ISG GigE Vision™ Host Application Interface Developer's Guide

The purpose of this document is to describe the way in which a user can create a Windows Visual Studio Application to control and acquire video from an ISG GigeVision™ camera. Two independent libraries are used for this purpose. One is the GenICam™ library which is a reference implementation that is supplied by the EMVA standards organization. This provides an interface to the full feature set of the camera. Installation of the GenApi software environment is required in order to operate the ISG API. Upon installation of the GenApi, a suite of header files are installed in the Program Files\GeniCam_ver_xx include path, to provide all of the class and type definitions that are required. A pair of GenICam™ standard documents have been included within the ISG software disk that provides details on the GenICam interface. Document one is the GenICam_Standard_vxx.pdf document. Among other things, it provides a description of the interfaces that are available for each data type defined by the specification (see section 2.9). Document two in the software packet is the GenICam_SFNC_xx.pdf file. That document defines all of the features that are common to a digital video camera, and therefore most cameras such as the ISG device, conform to these functions, along with some additional custom controls. The second library that makes up the API is provided by ISG and takes care of the GigeVision transport layer specific interface to the camera. This library and its header files are installed as part of the ISG demonstration GUI and are located in a subfolder of the GUI install location.

Here is the quick start guide to a set of commands that can be used to control an ISG GigeVision™ camera via the GenICam/GigeVisionLib library interface.

The ISG sample GUI is referenced as a source code example of these operations.

1) Discovery and Connection to the camera

In order to find a camera on the host network, a GigeVision defined discovery operation is started. This is shown in the "OnBnClickedButtonDiscovery" button in the GUI. Note that when the Discovery button is clicked, a new CGigEInterface (as defined by the ISG API) object is created. The constructor of this object will automatically perform network discovery and will find any GigeVision cameras on the network. Upon successful discovery, the GUI will display the MAC address of the connected camera. Note that the IP address of the host network card is passed as a parameter to the CGigEInterface class. By default the camera is setup with IP address 192.168.2.50. Therefore the GUI example is shown using a host card address of 192.168.2.1 and this is represented as the hex value 0x0102A8C0, which is the

proper format for the m_iPAddress parameter of the CGigEInterface. As shown in the sample code, the other parameters include a 'GigEDevice' object which holds address information about the camera as well as an integer parameter for receiving status. An alternative constructor for CGigEInterface takes a GigEDevice list as a parameter. This option is available to handle the case were multiple cameras are connected. The list of connected cameras will be made available for parsing. Once a camera has been successfully discovered (status parameter should be zero), the next step of making an exclusive connection can be performed.

The device is "connected" to the camera as shown in the "OnBnClickedButtonConnect" function of the GUI. This is done via a CreateControlChannel operation. A CDevPort object is passed to this function and it receives device setting for the camera. Once this path has been established, the CDevPort object can then be used to access resources in the camera. As shown in the code, a generically named "HostXMLFile.xml" is generated by the CreateControlChannel call, and created in the current working directory of the application. This file is then used in order to attach the GenApi defined CNodeMapRef object to the camera's supported resources via a LoadXMLFromFile member function. Refer to the GenICam™ standard document for a description of the variable type categories that are defined which allow the full functionality of the camera to be controlled. Also reference the GenICam SFNC (standard function naming convention) document for a complete listing of all of the common features that are available for control via the GenApi. Note that subsets of these, as listed in the XML file of the camera, are available in the ISG device.

2) Using the control interface of the GenApi to set camera features

The ISG GUI provides the complete list of features that are supported in the camera via a CTreeCtrl variable as defined by the Microsoft Foundation Class (MFC) library. The DumpFeatures function of the GUI allows a user to traverse the entire list of features, which always begin at the "Root" node of a camera. When the root node is opened, the top level feature categories are then displayed. As a subsequent control set is opened, the next level of feature detail becomes available. Eventually the user will be given a set of selections that map to register control of the camera.

Each of these controls has a specific, GenICam™ defined, data type associated with it. Examples of these data types include IInteger, IFoat, IString, and ICommand, each of which provides their own specific methods of getting and setting data. There are also some common member functions among the various data types. For instance, most provide both a SetValue and GetValue method for writing integer or floating point data to the camera. Other functionality such as "execute", is specific to a single data type, in this case the ICommand type.

Following the CTreeCtrl model of the GUI provides a very convenient way to display the data types associated for each of the camera features automatically. It also enables a user to interact with all of the control data as shown in the edit boxes, drop list boxes and command buttons on the right hand side of the Camera Feature Control group box.

3) Camera video setup routines and data interfacing

The previous section described the way in which an application writer can display and interface to the entire camera feature set. There are also examples in the GUI of setting up a minimal number of features that will allow a custom image frame size to be delivered. The OnBnClickedButtonStartCamera function can be referenced for this purpose. In that function, the height and width resources of the camera are accessed in order to read their state. This access is available using the GetValue method, via an integer pointer variable which was assigned using the GetNode and the respective resource name. In the same way, the SetValue method could have instead been used in order to setup a desired custom frame. The StartCamera function then proceeds to go through the process of setting up for video transmission. This is done by opening and starting a data stream via the GigeVision.lib API CreateStreamChannel method and then executing the "AcquisitionStart" command. CreateStreamChannel will by default setup GigeVision stream channel zero within the camera. The ViewThreadMethod process is kicked off in the GUI in order to receive the video data. Inside this thread, the "AcquireBufferData" function will block until an image becomes available within the data buffer pointer. Note that in acquiring a frame, the GUI provides only a pointer to an image buffer that will be allocated within the GigeVision Library. The GUI application owns this image buffer (i.e. it will not be overwritten by the Library) until such time as the application calls function "FreeBufferData". The FreeBufferData call allows the image buffer to be reused by the library.

To summarize, the following methods of the ISG GigeVision Library represent the minimum set that are needed in order to operate the camera.

```
CGigEInterface(GigEDevice& m_pCurrentDevice,unsigned int m_iIPAdress,int& status );
int CGigEInterface::CreateControlChannel(CDevPort &m_pDevicePort);
int CGigEInterface::CreateStreamChannel(uint32_t pktSize,uint32_t pktDelay);
int CGigEInterface::AcquireBufferData(uint8_t* pData, uint32_t nDataSize,uint32_t
nTimeOut);
int CGigEInterface::FreeBufferData();
```

Refer to the GigEInterface.h header file for a brief description of the parameters, function and return value of these methods.

All other camera functionality is provided via the GenApi reference library. The sample GUI is intended as a substantial reference for complete control over all features of the camera via the tree structure that is populated by using the camera XML file.

JG-10-10-2014