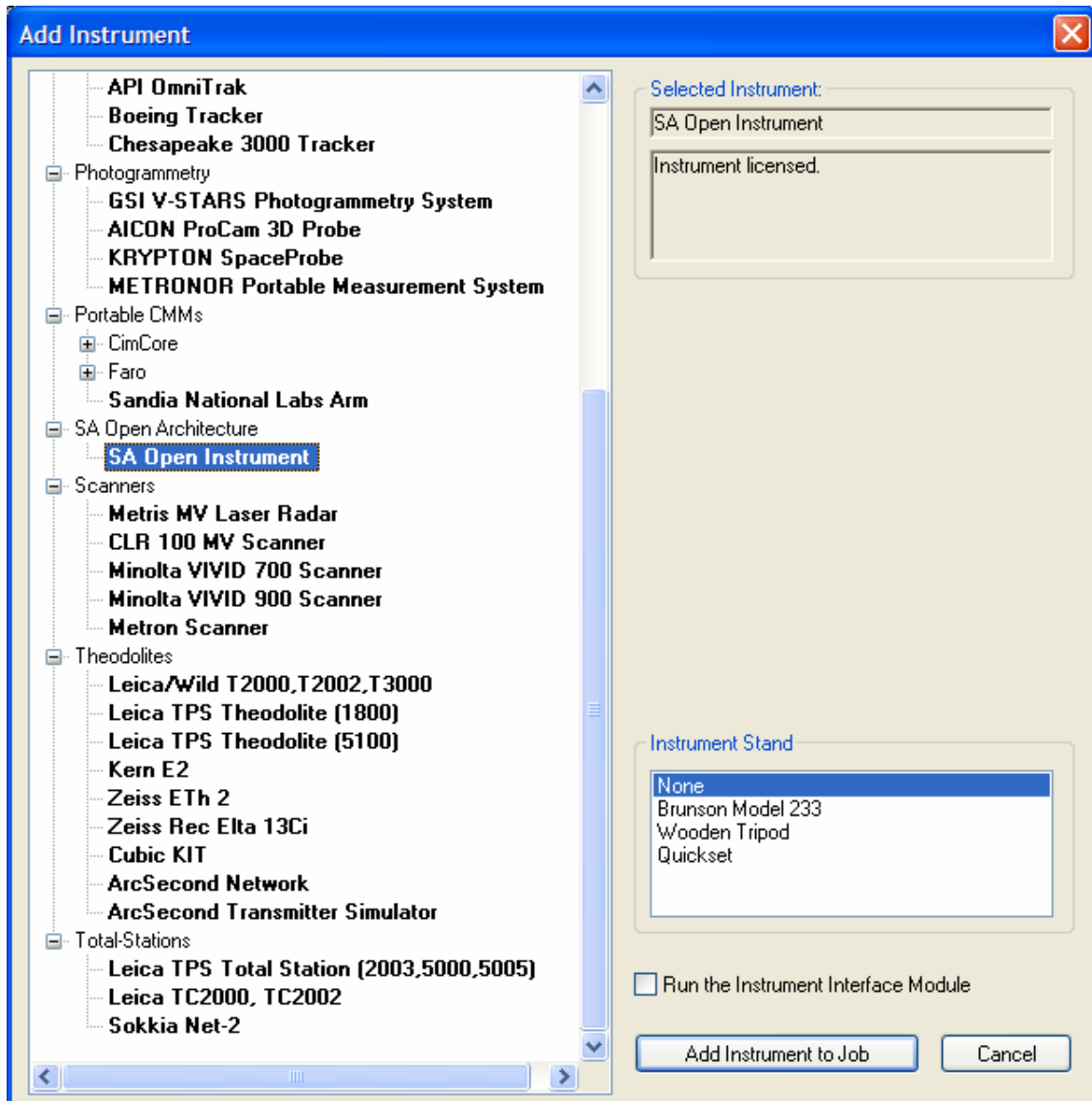


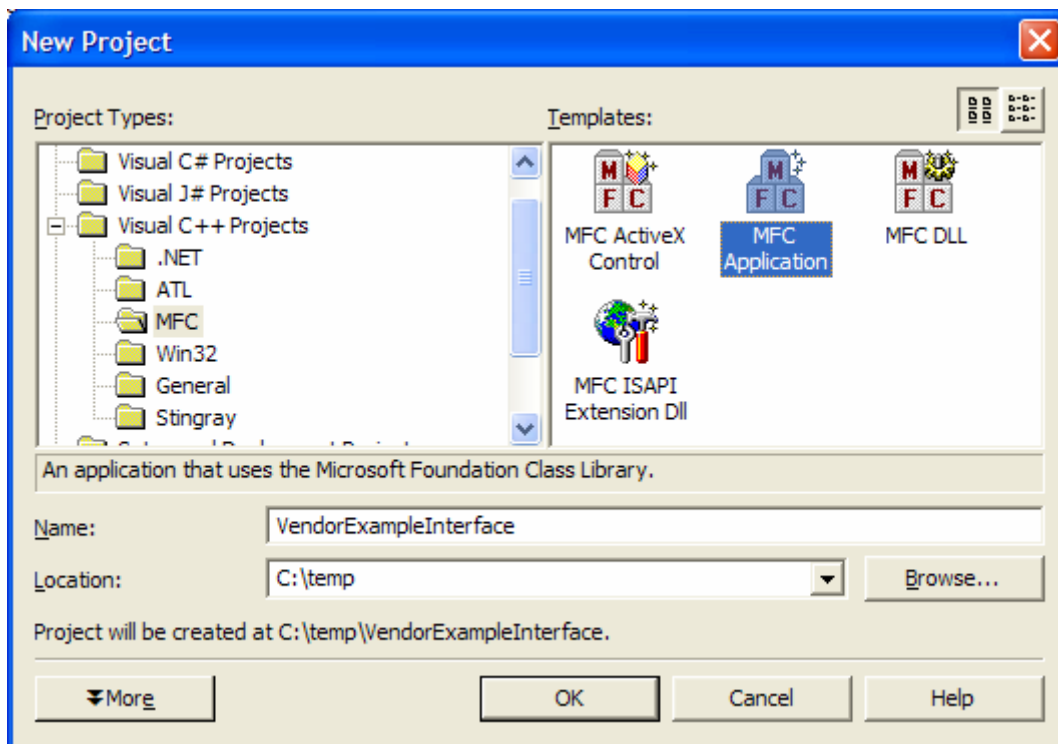
SA Open Instrument

The Spatial Analyzer (SA) Open Instrument interface is an ActiveX control that allows you to create your own instrument interface and send measurement information to SA. You must be running a recent version of SA that supports the Open Architecture / Instrument. When viewing the Add Instrument dialog within SA, the instrument is shown below:

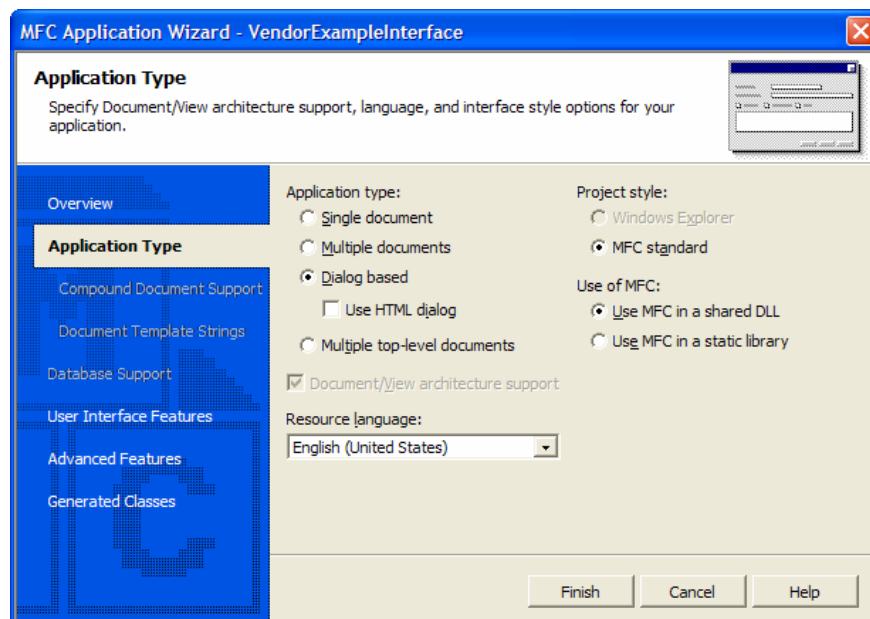


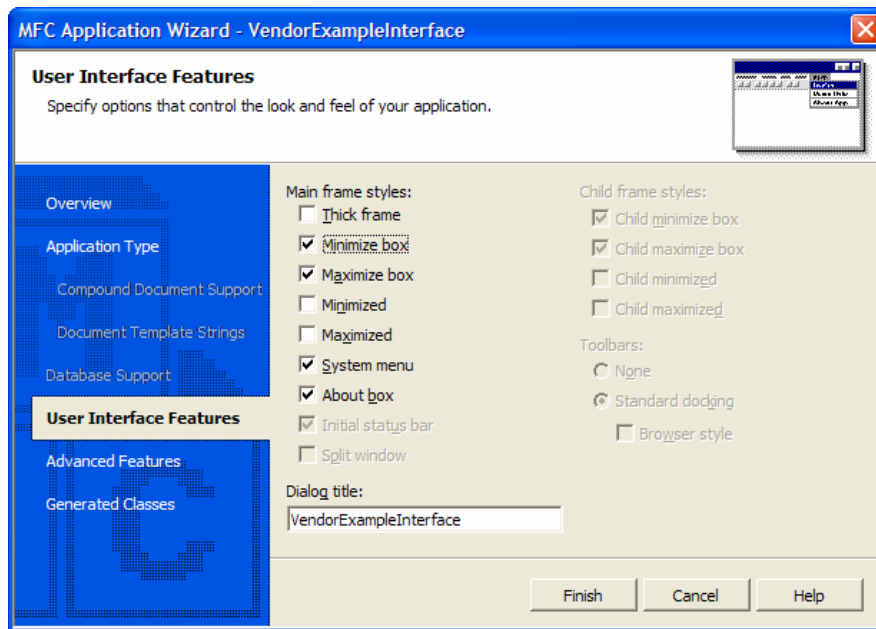
Creating the Interface (Visual C++)

1. Create a new dialog project

