ASD Inc.
2555 55th Street, Suite 100
Boulder, CO 80301

Phone: (303) 444-6522
Fax: (303) 444-6825
Email: support@asdi.com

Indico Version 8 File Format

Table of Contents

Introduction

Overview

The Indico file format is the format for storing both raw data as well as reference data. This format is created and used by the Indico, RS3 and 21CFR software. The following specification gives a detailed description of the structure for version 8 of this format.

Data Format

The ASD Indico file format is native to Windows and there for Intel processors, all data values are stored in Little-Endian (least significant byte first) order.

Variable length strings are stored in BSTR format. A BSTR (Basic string or binary string) is a string data type that is used by COM, Automation, and Interop functions. A BSTR is a composite data type that consists of a length prefix and the data string. The length prefix is a 4 byte integer that defines the length of the string. The data string is followed immediately after the length prefix.

Variable length arrays are stored in SAFEARRAY format.

File Structure

The Indico file layout consists of 12 sections: The following figure displays how these sections are laid out.

Basic Indico File Layout

Spectrum File Header
Spectrum Data
Reference File Header
Reference Data
Classifier Data
Dependent Variable Data
Calibration Header
Base Data
Lamp Data
Fiber Optic Data
Audit Log
Signature

Spectrum File Header

The spectrum file header section is the first section and consists of 484 bytes of data. The following table details the offset and data type format.

Offset	Size	Type	Description	Comment
	3	char	co[3];	// File Version - as6
3	157	char	comments[157];	// comment field
160	18	struct tm	when;	// time when spectrum was saved
178	1	byte	program_version;	// ver. of the programcreatinf this file. // major ver in upper nibble, min in lower
179	1	byte	file_version;	// spectrum file format version
180	1	byte	itime;	// Not used after v2.00
181	1	byte	dc_corr;	// 1 if DC subtracted, 0 if not
182	4	time_t (==long)	dc_time;	// Time of last dc, seconds since 1/1/1970
186	1	byte	data_type;	// see *_TYPE below
187	4	time_t (==long)	ref_time;	// Time of last wr, seconds since 1/1/1970
191	4	float	ch1_wavel;	// calibrated starting wavelength in nm
195	4	float	wavel_step;	// calibrated wavelength step in nm
199	1	byte	data_format;	// format of spectrum.
200	1	byte	old_dc_count;	// Num of DC measurements in the avg
201	1	byte	old_ref_count;	// Num of WR in the average
202	1	byte	old_sample_count;	// Num of spec samples in the avg
203	1	byte	application;	// Which application created APP_DATA
204	2	ushort	channels;	// Num of channels in the detector
206	128	APP_DATA	app_data;	// Application-specific data
334	56	GPS_DATA	gps_data;	// GPS position, course, etc.
390	4	ulong	it;	// The actual integration time in ms
394	2	int	fo;	// The fo attachment's view in degrees
396	2	int	dcc;	// The dark current correction value
398	2	uint	calibration;	// calibration series
400	2	uint	instrument_num;	// instrument number
402	4	float	ymin;	// setting of the y axis' min value
406	4	float	ymax;	// setting of the y axis' max value
410	4	float	xmin;	// setting of the x axis' min value
414	4	float	xmax;	// setting of the x axis' max value
418	2	uint	ip_numbits;	// instrument's dynamic range
420	1	byte	xmode;	// x axis mode. See *_XMODE
421	4	byte	flags[4];	// flags(0) = AVGFIX'ed // flags(1) see below
425	2	unsigned	dc_count;	// Num of DC measurements in the avg
427	2	unsigned	ref_count;	// Num of WR in the average
429	2	unsigned	sample_count;	// Num of spec samples in the avg
431	1	byte	instrument;	// Instrument type. See defs below
432	4	ulong	bulb;	// The id number of the cal bulb
436	2	uint	swir1_gain;	// gain setting for swir 1
438	2	uint	swir2_gain;	// gain setting for swir 2
440	2	uint	swir1_offset;	// offset setting for swir 1
442	2	uint	swir2_offset;	// offset setting for swir 2
444	4	float	splice1_wavelength;	// wavelength of VNIR and SWIR1 splice
448	4	float	splice2_wavelength;	// wavelength of SWIR1 and SWIR2 splice
452	27	float	SmartDetectorType	// Data from OL731 device
479	5	byte	spare[5];	// fill to 484 bytes

