SEIComm Instrument Interface Library SDK for .NET

This document describes the SEIComm software development kit and application programming interface for Spectral Evolution's range of spectroradiometer and spectrometer instruments. SEIComm.dll is a dynamic link library built under .NET that encapsulates much of the low-level interface with the instruments, providing a simple interface for custom Windows software implementations.

1. Requirements

In its initial implementation, SEIComm utilized the .NET SerialPort object exclusively, in tandem with Windows virtual COM port (VCP) drivers, for interfacing with instruments via USB, Bluetooth, or Ethernet.

After encountering intermittent data retrieval issues with SerialPort in tandem with USB, we experimented with using FTDI's (manufacturer of the USB module used in our instruments) native communications library instead of the SerialPort implementation, and found the performance more robust. As such, this implementation of SEIComm utilizes both approaches, SerialPort/WinVCP for Bluetooth and Ethernet and FTDI native for USB.

This requires a) that the FTDI drivers are installed on the PC prior to using SEIComm, and b) that the application built in Visual Studio contains references to both

SEIComm.dll and associated DLLs (See Building a New App below), and FTD2XX_NET.dll (FTDI's C#.NET wrapper for their native Win32 library routines), all included in this distribution.

Applications built with SEIComm will generate exceptions if the FTDI drivers are not present on the given PC. Many Windows PC's already come with FTDI's driver installed, but for those that do not have it the FTDI driver installer can be obtained from https://ftdichip.com/drivers/vcp-drivers/. We recommend using the Setup Executable linked from the Comments column in the matrix in the lower portion of that webpage listed under Windows (Desktop): https://ftdichip.com/wp-content/uploads/2021/08/CDM212364_Setup.zip.

To install the driver, run the installer .exe file and when complete connect your powered SEI instrument to the PC with USB cable. The driver will complete installation when the instrument is recognized. Once the driver is activated, applications using SEIComm will run properly.

2. Sample Project

This distribution includes a Visual Studio solution and project called SEIComm Example Program, which is a simple C#/.NET Framework V4.8 Windows Forms project, built under Visual Studio 2022. It illustrates much of the basic API functionality, namely

- Determining the ports that are available for USB or VCP communications
- Selecting and opening a port for communications with the SEI instrument
- Initializing the instrument, namely retrieving static parameters, wavelength and radiometric calibration data stored in the instrument's flash and dynamic parameters such as averaging/integration from the instrument's microprocessor
- Sending new values for some of those parameters to the instrument
- Requesting and displaying spectral scan data
- Closing the communications port

3. Building a new app

For new .NET Framework applications built in Visual Studio, once you have created your initial solution and project files, right-click the References heading and select "Add Reference", then browse to select SEIComm.dll and supporting DLL files (FTD2XX_NET.dll, NewProc2018.dll, NewUtility.dll, SEIBasic.dll). These files are all included in the sample project in a folder called DLLStorage.

Your project source code should declare a member variable of the class SpectralEvolution.SEIComm. The SEIComm object is used for all subsequent communications with the instrument. You can include a "using SpectralEvolution;"

directive at the top of your main code module to specify the namespace SpectralEvolution and avoid some typing.

Once declared, the Form constructor should initialize the SEIComm object, and optionally subscribe to events and generate event handlers. The StatusResponse event provides text updates during lengthy operations (especially the Initialize method) and asynchronous scan methods generate another event when data is available (ScanResponse) or an error occurs (CommErrorResponse).

Both synchronous and asynchronous scan methods are included. The GetAsync___Scan methods are recommended for WinForms applications over the GetBlocking____Scan methods, as scan responses from the instrument can take multiple seconds to complete, and the asynchronous methods allow the GUI thread to remain responsive to user input. For non-GUI based applications the synchronous (blocking) functions should be sufficient.

The SEIComm object, once initialized, contains member variables of other classes included in the library (Config, Param, Wavelength, RadioCal) which have metadata property values that can be referenced as needed. Instances of the SpectralData class are returned by the various scan methods, and contain the arrays of raw and processed spectral data that your application can then copy, display, save to file, etc. as the user sees fit.

