

Agilent Technologies Instrument Interface

Application Programming Interface Guide

Submitted to Agilent Technologies by
DLS Solutions, Inc.

January 30, 2018



Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 2 of 97	Revision: 1.22

Revision History

Revision	Author	Date	Description of Changes
0.1	Jeff Levi	12/1/07	Initial Version/Template layout.
0.2	Chad Lewis	1/9/08	Updated to include interface descriptions for all MLInstrumentInvoke C# interface calls.
1.0	Chad Lewis	1/15/08	Completed/Reviewed version 1 of document.
1.1	Chad Lewis	2/13/08	Added documentation of the StartCoaddedIGram And dpnrGetCoaddedIGram methods.
1.2	Chad Lewis	3/26/08	Added documentation for FTIRInst_RegisterButton1 and FTITInst_RegisterButton2 as well as updated the definition for the GetPathLenEx method.
1.3	Jeff Levi	3/26/08	Separated documentation of the GetCoaddedIgramNotify function. Updated the Document Properties Revision number.
1.4	Chad Lewis	9/16/08	Added documentation for new function calls.
1.5	Chad Lewis	1/15/09	Updated diag structs to include instrument shutdown flags
1.6	Chad Lewis	1/22/09	Added documentation for FTIR_RegisterStatus
1.7	Jonathan Lowthert	2/9/09	Added entry for FTIR_RegisterStatusEvents.
1.8	Jonathan Lowthert	3/5/09	Corrected FTIRInst_CheckProgressStruct() return value.
1.9	Jonathan Lowthert	3/9/09	Updated return values to match instrument.
1.10	Jonathan Lowthert	3/23/09	Added FTIRInst_SetAppLedState() function.
1.11	Jonathan Lowthert	5/1/09	Updated MLVersion and MLDiag structures.
1.12	Jonathan Lowthert	6/2/09	Added FTIRInst_SetTargetDeviceXxx() functions and related instructions.
1.13	Jonathan Lowthert	9/16/09	Added functions relating to OEM-specific external I2C devices (ADC, GPIO, Temperature).
1.14	Jonathan Lowthert	9/22/09	Added further documentation for FTIRInst_GetVersionEx() serial number.
1.15	Jonathan Lowthert	1/11/2010	Modified FTIRInst_GetExtTemps() and added FTIRInst_GetExtTemp().
1.16	Jonathan Lowthert	2/23/2010	Corrected documentation for FTIRInst_GetPathLenEx().
1.17	Jonathan Lowthert	3/8/2010	Added documentation for FTIRInst_GetOemNvmemData() and FTIRInst_SetOemNvmemData().
1.18	Jeff Levi	7/6/10	Corrected RegisterButton1 and RegisterButton2 prototypes
1.19	Jonathan Lowthert	11/16/2010	Added documentation for new functions related to laser temperature compensation.
1.20	Jonathan Lowthert	2/28/2011	Added more documentation for laser corrections. Also integrated Network Supplement document.
1.21	Jonathan Lowthert	3/15/2011	Updated MLSyncdVals to have int values instead of long so the definitions will work for both C and C#.
1.22	Jeff Levi	10/19/2012	Updated to Agilent Technologies Adjusted to EXCLUDE FTIRInst_SoftReset (obsolete) Adjusted to INCLUDE FTIRInst_dpnrGetLiveIGram

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 3 of 97	Revision: 1.22

Table of Contents

Agilent Technologies Instrument Interface Overview	5
Figure 1 : Architectural block diagram of Interface	6
Data Types and Notation	7
FTIRInst DLL	10
<i>Summary of Function Interfaces</i>	10
Typical Usage Patterns	12
FTIRInst_SetTargetDeviceUsb	13
FTIRInst_SetTargetDeviceSerialPort	14
FTIRInst_SetTargetDeviceNetwork	15
FTIRInst_Init	16
FTIRInst_Deinit	17
FTIRInst_SetComputeParams	18
FTIRInst_dpPtrStartSingleBeam	20
FTIRInst_dpPtrStartSpectrum	22
FTIRInst_dpPtrGetLiveSpectrum	24
FTIRInst_dpPtrGetSingleBeam	26
FTIRInst_dpPtrGetBackground	28
FTIRInst_dpPtrGetClean	30
FTIRInst_dpPtrGetSpectrum	31
FTIRInst_dpPtrGetRatioSpectrum	33
FTIRInst_KillCollection	34
FTIRInst_SetLaserWaveNumber	35
FTIRInst_SetPathLen	36
FTIRInst_GetLaserWaveNumber	37
FTIRInst_GetPathlenEx	38
FTIRInst_GetVersion	40
FTIRInst_GetVersionEx	41
FTIRInst_GetStatus	43
FTIRInst_GetStatusEx	45
FTIRInst_CheckProgress	47
FTIRInst_CheckProgressEx	48
FTIRInst_CheckProgressStruct	49
FTIRInst_StartCoaddedIGram	50
FTIRInst_StartCoaddedIGramNotify	52
FTIRInst_dpPtrGetCoaddedIGram	54
FTIRInst_RegisterButton1	56
FTIRInst_RegisterButton2	57
FTIRInst_dpPtrSetBackground	58

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 4 of 97	Revision: 1.22

FTIRInst_dpPtrGetLiveSingleBeam.....	60
FTIRInst_dpPtrGetLiveIGram.....	62
FTIRInst_GetIrGain.....	64
FTIRInst_RegisterStatus.....	66
FTIRInst_RegisterStatusEvents.....	68
FTIRInst_SetAppLedState.....	69
FTIRInst_I2cAdc_GetReadings	70
FTIRInst_I2cIo_SetPinDirs	71
FTIRInst_I2cIo_SetPinVals.....	72
FTIRInst_I2cIo_GetPinVals	73
FTIRInst_GetExtTemps.....	74
FTIRInst_GetExtTemp	75
FTIRInst_GetOemNvmemData.....	76
FTIRInst_SetOemNvmemData.....	77
FTIRInst_GetLaserTempCompVals.....	78
FTIRInst_SetLaserTempCompVals	79
FTIRInst_EnableLaserTempCompensation	80
FTIRInst_SetLaserStandardizationType.....	81
FTIRInst_GetSyncdVals.....	82
Network Interface	83
Data Types and Notation	84
FTIR Device General Network Operation.....	88
General Network Connectivity	88
Limited Network Bandwidth	88
Wireless Network Connections.....	88
Firewall Settings	89
Connecting and Disconnecting	89
Startup Connection Interface and Fallback.....	90
FTIRInst DLL – Network Functions	92
<i>Summary of Function Interfaces</i>	92
Typical Usage Patterns	93
Setting New Network Values.....	93
FTIRInst_GetNetConfig	94
FTIRInst_SetNetConfig.....	96

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 5 of 97	Revision: 1.22

Agilent Technologies Instrument Interface Overview

The main component for interfacing to the Agilent Technologies MicroLab™ Instruments is through the FTIRInst.DLL interface DLL. This DLL provides a set of C-Callable, high-level language interface calls for communicating with the Agilent Technologies' instrumentation products. The DLL interface can be invoked from VB, C, and C# modules, and is compatible with Windows, .NET framework, and Windows CE .NET Compact framework platforms. The interface described in this document will detail the functional interface for use by a C# InteropService client wrapper class.

The interface DLL will encapsulate and wrap all details of the Driver interface layer below the DLL level. A simulation DLL (FTIRInst_sim.DLL) provides a software only test solution, to provide an interface plug-in replacement that works seamlessly with applications built for the FTIRInst target interface.

Note: The simulation DLL may not provide all of the functionality described in this document, and the return values may differ from those that the live instrument DLL provides.

The wrapper class for C# provides a number of interfaces designed for ease of use in interfacing with C# client applications. These routines generally wrap the DLL routines, provide data type transformations, and or combine a number of DLL routines into one call.

For network configuration of network-capable FTIR devices, please see the separate document, *Agilent Technologies Instrument Interface: Network Supplement*. Access remains through the same DLLs.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 6 of 97	Revision: 1.22

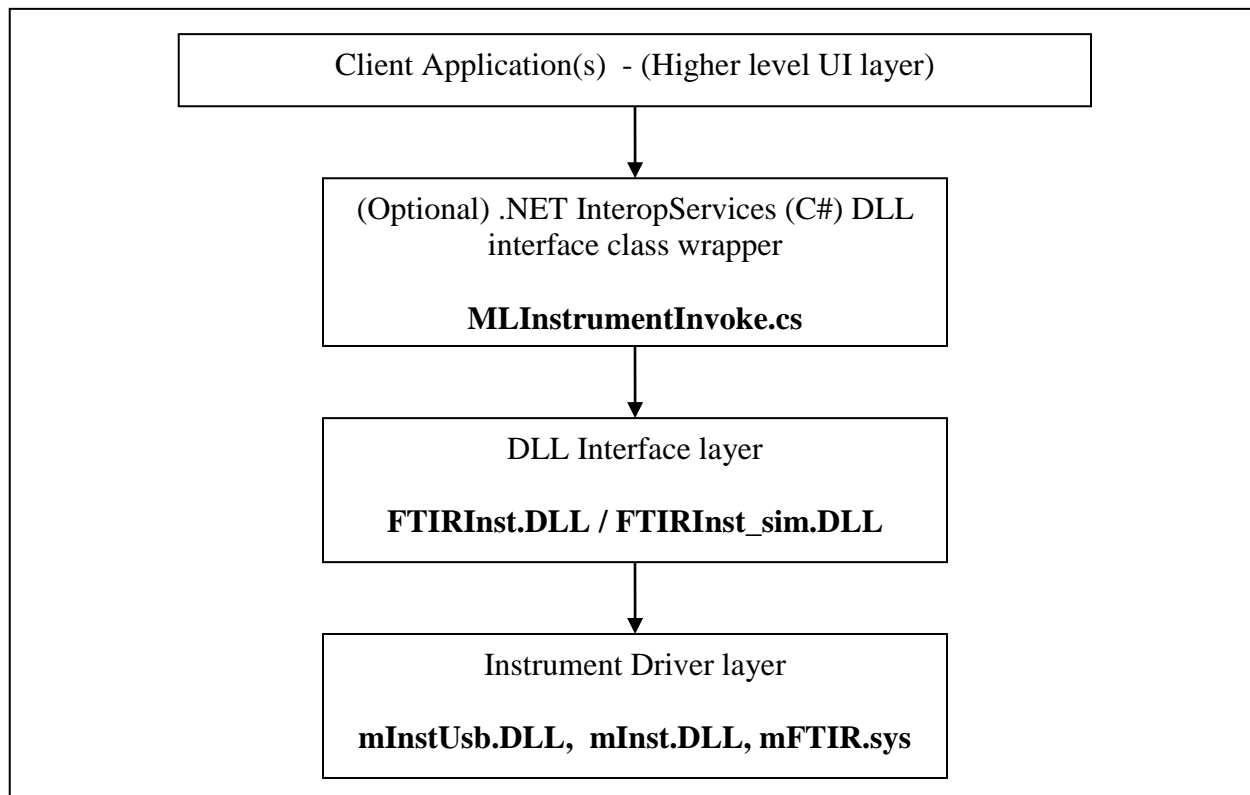


Figure 1 : Architectural block diagram of Interface

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 7 of 97	Revision: 1.22

Data Types and Notation

The data types used in this document are the types as defined in the .NET framework:

Standard data types

int	32-bit integer (typically a long or DWORD in older style languages)
short	16-bit integer
float	32-bit floating point value (typically a float or single precision value in older style languages)
double	64-bit floating point value (typically a double precision value in older style languages)
ref and or array[]	This is the .NET notation convention for a reference to a data type or a reference to an array of values (typically a pointer or ByRef value in older style languages)
public and private	Protection and accessibility level of a variable or function