Definitions:

Spectrum data type (variable data_type at byte offset 186):

```
#define RAW_TYPE         (byte)0
#define REF_TYPE         (byte)1
#define RAD_TYPE         (byte)2
#define NOUNITS_TYPE     (byte)3
#define IRRAD_TYPE       (byte)4
#define QI_TYPE          (byte)5
```

```
#define TRANS_TYPE      (byte)6
#define UNKNOWN_TYPE    (byte)7
#define ABS_TYPE        (byte)8
```

Spectrum data format (variable data_format at byte offset 199):

```
#define FLOAT_FORMAT    (byte)0
#define INTEGER_FORMAT  (byte)1
#define DOUBLE_FORMAT   (byte)2
#define UNKNOWN_FORMAT  (byte)3
```

Instrument type that created spectrum (variable instrument at byte offset 431):

```
#define UNKNOWN_INSTRUMENT          (byte)0
#define PSII_INSTRUMENT             (byte)1
#define LSVNIR_INSTRUMENT           (byte)2
#define FSVNIR_INSTRUMENT           (byte)3
#define FSFR_INSTRUMENT             (byte)4
#define FSNIR_INSTRUMENT            (byte)5
#define CHEM_INSTRUMENT             (byte)6
#define FSFR_UNATTENDED_INSTRUMENT  (byte)7


struct tm
{
  int   tm_sec;                                 // seconds [0,61]
  int   tm_min;                                 // minutes [0,59]
  int   tm_hour;                                  // hour [0,23]
  int   tm_mday;                                  // day of month [1,31]
  int   tm_mon;                                 // month of year [0,11]
  int   tm_year;                                  // years since 1900
  int   tm_wday;                                  // day of week [0,6] (Sunday = 0)
  int   tm_yday;                                  // day of year [0,365]
  int   tm_isdst;                                 // daylight savings flag
};


typedef long time_t;


APP_DATA - This is a 128 byte field that is used for storing results produced by various
real-time processing routines.

struct GPS_DATA
{
      double    true_heading;
      double    speed;
      double    latitude, longitude;
      double    altitude;
    struct
    {
      unsigned havecomm : 1;
      unsigned terrain : 2;
      unsigned datum   : 6;
      unsigned dist_sp_units : 2;
      unsigned alt_units : 2;
      unsigned mag_var : 2;
      unsigned nav : 1;
    } flags; // these are bit fields totaling to 2 bytes
      char      hardware_mode;
      time_t  timestamp;
    struct
    {
      unsigned corrected : 1;
      unsigned filler    : 15;
    } flags2; // these are bit fields totaling to 2 bytes
      unsigned char satellites[5];
      char      filler[2];
}

flags
    flags(0)
    flags(1)    vnir saturation =1
                swir1 satruation = 2
                swir2 saturation = 3
                Tec1 alarm= 8
                Tec2 alarm = 16
```

```
struct SmartDetectorType
{
  int serial_number;                    // Serial Number
  float Signal                          // Signal
  float dark                            // Dark Signal
  float ref                             // Ref Signal
  short Status                          // Smart Detector Status
  byte avg                              // Averaging
  float humid                           // Humidity
  float temp                            // Temperature
}
```

Spectrum Data

The spectrum data section consists of byte 485 to channels as defined in byte 204 in the spectrum file header. The following table details the offset and data type format.

Offset	Size	Type	Description	Comment
485	channels	double	Spectrum	// Spectrum data to size of channels

Reference File Header

The reference file header section consists of Spectrum Data Size + 1 to the size of Reference File Header. The following table details the offset and data type format.

Offset	Size	Type	Description	Comment
Spectrum Data size + 1	2	bool	ReferenceFlag	// Reference been taken
3	8	date	ReferenceTime	// Time Reference was taken
11	8	date	SpectrumTime	// Time Spectrum was taken
19	n	string	SpectrumDescripton	// Description of Spectrum

Reference Data

The reference data section consists of Reference File Header size + 1 to channels as defined in byte 204 in the spectrum file header. The following table details the offset and data type format.

Offset	Size	Type	Description	Comment
Reference File Header size + 1	channels	double	Reference	// Reference data to size of channels

Classifier Data

The classifier data section consists of Reference Data size + 1 to size of Classifier Data. The following table details the offset and data type format of Classifier Data.

Offset	Size	Type	Description	Comment
Reference Data size + 1	1	byte	yCode	// Type of Classifier Data - 0=SAM, 1=GALACTIC, 2=CAMOPREDICT, 3=CAMOCLASSIFY, 4=PCAZ, 5=INFOMETRIX
1	1	byte	yModelType	// Type of Model Quantify/Classify or both
2	n	String	stitle	// Title of Classifier
n+1	n	String	sSubTitle	// SubTitle of Classifier
n+1	n	String	sProductName	// Product Name
n+1	n	String	sVendor	// Vender Name
n+1	n	String	sLotNumber	// LotNumber of Sample
n+1	n	String	sSample	// Sample Description
n+1	n	String	sModelName	// Model Description
n+1	n	String	sOperator	// Operator Name
n+1	n	String	sDateTime	// Date/time sample taken
n+1	n	String	sInstrument	// Instrument Name
n+1	n	String	sSerialNumber	// Serial Number of Instrument
n+1	n	String	sDisplayMode	// Display Mode
n+1	n	String	sComments	// Comments for sample
n+1	n	String	sUnits	// Units of Concentration
n+1	n	String	sFilename	// File Name for sample
n+1	n	String	sUserName	// User Name
n+1	n	String	sReserved1	// Reservered
n+1	n	String	sReserved2	// Reservered
n+1	n	String	sReserved3	// Reservered
n+1	n	String	sReserved4	// Reservered
n+1	2	integer	iConstituentCount	// Number of Constituents
n+3		ConstituentType	actConstituent()	// See definition below.

Definitions:

```
ConstituentType
{       'Items in the Material Report
   ctConstituentName As String
   ctPassFail As String
   ctMDistance As Double
   ctMDistanceLimit As Double
   ctConcentration As Double
   ctConcentrationLimit As Double
   ctFRatio As Double
   ctResidual As Double
   ctResidualLimit As Double
   ctScores As Double
   ctScoresLimit As Double
   ctModelType As Long
   ctReserved1 As Double
   ctReserved2 As Double
}
```

Dependent Variables

The dependent variables section consists of Classifier Data size + 1 to size of Dependent Variables size. The following table details the offset and data type format of Dependent Variables.

Offset	Size	Type	Description	Comment
Classifier Data size + 1	1	bool	SaveDependentVariables	// Has reference been taken
1	2	integer	DependentVariableCount	// Number of dependent variables
4	n	String	DependentVariableLabels()	// Names of dependents variables
n+1	4	float	DependentVariables()	// Values of dependent variables

Calibration Header

The calibration header defines the calibration data to follow. The count field defines the number of calibration buffers contained in the file. The CalBuffer holds data about each calibration buffer. The following table details the offset and data type format of Calibration Header.

Offset	Size	Type	Description	Comment
Dependent Variables size + 1	1	byte	Count	// Number of calibration buffers in the file.
1	29	CalBuffer	Structure for each calibration buffer	// Defines the Type, Name, Integration Time and Gains of buffer

Definitions:

```
typedef enum _CAL_TYPE
{
   ABS,                                          // Absolute Reflectance File
   BSE,                                          // Base File
   LMP,                                          // Lamp File
   FO,                                           // Fiber Optic File
} CALIBRATION_TYPE;

struct
{
   byte         cbType                           // ABS, BSE, LMP or FO
   char         cbName[20]                      // Name of file
   long         cbIT                             // Integration Time in ms of buffer
   int          cbSwir1Gain                     // Swir1 Gain of buffer
   int          cbSwir2Gain                     // Swir2 Gain of buffer
} CAL_BUFFER;
```

Base Calibration Data

This section consists of either absolute reflectance or base calibration data. The cbType field of the calibration header for the first element will define the type. The following table details the offset and data type format.

Offset	Size	Type	Description	Comment
Calibration Header size + 1	channels	double	Absolute Reflectance or Base file	// data to size of channels defined in the Spectrum Header.

Lamp Calibration Data

This section consists the lamp data. The cbType field of the calibration header for the second element will define the type. The following table details the offset and data type format.

Offset	Size	Type	Description	Comment
Base calibration data size + 1	channels	double	Lamp file	// data to size of channels defined in the Spectrum Header.

Fiber Optic Data

This section consists the fiber optic data. The cbType field of the Calibration header for the second element will define the type. The following table details the offset and data type format.

Offset	Size	Type	Description	Comment
Lamp calibration data size + 1	channels	double	Fiber optic file	// data to size of channels defined in the Spectrum Header. The type of fiber optic is defined in the fo field in the Spectrum Header.

Audit Log

This section defines the Audit Log for each signature event. The log event is defined by the <Audit_Event> and </Audit_Event> tags. Within the audit event are tags that define the log event. Below is a sample of an audit log.

Offset	Size	Type	Description	Comment
0	4	long	Count	// Number of log events in the string array .
4	n	String Array	AuditEvents	// String array for each audit event

Definitions:

```
<Audit_Event>
<Audit_Application>Indico Pro</Audit_Application>
<Audit_AppVersion> 6.0</Audit_AppVersion>
<Audit_Name>Bryon Bending</Audit_Name>
<Audit_Login>\\ASDI\bryon.bending</Audit_Login>
<Audit_Time>2009/12/12 14:11:22 GMT</Audit_Time>
<Audit_Source>c:\ASD\Data\Sample.asd</Audit_Source>
<Audit_Function>Initial Collection</Audit_Function>
<Audit_Notes>Sample prepared and collected following SOP # TG101</Audit_Notes>
</Audit_Event>

<Audit Event>
<Audit_Application>Spectral Viewer</Audit_Application>
<Audit_AppVersion> 1.0</Audit_AppVersion>
<Audit_Name>Don Campbell</Audit_Name>
<Audit_Login>\\ASDI\don.campbell</Audit_Login>
<Audit_Time>2009/12/12 15:11:22 UTC</Audit_Time>
<Audit_Source>c:\ASD\Data\Sample.asd</Audit_Source>
<Audit_Function>Approval</Audit_Function>
<Audit_Notes>Sample approved</Audit_Notes>
</Audit Event>
```

Signature

The section defines the electronic signature of the file. The signature details the user who signed the file, when the file was signed, source file and a reason for signing. The electronic signature uses asymmetric cryptography. Asymmetric Cryptography uses both a private and public key. The private key is used to encrypt the record, while the public key is used to decrypt the record. The signature in the record will consist of the private key and a hash of the record. To detect an altered record a user must compute the hash of the record and compare it to the signature with the signer's public key.

Offset	Size	Type	Description	Comment
0	1	byte	Signed	// 0 – Unsigned 1 - Signed
2	8	date	SignatureTime	// Date and Time File was signed. Value is stored in UTC time.
11	n	string	UserDomain	// Users Login domain
n+1	n	string	UserLogin	// Users Login
n+1	n	string	UserName	// Users Name
n+1	n	String	Source	// Source file at time of signature
n+1	n	string	Reason	// Reason for signature
n+1	n	string	Notes	// Additional notes for the signature
n+1	n	string	PublicKey	// User Public Key
n+1	128	String(128)	Signature	// User Signature – Hash of Record + Private Key