4. API Reference

4.1. SEIComm Class:

Required. This is the basic element used for SEI instrument interface. It encapsulates communications with the instrument and handles most processing functions.

The application should declare a single object instance of the SEIComm class in the main processing code and use the methods and event handlers to issue commands and read back responses.

4.1.1. Constructor

SEIComm() Default constructor, no parameters

4.1.2. Member variables

CNFG (see discussion of Config class below), Static configuration parameters stored in the instrument's flash memory. Set up by the Initialize() method.

IAP (new for V1.1) Instrument-specific settings used by SEIComm to report detector temperature and voltage alarms described in the Properties section below.

PARM (see discussion of Param class below), Dynamic parameters stored in instrument RAM. Set up by the Initialize(), SetAveraging(), SetAutoIntegration() and SetFixedIntegration() methods.

RCArray (see discussion of RadioCal class below) Array of RadioCal objects referencing radiometric calibration coefficient data sets stored in the flash memory of spectroradiometer instruments. Used for converting the raw spectral data to radiance/irradiance depending on the optics. Set up by the Initialize() method.

SCD (see discussion of SEICommData class below), Information about the last communications interaction with the instrument (useful for debug). Updated whenever a command is issued.

WVL (see discussion of Wavelength class below), Instrument-specific wavelength calibration array data stored in instrument flash. Set up by the Initialize() method.

4.1.3. Enumerations

SEIErrorType {TIMEOUT, PARSE_ERROR, PORT_ERROR}
SEIPortType {FTDI_USB, DOTNET_VCP, INVALID}

4.1.4. Properties (Read-only)

double InGaAs1HiAlarmTemperature: High-temperature limit for the thermoelectric (TE) cooler for the InGaAs1 (SWIR1) detector typically covering 1000-1900nm on multi-detector systems. A fault condition (temperature reading above this value) during a scan will cause a fault alarm flag to be set in the returned SpectralData object. Value is dependent on instrument model.

double InGaAs2HiAlarmTemperature: High-temperature limit for the thermoelectric (TE) cooler for the InGaAs2(SWIR2) detector typically covering 1900-2500nm on multi-detector systems. A fault condition (temperature reading above this value) during a scan will cause a fault alarm flag to be set in the returned SpectralData object. Value is dependent on instrument model.

bool IsOpen: TRUE if the communications port is currently open, FALSE otherwise.

double LowAlarmVoltage Minimum operating voltage for the instrument to maintain data quality. A fault condition (voltage below this value) during a scan will cause a fault alarm flag to be set in the returned SpectralData object. Value is dependent on instrument model.

double TECSiliconHiAlarmTemperature: High-temperature limit for systems that contain TE-cooled silicon detectors covering the VNIR range typically from 350-1000nm, e.g. SR-4500. A fault condition (temperature reading above this value) during a scan will cause a fault alarm flag to be set in the returned SpectralData object. Value is dependent on instrument model.

4.1.5. Methods

Most of the methods used by SEIComm have a return parameter of the SEIStatus enumeration type. SEI_OK means the method was successful, other values reflect error conditions as reported by the underlying FTDI library routines or other states.

Close()
Closes the currently-open VCP or FTDI communications port.
Parameters: none
Returns: void

CycleUSB()
Calls the FTDI library's CyclePort function in case of USB connection issues.
Parameters: none
Returns: SEIStatus enum (SEI_OK if successful)

GetAllRadCals()
Populates the RCArray of radiometric calibration data from instrument flash. This is typically handled during Initialization but can be called as needed.
Parameters: none
Returns:SEIStatus enum (SEI_OK if successful)

GetAsyncCalibratedScan(RadioCal RC) + 3 overloads
Requests calibrated spectroradiometer data (spectral radiance or irradiance) from the instrument based on the parameter RC. The method returns immediately and the ScanResponse event fires when the data is available, or CommErrorResponse if a timeout or parsing failure occurs.
Parameters: RC (typically one element indexed from the RCArray member),
Optional: uint Timeout – value in milliseconds allowing user override of default message timeout.
bool Verbose – flag to enable/disable StatusUpdate messages.
Returns: SEIStatus (SEI_OK if command issued successfully)