Enumerations (specific sets of values stored as 32-bit integer values as follows)

```

PHASETYPE { PT_MERTZ = 1, PT_FORMAN = 2, PT_FORMANRES = 3 };
APODTYPE { APOD_NONE = 0, APOD_BOXCAR = APOD_NONE, APOD_TRIANGULAR = 1,
APOD_WEAKNORTONBEER = 2, APOD_MEDIUMNORTONBEER = 3,
APOD_STRONGNORTONBEER = 4, APOD_HAPPGENZEL = 5, APOD_BESSEL = 6,
APOD_COSINE = 7, APOD_HANNING = APOD_COSINE, };
FTIR_STATE { FTIR_Init = 0, FTIR_Collecting = 1, FTIR_DataReady = 2,
FTIR_Aborting = 3, FTIR_Error = 4, };
PHASEPOINTS { PP_128 = 128, PP_256 = 256, PP_512 = 512, PP_1024 = 1024 };
OFFSETCORRECTTYPE { OT_NONE = 0, OT_ALL = 1, OT_ENDS = 2 };
ZFFTYPE { ZFF_NONE = 0, ZFF_2 = 1, ZFF_4 = 2, ZFF_8 = 3, ZFF_16 = 4 };
SAMPLINGTECHNOLOGYTYPE { ST_NONE = 0, ST_ATRSINGLE = -1, ST_ATRTRIPLE = -3,
ST_ATRNINEBOUNCE = -9, ST_TRANSMISSIONCELL = 1, ST_GASCELL = 2,
ST_REFLECTANCE = 3, };
ML_INSTRUMENT_TYPE { eInstrumentType_Undefined = 0,
eInstrumentType_ML = 1, eInstrumentType_MLP = 2,
eInstrumentType_MLX = 3, eInstrumentType_Exoscan = 4, };
DATATYPE { XT_ARB = 0, XT_WN = 1, XT_uM = 2, XT_nM = 3, XT_Seconds = 4,
XT_Minutes = 5, XT_MassCharge = 9, XT_RAMSHFT = 13,
XT_Points = 22, XT_Hours = 30, XT_AMU = 50, XT_Custom = 51 };
DATATYPE { YT_ARB = 0, YT_IGRAM = 1, YT_Abs = 2, YT_Percent = 11,
YT_Intensity = 12, YT_RelAbundance = 13, YT_Trans = 128,
YT_Refl = 129, YT_Custom = 51, YT_Abundance = 52, };
REJECTREASON { RR_GOOD = 0, RR_20PCT = 0x00010004,
RR_CENTERBURST = 0x00010005, RR_HW_UNSTABLE = 0x00010010 };

```

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 8 of 97	Revision: 1.22

Data structures (structures of information, usually passed by reference, as follows)

```

struct _instrumentMLDiag
{
    public int nVersion;           // struct version (100 to 102)
    public int nEnergyStatus;      // Height of Center burst
    public int nLaserStatus;
    public int numTemps;
    public int nBatteryMinutes;
    public int nBatteryPct;
    public int nBatteryState;      // bits: 1=connected, 2=ac connected,
                                   // 4=charging, 16=fully charged

    public float fSourceCurrentStatus;
    public float fSourceVoltageStatus;
    public float fSpare;
    public float fTempCPU;         // Cpu board temperature
    public float fTempPower;       // Power board temperature
    public float fTempIR;          // IR board temperature
    public float fTempDetector;    // Detector temperature

    // MLDiag 102+
    public Int32 nSystemStatus;     // System Status
    public Int32 nShutdownReason;  // System Shutdown Reason
};

struct _instrumentMLVersion
{
    public int nVersion;           // struct version (100 to 103)
    public int fwRev;              // firmware rev
    public int dllRev;             // dll rev
    public int nReserved0;         // reserved; return value is undefined
    public int instrType;          // ML_INSTRUMENT_TYPE enum value
    public int sampleTechType;     // negative ==> ATR
    public int atrType;            // 1, 3, 9 (for ATR type sampleTechs)
    public int spare;              // always returned as 0
    public double dLaserWN;        // in WN
    public double dBasePathLength; // Transmission/gascell sampleTechs in mm
    public double dAdjPathLength;  // Transmission/gascell sampleTechs in mm

    // MLVersion 101+
    // Serial number (in WCHAR-compatible format)
    public short serialNo01;
    public short serialNo02;
    public short serialNo03;
    public short serialNo04;
    public short serialNo05;
    public short serialNo06;
    public short serialNo07;
    public short serialNo08;
    public short serialNo09;
    public short serialNo10;

```


Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 9 of 97	Revision: 1.22

```

public short serialNo11;
public short serialNo12;
public short serialNo13;
public short serialNo14;
public short serialNo15;
public short serialNo16;
public short serialNo17;

// MLVersion 102+
public int nCpuBrdRev;
public int nPwrBrdRev;
public int nIrBrdRev;
public int nLasBrdRev;

// MLVersion 103+
public int nUpdFwRev;
public int nBootloaderFwRev;
public int nFpgaRev;
};

struct _progress
{
    public int nStructSize;    // size bytes of struct (initially = 28)
    public FTIR_STATE state;
    public int currentUnits;
    public int totalUnits;
    public int recentRejected;
    public int rejectReason;    // reason, or Good if the last scan good
    public int numRejectsSame; // num consec rejects w same rejectReason
};

struct MLSyncdVals
{
    public int nVersion;        // struct version; currently 101
    public int bBlockTempValid;
    public float fBlockTemp;
    public float fAdjustedLaserFreq;
    public int nAdjustEnabled;
    public int nLaserStandardizationType;
};

```

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 10 of 97	Revision: 1.22

FTIRInst DLL

Summary of Function Interfaces

```

int FTIRInst_SetTargetDeviceUsb(wchar_t *pDeviceName);
int FTIRInst_SetTargetDeviceSerialPort(long nPort);
int FTIRInst_SetTargetDeviceNetwork(wchar_t *pDeviceName);
int FTIRInst_Init();
int FTIRInst_Deinit();
int FTIRInst_SetComputeParams(PHASEPOINTS ppoints, PHASETYPE ptype, APODTYPE
    papod, APODTYPE iapod, ZFFTYPE zff, OFFSETCORRECTTYPE offset);

int FTIRInst_dptraStartSingleBeam(int numScans, ref double from, ref double to, int res, int
    bAutoSetBkg, int bAutoSetClean);
int FTIRInst_dptraStartSpectrum(int numScans, ref double from, ref double to, int res,
    DATAATYPE xtype, DATAATYPE ytype, int bAutoSetUnknown);
int FTIRInst_dptraGetLiveSpectrum(ref double from, ref double to, int res, DATAATYPE
    xtype, DATAATYPE ytype, double[] array, int size, ref double actualFrom, ref
    double actualTo, ref int actualRes);

int FTIRInst_dptraGetSingleBeam(double[] array, int size, ref double actualFrom, ref double
    actualTo, ref int actualRes);
int FTIRInst_dptraGetBackground(double[] array, int size, ref double actualFrom, ref double
    actualTo, ref int actualRes);
int FTIRInst_dptraGetClean(double[] array, int size, ref double actualFrom, ref double
    actualTo, ref int actualRes);
int FTIRInst_dptraGetSpectrum(double[] array, int size, ref double actualFrom, ref double
    actualTo, ref int actualRes);
int FTIRInst_dptraGetRatioSpectrum(double[] bkgarray, double[] smparray, double[] outarray,
    int size, DATAATYPE ytype);

int FTIRInst_KillCollection();

```

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 11 of 97	Revision: 1.22

```

int FTIRInst_SetLaserWaveNumber(ref float newLaser);
int FTIRInst_SetPathlen(ref float newPathlength);
int FTIRInst_GetLaserWaveNumber(ref float curLaser);
int FTIRInst_GetPathlenEx(ref _instrumentMLVersion, ref float curPathlength);

int FTIRInst_GetVersion(ref int fwRev, ref int dllRev, ref int serialNo);
int FTIRInst_GetVersionEx(ref _instrumentMLVersion _vInfo);
int FTIRInst_GetStatus(ref int nEnergyStatus, ref float fBatteryStatus, ref float
    fSourceCurrentStatus, ref float fSourceVoltageStatus, ref int nLaserStatus, ref float
    fDetectorStatus);
int FTIRInst_GetStatusEx(ref _instrumentMLDiag _dStatus);

FTIR_STATE FTIRInst_CheckProgress(ref int currentUnits, ref int totalUnits);
FTIR_STATE FTIRInst_CheckProgressEx(ref int currentUnits, ref int totalUnits, ref int
    rejectedScans);
int FTIRInst_CheckProgressStruct(ref _progress pProgress);

int FTIRInst_dpPtrGetLiveIGram(int res, double[] array, int size, ref int pActualFrom, ref int
    pActualTo, ref int pActualRes);
int FTIRInst_StartCoaddedIGram(int numScans, int nRes, int nPhasePts);
int FTIRInst_dpPtrGetCoaddedIGram(double[] pArray, int nArraySize);
int FTIRInst_dpPtrSetBackground(double[] pArray, int nSize, double from, double to, int nRes)
int FTIRInst_dpPtrGetLiveSingleBeam(int res, double[] pArray, long size, ref double
    actualFrom, ref double actualTo, ref int actualRes);
int FTIRInst_GetIrGain (ref int nVal);
int FTIRInst_SetIrGain (int res, uint flags);
int FTIRInst_RegisterStatus (IntPtr whandle int wm_MessageID);
int FTIRInst_SetAppLedState (int nLedState);

```

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 12 of 97	Revision: 1.22

Typical Usage Patterns

1. Call one of the **SetTargetDeviceXxx()** functions to choose the interface to connect over, and to provide any additional identifying information that will allow a connection to a device. If none of these functions are called, then the DLL interface will default to connecting to the first USB device that is found.
2. Call **FTIRInst_Init** and check return value to make sure instrument connects properly.
3. <optional> check version numbers
4. Call **FTIRInst_SetComputeParams** with required settings
5. Call **FTIRInst_dptraStartSingleBeam** to start the data collection sequence.
6. Monitor instrument status with **FTIRInst_CheckProgressStruct**. When instrument state changes to FTIR_DataReady a single beam is ready.
7. Call **FTIRInst_dptraGetSingleBeam** to determine the size of the memory array that will be needed for the returned single beam. This is done by setting the 'array' parameter to zero (and all other parameters to valid values).
8. Allocate a memory array of sufficient size to receive the single beam result and call **FTIRInst_dptraGetSingleBeam** again but this time with the 'array' parameter set to point to the memory array.
9. If you want to get another single beam go back to (4)
10. If you want to collect a spectrum, collect a single beam with the 'setAsBackground' flag on, then call the same sequence (4,5,6,7) but substitute **FTIRInst_dptraStartSpectrum** and **FTIRInst_dptraGetSpectrum** for the single beam calls.
11. You can collect and monitor spectra by calling **FTIRInst_dptraGetLiveSpectrum**. This automatically (and temporarily) switches to numCoadds == 1 and returns when the next spectrum is available. This would typically be used to display a live spectrum for the user or to monitor for sample contact, etc. Each time **FTIRInst_dptraGetLiveSpectrum** is called a new spectrum is returned. If you call it before the next is ready, the function will not return until a fresh spectrum is available. Terminate collection of the live spectra by calling **FTIRInst_KillCollection**.
12. To change collection parameters, go back to (3)
13. Prior to exiting the application be sure to call **FTIRInst_Deinit** for an efficient shutdown.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 13 of 97	Revision: 1.22

FTIRInst_SetTargetDeviceUsb

The FTIRInst_SetTargetDeviceUsb function should be called prior to calling FTIRInst_Init if it is desired to connect to an FTIR device over the USB interface.

C# Declaration

```
int FTIRInst_SetTargetDeviceUsb(wchar_t *pDeviceName);
```

C++ Declaration

```
long FTIRInst_SetTargetDeviceUsb(wchar_t *pDeviceName);
```

Parameters

pDeviceName

[in] A placeholder for a pointer to a wide-character string that gives an identifying name to the FTIR device to connect to. **This is not currently used, and should be set to 0 (null).**

Return Values

This function returns 0 if successful, otherwise an error code is returned.

-1 == General Error

Remarks

USB is the default connection method. Currently, the first USB device that is found is the one that a connection is made to; multiple USB FTIR devices are not currently supported. In the future, this function may allow a caller to choose among multiple USB FTIR devices. For the time being, the pDeviceName pointer is ignored, and it is recommended that a 0 (null) value be passed in.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 14 of 97	Revision: 1.22

FTIRInst_SetTargetDeviceSerialPort

The FTIRInst_SetTargetDeviceSerialPort function should be called prior to calling FTIRInst_Init if it is desired to connect to an FTIR device over a serial port (or virtual serial port) interface; Bluetooth connections are made using a virtual serial port.

C# Declaration

```
int FTIRInst_SetTargetDeviceSerialPort(int nPort);
```

C++ Declaration

```
long FTIRInst_SetTargetDeviceSerialPort(long nPort);
```

Parameters

nPort

[in] Serial port number to connect over; may be virtualized.