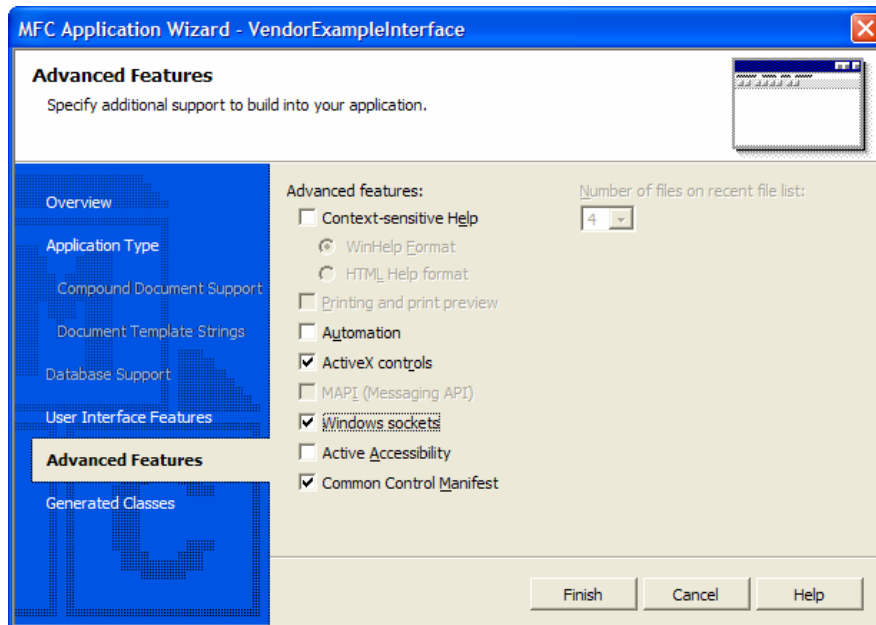


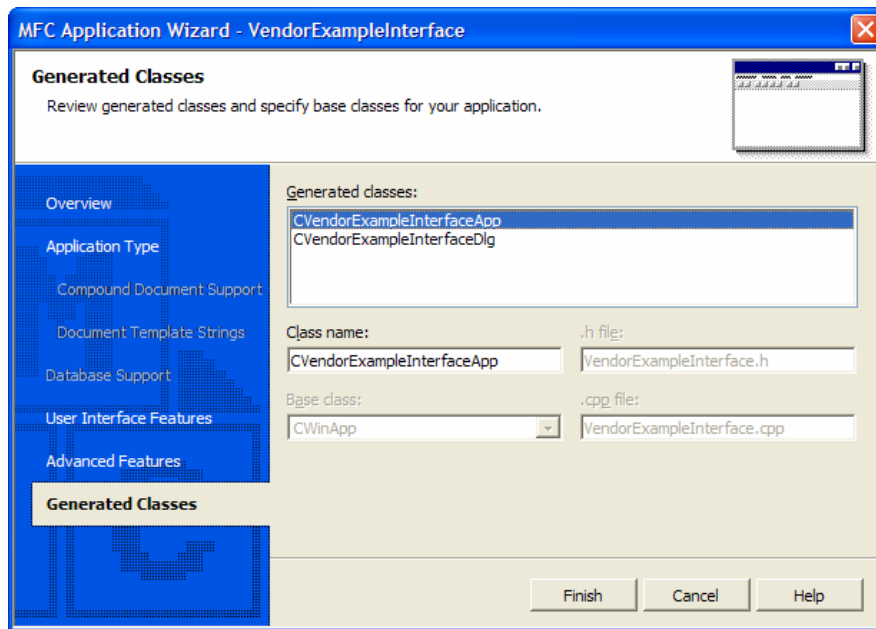
2. Setup the initial wizard configuration