GetAsyncRatioScan(SpectralData RefSD, bool UsingContactProbe) + 3 overloads
Requests ratio spectrum (reflectance/transmittance) of unknown target as compared with the reference spectrum in RefSD. The method returns immediately and the ScanResponse event fires when the data is available, or CommErrorResponse if a timeout or parsing failure occurs.
Parameters: RefSD (spectral data of nominal reference for ratio calculation),
UsingContactProbe (when TRUE, small scaling shifts may be applied to the silicon and InGaAs2 portions of the multi-detector spectrum in order to minimize artifacts in the overlapping regions,
Optional: uint Timeout – value in milliseconds allowing user override of default message timeout.
bool Verbose – flag to enable/disable StatusUpdate messages.
Returns: SEIStatus (SEI_OK if command issued successfully)

GetAsyncReferenceScan() + 3 overloads
Requests reference spectrum (e.g., of white diffuse reflectance plate) for use with subsequent ratio measurements. The method returns immediately and the ScanResponse event fires when the data is available, or CommErrorResponse if a timeout or parsing failure occurs.
Parameters: none required
Optional: uint Timeout – value in milliseconds allowing user override of default message timeout.
bool Verbose – flag to enable/disable StatusUpdate messages.
Returns: SEIStatus (SEI_OK if command issued successfully)

GetBlockingCalibratedScan(out SpectralData SD, RadioCal RC) + 1 overload
Requests calibrated spectroradiometer data (spectral radiance or irradiance) from the instrument based on the parameter RC and blocks the execution thread until data is returned from the instrument or timeout/error occurs.
Parameters: SD (out parameter, spectral data returned from instrument)

RC (RadioCal to apply to the data for spectral radiance/irradiance output)
Optional: uint Timeout – value in milliseconds allowing user override of default message timeout.
Returns: SEIStatus (SEI_OK if completed scan successfully)

GetBlockingRatioScan(out Spectral Data TgtSD, ref SpectralData RefSD, bool UsingContactProbe) + 1 overload
Requests ratio spectrum (reflectance/transmittance) of unknown target as compared with the reference spectrum in RefSD and blocks the execution thread until data is returned from the instrument or timeout/error occurs.
Parameters: TgtSD (out parameter, ratio spectral data returned from instrument)
RefSD (spectral data of nominal reference for ratio calculation)
UsingContactProbe (when TRUE, small scaling shifts may be applied to the silicon and InGaAs2 portions of the multi-detector spectrum in order to minimize artifacts in the overlapping regions.
Optional: uint Timeout – value in milliseconds allowing user override of default message timeout.
Returns: SEIStatus (SEI_OK if completed scan successfully)

GetBlockingReferenceScan(out SpectralData RefSD) + 1 overload
Requests reference spectrum (e.g., of white diffuse reflectance plate) for use with subsequent ratio measurements and blocks the execution thread until data is returned from the instrument or timeout/error occurs.
Parameters: RefSD (out parameter, spectral data returned from instrument)
Optional: uint Timeout – value in milliseconds allowing user override of default message timeout.
Returns: SEIStatus (SEI_OK if completed scan successfully)

GetUSBPorts(out string[] Names, out string[] Descriptions)
Uses the FTD2XX_NET library to obtain arrays of COM port names and internal Descriptions of any connected FTDI USB ports. The application can use these to distinguish the USB-based virtual COM ports from virtual COM ports assigned via Bluetooth or Ethernet. An element from either Names or Descriptions can be used as the input string to the Open method.
Parameters: Names (out parameter, array of COM port names assigned to FTDI USB ports
Descriptions (out parameter, array of descriptions of FTDI ports)
Returns: void

Initialize()

Performs the SEI instrument initialization procedure, retrieving data from the instrument's flash memory and microprocessor. When complete, the member variables CNFG (configuration string), PARM (current instrument operating parameters), WVL (wavelength calibration) and RCArray (radiometric calibration coefficient sets) will be populated and available for use.
The method returns immediately. Subscribing to the InitializationComplete event lets the application know when to proceed with data acquisition, and the StatusResponse event provides text string feedback of the initialization progress.
If Initialize() is called but the InitializationComplete event is not subscribed, wait until the CNFG, PARM, Wvl and RCArray have been populated before requesting scan data. .
Parameters: none
Return SEIStatus (SEI_OK if request issued successful)

Open(string PortNameOrDescription)
Attempts to open the specified port. If the PortNameOrDescription does not begin with "COM" it is assumed to be the description of an FTDI USB device and opened with the FTD2XX library. If it does begin with "COM" it is tested to find out if it's an FTDI USB or Windows VCP port, and is then handled accordingly.

Parameter: PortName (string, either COM## or FTDI device description)
Returns: SEIStatus (SEI_OK if successful)

OpenWinVCP(string PortName)
Attempts to open the Windows SerialPort (typically used with a Bluetooth adapter with Windows VCP (Virtual COM Port) drivers, without using FTDI's library routines.
Parameter: PortName (string, typically COM##)
Returns: SEIStatus (SEI_OK if successful)

RescanUSB()
Attempts to resolve USB communication issues by invoking the Rescan method from the FTDI library (FTD2XX_NET.dll)
Parameters: none
Returns: SEIStatus (SEI_OK if command was issued successfully)

ResetUSBDevice()
Attempts to resolve USB communication issues by invoking the ResetDevice method from the FTDI library (FTD2XX_NET.dll)
Parameters: none
Returns: SEIStatus (SEI_OK if command was issued successfully)

ResetUSBPort()
Attempts to resolve USB communication issues by invoking the ResetPort method from the FTDI library (FTD2XX_NET.dll)
Parameters: none
Returns: SEIStatus (SEI_OK if command was issued successfully)

SetAutoIntegration()
Directs the connected instrument to perform scans using automatic integration (gain).
Parameters: none
Returns: SEIStatus (SEI_OK if successful)

SetAveraging(int NewValue)

Adjusts the averaging setting being used for subsequent scans. Higher numbers produce smoother results but scans take longer to complete.
Parameters: NewValue (int, averaging setting typically from 1-100)
Returns: SEIStatus (SEI_OK if successful)

SetFixedIntegration(int DetectorCount, int WhichDetector, int NewValue)
Directs the connected instrument to perform scans using the specified new fixed integration value for the specified detector. In multi-detector systems, changing from automatic to fixed integration will require new values for each detector. For example, to set a 3-detector system to the fixed setting [5, 10, 20], the application should call

```
SetFixedIntegration(3, 1, 5);
SetFixedIntegration(3, 2, 10);
SetFixedIntegration(3, 3, 20);
```

Parameters: DetectorCount (int, number of detectors in the instrument: 1, 2, or 3)
WhichDetector (int, one-based index of detector to set
For Silicon use 1, InGaAs1 use 2, and InGaAs2 use 3
NewValue (int, new integration setting between 1 and the limit specified by the MaxInteg_ properties in CNFG
Returns: SEIStatus (SEI_OK if successful)

4.1.6. Events

CommErrorResponse (SEIErrorArgs)

Event generated when an error in communication occurs, whether related to port access or to a parsing error. The SEIErrorArgs parameter has the following property values:

ErrorMessage (type string) Description of the error
ErrorType (type SEIErrorType) Category description, either PARSE_ERROR, PORT_ERROR, or TIMEOUT.

InitializationComplete (no arguments)
Event generated when the SEIComm object has completed the multi-step Initialize() method. Once this event is triggered the Config, Param, RadioCal and Wavelength data have been successfully retrieved and SEICom can proceed with acquiring spectral scan data.

ScanResponse (SEISpectralDataEventArgs)
Event generated once the instrument has completed acquiring data via one of the GetAsync____Scan methods. The SEISpectralDataEventArgs parameter contains a member variable SD of type SpectralEvolution.SpectralData (defined in a later section), containing the raw spectral data as well as metadata such as detector temperature(s), voltage, integration settings, etc.