Return Values

This function returns 0 if successful, otherwise an error code is returned.

-1 == General Error

Remarks

Some FTIR devices are configured with a Bluetooth interface, and can be connected to using a virtual serial port (aka virtual COM port). The pairing between the host device (e.g. PC) and FTIR device must be done outside of the purview of the FTIR DLL, and a virtual serial port assigned. Subsequently the device can be connected to by calling this function with the serial port number that was assigned.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 15 of 97	Revision: 1.22

FTIRInst_SetTargetDeviceNetwork

The FTIRInst_SetTargetDeviceNetwork function should be called prior to calling FTIRInst_Init if it is desired to connect to an FTIR device over the network interface, either wired Ethernet or wireless.

C# Declaration

```
int FTIRInst_SetTargetDeviceNetwork(wchar_t *pDeviceName);
```

C++ Declaration

```
long FTIRInst_SetTargetDeviceNetwork(wchar_t *pDeviceName);
```

Parameters

pDeviceName

[in] A pointer to a wide-character string that gives the network name of the FTIR device to connect to. This may be an IP address in dotted-notation, or a hostname if such a name can be resolved on the local network.

Return Values

This function returns 0 if successful, otherwise an error code is returned.

-1 == General Error

Remarks

It is recommended that the device name (hostname) string is a string representation of the IP address of the target FTIR device; for example, "192.168.1.2". Care must be taken that the target FTIR network device is reachable by the client machine—the system routing tables and firewall must be configured to provide a path to the device; pinging the device can assist in ensuring that it is reachable.

It is possible to pass a network device name (hostname) as the input string, but the name must be resolvable by the client machine. Early development should avoid using hostnames and use IP addresses.

This function is called to connect over wired Ethernet or wireless—the local network configuration will dictate whether and how the FTIR device is reached. The FTIR device must have its network configuration set appropriately to appear reachable on the network.

Although the FTIR DLL provides a manner of accessing network devices, it is not able to provide any network support functionality—please see your network administrator for more assistance.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 16 of 97	Revision: 1.22

FTIRInst_Init

The FTIRInst_Init function should be called prior to using any other functions in this component DLL

C# Declaration

```
int FTIRInst_Init();
```

C++ Declaration

```
long FTIRInst_Init();
```

Parameters

None

Return Values

This function returns 0 if successful, otherwise an error code is returned. Any other value indicates that the instrument failed to initialize properly.

-1 == Internal Error

-2 == Cannot connect to the instrument. The instrument is probably off or disconnected.

Remarks

Be sure to check the return value. No other FTIRInst functions will succeed if this fails.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 17 of 97	Revision: 1.22

FTIRInst_Deinit

The FTIRInst_Deinit function should be called at the conclusion of use of the DLL. If this is not called before exiting, the DLL cannot clean up before terminating, resulting in a very slow application shutdown.

C# Declaration

```
int FTIRInst_Deinit();
```

C++ Declaration

```
long FTIRInst_Deinit();
```

Parameters

None

Return Values

This function always returns 0.

Remarks

None

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 18 of 97	Revision: 1.22

FTIRInst_SetComputeParams

The FTIRInst_SetComputeParams function should be called prior to using any of the data collection calls, in order to set the interferogram compute parameters. Default values will be used if no changes are made to the instrument through this call.

C# Declaration

```
int FTIRInst_SetComputeParams(
    PHASEPOINTS ppoints,
    PHASETYPE ptype,
    APODTYPE papod,
    APODTYPE iapod,
    ZFFTYPE zff,
    OFFSETCORRECTTYPE offset
);
```

C++ Declaration

```
long FTIRInst_SetComputeParams(
    PHASEPOINTS ppoints,
    PHASETYPE ptype,
    APODTYPE papod,
    APODTYPE iapod,
    ZFFTYPE zff,
    OFFSETCORRECTTYPE offset
);
```

Parameters

ppoints

[in] The number of phase points to be used for the COMPUTE algorithm

ptype

[in] Phase correction type. (Currently only Mertz phase correction is supported).

papod

[in] The phase apodization type.

iapod

[in] The interferogram apodization type.

zfftype

[in] The type of zero fill factor to be used in the COMPUTE.

offset

[in] The type of offset correction to be used in the COMPUTE.

Return Values

If successful, this function returns a 1 value if successful. A value of 0 is returned for failure.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 19 of 97	Revision: 1.22

Remarks

Default values are : 512 Phasepoints, Mertz Correction, Triangular phase apodization, HappGenzel interferogram apodization, NO zero fill factor, NO offset correction.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 20 of 97	Revision: 1.22

FTIRInst_dptraStartSingleBeam

The FTIRInst_dptraStartSingleBeam function is called to start the single beam collection of data.

C# Declaration

```
int FTIRInst_dptraStartSingleBeam (
    int numScans,
    ref double from,
    ref double to,
    int res,
    int bAutoSetBkg,
    int bAutoSetClean
);
```

C++ Declaration

```
long FTIRInst_dptraStartSingleBeam (
    long numScans,
    double * from,
    double * to,
    long res,
    long bAutoSetBkg,
    long bAutoSetClean
);
```

Parameters

numScans

[in] The number of scans to be completed.

from

[in] The starting wave number in the spectral range.

to

[in] The ending wave number in the spectral range.

res

[in]. The resolution.

bAutoSetBkg

[in] When this is non-zero, the next single beam collected will be kept as the new 'background' reference single beam. This will be used in the calculation of a spectrum

bAutoSetClean

[in] When this is non-zero, the next single beam collected will be kept as the new 'clean' reference single beam.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 21 of 97	Revision: 1.22

Return Values

Negative return values are error codes.

-1 == the instrument is not connected (or **FTIRInst_Init** has not been called).

-2 == specified resolution value is not valid

-3 == instrument is not in a valid state to start data collection. State must not be **FTIR_Collecting** or **FTIR_Aborting**

If successful, this function returns a positive value. This value corresponds to the number of data points that will be returned later when calling **FTIRInst_dptrGetSingleBeam**. The array size can also be obtained from **FTIRInst_dptrGetSingleBeam**.

Remarks

A single beam in a non-background corrected spectrum. You must acquire a single beam and tag it as the background before you can use **FTIRInst_dptrStartSpectrum**. The background single beam is used in the calculation of the spectrum.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 22 of 97	Revision: 1.22

FTIRInst_dptraStartSpectrum

The FTIRInst_dptraStartSpectrum function is called to start the collection of a spectrum. There must be a valid tagged single beam before a spectrum can be calculated.

C# Declaration

```
int FTIRInst_dptraStartSpectrum(
    int numScans,
    ref double from,
    ref double to,
    int res,
    DATAATYPE xtype,
    DATAATYPE ytype,
    int bAutoSetUnknown
);
```

C++ Declaration

```
long FTIRInst_dptraStartSpectrum(
    long numScans,
    double * from,
    double * to,
    long res,
    DATAATYPE xtype,
    DATAATYPE ytype,
    int bAutoSetUnknown
);
```

Parameters

numScans
[in] The number of scans to be completed.

from
[in] The starting wave number in the spectral range.

to
[in] The ending wave number in the spectral range.

res
[in]. The resolution.

xtype
[in]. Units of the x-axis of the spectrum.

ytype
[in] Units of the y-axis of the spectrum.

bAutoSetUnknown
[in] (Reserved for future use. Set to zero.)

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 23 of 97	Revision: 1.22

Return Values

If successful, this function returns the number of points in each scan. If no background data is present a -3 is returned. If background data is available but is incompatible a -4 is returned. -1 is returned if the scan cannot be started for any other reason.

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 24 of 97	Revision: 1.22

FTIRInst_dpnrGetLiveSpectrum

The FTIRInst_dpnrGetLiveSpectrum function is called to get the data from the last good collected spectrum.

C# Declaration

```
int FTIRInst_dpnrGetLiveSpectrum(
    ref double from,
    ref double to,
    int res,
    DATAXTYPE xtype,
    DATAYTYPE ytype,
    double[] array,
    int size,
    ref double actualFrom,
    ref double actualTo,
    ref int actualRes
);
```

C++ Declaration

```
long FTIRInst_dpnrGetLiveSpectrum(
    double* from,
    double* to,
    long res,
    DATAXTYPE xtype,
    DATAYTYPE ytype,
    double* array,
    long size,
    double* actualFrom,
    double* actualTo,
    long* actualRes
);
```

Parameters

from

[in] The starting wave number in the spectral range.

to

[in] The ending wave number in the spectral range.

res

[in] The resolution.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 25 of 97	Revision: 1.22

xtype
[in] The unit type of the x-axis of the interferogram

ytype
[in] The unit type of the y-axis of the interferogram

array
[out] The array of doubles containing the spectral data.

size
[out] the length of the array.

actualFrom
[out] The actual starting wave number from the spectral range.

actualTo
[out] The actual ending wave number from the spectral range.

actualRes
[out] The actual resolution.

Return Values

If successful, this function returns the length of the data array. If no background data is present a -3 is returned. If background data is available but is incompatible a -4 is returned. -1 is returned if the scan cannot be started for any other reason.

Remarks

This initiates the collection of a sequence of live spectra for monitoring. A background single beam must be available. The first time this is called, the system automatically (and temporarily) switched numScans to 1, and then waits for the completion of one scan, computes a spectrum and returns. Subsequent calls wait for the next spectrum, then return. If scan has been completed before the second call this function will return that spectrum immediately, but it will not return the same spectrum on consecutive calls.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 26 of 97	Revision: 1.22

FTIRInst_dpPtrGetSingleBeam

The FTIRInst_dpPtrGetSingleBeam function is called to get a completed single beam.

C# Declaration

```
int FTIRInst_dpPtrGetSingleBeam(
    double[] array,
    int size,
    ref double actualFrom,
    ref double actualTo,
    ref int actualRes);
```

C++ Declaration

```
long FTIRInst_dpPtrGetSingleBeam(
    double* array,
    long size,
    double* actualFrom,
    double* actualTo,
    long* actualRes);
```

Parameters

array

[out] The array of data.

size

[out] the length of the array.

actualFrom

[out] The actual starting wave number from the spectral range.

actualTo

[out] The actual ending wave number from the spectral range.

actualRes

[out]. The actual resolution.

Return Values

If successful, this function returns the length of the data array. On error a zero or negative value is returned.

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 27 of 97	Revision: 1.22

After initiating the collection of a single beam (using **FTIRInst_dptraStartSingleBeam**) and monitoring for FTIR_DataReady (using **FTIRInst_CheckProgressStruct**) the completed single beam may be retrieved using this call.

If you call **FTIRInst_dptraGetSingleBeam** with the array parameter set to zero and all other parameters set properly, the return value will be the length (in number of data points) of the output array. You can then allocate an appropriately sized array to receive the result data.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 28 of 97	Revision: 1.22

FTIRInst_dpnrGetBackground

The FTIRInst_dpnrGetBackground function retrieves the instrument's stored background spectrum.

C# Declaration

```
int FTIRInst_dpnrGetBackground(
    double[] array,
    int size,
    ref double actualFrom,
    ref double actualTo,
    ref int actualRes);
```

C++ Declaration

```
long FTIRInst_dpnrGetBackground(
    double* array,
    long size,
    double* actualFrom,
    double* actualTo,
    long* actualRes);
```

Parameters

array

[out] The array of data.

size

[out] the length of the array.

actualFrom

[out] The actual starting wave number from the spectral range.

actualTo

[out] The actual ending wave number from the spectral range.

actualRes

[out]. The actual resolution.

Return Values

If successful, this function returns the length of the data array. On error a zero or negative value is returned.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 29 of 97	Revision: 1.22

Remarks

If you call **FTIRInst_dptraGetBackground** with the array parameter set to zero and all other parameters set properly, the return value will be the length (in number of data points) of the output array. You can then allocate an appropriately sized array to receive the result data.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 30 of 97	Revision: 1.22

FTIRInst_dpnrGetClean

The FTIRInst_dpnrGetClean function retrieves the instrument's stored Clean spectrum.

C# Declaration

```
int FTIRInst_dpnrGetClean(
    double[] array,
    int size,
    ref double actualFrom,
    ref double actualTo,
    ref int actualRes);
```

C++ Declaration

```
long FTIRInst_dpnrGetClean(
    double* array,
    long size,
    double* actualFrom,
    double* actualTo,
    long* actualRes);
```

Parameters

array

[out] The array of data.

size

[out] the length of the array.

actualFrom

[out] The actual starting wave number from the spectral range.

actualTo

[out] The actual ending wave number from the spectral range.

actualRes

[out]. The actual resolution.

Return Values

If successful, this function returns the length of the data array. On error a zero or negative value is returned.

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 31 of 97	Revision: 1.22

If you call **FTIRInst_dpnrGetClean** with the array parameter set to zero and all other parameters set properly, the return value will be the length (in number of data points) of the output array. You can then allocate an appropriately sized array to receive the result data.

FTIRInst_dpnrGetSpectrum

The FTIRInst_dpnrGetSpectrum function retrieves the instrument's stored spectrum.

C# Declaration

```
int FTIRInst_dpnrGetSpectrum(
    double[] array,
    int size,
    ref double actualFrom,
    ref double actualTo,
    ref int actualRes);
```

C++ Declaration

```
long FTIRInst_dpnrGetSpectrum(
    double* array,
    long size,
    double* actualFrom,
    double* actualTo,
    long* actualRes);
```

Parameters

array

[out] The array of data.

size

[out] the length of the array.

actualFrom

[out] The actual starting wave number from the spectral range.

actualTo

[out] The actual ending wave number from the spectral range.

actualRes

[out]. The actual resolution.

Return Values

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 32 of 97	Revision: 1.22

If successful, this function returns the length of the data array. On error a zero or negative value is returned. If the array that is passed in is too small, then a value of -9 is returned.

Remarks

Null can be passed in for the array parameter and the length of the data will be returned.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 33 of 97	Revision: 1.22

FTIRInst_dpPtrGetRatioSpectrum

The FTIRInst_dpPtrGetRatioSpectrum function returns the ratio of the background spectrum with the sample spectrum.

C# Declaration

```
int FTIRInst_dpPtrGetRatioSpectrum(
    double[] bkgarray,
    double[] smparray,
    double[] outarray,
    int size,
    DATATYPE ytype);
```

C++ Declaration

```
long FTIRInst_dpPtrGetRatioSpectrum(
    double* bkgarray,
    double* smparray,
    double* outarray,
    long size,
    DATATYPE ytype);
```

Parameters

bkgarray

[in] The array of doubles containing the background spectrum data.

smparray

[in] The array of doubles containing the sample spectrum data.

outarray

[out] The array of doubles containing the return spectrum.

size

[out] The length of the outarray.

ytype

[in] The unit type of the y-axis of the spectrum.

Return Values

The function always returns size, as passed in, if successful. A value of 0 is returned for failure.

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 34 of 97	Revision: 1.22

FTIRInst_KillCollection

The FTIRInst_KillCollection function stops the collection of data within the instrument.

C# Declaration

```
int FTIRInst_KillCollection();
```

C++ Declaration

```
long FTIRInst_KillCollection();
```

Parameters

None.

Return Values

This function returns 0 if successful, otherwise an error code is returned. *Currently no errors are defined.*

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 35 of 97	Revision: 1.22

FTIRInst_SetLaserWaveNumber

The FTIRInst_SetLaserWaveNumber sets the value of the Laser Wave number that is stored in the instrument's EEPROM and used in calculation of spectra.

C# Declaration

```
int FTIRInst_SetLaserWaveNumber(
    ref float newLaser
);
```

C++ Declaration

```
long FTIRInst_SetLaserWaveNumber(
    float* newLaser
);
```

Parameters

newLaser
[in] The value of the wave number.

Return Values

This function returns 0 if successful, otherwise an error code is returned. *Currently no errors are defined.*

Remarks

The actual wave number of the instrument laser is used in the calculation of spectra. This value after determination by a calibration procedure is stored in EEPROM in the instrument by calling this function. This function does not implement the calibration procedure; it simply supplies the value to the instrument for use and storage.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 36 of 97	Revision: 1.22

FTIRInst_SetPathLen

The FTIRInst_SetPathLen sets the value of the PathLength number that is stored within the instrument's EEPROM.

C# Declaration

```
int FTIRInst_SetPathlen(
    ref float newPathlength
);
```

C++ Declaration

```
long FTIRInst_SetPathlen(
    float* newPathlength
);
```

Parameters

newPathlength
[in] The value of the new pathlength.

Return Values

This function returns 0 if successful, otherwise an error code is returned. *Currently no errors are defined.*

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 37 of 97	Revision: 1.22

FTIRInst_GetLaserWaveNumber

The FTIRInst_GetLaserWaveNumber gets the current value of the laser wave number that is stored the instrument's EEPROM.

C# Declaration

```
int FTIRInst_GetLaserWaveNumber(  
    ref float curLaser  
);
```

C++ Declaration

```
long FTIRInst_GetLaserWaveNumber(  
    float* curLaser  
);
```

Parameters

curLaser
[out] The value of the laser wave number.

Return Values

This function returns 0 if successful, otherwise an error code is returned. *Currently no errors are defined.*

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 38 of 97	Revision: 1.22

FTIRInst_GetPathlenEx

The FTIRInst_GetPathlenEx gets the current value of the pathlength that is stored within the instrument's version information.

C# Declaration

```
int FTIRInst_GetPathlenEx(
    ref _instrumentMLVersion mlvers,
    ref double pathlen
);
```

C++ Declaration

```
long FTIRInst_GetPathlenEx(
    _instrumentMLVersion *mlvers,
    double *pPathlen
);
```

Parameters

mlvers

[in] The instrument version information.

pathlen

[out] The value of the current pathlength.

Return Values

This function returns 0 if successful, otherwise an error code is returned. *Currently no errors are defined.*

Remarks



Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 39 of 97	Revision: 1.22

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 40 of 97	Revision: 1.22

FTIRInst_GetVersion

The FTIRInst_GetVersion gets the current value of the firmware version, the dll version, and the serial number of the instrument.

C# Declaration

```
int FTIRInst_GetVersion(
    ref int fwRev,
    ref int dllRev,
    ref int serialNo);
```

C++ Declaration

```
long FTIRInst_GetVersion(
    long* fwRev,
    long* dllRev,
    long* serialNo);
```

Parameters

fwRev
[out] The current version of the instrument firmware.

dllRev
[out] The current version number of the instrument interface DLL

serialNo
[out] The serial number of the instrument.

Return Values

This function returns 1 if successful, otherwise an error code is returned. *Currently no errors are defined.*

Remarks

This method has been made obsolete by the FTIRInst_GetVersionEx function.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 41 of 97	Revision: 1.22

FTIRInst_GetVersionEx

The FTIRInst_GetVersionEx gets the current version information from the instrument.

C# Declaration

```
int FTIRInst_GetVersionEx(
    ref _instrumentMLVersion _vInfo
);
```

C++ Declaration

```
long FTIRInst_GetVersionEx(
    instrumentMLVersion* _vInfo
);
```

Parameters

vinfo

[out] Current version information from the instrument. The _instrumentMLVersion struct contains the same information as provided by the FTIRInst_GetVersion function and also contains info about the instrument type.

Return Values

This function returns 0 if successful, otherwise an error code is returned. A value of -1 is returned if the instrument is not connected, and a value of -2 is returned if the _vInfo pointer is null.

Remarks

The structure that is pointed to by _vInfo must be allocated by the caller, and the nVersion field must be filled in with the appropriate version of the structure. The version must match the size of the structure, since the function will fill in as many fields as the nVersion value dictates.

Note: There is no longer an integer serial number field in the structure—the serial number is always stored in the wide-character serialNoXX array, allowing any alphanumeric character to appear in a serial number.

To access the serialNoXX array, the caller may need to translate from wide characters (Unicode) to 'multi-byte' (generally 8-bit ASCII); this can be done using the WideCharToMultiByte() Windows function. For those callers already using Unicode, they can reference the string directly, starting at the first character in the structure; it may be easier to reference the serialNoXX array as wchar_t serialNo[17] instead of individual characters, depending on the development environment.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 42 of 97	Revision: 1.22

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 43 of 97	Revision: 1.22

FTIRInst_GetStatus

The FTIRInst_GetStatus gets the current status information from the instrument.

C# Declaration

```
int FTIRInst_GetStatus(
    ref int nEnergyStatus,
    ref float fBatteryStatus,
    ref float fSourceCurrentStatus,
    ref float fSourceVoltageStatus,
    ref int nLaserStatus,
    ref float fDetectorStatus);
```

C++ Declaration

```
long FTIRInst_GetStatus(
    int* nEnergyStatus,
    float* fBatteryStatus,
    float* fSourceCurrentStatus,
    float* fSourceVoltageStatus,
    int* nLaserStatus,
    float* fDetectorStatus);
```

Parameters

nEnergyStatus
[out] The current energy status from the instrument.

fBatteryStatus
[out] The current battery status.

fSourceCurrentStatus
[out] The source current.

fSourceVoltageStatus
[out] The source voltage.

nLaserStatus
[out] The current status of the laser.

nDetectorStatus
[out] The current status of the detector.

Return Values

This function returns 1 if successful, otherwise a 0 is returned for failure.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 44 of 97	Revision: 1.22

Remarks

This function has been made obsolete by the FTIRInst_GetStatusEx function.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 45 of 97	Revision: 1.22

FTIRInst_GetStatusEx

The FTIRInst_GetStatusEx gets the extended status information from the instrument.

C# Declaration

```
int FTIRInst_GetStatusEx(
    ref _instrumentMLDiag _dStatus
);
```

C++ Declaration

```
long FTIRInst_GetStatusEx(
    instrumentMLDiag* _dStatus
);
```

Parameters

_dStatus
[out] An _instrumentMLDiag struct containing instrument status.

Return Values

This function returns 1 if successful, otherwise an error code is returned. A -1 is returned if the instrument is not connected, and a -2 is returned if the _dStatus pointer is null.

Remarks

The structure that is pointed to by _dStatus must be allocated by the caller, and the nVersion field must be filled in with the appropriate version of the structure. The version must match the size of the structure, since the function will fill in as many fields as the nVersion value dictates.

System Status Values:

SYSSTAT_UNCONNECTED	0x00000001 // std startup state
SYSSTAT_CONNECTED	0x00000002
SYSSTAT_CONNECTION_LOST	0x00000100 //was connected, but lost
connection	
SYSSTAT_SHUTTINGDOWN	0x00001000
SYSSTAT_SHUTDOWN	0x00002000 // implies now unconnected



Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 46 of 97	Revision: 1.22

System Shutdown Values:

SHDOWN_NOTVALID	0x00000000
SHDOWN_USERBUTTON	0x00000001
SHDOWN_BATTERYCRITICAL	0x00000010

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 47 of 97	Revision: 1.22

FTIRInst_CheckProgress

The FTIRInst_CheckProgress function returns the current state of the instrument.

C# Declaration

```
FTIR_STATE FTIRInst_CheckProgress(
    ref int currentUnits,
    ref int totalUnits
);
```

C++ Declaration

```
FTIR_STATE FTIRInst_CheckProgress(
    long* currentUnits,
    long* totalUnits
);
```

Parameters

currentUnits

[out] The current number of successful scans completed since the FTIRInst_dpPtrStartSpectrum was called.

totalUnits

[out] The total number of scans to be completed before the spectrum can be computed.

Return Values

The function returns one of the following values:

```
FTIR_Init = 0,
FTIR_Collecting = 1,
FTIR_DataReady = 2,
FTIR_Aborting = 3,
FTIR_Error = 4
```

Remarks

This function has been made obsolete by the CheckProgressStruct function.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 48 of 97	Revision: 1.22

FTIRInst_CheckProgressEx

The FTIRInst_CheckProgressEx function returns the current state information from the instrument.

C# Declaration

```
FTIR_STATE FTIRInst_CheckProgressEx(
    ref int currentUnits,
    ref int totalUnits,
    ref int rejectedScans);
```

C++ Declaration

```
FTIR_STATE FTIRInst_CheckProgressEx(
    long* currentUnits,
    long* totalUnits,
    long* rejectedScans);
```

Parameters

currentUnits

[out] The current number of successful scans completed since the FTIRInst_dpTrStartSpectrum was called.

totalUnits

[out] The total number of scans to be completed before the spectrum can be computed.

rejectedScans

[out] The total number of rejected scans out of the last 10 scans to be completed.

Return Values

The function returns one of the following values:

```
FTIR_Init = 0,
FTIR_Collecting = 1,
FTIR_DataReady = 2,
FTIR_Aborting = 3,
FTIR_Error = 4
```

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 49 of 97	Revision: 1.22

This function has been made obsolete by the CheckProgressStruct function.

FTIRInst_CheckProgressStruct

The FTIRInst_CheckProgressStruct function returns the current state information from the instrument in the form of a `_progress` Struct.

C# Declaration

```
int FTIRInst_CheckProgressStruct(
    ref _progress pProgress
);
```

C++ Declaration

```
long FTIRInst_CheckProgressStruct(
    progress* pProgress
);
```

Parameters

pProgress

[out] The `_progress` struct containing information about the progress of the current run.

Return Values

This function returns the size of the `_progress` structure that is returned; this value is returned both when a valid pointer is passed in, as well as if a null pointer is passed in.

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 50 of 97	Revision: 1.22

FTIRInst_dpPtrGetLiveIGram

The start call takes the number of scans to coadd, the resolution (in wavenumber: 2, 4, 8, 16), and the number of phase pts to the left of the centerburst; valid values for nPhasePts are the standard power-of-two values (128, 256, 512, 1024).

C# Declaration

```
int FTIRInst_dpPtrGetLiveIGram(
    int res,
    double[] pDataArray,
    int size,
    ref int pActualFrom,
    ref int pActualTo,
    ref int pActualRes
);
```

C++ Declaration

```
long FTIRInst_dpPtrGetLiveIGram (long res, double* pDataArray, long size,
    long* pActualFrom, long* pActualTo, long* pActualRes);
```

Parameters

res

[in] The desired resolution for the Interferogram data (In wavenumber: 2, 4, 8, 16)

pDataArray

[out] The output data points for the interferogram

size

[in] The maximum size of the output data array

pActualFrom

[out] The index in the output array of the first data point (usually 0)

pActualTo

[out] The index in the output array of the last data point

pActualRes

[out] The output resolution of the interferogram

Return Values

The return value will be the number of points filled into the output array for success, and a negative number for failure. NULL can be passed in for pDataArray, which will result in the size of the expected array to be returned.

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 51 of 97	Revision: 1.22

FTIRInst_StartCoaddedIGram

The start call takes the number of scans to coadd, the resolution (in wavenumber: 2, 4, 8, 16), and the number of phase pts to the left of the centerburst; valid values for nPhasePts are the standard power-of-two values (128, 256, 512, 1024).

C# Declaration

```
int FTIRInst_StartCoaddedIGram(
    int numScans,
    int nRes,
    int nPhasePts
);
```

C++ Declaration

```
long FTIRInst_StartCoaddedIGram(long numScans, long nRes, nPhasePts);
```

Parameters

numScans

[in] The number of scans to run before returning the completed coadded IGram.

nRes

[in] The resolution (In wavenumber: 2, 4, 8, 16)

nPhasePts

[in] The number of phase pts to the left of the centerburst.

Return Values

The return value will be 0 for success, and a negative number for failure.

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 52 of 97	Revision: 1.22

FTIRInst_StartCoaddedIGramNotify

The start call takes the number of scans to coadd, the resolution (in wavenumber: 2, 4, 8, 16), and the number of phase pts to the left of the centerburst; valid values for nPhasePts are the standard power-of-two values (128, 256, 512, 1024). The client is 'notified' of completion by signal of the passed in Operation System event handle.

C# Declaration

```
int FTIRInst_StartCoaddedIGramNotify(
    int numScans,
    int nRes,
    int nPhasePts
    IntPtr eventHandle);
```

C++ Declaration

```
long FTIRInst_StartCoaddedIGramNotify(long numScans, long nRes, nPhasePts,
    HANDLE hReadyEvent);
```

Parameters

numScans
[in] The number of scans to run before returning the completed coadded IGram.

nRes
[in] The resolution (In wavenumber: 2, 4, 8, 16)

nPhasePts
[in] The number of phase pts to the left of the centerburst.

hReadyEvent
[in] Handle of a Windows Event to be set by the by the DLL when the next IGram is available.

Return Values

The return value will be 0 for success, and a negative number for failure.

Remarks

If a valid handle to an Operating System event is passed in the DLL will call SetEvent on that handle when the requested igrm is available. The caller is responsible for creating the event, insuring that it is not set when the call is made and destroying the event when no longer needed. The system will

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 53 of 97	Revision: 1.22

only set the event once per call to StartCoaddedIGramNotify. After being notified, the application must call StartCoaddedIGramNotify again to request another igram and another notification.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 54 of 97	Revision: 1.22

FTIRInst_dpPtrGetCoaddedIGram

This function fills in the array pointed to by pArray with Igram data. Calling this function with (pArray==0) will return the number of entries in the current array. If (pArray!=0), then the nArraySize value MUST be set to the size of the pArray buffer that is being passed in, in elements. This will be used to verify that the array is large enough.

C# Declaration

```
int FTIRInst_dpPtrGetCoaddedIGram(
    double[] pArray,
    int nArraySize
);
```

C++ Declaration

```
long FTIRInst_dpPtrGetCoaddedIGram (
    double *pArray,
    long nArraySize
);
```

Parameters

pArray

[out] The array that is to be populated the the coadded IGram data.

nArraySize

[in] The length of the pArray array that is being passed.

Return Values

The return value is the number of elements in the array, which may be less (and should be) than the nArraySize value that is passed in. A value of -9 is returned if the size of the passed-in array is too small, per the nArraySize argument.

Note: It is not guaranteed that the returned array size will always be the same from coadd to coadd. If there is a shift in the position of the centerburst, it may be possible to get more or less points. See remarks below.

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 55 of 97	Revision: 1.22

Note that the number of elements in an Igram array is NOT equal to the number of points dictated by nRes and nPhasePts in the StartCoaddedIgram() call. There are a number of padding points that may be added both to the left and right of the Igram, to ensure that a full set of data is provided. These padding points will always be returned in the GetCoaddedIgram() call, and can be dealt with as desired by the calling application.

The array size checking is required because there is a rare chance that the size might change, due to receiving a new coadded Igram, between a Get call with (pArray==0) and a subsequent call with (pArray!=0). By design the system will return a size that is approximately 100 data points larger than necessary. Then if a small fluctuation happens in the size of the igram, the buffer will have adequate capacity to handle it. The return value from this function will reflect the actual number of data points filled into the array.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 56 of 97	Revision: 1.22

FTIRInst_RegisterButton1

This function is called to register the Windows message that is returned when the instrument's trigger is pulled.

C# Declaration

```
int FTIRInst_RegisterButton1(
    IntPtr whandle
    ref int wm_MessageID
);
```

C++ Declaration

```
long FTIRInst_RegisterButton1 (
    HWND whandle,
    long* wm_MessageID
);
```

Parameters

whandle

[in] The handle of the form that is going to handle the message.

wm_MessageID

[out] The ID of the Windows message that is posted when the trigger on the instrument is pulled.

Return Values

This method always returns a 0.

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 57 of 97	Revision: 1.22

FTIRInst_RegisterButton2

This function is called to register the Windows message that is returned when the instrument's navigation buttons are pushed.

C# Declaration

```
int FTIRInst_ResigerButton1(
    IntPtr whandle
    ref int wm_MessageID
);
```

C++ Declaration

```
long FTIRInst_ResigerButton1 (
    HWND whandle,
    long* wm_MessageID
);
```

Parameters

whandle

[in] The handle of the form that is going to handle the message.

wm_MessageID

[out] The ID of the Windows message that is posted when either of the side buttons on the Exoscan are pushed.

Return Values

This method always returns a 0.

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 58 of 97	Revision: 1.22

FTIRInst_dpPtrSetBackground

The FTIRInst_dpPtrSetBackground sends background data to the instrument. This background data is then used when the instrument creates spectrums when either calling FTIRInst_dpPtrGetLiveSpectrum or FTIRInst_dpPtrStartSpectrum.

C# Declaration

```
int FTIRInst_SetBackground (
    double[] pArray,
    int nSize,
    double from,
    double to,
    int nRes);
```

C++ Declaration

```
long FTIRInst_SetBackground (
    double* pArray,
    long nSize,
    double* from,
    double* to,
    long nRes);
```

Parameters

pArray [in] The array of data containing the background information.

nSize [in] The size of the data being passed in.

from [in] The starting x Value.

to [in] The ending x Value.

nRes [in] The resolution.



Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 59 of 97	Revision: 1.22

Return Values

This function returns 0 if successful, otherwise an error code is returned.

- -2: Could not access target object buffer.
- -1: Memory allocation error.

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 60 of 97	Revision: 1.22

FTIRInst_dpPtrGetLiveSingleBeam

The FTIRInst_dpPtrGetLiveSingleBeam function is called to get the data from the last good collected single beam.

C# Declaration

```
int FTIRInst_dpPtrGetLiveSingleBeam (
    int res,
    double[] pArray,
    int size,
    ref double actualFrom,
    ref double actualTo,
    ref int actualRes);
```

C++ Declaration

```
long FTIRInst_dpPtrGetLiveSingleBeam (
    long res,
    double* pArray,
    long size,
    double* actualFrom,
    double* actualTo,
    long actualRes);
```

Parameters

res

[in] The resolution of the single beam.

pArray

[out] The array that is to be filled with the single beam data.

size

[in] The size of the array being passed in.

actualFrom

[out] The actual From value.

actualTo

[out] The actual To value.

actualRes

[out] The actual resolution.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 61 of 97	Revision: 1.22

Return Values

This function returns the size of the array if successful. Otherwise an error code is returned.

- -4: Could not acquire the single beam.
- -9: *pArray* is too small to hold data.
- -11: Device is no longer connected.

Remarks

To get the size of the data, a null should be passed in as the *pArray* argument. This will trigger the function to return the size of the array. An array of the correct size should then be allocated and the function should be called a second time while passing the allocated array in as the *pArray* argument.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 62 of 97	Revision: 1.22

FTIRInst_dpPtrGetLiveIGram

The FTIRInst_dpPtrGetLiveIGram function is called to get the data from the last good collected IGram.

C# Declaration

```
int FTIRInst_dpPtrGetLiveIGram (
    int res,
    double[] pArray,
    int size,
    ref int actualFrom,
    ref int actualTo,
    ref int actualRes);
```

C++ Declaration

```
long FTIRInst_dpPtrGetLiveIGram (
    long res,
    double* pArray,
    long size,
    long* actualFrom,
    long* actualTo,
    long actualRes);
```

Parameters

res
[in] The resolution of the IGram.

pArray
[out] The array that is to be filled with the IGram data.

size
[in] The size of the array being passed in.

actualFrom
[out] The actual From value.

actualTo
[out] The actual To value.

actualRes
[out] The actual resolution.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 63 of 97	Revision: 1.22

Return Values

This function returns the size of the array if successful. Otherwise an error code is returned.

- -4: Could not acquire the interferogram.
- -9: Returned if the *pArray* parameter is too small for the data. Could also return this value if the *size* parameter is too small.
- -11: Device is no longer connected.

Remarks

To get the size of the data, a null should be passed in as the *pArray* argument. This will trigger the function to return the size of the array. An array of the correct size should then be allocated and the function should be called a second time while passing the allocated array in as the *pArray* argument.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 64 of 97	Revision: 1.22

FTIRInst_GetIrGain

The FTIRInst_GetIrGain function is called to get the current Ir Gain value from the instrument.

C# Declaration

```
int FTIRInst_GetIrGain(
    ref int nVal );
```

C++ Declaration

```
long FTIRInst_GetIrGain(
    long* nVal );
```

Parameters

nVal

[out] The value of the Ir Gain variable in the Instrument.

Return Values

This function returns a 0 if successful. On any error will return a -1.

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 65 of 97	Revision: 1.22

FTIRInst_SetIrGain

The FTIRInst_SetIrGain function is called to set the current Ir Gain value in the instrument.

C# Declaration

```
int FTIRInst_SetIrGain(
    int nVal,
    uint flags );
```

C++ Declaration

```
long FTIRInst_SetIrGain(
    long nVal,
    unsigned long flags );
```

Parameters

nVal

[in] The value of the Ir Gain variable that the Instrument will use when scanning.

flags

[in] A flag to tell the instrument give the instrument additional commands. A '0' value will do nothing. '1' will set the value to non-volatile memory.

Return Values

This function returns a 0 if successful. On any error will return a -1.

Remarks

Passing in an -1 as the nVal argument will make the instrument use the factory set default value as the gain.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 66 of 97	Revision: 1.22

FTIRInst_RegisterStatus

This function is called to register the Windows message that is returned when the instrument's progress changes.

C# Declaration

```
int FTIRInst_RegisterStatus(
    IntPtr whandle
    int wm_MessageID
);
```

C++ Declaration

```
long FTIRInst_RegisterStatus (
    HWND whandle,
    long wm_MessageID
);
```

Parameters

whandle

[in] The handle of the form that is going to handle the message.

wm_MessageID

[out] The ID of the Windows message that is posted when the instrument progress or state changes.

Return Values

This method always returns a 0.

Remarks

The wParam parameter of the message that is posted will contain the current status of the instrument. The available values are :

SYSSTAT_UNCONNECTED	0x00000001	// std startup state
SYSSTAT_CONNECTED	0x00000002	
SYSSTAT_CONNECTION_LOST	0x00000100	
SYSSTAT_SHUTTINGDOWN	0x00001000	
SYSSTAT_SHUTDOWN	0x00002000	

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 67 of 97	Revision: 1.22

The lParam parameter contains the current progress of the instrument. The upper 16 bits contains the current progress value and the lower 16 bits contains the total progress.

If the instrument status is currently SYSSTAT_CONNECTION_LOST all get live signal calls will return -11. Most other calls will return a -4 error message. In general, all calls to the FTIRInst assembly should cease while the instrument has any status but SYSSTAT_CONNECTED.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 68 of 97	Revision: 1.22

FTIRInst_RegisterStatusEvents

This function is called to register for events to be triggered when the instrument's progress or status changes.

C# Declaration

```
int FTIRInst_RegisterStatusEvents(
    IntPtr hEvent
);
```

C++ Declaration

```
long FTIRInst_RegisterStatusEvents(
    HANDLE hEvent
);
```

Parameters

hEvent

[in] The handle of the event that will be set by the framework when the instrument progress of state has changed.

Return Values

This method always returns a 0.

Remarks

The hEvent parameter gives the handle of an event that will be set by the framework when the instrument progress or state changes. The progress will change as scans are coadded, and the state will change if the instrument is shut down or loses its physical connection.

The client that calls this function should wait for the event to be set, typically using WaitForSingleObject() or one of its relatives. After the client is done processing the event, it is responsible for calling ResetEvent() so that a new event may be sent.

As part of processing this event, the client will need to ascertain what progress and/or state has changed; this information may be garnered by calling FTIRInst_CheckProgressStruct() or FTIRInst_GetStatusEx().

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 69 of 97	Revision: 1.22

FTIRInst_SetAppLedState

This function is called to set the state of the ‘Application’ LED; possible states are Off, Red, and Amber.

C# Declaration

```
int FTIRInst_SetAppLedState(
    int nLedState);
```

C++ Declaration

```
long FTIRInst_SetAppLedState(
    long nLedState);
```

Parameters

nLedState

[in] The state of the application LED.

Possible states, and their integer values, are:

Off	1
Amber	2
Red	3

Return Values

This method returns a 0 if the call was successful, and a -1 if there was an error.

Remarks

The application LED is set to Amber when the FTIR device is first started—this is to signify that the instrument is not yet communicating with a host. After host software has connected to the device, it may control the LED as it deems appropriate; one example would be to set the LED to the Off state to denote that host software is communicating with the device. It may also be desirable to set the LED to the Red state when the application has found that an error has occurred.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 70 of 97	Revision: 1.22

FTIRInst_I2cAdc_GetReadings

This function is called to retrieve the readings from the external I2C analog-to-digital converter (ADC).

C# Declaration

```
int FTIRInst_I2cAdc_GetReadings(
    int *pArray);
```

C++ Declaration

```
long FTIRInst_I2cAdc_GetReadings(
    long *pArray);
```

Parameters

pArray

[out] A pointer to a caller-allocated array of 8 32-bit values; the values will be filled in by this function if successful. Only the least-significant 12 bits of each value are relevant, as the supported ADC provides only 12 bits of data.

Return Values

This method returns a 0 if the call was successful, and a -1 if there was an error.

Remarks

This function is specific to the Burr-Brown ADS7828 ADC, which is an I2C device that is externally attached to the FTIR device I2C bus, and configured with an address of 0x94 (0x95 with the Read bit set). This device provides 8 channels of analog-to-digital conversion.

The FTIR device will occasionally – about once every ten seconds – sample the values in the ADC, and cache them for return through this function; calling this function does not initiate an ADC conversion.

Only the least-significant 12 bits of each 32-bit value are relevant, since the ADS7828 is a 12-bit ADC that provides only 12 bits of resolution. It is the responsibility of the caller to interpret the 12 bits of information, converting the bits into a voltage, and subsequently understanding what that voltage means.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 71 of 97	Revision: 1.22

FTIRInst_I2clo_SetPinDirs

This function is called to set the directions of the PCA9555 I/O lines.

C# Declaration

```
int FTIRInst_I2cIo_SetPinDirs(
    int vals);
```

C++ Declaration

```
long FTIRInst_I2cIo_SetPinDirs(
    long vals);
```

Parameters

vals

[in] Each of the least-significant 16 bits dictates the signal direction for a pin. Possible bit states, and their integer values, are:

Output	0
Input	1

Return Values

This method returns a 0 if the call was successful, and a -1 if there was an error.

Remarks

This function is specific to the Philips PCA9555 16-bit port expander. All 16 bits of the port expander are used, and positioned in the least-significant bits of the 32-bit values used in this API. Port 0 is in the LSB, while Port 1 is in the next most significant byte.

MSB		LSB	
0	0	Port 1	Port 0

By default, the PCA9555 defines all pins as inputs upon power-up or reset. Hardware must deal with such startup conditions appropriately. The PCA9555 has a weak (100 kohm) pull-up resistor on each pin that pull each input pin high when undriven.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 72 of 97	Revision: 1.22

FTIRInst_I2clo_SetPinVals

This function is called to set the drives of the PCA9555 I/O lines.

C# Declaration

```
int FTIRInst_I2cIo_SetPinVals(
    int vals);
```

C++ Declaration

```
long FTIRInst_I2cIo_SetPinVals(
    long vals);
```

Parameters

vals

[in] Each of the 16 least-significant bits dictates the drive for a pin defined as an output. Possible bit states, and their integer values, are:

Drive Low	0
Drive High	1

Return Values

This method returns a 0 if the call was successful, and a -1 if there was an error.

Remarks

A value written to a pin defined as an input is not relevant as long as the direction remains set to input. The caller is responsible for managing any electrical issues related to driving pins and changing their directions.

See FTIRInst_I2cIo_SetPinDirs for additional information.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 73 of 97	Revision: 1.22

FTIRInst_I2clo_GetPinVals

This function is called to get the values read from the PCA9555 I/O lines.

C# Declaration

```
int FTIRInst_I2cIo_GetPinVals(
    long *pVals);
```

C++ Declaration

```
long FTIRInst_I2cIo_GetPinVals(
    long *pVals);
```

Parameters

pVals

[out] A pointer to a caller-allocated 32-bit value that will receive the input values sensed by the PCA9555. Each of the 16 least-significant bits corresponds with a pin, and is set to describe its read state.

Possible states, and their integer values, are:

Read Low	0
Read High	1

Return Values

This method returns a 0 if the call was successful, and a -1 if there was an error.

Remarks

The FTIR device will occasionally – about once every ten seconds – sample the values in the PCA9555, and cache them for return through this function. Calling this function does not initiate a capture of data; this makes looking for momentary events (e.g. button presses) infeasible.

The bit values for pins that are defined as inputs are more relevant for this function than those defined as outputs. Note that the value read for a pin may differ from what is driven, even for pins defined as outputs, if external circuitry is driving the pin more strongly than the PCA9555 is.

See FTIRInst_I2cIo_SetPinDirs for additional information.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 74 of 97	Revision: 1.22

FTIRInst_GetExtTemps

This function is called to retrieve the temperatures measured by up to four temperature sensors attached to the external I2C bus.

C# Declaration

```
int FTIRInst_GetExtTemps(
    float *pfTemp1, float *pfTemp2, float *pfTemp3, *pfTemp4);
```

C++ Declaration

```
long FTIRInst_GetExtTemps(
    float *pfTemp1, float *pfTemp2, float *pfTemp3, *pfTemp4);
```

Parameters

pfTemp1

[out] A pointer to a caller-allocated 32-bit float value that will be filled with the temperature of sensor 1 (I2C address 0x98); in case of an error, the value remains as it was when passed in.

pfTemp2

[out] A pointer to a caller-allocated 32-bit float value that will be filled with the temperature of sensor 2 (I2C address 0x9A); in case of an error, the value remains as it was when passed in.

pfTemp3

[out] A pointer to a caller-allocated 32-bit float value that will be filled with the temperature of sensor 3 (I2C address 0x9C); in case of an error, the value remains as it was when passed in.

pfTemp4

[out] A pointer to a caller-allocated 32-bit float value that will be filled with the temperature of sensor 4 (I2C address 0x9E); in case of an error, the value remains as it was when passed in.

Return Values

This method returns a bit-code denoting which temperature sensor values have valid data. If Temp1 is valid, then 0x01 (the LSbit) is ORed in; if Temp2 is valid, then 0x02 is ORed in; if Temp3 is valid then 0x04 is ORed in; if Temp4 is valid then 0x08 is ORed in.

A negative number (-1, -2) is returned in case of an error.

Remarks

This function is specific to LM75 (and compatible) temperature sensors, which are I2C devices that are attached to the FTIR device external I2C bus. Sensor 1 has an I2C address of 0x98, Sensor 2 has an I2C address of 0x9A, Sensor 3 has an I2C address of 0x9C, and Sensor 4 has an I2C address of 0x9E.

Temperature values are read from the temperature sensors no more frequently than every 5 seconds, so no change in values will be seen if this function is called more frequently.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 75 of 97	Revision: 1.22

FTIRInst_GetExtTemp

This function is called to retrieve the temperature measured by an OEM temperature sensor attached to the external I2C bus.

C# Declaration

```
int FTIRInst_GetExtTemp(
    float *pfTemp);
```

C++ Declaration

```
long FTIRInst_GetExtTemp(
    float *pfTemp);
```

Parameters

pfTemp

[out] A pointer to a caller-allocated 32-bit float value that will be filled with the temperature of sensor 1 (I2C address 0x96); in case of an error, the value remains as it was when passed in.

Return Values

This method returns a 1 if the call was successful, and a negative value (-1, -2) if there was an error.

Remarks

This function is specific to the LM76 temperature sensor, which is an I2C device that can be attached externally to the FTIR device I2C bus by an OEM. The temperature sensor must be of type LM76 (or identical), and must have an I2C address of 0x96.

Temperature values are read from the temperature sensors no more frequently than every 5 seconds, so no change in value will be seen if this function is called more frequently.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 76 of 97	Revision: 1.22

FTIRInst_GetOemNvmemData

This function is called to retrieve the OEM data stored in the device's non-volatile memory.

C# Declaration

```
int FTIRInst_GetOemNvmemData(
    byte[] aData,
    int nBufSize);
```

C++ Declaration

```
long FTIRInst_GetOemNvmemData(
    unsigned char *pData,
    long nBufSize);
```

Parameters

pData

[out] A pointer to a caller-allocated buffer that will be filled with the OEM data that is stored inside of a device.

nBufSize

[in] An integer value that represents the size of the buffer that is being passed in.

Return Values

This method returns the number of bytes retrieved if the call was successful; this value is always 128.

A negative value is returned if there was an error; -1 for a general error, -2 for nBufSize too small.

Remarks

Each FTIR device has non-volatile memory, a portion of which can be used to store OEM data; this memory will persist across power losses to the device.

The OEM is solely responsible for managing this data. When an FTIR device is first provided to an OEM, the contents of the OEM NVMEM should contain bytes of all 0xFF. The OEM must choose how to organize his data, must fill the NVMEM with such data, and must interpret the data after reading it from the device. Data is always handled as a single, atomic, 128 B package, for both get and set.

The OEM NVMEM data is fixed at 128 bytes. The nBufSize value that is passed in to this function must be at least 128 B; a larger value will allow the function to continue, while a smaller value will return an error. In general, nBufSize should always be 128.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 77 of 97	Revision: 1.22

FTIRInst_SetOemNvmemData

This function is called to set the data that is stored in the device's non-volatile memory.

C# Declaration

```
int FTIRInst_SetOemNvmemData (
    byte[] aData,
    int nBufSize);
```

C++ Declaration

```
long FTIRInst_SetOemNvmemData(
    unsigned char *pData,
    long nBufSize);
```

Parameters

pData

[out] A pointer to a caller-allocated buffer that contains the OEM data that is to be stored inside of a device's non-volatile memory.

nBufSize

[in] An integer value that represents the size of the buffer that is being passed in.

Return Values

This method returns the number of bytes written if the call was successful; this value is always 128.

A negative value is returned if there was an error; -1 for a general error, -2 for nBufSize too small.

Remarks

Each FTIR device has non-volatile memory, a portion of which can be used to store OEM data; this memory will persist across power losses to the device. See the GetOemNvmemData() documentation for more details.

The OEM NVMEM data is fixed at 128 bytes. The nBufSize value that is passed in to this function should be 128 (bytes); a larger value will allow the function to continue, but only the first 128 bytes will be stored in non-volatile memory; a smaller value will return an error. In general, nBufSize should always be 128.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 78 of 97	Revision: 1.22

FTIRInst_GetLaserTempCompVals

The FTIRInst_GetLaserTempCompVals retrieves the current values used to temperature-compensate the laser wave number of the instrument. These values are relevant only for record-keeping.

C# Declaration

```
int FTIRInst_GetLaserTempCompVals(
    ref float fSlope,
    ref float fIntercept);
```

C++ Declaration

```
long FTIRInst_GetLaserTempCompVals(
    float* pfSlope
    float* pfIntercept);
```

Parameters

fSlope

[out] The value of the slope value used to temperature-compensate the laser wave number.

fIntercept

[out] The value of the slope value used to temperature-compensate the laser wave number.

Return Values

This function returns 0 if successful, otherwise an error code is returned. *Currently no errors are defined.*

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 79 of 97	Revision: 1.22

FTIRInst_SetLaserTempCompVals

The FTIRInst_SetLaserTempCompVals sets the current values used to temperature-compensate the laser wave number of the instrument. These values should only be set after a thorough calibration process.

C# Declaration

```
int FTIRInst_SetLaserTempCompVals(
    ref float fSlope
    ref float fIntercept);
```

C++ Declaration

```
long FTIRInst_SetLaserTempCompVals(
    float* fSlope,
    float* fIntercept);
```

Parameters

fSlope

[out] The value of the slope value used to temperature-compensate the laser wave number.

fIntercept

[out] The value of the slope value used to temperature-compensate the laser wave number.

Return Values

This function returns 0 if successful, otherwise an error code is returned. *Currently no errors are defined.*

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 80 of 97	Revision: 1.22

FTIRInst_EnableLaserTempCompensation

The FTIRInst_EnableLaserTempCompensation enables or disables temperature compensation of the laser frequency.

C# Declaration

```
int FTIRInst_EnableLaserTempCompensation(
    long bEnable);
```

C++ Declaration

```
long FTIRInst_EnableLaserTempCompensation(
    long bEnable);
```

Parameters

bEnable

[in] A boolean value (0 = false; 1 = true) denoting whether to enable (true) or disable (false) the laser temperature compensation

Return Values

This function returns 0 if successful, otherwise an error code is returned.

Remarks

Each FTIR instrument is calibrated so as to find the specific laser frequency of the system. However, this frequency can vary based on the temperature inside of the FTIR device. When Laser Temperature Compensation is enabled, single-beams and/or spectra will be provided that compensate for the change in laser frequency by adjusting for the internal device temperature.

As of 2011-02-28, Laser Temperature Compensation is turned off by default. However, the client must not rely on default behavior, and should make a call to this function to ensure that compensation is enabled or disabled per the client's requirements. This value is NOT stored in the NVMEM of the FTIR device, so this call must be made every time a connection is made to the FTIR device.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 81 of 97	Revision: 1.22

FTIRInst_SetLaserStandardizationType

The FTIRInst_GetSyncdVals retrieves a structure filled with time-sensitive values, these values having been used in computations for the most-recently retrieved data object (e.g. spectrum).

C# Declaration

```
int FTIRInst_SetLaserStandardizationType(
    long nType);
```

C++ Declaration

```
long FTIRInst_SetLaserStandardizationType(
    long nType);
```

Parameters

nType

[in] A value to set the standardization type; the following values are valid:
 0 = No standardization is performed
 1 = Standardization is performed

Return Values

This function returns 0 if successful, otherwise an error code is returned.

Remarks

If Laser Standardization is enabled, then all single-beam and spectra are provided with a consistent frequency range.

As of 2011-02-28, standardization is disabled by default. However, the client must not rely on the default behavior, and should call this function to enable or disable the feature as desired. This value is NOT stored in the NVMEM of the FTIR device, so this call must be made every time a connection is made to the FTIR device.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 82 of 97	Revision: 1.22

FTIRInst_GetSyncdVals

The FTIRInst_GetSyncdVals retrieves a structure filled with time-sensitive values, these values having been used in computations for the most-recently retrieved data object (e.g. spectrum).

C# Declaration

```
int FTIRInst_GetSyncdVals(
    ref MLSyncdVals syncdVals);
```

C++ Declaration

```
long FTIRInst_GetSyncdVals(
    MLSyncdVals* syncdVals);
```

Parameters

syncdVals
[out] A structure containing the synchronized values.

Return Values

This function returns 0 if successful, otherwise an error code is returned. *Currently no errors are defined.*

Remarks

The structure that is pointed to by syncdVals must be allocated by the caller, and the nVersion field **must** be filled in with the appropriate version of the structure. The version must match the size of the allocated structure, since the function will fill in as many fields as the nVersion value dictates.

This function can be used by the client when calculating its own single-beam (i.e. using the fAdjustedLaserFreq member to know the frequency range), or to verify that the proper settings are being used.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 83 of 97	Revision: 1.22

Network Interface

The following sections detail the network-based interface to the FTIR instrument. This includes the general behavior of the device, as well as the API functions used to configure the network interface.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 84 of 97	Revision: 1.22

Data Types and Notation

The data types used in this document are the types as defined in the .NET framework:

Standard data types

int	32-bit integer (typically a long or DWORD in older style languages)
short	16-bit integer
float	32-bit floating point value (typically a float or single precision value in older style languages)
double	64-bit floating point value (typically a double precision value in older style languages)
ref and or array[]	This is the .NET notation convention for a reference to a data type or a reference to an array of values (typically a pointer or ByRef value in older style languages)

Enumerations (specific sets of values stored as 32-bit integer values as follows)

```
enum InterfaceType
{
    Net_If_Disabled = 0,
    Net_If_Wired = 1,
    Net_If_Wireless = 2
};
enum AddrAllocMethod
{
    IpAddr_Static = 0,
    IpAddr_Dhcp = 1,
    IpAddr_AutoIp = 2
};
enum SecurityType
{
    Net_Security_None = 0,
    Net_Security_Wep = 1,
    Net_Security_Wpa = 2,
    Net_Security_Wpa2 = 3
};
enum NetEncryptFlags
{
    NET_ENCRYPT_AES = 0x00000001,    // aka CCMP
    NET_ENCRYPT_TKIP = 0x00000002,
    NET_ENCRYPT_WEP = 0x00000004
};
enum WiFiAuthType
{
```



Title: Agilent Technologies Instrument Interface

Stored as Document Name:
AgilentInterface.doc

Last Modified:
1/30/2018 4:01:00 PM

Page 85 of 97

Revision: 1.22

```
Net_Auth_Open = 0,
Net_Auth_Shared = 1,
Net_Auth_Psk = 2,
Net_Auth_Ieee8021x = 3
};
enum WiFiKeyType
{
    Net_Key_Passphrase = 0,
    Net_Key_Hex = 1
};
enum WiFiKeySize
{
    Net_Key_Size64 = 0,
    Net_Key_Size128 = 1
};
enum EntAuthType
{
    Net_Auth_Undefined = 0,
    Net_Auth_Eaptls = 1,
    Net_Auth_Eapttls = 2,
    Net_Auth_Leap = 3,
    Net_Auth_Peap = 4
};

enum EntInnerAuthType
{
    Net_InnerAuth_Undefined = 0,
    Net_InnerAuth_Eap_Mschapv2 = 1,
    Net_InnerAuth_Mschapv2 = 2,
    Net_InnerAuth_Mschap = 3,
    Net_InnerAuth_Chap = 4,
    Net_InnerAuth_Pap = 5,
    Net_InnerAuth_Eap_Md5 = 6
};
```

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 86 of 97	Revision: 1.22

Data structures (structures of information, usually passed by reference, as follows)

```
typedef struct
{
    u32                nStructLen;           // size of this struct (in bytes)
    InterfaceType      nInterface;          // active interface
} NetSettings;

typedef struct
{
    u32                nStructLen;
    AddrAllocMethod    nAllocType;
    u32                nIpAddress;           // 32-bit IPv4 address
    u32                nNetMask;            // 32-bit mask
    u32                nDefaultGateway;     // 32-bit IPv4 address
} NetAddrSettings;

typedef struct
{
    u32                nStructLen;
    u8                 szSsid[36];           // 32-char null-term'd string
    SecurityType       nSecurityType;
    WiFiAuthType       nAuthentication;
    WiFiKeyType        nKeyType;            //
    WiFiKeySize        nKeySize;            // valid for WEP only
    u32                nTxKeyIndex;         // WEP only; values are 1, 2, 3, 4
    u8                 arKeyBytes[64];      // see Note 5
} WiFiSecurity;

typedef struct
{
    u32                nStructLen;
    EntAuthType        nProtocol;
    EntInnerAuthType   nInnerAuth;         // for TTLS or PEAP
    u8                 szUserName[64];      // for TTLS, LEAP, PEAP
    u8                 szPassword[64];     // for TTLS, LEAP, PEAP
} WiFiEnterpriseSecurity;
```

Default Values

When a networked device is first delivered, it is configured with the following default values, which may be necessary to connect to the device. The NetAddrSettings default values are the same for both the wired and wireless interfaces, although since the wired interface is the default active interface, in practice the default values are relevant only to the wired interface.

Field	Structure Field	Default Value
Interface	NetSettings.nInterface	Wired
Wired IP Addr Type	NetAddrSettings.nAllocType	Static

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 87 of 97	Revision: 1.22

Wired IP Address	NetAddrSettings.nIpAddress	192.168.1.2
Wired Mask	NetAddrSettings.nNetMask	255.255.255.0
Wired Gateway	NetAddrSettings.nDefaultGateway	192.168.1.1

Notes

- (1) All structures **MUST** have their nStructLen member filled in before making any calls, as this field is used to determine the version of the structure being used, and how much memory the caller allocated for the structure.
- (2) IP Addresses are all given in IPv4 format; IPv6 is not supported. This limitation should not be an issue, as FTIR devices should be contained in a private network, with private IP addresses as defined per RFC 1918.
- (3) IP Addresses and network masks are all given with the most-significant byte (MSB) being the first octet in the IP address.
- (4) The WiFiSecurity szSsid[] field must contain a null-terminated string of valid alphanumeric characters.
- (5) The WiFiSecurity arKeyBytes[] field varies depending on the settings of nKeyType and nKeySize (WEP only) fields. If nKeyType is set to Net_Key_Passphrase, then arKeyBytes[] can contain an ASCII string of up to 63 characters, with a terminating null. If nKeyType is set to Net_Key_Hex, then the array contents will vary depending on the security type and the key length. For WPA/WPA2, arKeyBytes[] contains 64 hex digits, i.e. 64 characters in the range [0-9, A-F]. For WEP, arKeyBytes[] contains 4 back-to-back keys, each of either 10 or 26 hex digits, depending on nKeySize. Note that for the hex keys that no terminating-null is applicable, especially since the array is only 64 bytes long.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 88 of 97	Revision: 1.22

FTIR Device General Network Operation

General Network Connectivity

Limited Network Bandwidth

Communication over a network interface to an FTIR device is slower than communication over a USB connection, so a software developer will have to understand the limitations to make the best use of a network connection. Note that this is not a limitation of network communications in general, but rather a limitation of the FTIR device's network interface. Some best practices follow.

1. Collect coadded data objects. This allows the FTIR device to coadd the data internally without transferring; data is coadded at the full scan rate of the device (e.g. 140 scans/min).
2. When collecting data objects for single scans, expect for transfer of the data from the FTIR device to take longer than an actual scan may take. For example, an FTIR device may scan at 140 scans/min, but a network connection may only be capable of transferring 16 to 27 data objects per minute. Thus, while data objects for single scans are useful for giving an end user some feedback, they should never be used for actual data collection.

Wireless Network Connections

A network-enabled FTIR device can be configured to connect over a wired Ethernet or a Wi-Fi connection. For the most reliable data transfer, a wired Ethernet connection is recommended; a variety of environmental factors may affect wireless connections.

To increase the reliability of a wireless connection, consider the following guidelines.

1. Aim the window of the FTIR device towards the wireless access point that it is configured to connect to.
2. Minimize signal degradation by limiting walls, furniture, &c between the device and the access point.
3. Although the 802.11 specification lists a maximum range of up to 300 feet, it is a maximum; a more realistic distance inside a building may only be 100 feet. Every site has different environmental influences that will limit the wireless range.
4. Be aware of and minimize interference caused by other wireless and electrical equipment. Be especially aware of generators, microwaves, Bluetooth devices, cordless phones, and any wireless devices that operate in the 2.4 GHz ISM band.
5. Ensure that the wireless access point that you are trying to connect to is on a different channel than other Wi-Fi networks in the area. For best results, use channels 1, 6, and 11 for different networks to reduce overlaps in the wireless spectrum.
6. Instead of many Wi-Fi networks with overlapping channels, it is better to consolidate multiple Wi-Fi networks into fewer, non-overlapped, networks to prevent interference.
7. Use a high-gain directional antenna on the access point, aimed at the FTIR device.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 89 of 97	Revision: 1.22

8. Move the FTIR device and access point closer together. If this is not possible, then try using a Wi-Fi repeater between the two devices.
9. If wireless communication is still not reliable, then the user will be forced to either re-locate to a more conducive environment or to use the wired Ethernet interface.

Firewall Settings

In order to communicate with the FTIR device over the network, host-side software may need to navigate through a firewall. Connections are always initiated by the host device (e.g. PC); most host firewall software should allow (or be configurable to allow) such outgoing network connections.

For firewalls that require specific ports to be opened for access, it may be necessary to know that the FTIR device is communicated with over TCP port 10001.

Connecting and Disconnecting

The FTIRInst DLL provides feedback on the current status of the connection to a device. There are a number of ways for an application to be aware of the device status. The FTIRInst_GetStatusEx() function provides an MLDiag object with the nSystemStatus field, which gives the status of a connection to an FTIR device; this function can be polled periodically by the application. The FTIRInst_RegisterStatus() and FTIRInst_RegisterStatusEvents() functions allow an application to be informed of connection changes via Windows messages and Windows events respectively.

The following pointers may help to understand how connections are dealt with by the underlying components, and thus inform the application on how best to deal with them.

1. After the FTIR device is powered on, it will take time for the device to boot, initialize its core firmware, configure the networking hardware, and connect to a network. A minimum of 60 seconds should be allowed for such steps to occur. A longer wait may be required when connecting to a wireless network, and/or when using DHCP to allocate IP addresses.
2. In general, it will take the FTIRInst 60 seconds to determine that a network connection has been lost. This timeout allows for wireless communication to continue during multiple retries.
3. Once FTIRInst has determined that a connection has been lost, it will change the state of the connection to Lost—this state is not the same as Unconnected. While in the Lost state, FTIRInst will continue to try to reconnect to the same device at the same IP address.
 - a. The application can choose to terminate the connection by calling FTIRInst_Deinit(), thus placing the connection back into the Unconnected state; the application can then handle attempts to reconnect by itself, by calling FTIRInst_Init().
 - b. The application can wait for FTIRInst to reconnect to the device, upon which the state will be changed to Connected.
4. After a reconnection has occurred, the application should reinitialize the FTIR device to the settings that are desired. The reason for the loss of connection cannot always be assumed—if

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 90 of 97	Revision: 1.22

an FTIR device was powered off for a short period of time and then powered back on, it will appear to reconnect no differently than a device that lost a network connection.

- a. For FTIR devices that have a supported front-panel power button, status notifications of user-initiated power downs are possible. But, this does not work for direct removals of line power, or for devices that do not have front-panel power button.

Startup Connection Interface and Fallback

When first powered on, the FTIR device will attempt to find a connection on the primary interface, which is the interface that has been set through a previous SetNetConfig() call and stored in the non-volatile memory of the device. In case a valid physical connection is not made, the device will attempt to find a connection on other interfaces, or with other settings.

Note that once any valid physical connection is made after startup, then that connection and its associated settings will be deemed the active interface, and all communication must take place through that interface until the device is power cycled. This means that the connection fallback mechanism is NOT valid once an active connection is found.

For the wired Ethernet interface, a physical connection is deemed valid (and thus active) whenever an Ethernet connection is found; if the device is plugged in to an Ethernet device (hub, switch, &c), and a Link light is lit, then the device has found a valid physical connection. Note that it is still possible that the device may not have a valid IP configuration (e.g. IP address, mask, gateway), and thus it may still not be possible to communicate with the device. It is the network technician's responsibility to configure the device properly such that once it gains a physical Ethernet connection that it will be able to be communicated with.

For the wireless Ethernet interface (Wi-Fi), a physical connection requires more. The device must be able to find and connect to the configured wireless network, so all of the wireless settings must be correct. Once the FTIR device has been able to connect to and log onto the wireless network, then it is deemed to have a physical connection. Note that the IP configuration must still be set properly after a physical connection is made.

The FTIR device is designed with a fallback mechanism that will allow the device to cycle through different interfaces and configurations when a physical connection is not made; this allows for finding of a mis-configured device, as well as when a previously-configured interface is no longer valid.

The device will attempt to connect with a different configuration every 5 minutes. The first attempt will be with the settings that were made via the SetNetConfig() call, targetted at the primary interface as defined in the NetSettings.nInterface field. If no physical connection is made after 5 minutes, then the device will try the other interface type, for example it will switch from wired Ethernet to wireless; the SetNetConfig() settings will also be used for this interface.

If the other interface does not result in a physical connection after 5 minutes, then these two configurations will be cycled through one more time, bringing to total time to 20 minutes.

If no physical connection has been made after 20 minutes, then the device will fall back to using factory default values. First, the interface type defined by nInterface is tried, but with the default values. After 5 minutes the other interface will be tried. This swap will continue from then on

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 91 of 97	Revision: 1.22

until a physical connection is made. If it is desired to restart the process, then power will need to be cycled on the FTIR device.

For FTIR devices that provide LED feedback, during each phase of the connection attempt, the Connection LED will blink with a different pattern to represent which interface is being tried; once a connection is made the LED will glow solid green. When attempting to connection over the wired Ethernet interface, the blink pattern will be 500 ms on – 500 ms off. When attempting to connect over the wireless interface, the blink pattern will be 250 ms on – 750 ms off; the wireless interface will flash the LED for a shorter time.

The following table shows the sequence of interfaces and settings that are attempted.

Time	Interface	Settings	LED
0 to 5 minutes	Primary	User-defined (SetNetConfig)	Green (blinking)
5 to 10	Secondary	User-defined	Green (blinking)
10 to 15	Primary	User-defined	Green (blinking)
15 to 20	Secondary	User-defined	Green (blinking)
20 to 25	Primary	Default	Amber (blinking)
25 to 30	Secondary	Default	Amber (blinking)
30+	Alternate	Default	Amber (blinking)

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 92 of 97	Revision: 1.22

FTIRInst DLL – Network Functions

Summary of Function Interfaces

```

long FTIRInst_GetNetConfig(
    NetSettings *pNetSettings,
    NetAddrSettings *pNetAddrSettingsWired,
    NetAddrSettings *pNetAddrSettingsWireless,
    WiFiSecurity *pWifiSecurity,
    WiFiEnterpriseSecurity *pWifiEntSecurity);
long FTIRInst_SetNetConfig(
    NetSettings *pNetSettings,
    NetAddrSettings *pNetAddrSettingsWired,
    NetAddrSettings *pNetAddrSettingsWireless,
    WiFiSecurity *pWifiSecurity,
    WiFiEnterpriseSecurity *pWifiEntSecurity);

```

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 93 of 97	Revision: 1.22

Typical Usage Patterns

Setting New Network Values

1. Call **FTIRInst_Init** and check return value to make sure instrument connects properly.
2. <Optional> check version numbers
3. Allocate memory for the network configuration structures.
4. <Optional> Call **FTIRInst_GetNetConfig** to retrieve the current settings into the structures.
5. Modify the structures to contain the new configuration information.
6. Call **FTIRInst_SetNetConfig** to send the new configuration to the device; the new settings are stored in non-volatile memory, and ready to be applied.
7. Call **FTIRInst_Deinit**, which causes the device to apply the new network settings and disconnects from the device.
8. Wait for 30 to 60 seconds, to allow the device to apply the settings and reset the network connection. A longer wait may be required when allocating IP addresses dynamically (using DHCP), and when connecting over a wireless network.
9. Call **FTIRInst_Init** to connect back with the device.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 94 of 97	Revision: 1.22

FTIRInst_GetNetConfig

The FTIRInst_GetNetConfig function returns the current network settings of the attached device. All structures need to be initialized with the proper value of nStructLen before being passed to this function.

C# Declaration

```
int FTIRInst_GetNetConfig(
    NetSettings          *pNetSettings,
    NetAddrSettings      *pNetAddrSettingsWired,
    NetAddrSettings      *pNetAddrSettingsWireless,
    WiFiSecurity          *pWifiSecurity,
    WiFiEnterpriseSecurity *pWifiEntSecurity);
```

C++ Declaration

```
long FTIRInst_GetNetConfig(
    NetSettings          *pNetSettings,
    NetAddrSettings      *pNetAddrSettingsWired,
    NetAddrSettings      *pNetAddrSettingsWireless,
    WiFiSecurity          *pWifiSecurity,
    WiFiEnterpriseSecurity *pWifiEntSecurity);
```

Parameters

pNetSettings
[out] Pointer to a caller-allocated object to receive the current settings.

pNetAddrSettingsWired
[out] Pointer to a caller-allocated object to receive the current settings.

pNetAddrSettingsWireless
[out] Pointer to a caller-allocated object to receive the current settings.

pWifiSecurity
[out] Pointer to a caller-allocated object to receive the current settings.

pWifiEntSecurity
[out] Pointer to a caller-allocated object to receive the current settings.

Return Values

0 = Success

-1 = Failure

Remarks

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 95 of 97	Revision: 1.22

None.

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 96 of 97	Revision: 1.22

FTIRInst_SetNetConfig

The FTIRInst_SetNetConfig function sends new network settings to the attached device. All structures need to be initialized with the proper value of nStructLen, as well as all values, before being passed to this function.

This function will send the value to the FTIR device, but these values will not be applied to the device until after the caller has disconnected from the device. This allows the networking module within the FTIR device to change to the new settings. The caller can then reconnect to the FTIR device by addressing it with its new settings. The most obvious requirement for such a sequence is a change in IP address.

C# Declaration

```
int FTIRInst_SetNetConfig(
    NetSettings          *pNetSettings,
    NetAddrSettings      *pNetAddrSettingsWired,
    NetAddrSettings      *pNetAddrSettingsWireless,
    WiFiSecurity          *pWifiSecurity,
    WiFiEnterpriseSecurity *pWifiEntSecurity);
```

C++ Declaration

```
long FTIRInst_SetNetConfig(
    NetSettings          *pNetSettings,
    NetAddrSettings      *pNetAddrSettingsWired,
    NetAddrSettings      *pNetAddrSettingsWireless,
    WiFiSecurity          *pWifiSecurity,
    WiFiEnterpriseSecurity *pWifiEntSecurity);
```

Parameters

pNetSettings

[in] Pointer to a caller-allocated and filled object with new settings.

pNetAddrSettingsWired

[in] Pointer to a caller-allocated and filled object with new settings.

pNetAddrSettingsWireless

[in] Pointer to a caller-allocated and filled object with new settings.

pWifiSecurity

[in] Pointer to a caller-allocated and filled object with new settings.

pWifiEntSecurity

[in] Pointer to a caller-allocated and filled object with new settings.

Return Values

0 = Success

Title: Agilent Technologies Instrument Interface	
Stored as Document Name: AgilentInterface.doc	Last Modified: 1/30/2018 4:01:00 PM
Page 97 of 97	Revision: 1.22

-1 = Failure

Remarks

None.