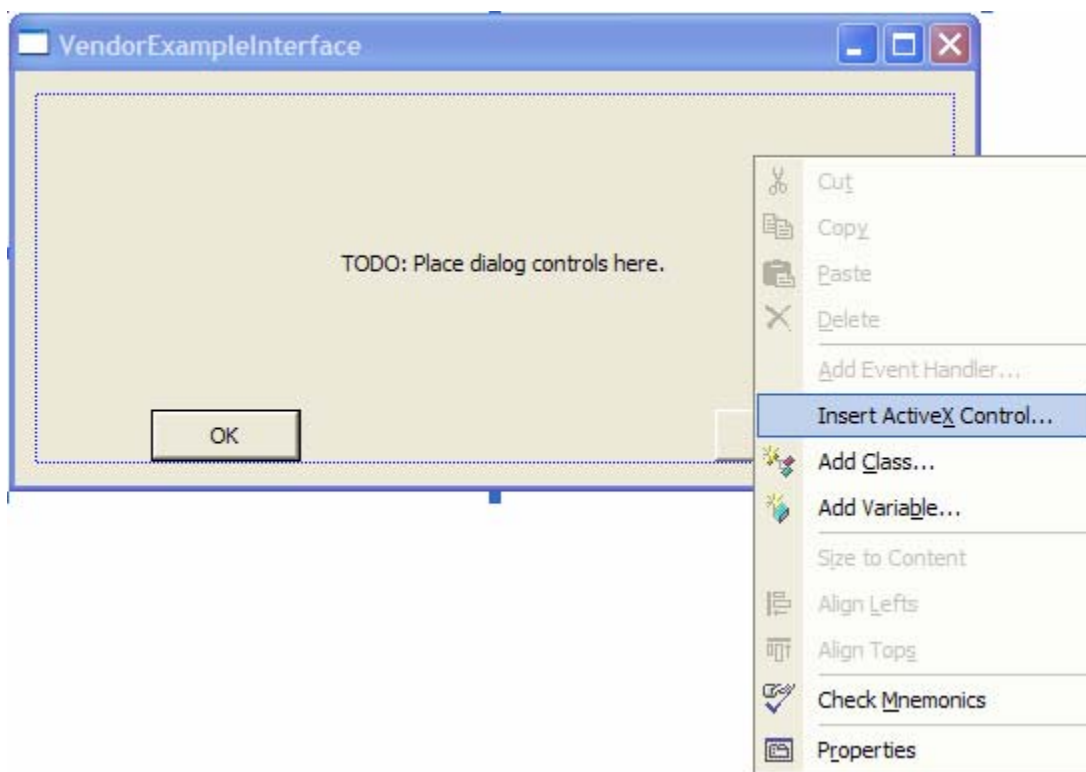
Be sure that “ActiveX controls” and “Windows sockets” are checked in the Advanced Features dialog:



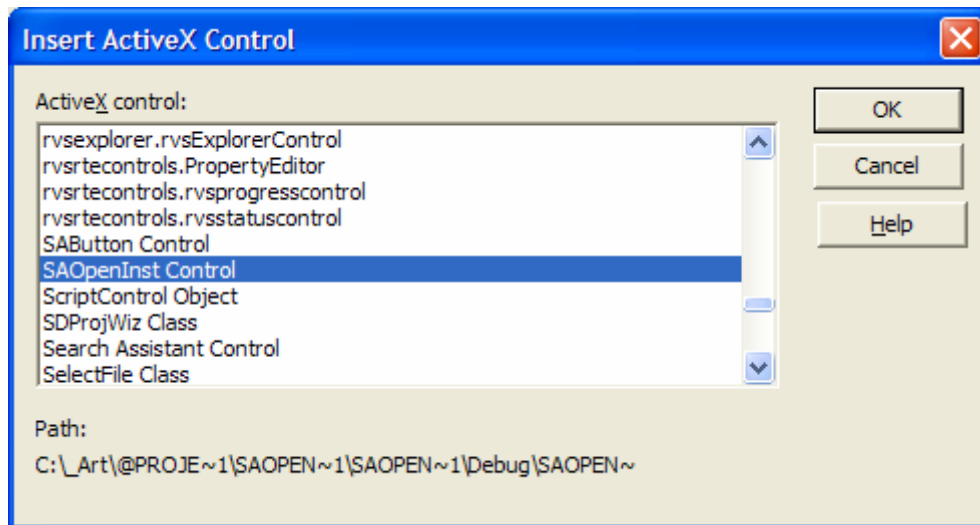


Click the “Finish” button when you’re ready to generate the project.

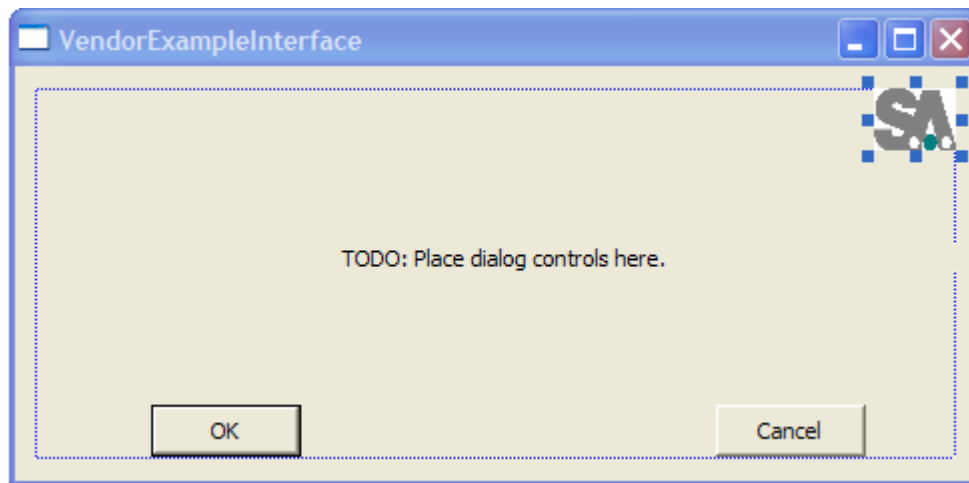
3. Insert the SA Open Instrument ActiveX control into your dialog:



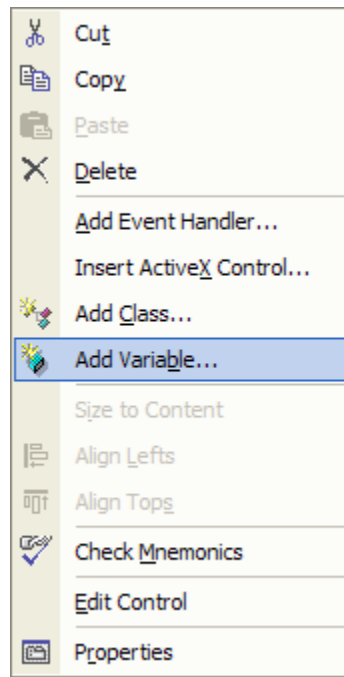
Select the “SAOpenInst Control” from the Insert ActiveX Control shown below:



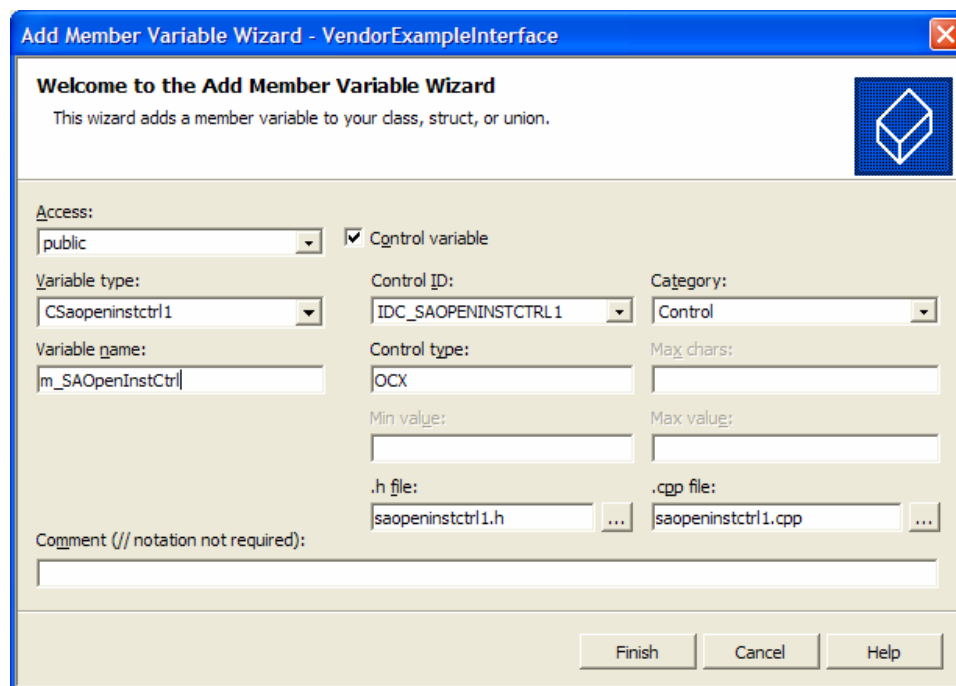
4. Move / Resize the control to the desired location within your dialog:



5. Now insert the control into your dialog .h and .cpp code:
 - Right Mouse click on the SAOpenInst Control and select the “Add Variable” menu item



- Fill in the wizard fields as desired. In particular the Access and Variable name fields.



- Click the “Finish” button.
 - Note: this will also create the saopeninstctrl1.h/cpp files necessary to communicate with the control.
6. With the control variable added to your dialog and the auto generated saopeninstctrl1.h/cpp files, you’re now ready to use the control’s methods to implement your interface!
 7. The control methods are as follows:

- **BOOL PopLogonDialog(void):**

Display the logon dialog to allow the user to select the desired SpatialAnalyzer connection.
 @param none
 @return TRUE for a successful logon, otherwise FALSE.

- **BOOL IsConnected(void):**

Determine if the interface is connected to SpatialAnalyzer
 @param none
 @return TRUE if connected, otherwise FALSE.

- **void ClearMeasurements(void):**

Clear the measurement buffer.
 @param none
 @return none

- **BOOL AddMeasurement(DOUBLE XValue, DOUBLE YValue, DOUBLE ZValue, DOUBLE XUncert, DOUBLE YUncert, DOUBLE ZUncert)**

Add a measurement to the buffer to be sent to SA.
 @param XValue the x-component of the measurement to be sent.
 @param YValue the y-component of the measurement to be sent.
 @param ZValue the z-component of the measurement to be sent.
 @param xUncertVal the x-uncertainty of the measurement to be sent.
 @param yUncertVal the y-uncertainty of the measurement to be sent.
 @param zUncertVal the z-uncertainty of the measurement to be sent.

@return TRUE if successfully, otherwise FALSE.

- **BOOL SendMeasurements(LPCTSTR collectionName, LPCTSTR groupName, LPCTSTR targetName, DOUBLE planarOffset, DOUBLE radialOffset)**

Send the pending measurement(s) to the current SA connection.
 @param collectionName the name of the desired collection within SA for the measurement.
 @param groupName the name of the desired group within SA for the measurement.
 @param targetName the name of the desired target within SA for the measurement.
 @param planarOffset the planar offset (inches).
 @param radialOffset the radial offset (inches).

@return TRUE if the measurement was successfully sent, otherwise FALSE.

- **void ClearCloud(void):**

Clear the cloud buffer.
 @param none
 @return none

- **BOOL AddCloudPoint(DOUBLE XValue, DOUBLE YValue, DOUBLE ZValue):**

Add a cloud point to the buffer to be sent to SA.
 @param XValue the x-component of the cloud point to be sent.
 @param YValue the y-component of the cloud point to be sent.
 @param ZValue the z-component of the cloud point to be sent.

@return TRUE if successfully, otherwise FALSE.

- `BOOL SendCloud(LPCTSTR cloudName, LPCTSTR collectionName):`

Send the pending cloud data to the current SA connection.

@param cloudName the name of the desired cloud within SA.

@param collectionName the name of the desired collection within SA for the cloud.

@return TRUE if the cloud was successfully sent, otherwise FALSE.

- `void OnSetColGroupTarget(LPCTSTR collectionName, LPCTSTR groupName, LPCTSTR targetName):`

Event handler that will be invoked to notify the user of a collection, group, target change request.