StatusResponse (SEIUpdateArgs)
Event generated by methods such as Initialize(). The SEIUpdateArgs parameter has a string member variable Status with text information about the instrument state that can be displayed in the GUI.

4.2. Config Class:

Contains static configuration parameters from instrument flash memory storage. Retrieved by the SEIComm.Initialize() method.

4.2.1. Properties (read-only)

string ConfigString: Configuration string stored in instrument flash and used to generate the elements in the Config object.

int DetectorCount: Number of detector arrays used in the instrument (1, 2, or 3)

string FirmwareVersion: Firmware build version for the instrument

bool HasInGaAs1: TRUE if the instrument uses an InGaAs SWIR1 detector, typically covering the wavelength range from 1000 to 1900 nm

bool HasInGaAs2: TRUE if the instrument uses an InGaAs SWIR2 detector, typically covering the wavelength range from 1900 to 2500 nm

bool HasSilicon: TRUE if the instrument uses a silicon detector, typically covering the VIS/NIR wavelength range from 350 to 1000nm

double HighWavelength: Nominal top wavelength limit supported by the instrument.

bool IsValid: TRUE if the ConfigString could be properly parsed, and indicates that the Config object can be used for subsequent scan processing.

double LowWavelength: Nominal bottom wavelength limit supported by the instrument.

int MaxAveraging: Maximum valid setting for the Averaging parameter, limit for SEIComm.SetAveraging method. Value is typically 100 for most instruments, 255 for some single-detector silicon-based systems like the PSR-1100.

int MaxInteg_InGaAs1: Maximum valid integration setting for InGaAs SWIR1 detector, -1 if the detector is not present. Should be used to limit integration setting in the SEIComm.SetFixedIntegration method.

int MaxInteg_InGaAs2: Maximum valid integration setting for InGaAs SWIR2 detector, -1 if the detector is not present. Should be used to limit integration setting in the SEIComm.SetFixedIntegration method.

int MaxInteg_Silicon: Maximum valid integration setting for silicon VIS/NIR detector, -1 if the detector is not present. Should be used to limit integration setting in the SEIComm.SetFixedIntegration method.

Int[] MaxIntegrationSettings: Array of integers (length 1, 2, or 3 depending on instrument) reflecting the maximum valid integration settings for each detector. Same information as in the MaxInteg_(detector) properties, presented slightly differently.

string Model: SEI instrument model name, e.g. “SR-3500”

string SerialNumber: SEI instrument serial number, e.g. “16A8001”

int[] ValidRadioCals: Array of integers reflecting the index numbers (1 through 8 possible) of instrument flash memory slots that contain factory-validated radiometric calibration data. Uncalibrated spectrometers (e.g. OreXpress, SM-3500) may return an

empty array. Calibrated spectroradiometers (e.g. SR-3500) may have any combination from [1] to [1,2,3,4,5,6,7,8]

4.3. Param Class:

Run-time dynamic parameters retrieved from the instrument's microcontroller, updated by calls to the SEIComm.Initialize, SetAveraging, SetAutoIntegration, and SetFixedIntegration methods.

4.3.1. Properties (read-only)

bool AutoIntegration (get, type bool) TRUE if the instrument is currently set for automatic integration (gain) adjustment.

int Averages (get, type int) Current averaging setting being used for spectral scans (typically 10 at startup)

int IntegInGaAs1 (get, type int) Current fixed integration setting for the InGaAs SWIR1 detector (not valid if AutoIntegration is TRUE). -1 if detector is not present.

int IntegInGaAs2 (get, type int) Current fixed integration setting for the InGaAs SWIR2 detector (not valid if AutoIntegration is TRUE). -1 if detector is not present.

int[] Integration (get, type int[])Array of integers (length 1, 2, or 3 depending on instrument) reflecting the current fixed integration settings for each detector. Not valid if AutoIntegration is TRUE.

int IntegSilicon (get, type int) Current fixed integration setting for the silicon VIS/NIR detector (not valid if AutoIntegration is TRUE), -1 if detector is not present.

bool IsValid (get, type bool) Set to TRUE if the most-recently-retrieved parameter set was parsed successfully. If FALSE the other property values should be disregarded.

bool SupportsLongAveraging (new in V1.1) set TRUE for instruments (such as SR-6500) that allow for averaging values past 100 for extreme smoothing. In these cases, legal values for SEIComm.SetAveraging() are [1-20] or [40, 60, 80, 100, …4660, 4680, 4700]. Note that selecting large values may take cause the instrument to be unresponsive for several minutes before scan responses are issued.

4.4. RadioCal Class:

Object representing a radiometric optical calibration data set stored in the flash memory of calibrated spectroradiometer instruments. The SEIComm object declares a member variable RCArray (an array of RadioCal objects) populated by the Initialize() method. One selected RadioCal is used as an input parameter for the GetAsyncCalibratedScan and GetBlockingCalibratedScan methods. The default ToString() method is overridden to display the RadioCal's name and type in comboboxes or other GUI controls.

4.4.1. Enumeration

CalibrationType {Radiance, Irradiance, None, Invalid}

4.4.2. Properties (read-only)

CalibrationType CalType: Descriptor for the type of calibration coefficients stored. Radiance: output spectra are calculated as spectral radiance in units of W/(m^2·sr·nm) Irradiance: output spectra are calculated as spectral irradiance in units of W/(m^2·nm) None, Invalid: no calibration is applied to output spectra. Results will be presented as digital count numbers normalized by the current detector integration setting.

bool IsValid: TRUE if the calibration data was retrieved and parsed successfully.

string Name: Up to six-character label for calibration data set as stored in flash memory, e.g. "LENS4", "Difuzr", "None".

4.4.3. Methods

ToString()

Overrides base ToString() method, returning the Name and CalType for display when the RadioCal object is referenced by a GUI element such as a combobox.

Parameters: none
Returns: text string e.g. "LENS4 (Radiance)"

4.5. SEICommData Class:

Data about the last serial communications interaction received from the instrument, predominantly for debugging purposes.

4.5.1. Members

uint ExpectedBytes: Number of bytes expected in the response from the last-issued command to the instrument.

uint ReceivedBytes: Number of bytes actually received since the last-issued command to the instrument.

4.6. Wavelength Class:

SEIComm member object that contains the wavelength calibration as stored in instrument flash, retrieved by the SEIComm.Initialize method.

4.6.1. Properties (read-only)

double[] InterpolatedWavelengths: Wavelength array in nanometers, monotonically increasing in 1.0 nm intervals and matched to processed output.

bool IsValid: TRUE if the wavelength data was successfully retrieved from the instrument and ready for matching to acquired spectral data.

double[] RawChannelWavelengths: Unsorted native detector channel wavelengths in nm, corresponding to the raw spectral data output, converted to doubles. Presented in detector order, e.g. 512 channels for silicon, then 256 for InGaAs1 and 256 for InGaAs2.

4.7. ScanType Enumeration

Used by SpectralData class to indicate type of scan to be processed:
ScanType { CALIBRATED, RATIO_REFERENCE, RATIO_TARGET, INVALID}

4.8. SpectralData Class:

SpectralData objects are returned as out parameters by the GetBlocking____Scan methods, or as members of the SEISpectralDataEventArgs parameter in the ScanResponse event generated by the GetAsync_____Scan methods.

The class contains the raw spectral data returned by the SEI instrument and processed output (e.g. with radiometric calibration applied, ratio relative to a reference spectrum calculated, output interpolated to 1nm spacing).

4.8.1. Properties (read-only)

bool AutoIntegration: TRUE if the scan was collected with automatic integration, FALSE if fixed.

int Averages: Averaging setting used for the current scan.

bool InGaAs1TempAlarm: TRUE if the instrument features a thermoelectrically-cooled InGaAs SWIR1 detector and the scan response reported a temperature in excess of the SEIComm.InGaAs1HiAlarmTemperature property.

double InGaAs1Temperature: Temperature reported by the TE-cooled InGaAs SWIR1 detector on the current scan, if the instrument features one.

bool InGaAs2TempAlarm: TRUE if the instrument features a thermoelectrically-cooled InGaAs SWIR2 detector and the scan response reported a temperature in excess of the SEIComm.InGaAs2HiAlarmTemperature property.

double InGaAs2Temperature: Temperature reported by the TE-cooled InGaAs SWIR2 detector on the current scan, if the instrument features one.

int InstrumentErrorCode Numeric error code specified in the scan response header (included primarily for SEI debug use)

int IntegInGaAs1: Integration value reported for the InGaAs SWIR1 detector on the current scan, if that detector is present. -1 if absent.

int IntegInGaAs2: Integration value reported for the InGaAs SWIR2 detector on the current scan, if that detector is present. -1 if absent.

int[] Integration: Array of 1, 2, or 3 integration values, depending on instrument detector layout.

int IntegSilicon: Integration value reported for the silicon VIS/NIR detector on the current sscan, if that detector is present. -1 if absent.

bool IsSaturated: TRUE if the instrument reported a detector saturation condition during the current scan acquisition.

bool IsValid: TRUE if the data returned by the instrument was parsed successfully.

double[] ProcessedSpectrumInterp: Array of processed spectral output data (calibrated or ratio), with detector data matched to the interpolated and monotonically-increasing ProcessedWavelengthsInterp.

double[] ProcessedWavelengthsInterp: Array of processed wavelength data in nanometers, interpolated to 1.0 nm spacing and monotonically increasing.

double[] RawChannelWavelengths: Array of raw channel center wavelengths rounded to nearest 0.1nm, grouped by detector.

short int[] RawSpectrumInDN: Array of unsorted 16-bit signed integer spectral digital number output from the instrument, grouped by detector as with RawChannelWavelengths.

SpectralEvolution.ScanType ScanType: Depending on the type of spectrum requested, either CALIBRATED, RATIO_REFERENCE or RATIO_TARGET.

double SystemTemperature: Internal temperature measured by the instrument, typically ranges from 20°- 40°C.

bool TECSiliconTempAlarm: TRUE if the instrument features a thermoelectrically-cooled silicon VIS/NIR detector and the scan response reported a temperature in excess of the SEIComm.TECSiliconHiAlarmTemperature property.

double TECSiliconTemperature: Temperature reported by the TE-cooled silicon VIS/NIR detector on the current scan if the instrument features one (e.g. SR-4500).

double Voltage: Supply voltage reported by the instrument on the current scan. Typical operating voltages are in the vicinity of 7.5V.

bool VoltageAlarm: TRUE if the instrument read the voltage on the current scan as being less than the SEIComm.LowAlarmVoltage property.

4.8.2. Methods

DeepClone()
Creates a copy of the SpectralData object, but instead of the shallow copy supported by the ICloneable interface, DeepClone makes separate copies of all array members as well.

Parameters: none
Returns object (cast as SpectralData)

ProcessRatio(SpectralData RefScan, bool ApplyProbeCorrection)
Allows for recalculating spectral ratio data (reflectance, transmittance, etc.). Calculates the ratio of the data contained in the current SpectralData object divided by the data in RefScan. The results are copied to ProcessedSpectrumInterp.

Parameters: RefScan (valid SpectralData object containing the spectrum of the reference to be used for the ratio calculation
ApplyProbeCorrection (when TRUE, applies small scaling shifts to the silicon and InGaAs2 portions of the multi-detector spectrum in order to minimize artifacts in the overlapping regions)
Returns: SEIStatus (SEI_OK if successful)

ProcessReference()
Allows for recalculating raw spectral scan data as nominal white plate reference data for subsequent ratio calculations.

Parameters: none
Returns: SEIStatus (SEI_OK if successful)

ProcessUsingCalibration(RadioCal RC)
Allows for recalculating calibrated spectroradiometer output with the selected RadioCal applied. If RC is of CalType Radiance or Irradiance, those data will be stored in the ProcessedSpectrumInterp array. Otherwise, that array will be populated with interpolated digital count numbers normalized by the detector integration value.

Parameters: RC (valid RadioCal object to be applied to the raw data)
Returns: SEIStatus (SEI_OK if successful)