.Temperature.Channel4
Sensor.Detector.Type
Sensor.SerialNumber

6.3.40 VNIR PFD with Framelink Feature List

Below is a list of all available features for *VNIR PFD with Framelink*.

Table 41: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter

Feature name
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Digital
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Shutter.IsToggle
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width

Feature name
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber
Shutter.Channel
Shutter.Channels
VCECLB.CameraConfig

6.3.41 VNIR PFD with NI Feature List

Below is a list of all available features for *VNIR PFD with NI*.

Table 42: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average

Feature name
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.Channel
Camera.Channels
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Digital
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height

Feature name
Camera.Sensor.Width
Camera.SerialNumber
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Shutter.IsToggle
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber
Shutter.Channel
Shutter.Channels

6.3.42 VNIR sCMOS with Bitflow Feature List

Below is a list of all available features for *VNIR sCMOS with Bitflow*.

Table 43: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Encoding
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback

Feature name
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.SimpleGainControl
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera1.ExposureTime
Camera1.FrameRate
Camera2.ExposureTime
Camera2.FrameRate
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Image.Height
Camera.Image.SizeBytes

Feature name
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.SerialNumber
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.TargetTemperature
Camera.Temperature
Camera.Trigger.Mode
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Scb.Channel
Scb.Channels
Scb.IsOpen
Scb.Temperature.Channel1
Scb.Temperature.Channel2

Feature name
Scb.Temperature.Channel3
Scb.Temperature.Channel4
Sensor.Detector.Type
Sensor.SerialNumber

6.3.43 XLNIR Feature List

Below is a list of all available features for *XLNIR*.

Table 44: Available features

Feature name
Acquisition.Acquiring
Acquisition.CalculatedFrameRate
Acquisition.Error
Acquisition.FrameCounter
Acquisition.FrameCounter.Reset
Acquisition.PostStartCallback
Acquisition.PostStopCallback
Acquisition.PreStartCallback
Acquisition.PreStopCallback
Acquisition.RingBuffer.Lag
Acquisition.RingBuffer.Size
Acquisition.RingBuffer.Sync
Acquisition.Start
Acquisition.Stop
Acquisition.Timeout
AcquisitionWindow.Height
AcquisitionWindow.Left
AcquisitionWindow.Top
AcquisitionWindow.Width
Camera.Binning.Average
Camera.Binning.HardwareModes
Camera.Binning.Spatial
Camera.Binning.Spectral
Camera.BitDepth
Camera.ByteDepth

Feature name
Camera.CalibrationPack
Camera.CalibrationPack.IsLoaded
Camera.CheckTemperatures
Camera.CloseShutter
Camera.Count
Camera.Debug
Camera.Debug.FrameNumber
Camera.ExposureTime
Camera.ExposureTime.Auto
Camera.FrameRate
Camera.FWHM
Camera.Gain.Analog
Camera.Image.Height
Camera.Image.SizeBytes
Camera.Image.SizePixels
Camera.Image.Transformation
Camera.Image.Width
Camera.IsConnected
Camera.MaxDataNumber
Camera.Model
Camera.MROI.Clear
Camera.MROI.Enable
Camera.MROI.IsHardware
Camera.MROI.MultibandString
Camera.OpenShutter
Camera.Pixel.Height
Camera.Pixel.Width
Camera.Sensor.Height
Camera.Sensor.Width
Camera.SerialNumber
Camera.Shutter.IsConnected
Camera.Shutter.IsOpen
Camera.Shutter.IsToggle
Camera.Temperature
Camera.Trigger.Mode

Feature name
Camera.Wavelength.Start
Camera.Wavelength.Stop
Camera.WavelengthTable
Detector.Image.Height
Detector.Image.Width
Grabber.Channel
Grabber.Channels
Initialize
IsInitialized
Sensor.Detector.Type
Sensor.SerialNumber
Shutter.Channel
Shutter.Channels
Xenics.CalibrationFile
Xenics.CalibrationFolder
Xenics.CalibrationRange
Xenics.TargetTemperature
Xenics.TEC

7 Functions

7.1 Function List

The Lumo Sensor SDK contains the following functions:

```
int SI_Load(const SI_WC* LicensePath)
int SI_Unload(void)
int SI_Open(int DeviceIndex, SI_H* Handle)
int SI_Close(SI_H Handle)
int SI_IsImplemented(SI_H Handle, const SI_WC* Feature, SI_BOOL*
 Implemented)
int SI_IsReadOnly(SI_H Handle, const SI_WC* Feature, SI_BOOL* ReadOnly)
int SI_IsWritable(SI_H Handle, const SI_WC* Feature, SI_BOOL* Writable)
int SI_IsReadable(SI_H Handle, const SI_WC* Feature, SI_BOOL* Readable)
int SI_RegisterFeatureCallback(SI_H Handle, const SI_WC* Feature,
 FeatureCallback Callback, void* Context)
int SI_UnregisterFeatureCallback(SI_H Handle, const SI_WC* Feature,
 FeatureCallback Callback)
int SI_SetInt(SI_H Handle, const SI_WC* Feature, SI_64 Value)
int SI_GetInt(SI_H Handle, const SI_WC* Feature, SI_64* Value)
int SI_GetIntMax(SI_H Handle, const SI_WC* Feature, SI_64* MaxValue)
int SI_GetIntMin(SI_H Handle, const SI_WC* Feature, SI_64* MinValue)
int SI_SetFloat(SI_H Handle, const SI_WC* Feature, double Value)
int SI_GetFloat(SI_H Handle, const SI_WC* Feature, double* Value)
int SI_GetFloatMax(SI_H Handle, const SI_WC* Feature, double* MaxValue)
int SI_GetFloatMin(SI_H Handle, const SI_WC* Feature, double* MinValue)
int SI_SetBool(SI_H Handle, const SI_WC* Feature, SI_BOOL Value)
int SI_GetBool(SI_H Handle, const SI_WC* Feature, SI_BOOL* Value)
int SI_SetString(SI_H Handle, const SI_WC* Feature, SI_WC* Value)
int SI_GetString(SI_H Handle, const SI_WC* Feature, SI_WC* Value, int
 StringLength)
int SI_GetStringMaxLength(SI_H Handle, const SI_WC* Feature, int*
 MaxStringLength)
int SI_GetEnumStringMaxLength(SI_H Handle, const SI_WC* Feature, int
 Index, int* MaxStringLength)
int SI_SetEnumIndex(SI_H Handle, const SI_WC* Feature, int Value)
int SI_SetEnumIndexByString(SI_H Handle, const SI_WC* Feature, const
 SI_WC* String)
int SI_SetEnumString(SI_H Handle, const SI_WC* Feature, SI_WC* String)
int SI_GetEnumIndex(SI_H Handle,const  SI_WC* Feature, int* Value)
int SI_GetEnumCount(SI_H Handle,const  SI_WC* Feature, int* Value)
int SI_GetEnumStringByIndex(SI_H Handle,const  SI_WC* Feature, int Index,
 SI_WC* String, int StringLength)
int SI_IsEnumIndexAvailable(SI_H Handle,const  SI_WC* Feature, int Index,
 SI_BOOL* Available)
int SI_IsEnumIndexImplemented(SI_H Handle,const  SI_WC* Feature, int
 Index, SI_BOOL* Implemented)
int SI_Command(SI_H Handle, const SI_WC* Feature)
int SI_CreateBuffer(SI_H Handle, SI_64 BufferSize, void**
 BufferAddressPtr)
int SI_DisposeBuffer(SI_H Handle, void* BufferAddress)
int SI_RegisterDataCallback(SI_H Handle, DataCallBack Callback, void*
 Context)
int SI_UnregisterDataCallback(SI_H Handle)
int SI_Wait(SI_H Handle, SI_U8* BufferAddress, SI_64* FrameSize, SI_64*
 FrameNumber,
SI_64 Timeout = SI_INFINITE)
int SI_GetFeatureType(SI_H Handle, const SI_WC* Feature, SI_WC*
 String, int StringLength)
```

```
const SI_WC* SI_GetErrorString(int Error)
```

7.2 SI_Load

```
int SI_Load(const SI_WC* LicensePath)
```

Loads the library and must be called before any other function.

7.3 SI_Unload

```
int SI_Unload(void)
```

Unloads the library, and must be called as the last function.

7.4 SI_Open

```
int SI_Open(int DeviceIndex, SI_H* Handle)
```

Opens a handle to a particular device. The device selected is specified by the DeviceIndex parameter. The DeviceIndex parameter is zero-based. The opened handle (if successful) is returned in the Handle parameter. This handle is used as the first parameter in all other functions.

7.5 SI_Close

```
int SI_Close(SI_H Handle)
```

Closes a previously opened handle to a device. The Handle parameter is the same handle returned by the `SI_Open` function. This function is usually called when there is no need to access any devices anymore.

7.6 SI_IsImplemented

```
int SI_IsImplemented(SI_H Handle, const SI_WC* Feature, SI_BOOL*
 Implemented)
```

Used to determine whether the feature specified by Feature parameter is implemented in the selected device. The Implemented parameter will contain either `SI_TRUE` or `SI_FALSE`, depending on whether the feature is implemented or not.

7.7 SI_IsReadOnly

```
int SI_IsReadOnly(SI_H Handle, const SI_WC* Feature, SI_BOOL* ReadOnly)
```

Used to determine whether the feature specified by the Feature parameter can be modified. The ReadOnly parameter will contain either `SI_TRUE` or `SI_FALSE`, depending on whether the feature is a read-only feature.

7.8 SI_IsImplemented

```
int SI_IsImplemented(SI_H Handle, const SI_WC* Feature, SI_BOOL*
 Implemented)
```

Used to determine whether the feature specified by Feature parameter is implemented in the selected device. The Implemented parameter will contain either `SI_TRUE` or `SI_FALSE`, depending on whether the feature is implemented or not.

7.9 SI_IsWritable

```
int SI_IsWritable(SI_H Handle, const SI_WC* Feature, SI_BOOL* Writable)
```

Used to determine whether the feature specified by the Feature parameter can currently be modified. The Writable parameter will contain either `SI_TRUE` or `SI_FALSE`, depending on whether the feature is writable or not. The difference between `SI_IsReadOnly` and `SI_IsWritable` is that a feature can be temporarily in read-only mode due to other features.

7.10 SI_IsReadable

```
int SI_IsReadable(SI_H Handle, const SI_WC* Feature, SI_BOOL* Readable)
```

Used to determine whether the feature specified by the Feature parameter is currently readable. The Readable parameter will contain either `SI_TRUE` or `SI_FALSE`, depending on whether the feature is readable or not. A feature can temporarily become non-readable due to other features.

7.11 SI_UnregisterFeatureCallback

```
int SI_UnregisterFeatureCallback(SI_H Handle, const SI_WC* Feature,
 FeatureCallback Callback)
```

Used to unregister a previously registered feature callback function. The parameters are:

- The Feature parameter will contain the feature to which you want to stop receiving notifications.
- The FeatureCallback parameter is the callback function previously registered with `SI_RegisterFeatureCallback`.

No more feature notifications will be sent after this function has successfully returned.

7.12 SI_SetInt

```
int SI_SetInt(SI_H Handle, const SI_WC* Feature, SI_64 Value)
```

Used to modify the value of a specified feature only if the feature is of the integer type.

Note:

Use `SI_64` as the type for the value to be set.

7.13 SI_GetInt

```
int SI_GetInt(SI_H Handle, const SI_WC* Feature, SI_64* Value)
```

Used to read the value of a specified feature only if the feature is of the integer type.

Note:

Use `SI_64` pointer as the type for the value to get.

7.14 SI_GetIntMax

```
int SI_GetIntMax(SI_H Handle, const SI_WC* Feature, SI_64* MaxValue)
```

Used to get the maximum allowed value for the specified integer feature.

Note:

Use `SI_64` pointer as the type for the value to get.

7.15 SI_GetIntMin

```
int SI_GetIntMin(SI_H Handle, const SI_WC* Feature, SI_64* MinValue)
```

Used to get the minimum allowed value for the specified integer feature.

Note:

Use `SI_64` pointer as the type for the value to get.

7.16 SI_SetFloat

```
int SI_SetFloat(SI_H Handle, const SI_WC* Feature, double Value)
```

Used to modify the value of a specified feature only if the feature is of the floating-point type.

Note:

Use double pointer as the type for the value to be set.

7.17 SI_GetFloat

```
int SI_GetFloat(SI_H Handle, const SI_WC* Feature, double* Value)
```

Used to read the value of a specified feature only if the feature is of the floating-point type.

Note:

Use double pointer as the type for the value to be get.

7.18 SI_GetFloatMax

```
int SI_GetFloatMax(SI_H Handle, const SI_WC* Feature, double* MaxValue)
```

Used to get the maximum allowed value for the specified floating-point feature.

Note:

Use double pointer as the type for the value to get.

7.19 SI_GetFloatMin

```
int SI_GetFloatMin(SI_H Handle, const SI_WC* Feature, double* MinValue)
```

Used to get the minimum allowed value for the specified floating-point feature.

Note:

Use double pointer as the type for the value to get.

7.20 SI_SetBool

```
int SI_SetBool(SI_H Handle, const SI_WC* Feature, SI_BOOL Value)
```

Used to set the value of a specified feature only if the feature is of the boolean type. The Value parameter can be either `SI_TRUE` or `SI_FALSE`.

7.21 SI_GetBool

```
int SI_GetBool(SI_H Handle, const SI_WC* Feature, SI_BOOL* Value)
```

Used to get the current value of a specified feature only if the feature is of the boolean type. The Value parameter will contain either `SI_TRUE` or `SI_FALSE`, depending on the current value of the feature.

7.22 SI_Command

```
int SI_Command(SI_H Handle, const SI_WC* Feature)
```

Used to execute a specified feature only if the feature is of the command type.

7.23 SI_SetString

```
int SI_SetString(SI_H Handle, const SI_WC* Feature, SI_WC* Value)
```

Used to modify the value of a specified feature only if the feature is of the string type. The Value parameter should contain a null terminated string.

Note:

Use `SI_WC` pointer as the type for the string value to be set.

7.24 SI_GetString

```
int SI_GetString(SI_H Handle, const SI_WC* Feature, SI_WC* Value, int
 StringLength)
```

Used to get the current string value of a specified feature only if the feature is of the string type. The parameters are:

- The Value parameter is the user allocated `SI_WC*` buffer to hold the characters returned by the function.
- The StringLength parameter should hold the maximum size of the Value parameter.

Note:

Remember to leave enough space for the null-termination character as well.

7.25 SI_GetStringMaxLength

```
int SI_GetStringMaxLength(SI_H Handle, const SI_WC* Feature, int*
 MaxStringLength)
```

Used to get the maximum allowed string length of a specified feature only if the feature is of the string type. This value can be used when retrieving a string value with the `SI_GetString` function.

7.26 SI_GetEnumStringMaxLength

```
int SI_GetEnumStringMaxLength(SI_H Handle, const SI_WC* Feature, int
 Index, int* MaxStringLength)
```

Used to get the maximum allowed string length of a specific index in a specified enumeration feature. The Index parameter refers to the index in the enumeration.

7.27 SI_SetEnumIndex

```
int SI_SetEnumIndex(SI_H Handle, const SI_WC* Feature, int Value)
```

Used to set the currently used index of a specified feature only if the feature is of the enumeration type. The Value parameter is the zero-based index in the enumeration. The Value parameter should be in between 0 and Count – 1 where Count is retrieved by using the `SI_GetEnumCount` function.

7.28 SI_SetEnumIndexByString

```
int SI_SetEnumIndexByString(SI_H Handle, const SI_WC* Feature, const
 SI_WC* String)
```

Used to set the currently used index by a string of a specified feature only if the feature is of the enumeration type, and only if the string exists in the enumeration. The String parameter is null-terminated string.

7.29 SI_SetEnumString

```
int SI_SetEnumString(SI_H Handle, const SI_WC* Feature, SI_WC* String)
```

Used to change the enumeration's string value in the currently selected index.

7.30 SI_GetEnumIndex

```
int SI_GetEnumIndex(SI_H Handle,const  SI_WC* Feature, int* Value)
```

Used to get the currently selected index of a specified feature only if the feature is of the enumeration type. The Value parameter will contain the zero-based current index in the enumeration feature.

7.31 SI_GetEnumCount

```
int SI_GetEnumCount(SI_H Handle,const  SI_WC* Feature, int* Value)
```

Used to get the total number of indices in a specified feature only if the feature is of the enumeration type.

7.32 SI_GetEnumStringByIndex

```
int SI_GetEnumStringByIndex(SI_H Handle,const  SI_WC* Feature, int Index,
 SI_WC* String, int StringLength)
```

Used to get the string value of a specific index in a specified feature only if the feature is of the enumeration type. The parameters are:

- The Index parameter refers to the index, the string of which is to be retrieved.
- The String parameter will contain a user allocated `SI_WC*` buffer to hold the retrieved string value.
- The StringLength parameter should hold the maximum size of the Value parameter.

Note:

Remember to leave enough space for the null-termination character as well.

7.33 SI_IsEnumIndexAvailable

```
int SI_IsEnumIndexAvailable(SI_H Handle,const  SI_WC* Feature, int Index,
 SI_BOOL* Available)
```

Used to determine whether the specific index can be selected in a specified feature only if the feature is of the enumeration type. The Available parameter will contain either `SI_TRUE` or `SI_FALSE`, depending on the availability of the index in the enumeration feature.

7.34 SI_IsEnumIndexImplemented

```
int SI_IsEnumIndexImplemented(SI_H Handle,const  SI_WC* Feature, int
 Index, SI_BOOL* Implemented)
```

Used to determine whether the specific index is implemented in the specified enumeration feature for the currently selected device.

7.35 SI_Wait

```
int SI_Wait(SI_H Handle, SI_U8* BufferAddress, SI_64* FrameSize, SI_64*
 FrameNumber, SI_64 Timeout = SI_INFINITE)
```

Used to retrieve image data from the camera. The parameters are:

- The BufferAddress parameter is the pointer to the buffer to hold a single frame. BufferAddress should be allocated with `SI_CreateBuffer` before calling `SI_Wait` and deallocated with `SI_DisposeBuffer` when it is not needed anymore.
- The FrameSize parameter will contain the size of the retrieved frame in bytes.
- The FrameNumber parameter will contain the frame number of the retrieved frame.
- The Timeout parameter can be specified to indicate how long in milliseconds you wish to wait for the next available frame. The function will wait until it either receives a new frame or the timeout expires. If the Timeout parameter is left empty , the function will return only when it receives a new frame.

The `SI_Wait` function is disabled if a DataCallback has been registered with the `SI_RegisterDataCallback` function.

In general, DataCallback is recommended for use whenever the user is familiar with it and if performance is important. `SI_Wait` can be used in simple use cases where, for example, data is collected using a loop and a simple operation is performed on it, without high performance requirements. Internally, `SI_Wait` is a slower implementation than data callback. However, for users who are not familiar with callback terminology, `SI_Wait` may be an easier function to approach.

7.36 SI_CreateBuffer (Deprecated)

```
int SI_CreateBuffer(SI_H Handle, SI_64 BufferSize, void**
 BufferAddressPtr)
```

Used to create image buffers to be used with the SDK.

Deprecated: Instead use standard allocation functions such as malloc or new.

7.37 SI_DisposeBuffer (Deprecated)

```
int SI_DisposeBuffer(SI_H Handle, void* BufferAddress)
```

Used to dispose image buffers created with the `SI_CreateBuffer` function.

Deprecated: use standard deallocation functions such as free or delete instead.

7.38 SI_RegisterDataCallback

```
int SI_RegisterDataCallback(SI_H Handle, DataCallBack Callback, void*
 Context)
```

Used to register a user function to be called each time a new frame is retrieved. The DataCallback parameter is the callback function to be registered, and must follow the syntax:

```
int SI_IMPEXP_CONV DataCallbackFunction( SI_U8* Buffer, SI_64 FrameSize,
 SI_64 FrameNumber, void* Context)
```

The parameters are:

- The Buffer parameter will contain the frame data for the current frame.
- The FrameSize parameter will contain the size of the frame in bytes.
- The FrameNumber parameter will contain the number of the retrieved frame.
- The last parameter, Context, usually represents the pointer to the class that registered the callback for contextual purposes when the callback function is called.

Only one function can be registered at a time, and if a function is registered, the `SI_Wait` functionality is disabled to prevent both methods being used at the same time from different threads. DataCallback is recommended for use instead of `SI_Wait` whenever the user is familiar with it and if performance is important. All Lumo software uses DataCallback and, as a result, it is tested more extensively than `SI_Wait` in release tests.

7.39 SI_UnregisterDataCallback

```
int SI_UnregisterDataCallback(SI_H Handle)
```

Used to unregister a previously registered `DataCallback` function.

7.40 SI_GetFeatureType

```
int SI_GetFeatureType(SI_H Handle, const SI_WC* Feature, SI_WC* String,
 int StringLength)
```

Used to get the type of a specified feature as a text representation. The parameters are:

- The String parameter should be a user allocated `SI_WC*` string buffer and it should also have space for a null-termination character.

- The StringLength parameter should hold the maximum size of the String parameter.

8 Function Return Codes

Almost every function in the Lumo Sensor SDK returns an error code. These codes can be grouped into three main categories:

- No error.
- If an error has occurred, the returned value is negative, and indicates that the function has failed.
- In the case of a warning, the returned value is a positive, non-zero value, which indicates that something non-critical has happened in the function which may or may not affect the end results of the function, depending on the function used. If the function was successful, the returned value will be zero, and usually this is the only case where you can safely continue to the next statement.

8.1 Generic Error Codes

This section lists the generic Lumo Sensor SDK error codes.

Table 45: Generic Error Codes

Error	Error code	Description
siNoError	0	No error.
siFailed	-1	Failed.
siNotImplemented	-2	Not implemented.
siNotFound	-3	Not found.
siInvalidType	-4	Invalid type.
siInvalidHandle	-5	Invalid handle.
siInvalidPointer	-6	Invalid pointer.
siOutOfRange	-7	Out of range.
siInvalidDeviceType	-8	Invalid device type.
siInvalidParameter	-9	Invalid parameter.
siUnknownError	-10	Unknown error.
siNotSupported	-11	Not supported.
siOutOfMemory	-12	Out of memory.
siBufferTooSmall	-13	Buffer too small.
siNotInitialized	-14	Not initialized.
siInvalidFile	-15	Invalid file.
siOutOfRangeMax	-16	Value is greater than its maximum value.
siOutOfRangeMin	-17	Value is smaller than its minimum value.
siInvalidChecksum	-18	Invalid checksum.
siAbort	-19	Generic abort error.

Error	Error code	Description
siExceptionOccured	-20	An exception occurred and was handled.
siUnmappedOsError	-21	Unmapped OS Error.
siFailedToLoadDriver	-22	Failed to load driver.
siValueNotSet	-23	Value not set.
siNotAllowed	-24	Not allowed.
siInvalidCommand	-25	Invalid command.
siIncompleteCommand	-26	Incomplete command.

8.2 File Error Codes

This section lists the Lumo Sensor SDK file error codes.

Table 46: File Error Codes

Error	Error code	Description
siFileOpenFailed	-100	Could not open the file.
siFileCreateFailed	-101	Could not create the file.
siFileReadFailed	-102	Could not read the file.
siFileNotOpened	-103	File not opened.
siFileEOF	-104	Reached end of file.
siFileCRC	-105	File CRC error.
siFileWriteFailed	-106	File write failed.

8.3 XML Error Codes

This section lists the Lumo Sensor SDK XML error codes.

Table 47: File Error Codes

Error	Error code	Description
siXmlParseFailed	-200	The XML parsing failed.
siXmlStringNotSet	-201	The XML string is not set.
siXmlTagNotFound	-202	The XML tag was not found.

8.4 Sensor Profile Error Codes

This section lists the Lumo Sensor SDK sensor profile error codes.

Table 48: Sensor Profile Error Codes

Error	Error code	Description
siProfileInvalidDirectory	-300	The Specim Sensor Profiles (SSP) directory is invalid or does not exist.
siProfileInvalidDocumentType	-301	An invalid document type has been specified for this profile.
siProfileInvalidModuleDefinition	-302	An invalid module definition has been specified for this profile.
siProfileInvalidType	-303	An invalid type has been specified for this profile.
siProfileInvalidModule	-304	The profile has an invalid module. A module definition cannot be null.
siProfileNotOpen	-305	The profile handle is not opened.
siProfileMissingName	-306	The name of the profile is missing.
siProfileParseError	-307	The profile has some parsing error(s).
siProfileNotSet	-308	The profile is not set.

8.5 Feature Error Codes

This section lists the Lumo Sensor SDK feature error codes.

Table 49: Feature Error Codes

Error	Error code	Description
siFeatureNotFound	-400	The feature cannot be found.
siFeatureNotImplemented	-401	The feature is not implemented for this device.
siFeatureReadOnly	-402	The feature is read-only and cannot be written.
siFeatureNotWritable	-403	The feature it not writable at this point of time.
siFeatureNotReadable	-404	The feature is not readable at this point of time.
siFeatureValueOutOfRange	-405	The value specified for this feature is out of range.

Error	Error code	Description
siFeatureValueNotSet	-406	The value for this feature is not set.
siFeatureEmptyEnumeration	-407	The enumeration of the feature is empty.
siFeatureMissingFeaturesDefinition	-408	The profile features definition is missing.
siFeatureAlreadyExists	-409	The feature already exists.
siFeatureInvalidEnumerationString	-410	The enumeration string is invalid.
siFeatureGroupNotFound	-411	The feature group cannot be found.
siFeatureEnumStringNotFound	-412	The string was not found in the enumerated feature.
siFeatureMissingType	-413	The Feature type is missing.
siFeatureInvalidType	-414	The Feature type is invalid.
siFeatureMissingName	-415	The Feature missing name.
siFeatureLockTimeout	-416	The operation on the feature timed out.

8.6 Command Error Codes

This section lists the Lumo Sensor SDK command error codes.

Table 50: Command Error Codes

Error	Error code	Description
siCommandNotFound	-500	The command cannot be found.
siCommandNotAvailable	-501	The command is not available.
siCommandInvokeFailed	-502	The invoke command failed.

8.7 Module Error Codes

This section lists the Lumo Sensor SDK module error codes.

Table 51: Module Error Codes

Error	Error code	Description
siModuleNotFound	-600	The module cannot be found.
siModuleAlreadyDeclaredWithDifferentSignature	-601	The module is already declared, but with a different signature.

Error	Error code	Description
siModuleUnresolvedSymbol	-602	The module symbol could not be resolved.
siModuleNotLoaded	-603	The module is not loaded.
siModuleLoadFailed	-604	Loading the module failed.
siModuleNotRegistered	-605	The module is not registered.

8.8 Acquisition Error Codes

This section lists the Lumo Sensor SDK acquisition error codes.

Table 52: Acquisition Error Codes

Error	Error code	Description
siAcquisitionTimeout	-700	The acquisition has timed out.
siAcquisitionAborted	-701	The acquisition was aborted.
siAcquisitionBufferAlreadyAllocated	-702	Buffer already allocated. First, delete the memory using SI_DisposeBuffer.
siAcquisitionInvalidBuffer	-703	Invalid buffer. The buffer is either NULL or has not been created using SI_CreateBuffer.
siAcquisitionInProgress	-704	Acquisition is in progress.
siAcquisitionMultigrabbingInvalidWidth	-705	Multi-grabbing widths do not match.
siAcquisitionNotInProgress	-706	Acquisition is not in progress.

8.9 Header File Error Codes

This section lists the Lumo Sensor SDK header file error codes.

Table 53: Header File Error Codes

Error	Error code	Description
siHeaderInconsistentHeader	-800	Inconsistent header file.
siHeaderParseError	-801	The header file has a parsing error.
siHeaderKeywordAlreadyExists	-802	The header keyword already exists.
siHeaderInvalidType	-803	Invalid type for this keyword.

Error	Error code	Description
siHeaderInvalidData	-804	Invalid data for this keyword.

8.10 Buffering Error Codes

This section lists the Lumo Sensor SDK buffering error codes.

Table 54: Buffering Error Codes

Error	Error code	Description
siRingBufferNotInitialized	-900	Buffer is not initialized.
siRingBufferInvalidHandle	-901	Invalid reader handle.

8.11 Communication Error Codes

This section lists the Lumo Sensor SDK communication error codes.

Table 55: Communication Error Codes

Error	Error code	Description
siCommunicationFailure	-1100	Communication failure.
siCommunicationTimeout	-1101	Communication timeout.
siCommunicationOverrun	-1102	Communication buffer overrun.
siCommunicationCRC	-1103	Communication CRC error.
siCommunicationNotAcknowledged	-1104	Communication not acknowledged.
siCommunicationPortInUse	-1105	Communication port in use.
siCommunicationBaudRateNotSupported	-1106	Baud rate not supported.
siCommunicationBusy	-1107	Communication is busy.

8.12 Camera Error Codes

This section lists the Lumo Sensor SDK camera error codes.

Table 56: Camera Error Codes

Error	Error code	Description
siCameraInvalidChannel	-1200	Invalid camera channel.
siCameraSerialCommunicationFailure	-1201	Serial communication failure in camera.
siCameraCoolingNotCompleted	-1202	

Error	Error code	Description
siCameraConnectingError	-1203	Camera connection failure. The camera may be already in use.
siCameraNotFound	-1204	

8.13 SCB Error Codes

This section lists the Lumo Sensor SDK SCB error codes.

Table 57: SCB Error Codes

Error	Error code	Description
siSCBChecksumFailed	-1300	SCB Checksum error.
siSCBReadFailed	-1301	SCB Read failed.
siSCBWriteFailed	-1302	SCB Write failed.
siSCBFlashReadFailed	-1303	SCB Read flash failed.
siSCBFlashWriteFailed	-1304	SCB Write flash failed.
siSCBFlashEraseFailed	-1305	SCB Erase flash failed.
siSCBNotInitialized	-1306	SCB Not Initialized.

8.14 Licensing Error Codes (depracated)

This section lists the Lumo Sensor SDK licensing error codes.

Table 58: Licensing Error Codes

Error	Error code	Description
siLicensingInitError	-1400	Call LoadLicense successfully before using other functionality.
siLicensingErrorOpeningSharedMemory	-1401	Error opening a shared memory for licensing.
siLicensingHashNotSupported	-1402	Selected hash function is not supported.
siLicensingCipherNotSupported	-1403	Selected cipher function is not supported.
siLicensingEncryptionFailed	-1404	Licensing encryption failed.
siLicensingDecryptionFailed	-1405	Licensing decryption failed.
siLicensingFileNotFound	-1406	Licensing-related file not found.
siLicensingEmptyLicense	-1407	Empty license file.
siLicensingExpired	-1408	License expired.

Error	Error code	Description
siLicensingProductExpired	-1409	Product expired.
siLicensingModuleExpired	-1410	Module expired.
siLicensingFeatureExpired	-1411	Feature expired.
siLicensingApplicationNotFound	-1412	Application license not found.
siLicensingModuleNotFound	-1413	Module license not found.
siLicensingFeatureNotFound	-1414	Feature license not found.
siLicensingSyntaxError	-1415	Licensing syntax error.
siLicensingTimestampError	-1416	Licensing real-time timestamp error.

8.15 Generic Warning Codes

This section lists the generic Lumo Sensor SDK warning error codes.

Table 59: Generic Warning Error Codes

Error	Error code	Description
siAlreadyInitialized	1	Already initialized.
siAlreadyExists	2	Already exists.
siUnknownWarning	3	Unknown warning.
siIsEmpty	4	Trying to access an empty object.

8.16 Sensor Profile Warning Codes

This section lists the Lumo Sensor SDK sensor profile warning codes.

Table 60: Sensor Profile Warning Codes

Error	Error code	Description
siProfileNoDescrption	300	It is recommended, but not mandatory, to put a description for the sensor profile.

8.17 Feature Warning Codes

This section lists the Lumo Sensor SDK feature warning codes.

Table 61: Feature Warning Codes

Error	Error code	Description
siFeatureValueChanged	402	The feature value has changed implicitly.

8.18 Module Warning Codes

This section lists the Lumo Sensor SDK module warning codes.

Table 62: Module Warning Codes

Error	Error code	Description
`siModuleAlreadyRegistered`	`600`	The module is already registered.
`siModuleAlreadyLoaded`	`601`	The module is already loaded.

8.19 Buffering Warning Codes

This section lists the Lumo Sensor SDK buffering warning codes.

Table 63: Buffering Warning Codes

Error	Error code	Description
`siRingBufferWaitWriter`	`900`	The reader must wait for the writer to continue.

8.20 Calibration Warning Codes

This section lists the Lumo Sensor SDK calibration warning codes.

Table 64: Calibration Warning Codes

Error	Error code	Description
`siCalibrationMissing`	`1600`	Calibration is missing.

9 Code Examples

The C++ project examples can be located in `C:\Program Files\Specim\SDKs\Lumo_Sensor_SDK\<year_version>` where `<year_version>` is eg. `2023_625\examples` on a 32-bit machine and `C:\Program Files (x86)\Specim\SDKs\Lumo_Sensor_SDK\<year_version>` where `<year_version>` is eg. `2023_625\examples` on a 64-bit machine.

These examples are here to show you couple of basic things you can do with the SDK. Some of those things could be: listing all the available devices, changing the frame rate or grabbing incoming frames. The purpose of these examples is to help you understand the idea and the functionality of the SDK so that you are able to create your own more complex programs later on.

9.1 Enumerating Through Detected Devices

This example is a minimalistic program that demonstrates how to detect all the available sensor profiles. The program:

- Loads the SDK.
- Gets the sensor device count and prints it.
- Prints the names and descriptions of those devices.
- Unloads the SDK.

```
#include "stdafx.h"
#include "SI_sensor.h"
#include "SI_errors.h"



int _tmain(int argc, _TCHAR* argv[])
{
 // Create the necessary variables
 int nError = siNoError;
 SI_64 nDeviceCount = 0;
 SI_WC szDeviceName[4096];
 SI_WC szDeviceDescription[4096];

 // Load SDK and get the device count
 SI_CHK(SI_L(""));
 SI_CHK(SI_GetInt(SI_SYSTEM, L"DeviceCount", &nDeviceCount));

 wprintf(L"Device count: %d\n", nDeviceCount);


 // Iterate through each devices to print their name and description
 for (int n = 0; n < nDeviceCount; n++)
 {
  SI_CHK(SI_GetEnumStringByIndex(SI_SYSTEM, L"DeviceName", n,
 szDeviceName, 4096));
  SI_CHK(SI_GetEnumStringByIndex(SI_SYSTEM, L"DeviceDescription", n,
 szDeviceDescription, 4096));
  wprintf(L"Device %d:\n", n);
  wprintf(L"\tName: %s\n", szDeviceName);
  wprintf(L"\tDescription: %s\n", szDeviceDescription);
 }

 // Unload the library
 SI_CHK(SI_Unload());
```

```
 wprintf(L"\r\nPress any key to quit...\r\n");

 getchar();

Error:
 return 0;
}
```

9.2 Basic Set and Get Functionality and Waiting for Frames

This example demonstrates how to open a specific device and then change some of its parameters and grab 100 frames before closing down. The program:

- Loads the SDK.
- Lets the user to select a specific sensor device and then opens it.
- Initializes the device.
- Changes the frame rate to 25Hz and the exposure time to 3 ms.
- Creates a buffer for grabbing purposes.
- Grabs 100 frames and prints the incoming frame numbers.
- Disposes the created buffer.
- Closes the sensor device.
- Unloads the SDK.

```
// example2.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <windows.h>
#include "SI_sensor.h"
#include "SI_errors.h"


SI_H g_hDevice = 0;

int SelectDevice(void)
{
 int nError = siNoError;
 SI_64 nDeviceCount = 0;
 SI_WC szDeviceName[4096];
 int nIndex = -1;

 SI_CHK(SI_GetInt(SI_SYSTEM, L"DeviceCount", &nDeviceCount));
 wprintf(L"Device count: %d\n", nDeviceCount);

 // Iterate through each devices to print their name
 for (int n = 0; n < nDeviceCount; n++)
 {
  SI_CHK(SI_GetEnumStringByIndex(SI_SYSTEM, L"DeviceName", n,
 szDeviceName, 4096));
  wprintf(L"\t%d: %s\n", n, szDeviceName);
 }

 // Select a device
 wprintf(L"Select a device: ");
 scanf_s("%d", &nIndex);

 if ((nIndex >= nDeviceCount) || (nIndex == -1))
 {
  wprintf(L"Invalid index");
  return -1;
 }
```

```
Error:
 return nIndex;
}

int _tmain(int argc, _TCHAR* argv[])
{
 // Creates the necessary variables
 int nError = siNoError;
 int nDeviceIndex = -1;
 SI_U8* pFrameBuffer = 0;
 SI_64 nBufferSize = 0;
 SI_64 nFrameSize = 0;
 SI_64 nFrameNumber = 0;

 // Loads SDK and get the device count
 wprintf(L"Loading SDK...\n");
 SI_CHK(SI_Load(L""));

 // Select a device
 nDeviceIndex = SelectDevice();

 if (nDeviceIndex == -1)
 {
  return 0;
 }

 // Opens the camera handle
 SI_CHK(SI_Open(nDeviceIndex, &g_hDevice));
 SI_CHK(SI_Command(g_hDevice, L"Initialize"));

 // Sets frame rate and exposure
 SI_CHK(SI_SetFloat(g_hDevice, L"Camera.FrameRate", 25.0));
 SI_CHK(SI_SetFloat(g_hDevice, L"Camera.ExposureTime", 3.0));

 // Creates a buffer to receive the frame data
 SI_CHK(SI_GetInt(g_hDevice, L"Camera.Image.SizeBytes", &nBufferSize));
 SI_CHK(SI_CreateBuffer(g_hDevice, nBufferSize, (void**)&pFrameBuffer));

 // Starts the acquisition, acquires 100 frames and stops the acquisition
 SI_CHK(SI_Command(g_hDevice, L"Acquisition.Start"));

 for(int n = 0; n < 100; n++)
 {
  SI_CHK(SI_Wait(g_hDevice, pFrameBuffer, &nFrameSize, &nFrameNumber,
 1000));
  // Do something interesting with the frame pointer (pFrameBuffer)
  wprintf(L"Frame number: %d\n", nFrameNumber);
 }

 SI_CHK(SI_Command(g_hDevice, L"Acquisition.Stop"));


Error:
 if (SI_FAILED(nError))
 {
  char szInput[256] = "";
  wprintf(L"An error occurred: %s\n", SI_GetErrorString(nError));
  wprintf(L"Enter a character and press enter to exit\n");
  scanf_s("%s", &szInput, 255);

 }
 else
 {
  char szInput[256] = "";
  wprintf(L"It all went well!\n");
  wprintf(L"Enter a character and press enter to exit\n");
  scanf_s("%s", &szInput, 255);
```

```
 }

 // Cleanups the buffer, closes the camera and unloads SDK
 SI_DisposeBuffer(g_hDevice, pFrameBuffer);
 SI_Close(g_hDevice);
 SI_Unload();

 return SI_FAILED(nError) ? -1 : 0;
}
```

9.3 Feature Callbacks and Data Callback

This demonstrates the usage of different callbacks. Feature callbacks are used to inform the user that a feature value has changed so that the user does not need to query the values manually. The data callback is used to inform the user that a new frame has arrived from the sensor. The program will:

- Loads the SDK.
- Lets the user to select a specific sensor device and then opens it.
- Initializes the device.
- Registers a feature callback to handle both frame rate and exposure time. Also registers exposure time to another different callback. This means that every time the exposure time is changed both of those callbacks are called.
- Registers a data callback to handle the incoming frames.
- Allows the user to start and stop grabbing and prints the frame numbers.
- Closes the sensor device.
- Unloads the SDK.

```
// example3.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <windows.h>
#include "SI_sensor.h"
#include "SI_errors.h"


SI_H g_hDevice = 0;

int SI_IMPEXP_CONV onDataCallback(SI_U8* _pBuffer, SI_64 _nFrameSize,
 SI_64 _nFrameNumber, void* _pContext);


int SelectDevice(void)
{
 int nError = siNoError;
 SI_64 nDeviceCount = 0;
 SI_WC szDeviceName[4096];
 int nIndex = -1;

 SI_CHK(SI_GetInt(SI_SYSTEM, L"DeviceCount", &nDeviceCount));
 wprintf(L"Device count: %d\n", nDeviceCount);

 // Iterate through each devices to print their name
 for (int n = 0; n < nDeviceCount; n++)
 {
  SI_CHK(SI_GetEnumStringByIndex(SI_SYSTEM, L"DeviceName", n,
 szDeviceName, 4096));
  wprintf(L"\t%d: %s\n", n, szDeviceName);
 }

 // Select a device
 wprintf(L"Select a device: ");
```

```
 scanf_s("%d", &nIndex);

 if ((nIndex >= nDeviceCount) || (nIndex == -1))
 {
  wprintf(L"Invalid index");
  return -1;
 }

Error:
 return nIndex;
}


int SI_IMPEXP_CONV FeatureCallback1(SI_H Hndl, SI_WC* Feature, void*
 Context)
{
 if(wcscmp(Feature, L"Camera.ExposureTime") == 0)
 {
  wprintf(L"FeatureCallback1: Camera.ExposureTime\n");
 }
 else if(wcscmp(Feature, L"Camera.FrameRate") == 0)
 {
  wprintf(L"FeatureCallback1: Camera.FrameRate\n");
 }

 return 0;
}


int SI_IMPEXP_CONV FeatureCallback2(SI_H Hndl, SI_WC* Feature, void*
 Context)
{
 if(wcscmp(Feature, L"Camera.ExposureTime") == 0)
 {
  wprintf(L"FeatureCallback2: Camera.ExposureTime\n");
 }

 return 0;
}


int SI_IMPEXP_CONV onDataCallback(SI_U8* _pBuffer, SI_64 _nFrameSize,
 SI_64 _nFrameNumber, void* _pContext)
{
 wprintf(L"%d ", _nFrameNumber);
 return 0;
}


int _tmain(int argc, _TCHAR* argv[])
{
 // Create the necessary variables
 int nError = siNoError;
 int nDeviceIndex = -1;
 int nAction = 0;
 wchar_t szMessage[] = L"Select an action:\n\t0: exit\n\t1: start
 acquisition\n\t2: stop acquisition\n";

 // Load SDK and get the device count
 wprintf(L"Loading SDK...\n");
 SI_CHK(SI_L(""));

 // Select a device
 nDeviceIndex = SelectDevice();

 // Open the device and set the callbacks
 SI_CHK(SI_Open(nDeviceIndex, &g_hDevice));
```

```
 SI_CHK(SI_Command(g_hDevice, L"Initialize"));
 SI_CHK(SI_RegisterFeatureCallback(g_hDevice, L"Camera.FrameRate",
 FeatureCallback1, 0));
 SI_CHK(SI_RegisterFeatureCallback(g_hDevice, L"Camera.ExposureTime",
 FeatureCallback1, 0));
 SI_CHK(SI_RegisterFeatureCallback(g_hDevice, L"Camera.ExposureTime",
 FeatureCallback2, 0));
 SI_CHK(SI_RegisterDataCallback(g_hDevice, onDataCallback, 0));

 // Prompt commands
 wprintf(L"%s", szMessage);
 while(scanf("%d", &nAction))
 {
  if (nAction == 0)
  {
   wprintf(L"Bye bye!");
   break;
  }
  else if (nAction == 1)
  {
   wprintf(L"Start acquisition");
   SI_CHK(SI_Command(g_hDevice, L"Acquisition.Start"));
  }
  else if (nAction == 2)
  {
   wprintf(L"Stop acquisition");
   SI_CHK(SI_Command(g_hDevice, L"Acquisition.Stop"));
  }

  wprintf(L"%s", szMessage);
 }


Error:
 if (SI_FAILED(nError))
 {
  wprintf(L"An error occurred: %s\n", SI_GetErrorString(nError));
 }

 SI_Close(g_hDevice);
 SI_Unload();

 return SI_FAILED(nError) ? -1 : 0;
}
```

10 Appendix: Multiple Regions of Interest

The MROI features in Lumo Sensor SDK allow to output only subsets of the data.

MROI is done as the last stage of the imaging pipeline, so it happens after windowing, binning and transformation.

Prerequisite for enabling the MROI is to disable spectral binning. The MROI top offset should always be greater than previous offset + previous height. If you use AIE, you must include additional rows in the beginning and in the end of the MROI to prevent corruption of the intended MROI rows. For FX10/10e add three rows and for FX17/17e add four rows in the beginning and the end of the MROI. These extra rows must be discarded in data processing.

10.1 Example: Creating a Single Region of Interest

Create a single MROI using the MultibandString feature. First, disable MROI. Then, set the new MROI value, and finally enable MROI again.

```
// The example sets one MROI between indices 100-199.
SI_SetBool(Handle, L"Camera.MROI.Enable", false);
SI_SetString(Handle, L"Camera.MROI.MultibandString", L"100 100");
SI_SetBool(Handle, L"Camera.MROI.Enable", true);
```

10.2 Example: Creating Multiple Regions of Interest

Create multiple MROIs using the MultibandString feature. First, disable MROI. Then, set the new MROI values and separate the regions with a semicolon, and finally enable MROI again.

```
// The example sets two MROIs one between indices 80-119 and another one
 between indices 180-199.
SI_SetBool(Handle, L"Camera.MROI.Enable", false);
SI_SetString(Handle, L"Camera.MROI.MultibandString", L"80 40;180 20");
SI_SetBool(Handle, L"Camera.MROI.Enable", true);
```

11 Change History

Date	Version	Description
20 February 2024	1.6	• Support for Windows 11 • New features: Camera.Image.IsTestImage Camera.Image.ReadoutMode Camera.Reset
6 October 2023	1.5	• Updated screenshot titled Selecting the Components • Dalsa Xtium2-CL MX4 (FX10, FX17, GX17) entry added to grabber list
31 March 2023	1.4	• Added FX50 to Features per Sensor Profile and FX50 related features to feature list • Added GX17 to Features per Sensor Profile • Unified the SDK naming throughout the document • Added Pleora eBus compatibility • Updated Camera.Channel and Grabber.Channel feature descriptions
7 June 2021	1.3	• Updated MROI, Buffer management and FeatureList code examples • Removed SDK licensing • Corrected broken references • Removed MultiPointString feature • Added Acquisition.PreStart.Callback reference to Camera.FWHM, Camera.Image.SizeBytes, Camera.Image.SizePixels, Camera.Image.Width, and Camera.Image.Height • Updated error code lists and fixed the table layout • Added information about third party SDKs • Corrected some screenshots • Added warning code 402 • Other minor improvements