@param collectionName the desired collection name

@param groupName the desired group name

@param targetName the desired target name

@return none

- `void OnConnectToHardware(void):`

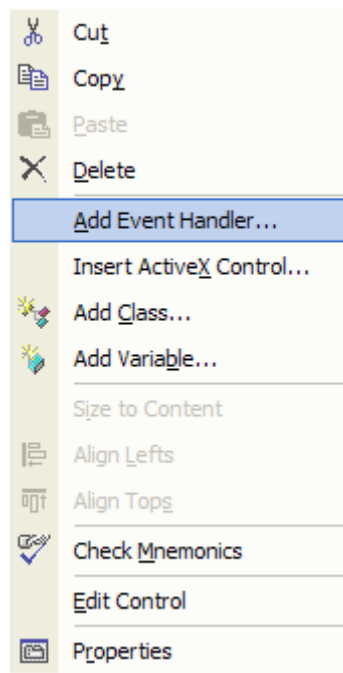
Event handler that will be invoked after a connection to SA has been established.

User may put hardware specific connection logic here.

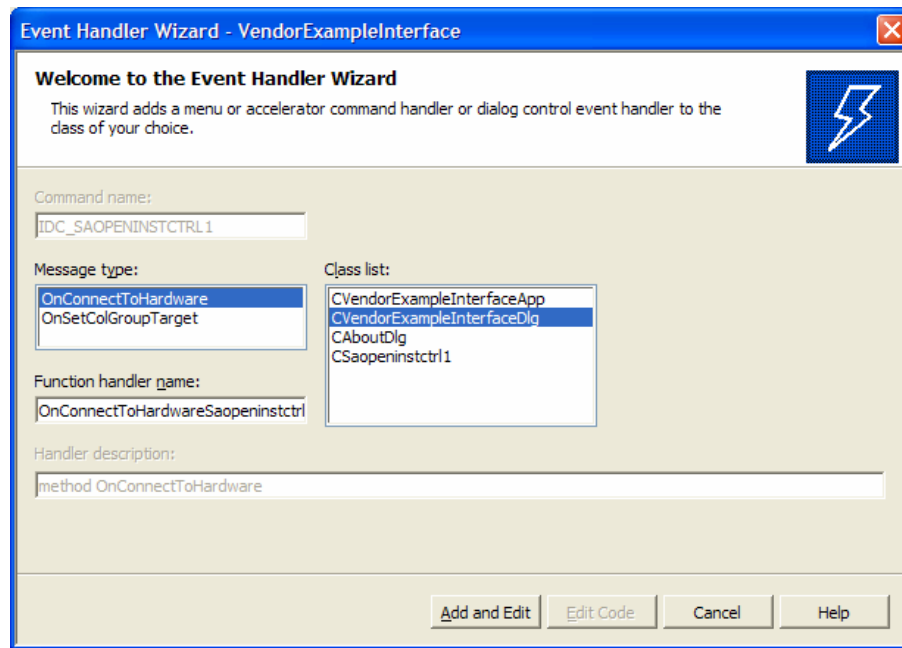
@param none

@return none

8. Event handlers are callbacks from the SAOpenInst Control into your application. To register with an event handler, right mouse click on the control and select the “Add Event Handler” menu item.



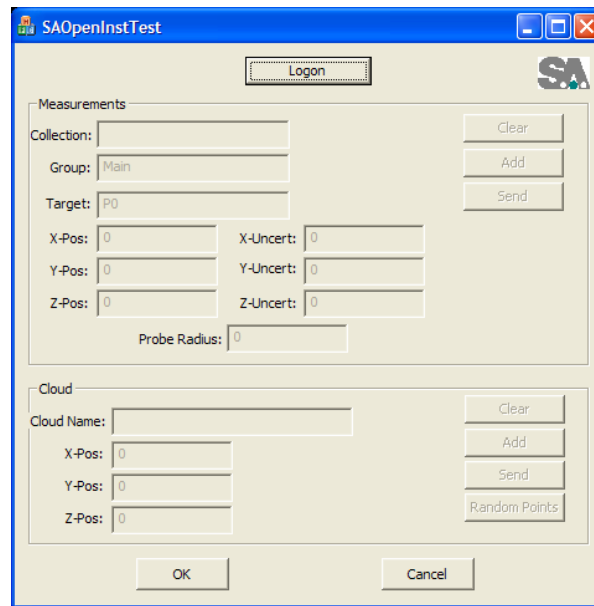
Select the desired event handler from the Event Handler Wizard dialog and Visual Studio will add the necessary code to your dialog:



9. Order of operation / things to keep in mind:

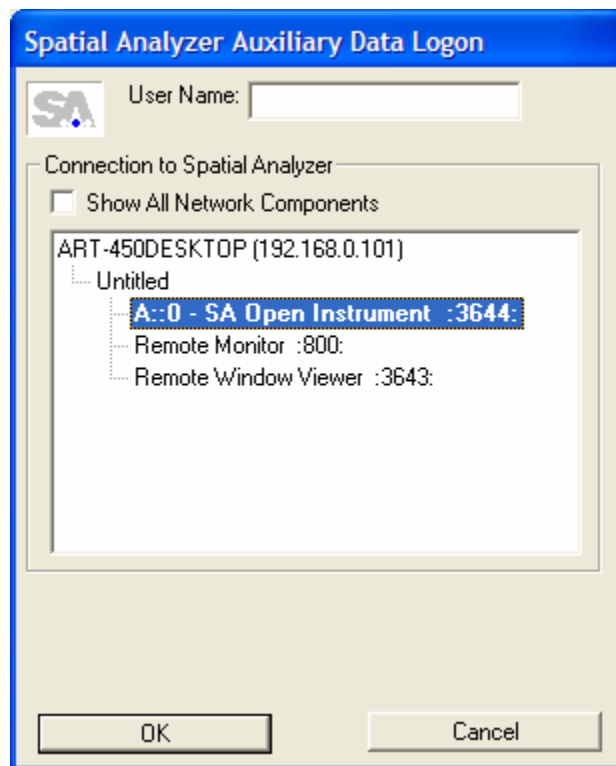
- PopLogonDialog must be called first to setup a connection with Spatial Analyzer and the desired open instrument.
- Always clear your measurements or cloud data after a successful send to SA. Thus, the proper order of operations would be:
 - i. AddMeasurement for each measurement that you wish to send to SA.
 - ii. SendMeasurements to send all buffered measurement data
 - iii. ClearMeasurements to clear all buffered measurement data once successfully sent to SA.

10. Be sure to check out the provided Visual Studio C++ example for a working example of the Open Instrument Interface. The dialog demonstrates sending measurement and cloud data to SA. The demo application looks like this:



The SAOpenInstTest dialog box is used for configuring measurements and cloud data. It features a 'Logon' button at the top center. Below it, the 'Measurements' section includes fields for 'Collection', 'Group' (set to 'Main'), 'Target' (set to 'P0'), and 'Probe Radius' (set to '0'). There are also input fields for X-Pos, Y-Pos, Z-Pos, X-Uncert, Y-Uncert, and Z-Uncert, all currently set to '0'. To the right of these fields are 'Clear', 'Add', and 'Send' buttons. The 'Cloud' section below has a 'Cloud Name' field and 'X-Pos', 'Y-Pos', 'Z-Pos' fields, all set to '0'. It also includes 'Clear', 'Add', 'Send', and 'Random Points' buttons. At the bottom are 'OK' and 'Cancel' buttons.

- With Spatial Analyzer running with a job containing an SA Open Instrument, press the Logon button. Select the desired instrument connection and press the “OK” button.



The Spatial Analyzer Auxiliary Data Logon dialog box is used for connecting to the Spatial Analyzer. It has a 'User Name' field at the top. Below it, the 'Connection to Spatial Analyzer' section includes a checkbox for 'Show All Network Components'. A tree view shows the network structure: 'ART-450DESKTOP (192.168.0.101)' is the root, with a sub-entry 'Untitled'. Under 'Untitled', the entry 'A::0 - SA Open Instrument :3644:' is selected and highlighted. Below this are 'Remote Monitor :800:' and 'Remote Window Viewer :3643:'. At the bottom are 'OK' and 'Cancel' buttons.

- Upon successful connection to SA, the dialog will activate all the measurement / cloud dialog widgets:

- Once connected to SA, enter point or cloud data. Add each item to the send buffer via the corresponding Add button. When ready to send to SA, press the corresponding Send button.
- This example also supports the OnSetColGroupTarget event handler / callback. If you were to run a Measurement Plan from within SA that contains the “Set Instrument Group and Target” command, you should see the demo application’s collection / group / target fields reflect the measurement plan command data.

11. Version Information: Right mouse click on the SAOpenInst ActiveX Control and select the About menu item for product version information:

