How to: Using exacqVision API Network I/O

Overview

This article explains the usage of basic and advanced I/O functionality in exacqVision API (evAPI).

Usage

The network I/O functions provided in evAPI are required to send requests and receive responses from an exacqVision server. After initializing the API instance and connecting to a server, the I/O functions are necessary to maintain communication with the server. If functions such as EVAPI_Select, EVAPI_Send and EVAPI_Recv are not used, the API will not send or receive any data with the server.

When using evAPI network I/O in a multi-threaded environment, it is necessary to put a mutex around the network I/O functions and any other API function calls on a particular API instance. This ensures the integrity of the underlying read and write buffers. The mutex is not necessary for API function calls in the callback because the callback is executed from within EVAPI_Select and EVAPI_Read. See the advanced I/O example below for more information.

Basic Network I/O

The purpose of the basic network I/O functionality provided with evAPI is to enable the end user to send and receive data with an exacqVision server without worrying about the underlying network details.

Functions:

EVAPI_Select: initiates a send and receive on the underlying network socket. The timeout value is the maximum amount of time the function will block before returning, although generally the function will return sooner. This function needs to be called frequently to ensure the receipt and sending of data with the server.

EVAPI_SetTimeout: specifies the blocking timeout in milliseconds for EVAPI_Login, EVAPI_SearchCamera, and EVAPI_SearchCameraMulti.

EVAPI_CreateWindow (Windows only): creates a window with a window procedure that receives messages on socket status via WSAAsyncSelect and provides a drawing surface for subscribed video streams. The messages received on socket status act as an I/O pump and don't require the use EVAPI_Select when subscribed to a video stream. To insure proper behavior, call this function right

Created On: 11th February 2021
KB Number: KB-00533-533-210211

before subscribing to a video stream as in the window sample in the SDK. If there is a delay in calling EVAPI_CreateWindow and subscribing to a video stream, it will be necessary to call EVAPI_Select once to get the I/O pump started.

Example:

```
Function Main
    Intialize evAPI (EVAPI_Init)
    Connect to server (EVAPI_Connect)

    // No need for select call when using the blocking login since it will be called //
    internally.
    Login to server (EVAPI_Login)

     While application running
          // If EVAPI_Select is not called after the EVAPI_PTZGotoPreset, then
        // the request will sit in the write buffer
        Make request to server (EVAPI_PTZGotoPreset)
        Perform network I/O (EVAPI_Select)
     End while

End Function Main
```

Advanced Network I/O

The purpose of the advanced network I/O functions provided with evAPI is to allow greater control over sending and receiving on the network socket.

Functions:

EVAPI_GetSocket returns the socket that evAPI is using to send and receive data from the exacqVision server.

EVAPI_IsWriting returns the amount of data in the buffer to be written to the socket.

EVAPI_Recv calls socket function recv on the evAPI socket and reads the data into a buffer in the API instance. Execution of this function will result in parsing of the incoming data prior to the function returning. The callback will be executed for any relevant data.

EVAPI_Send calls socket function send on the evAPI socket and sends data in the write buffer of the API instance.

EVAPI_SetNonBlocking sets the blocking status of the I/O functions. This currently affects only EVAPI_Login.

Example:

```
Function evCallback(Parameters: ParamTypeparam, int value, Payload* payload, size_tlength )
        Switch to handle ParamTypes
                Case Login
                        Set login to true
                End Case
                Case Config
                // Not necessary to mutex since there is a mutex around the read call
                Make evAPI function call
                End Case
        End Switch
End Function evCallback

Function Main
        Intialize evAPI (EVAPI_Init)
        Set callback to Function evCallback (EVAPI_SetCallback)
        Set to non-blocking (EVAPI_SetNonBlocking)
        Connect to server (EVAPI_Connect)
        Login to server (EVAPI_Login)

        // If EVAPI_Select is not called after the EVAPI_Login, then the
        // request will sit in the write buffer
        Perform network I/0 (EVAPI_Select)

        // Have to wait for the login to succeed or for a timeout before continuing
         While not login or not timeout
         End While
```

```
        If not login
                Goto End Function Main
        End if

  Spawn I/O Thread

        While application running
Lock mutex
                Call an evAPI function
Unlock mutex
        End while

End Function Main

I/O Thread
        Create empty read set of sockets (fd_set read)
        Create empty write set of sockets (fd_set write)

        Get underlying API socket (EVAPI_GetSocket)

        While thread running

                If there is data in the write buffer (EVAPI_IsWriting)
                        Add the socket to the write set (FD_SET)
                End if

                // Always add the socket to the read set
                Add the socket to the read set (FD_SET)

                Check to see if socket is ready to read and/or write (select)
```

If the socket is in the write set (FD_ISSET)
  Lock mutex
    Write the data (EVAPI_Write)
  Unlock mutex
End if

If the socket is in the read set (FD_ISSET)
  Lock mutex
    Read the data (EVAPI_Read)
  Unlock mutex
End if

End while

End I/O Thread