IVI-6.4: File Format Specification

January 27, 2014
Revision 0.97

Important Information

The IVI File Format Specification is authored by the IVI Foundation member companies. For a vendor membership roster list, please visit the IVI Foundation web site at www.ivifoundation.org.

The IVI Foundation wants to receive your comments on this specification. You can contact the Foundation through the web site at www.ivifoundation.org.

Warranty

The IVI Foundation and its member companies make no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The IVI Foundation and its member companies shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Trademarks

Product and company names listed are trademarks or trade names of their respective companies.

No investigation has been made of common-law trademark rights in any work.

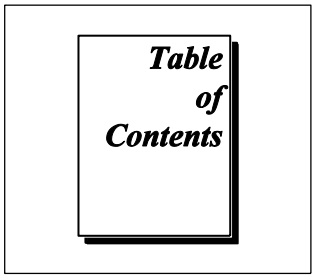

Table of Contents

Revision History

This section is an overview of the revision history of the IVI File Format Specification. Specific individual additions/modifications to the document in draft revisions are denoted with diff-marks, "|", in the right hand column of each line of text to which the change/modification applies.

Table 1. IVI File Format Specification Revision History

Revision Number	Date of Revision	Revision Notes
0.1		Initial draft.
0.2	May 8, 2013	Changes by Christian Hoelzl: • IviRange schema • Vendor specific schema • IviProperties schema • some extensions on the Trace schema • IviUnit schema • IviFunction schema
0.3	May 20, 2013	Changes by Damien Gray • Updated Trace definition • Removed Data definition (Explicit and Implicit data types make it unneeded) • Removed extra lines in tables Changes by Larry Ostheimer • Updated the section on Strings and changed type name in all sections from 'Utf8String' to 'String'. • Removed 'Name' attribute from IviDataGroup schema • Minor edits to IviFunction schema
0.4	May 22, 2013	• Updated section 1.2 – HDF Conventions • Added a new section, 2 – Schema Identification and moved vendor-specific schema to that section • Updated vendor-specific schema definition • Renamed section 3 – IVI File Format Schema Definitions • Updated section 4 – Usage Recommendations
0.5	May 24, 2013	• Split section 3 into two sections, 3 – Schema Definitions and 4 – Named Datatype Definitions • Removed Instrument Information Collection, Instrument Information and Test Information sections. We decided to have examples for such information in the vendor specific section. • Updated the Complex datatype definition and added a working MATLAB example
0.6	May 29, 2013	Changes from Damien Gray • Added missing attributes to implicit data and added clarification to definition of fixed point location Changes from Larry Ostheimer • Updated Timestamp, including a MATLAB example

Table 1. IVI File Format Specification Revision History

0.7	June 7, 2013	• Updated the example for IviSchema and IviSchemaVersion. • Changed the example for the Complex datatype to be a C function instead of MATLAB. • Updated the section for Vendor-specific schema and added an example
0.8	June 10, 2013	Changes by Christian Hoelzl • Updated IviFunction schema • Added sections 6 – 8 for Implicit Functions, LabVIEW Formula Parser and Measurement Functions (taken from specification version 2.0.5: Damien Gray, NI, One Dimensional Data Storage Using HDF5) • Changed IviUnits to IviUnit and updated all references. Also, schema member name changed from Units to Unit. Changes by Larry Ostheimer • Updated example for Timestamp – converted from MATLAB to C.
0.81	July 22, 2013	Changes by Larry Ostheimer • Some cleanup on chapter 3 and 4 section names • Updated IviDigital schema • Added BytesPerSymbol attribute • Added SymbolFormat attribute • Added a Symbol Format Datatype (for use by the SymbolFormat field of the IviDigital schema). • Removed LabVIEW Formula Parser section
0.82	July 25, 2013	Changes by Larry Ostheimer • Updated Summary of Contents section • Updated References section • Updated Encryption section
0.83	September 12, 2013	• Formatted sections 6 and 7 • Removed IviInstrumentInfo and IviTestInfo from the Data Group • Added examples for IviExplicit, IviImplicit and IviFunction • Schema updates for IviTrace, IviImplicit and IviRange
0.84	September 20, 2013	• Updated IviProperties section from Christian • Modified formatting for examples to use 8-point font to take up less space • Updated IviTrace schema. In particular, the Properties element • Added description for IviExplicit and IviImplicit • Added 'Language' attribute to IviFunction • Updated the 'Domain' element of IviImplicit • Updated the definition for dB and log
0.85	September 27, 2013	Changes from Damien Gray • Updated 1.2.5 – Strings • 3.2 – Trace; update to description, addition of IndependentMap attribute, addition of several

Table 1. IVI File Format Specification Revision History

		examples • Updates to descriptions for Explicit and Implicit data
0.86	October 9, 2013	• Moved Range and Concatenation sections up to group with Explicit and Implicit • Added an example for IviDataGroup • Added an example for IviRange • Minor update to example in Implicit Data • Updates to Trace schema
0.87	October 14, 2013	• Update to Trace schema • Updated section 6.1 • Suggestion to remove reference to h5dump.exe
0.88	October 15, 2013	• Edits from IVI Meeting in Dallas
0.89	October 28, 2013	• Added section for HDF5 library versions • Added Data Schemas section for grouping of schemas that represent data • Added Other Schemas section for grouping • Damien provided an update for Explicit Data
0.90	October 29, 2013	• Updated the vendor-specific schemas section, including an updated example. • Accepted revisions discussed so far. • Changed “Other Schemas” to “Supplemental Schemas”
0.91	November 11, 2013	• Reviewed changes and accepted insertions and deletions. • Only a few minor action items remain – search for <TBD> in document.
0.92	November 18, 2013	• Added reference to IEEE 1164 for digital logic values • Removed Count from IviFunction • Added Count to IviImplicit • Made Domain optional in IviImplicit
0.93	November 25, 2013	• Added circle example to IviImplicit • Added trademark symbols for language names under IviFunction • Added two new schema members to IviProperties
0.94	December 2, 2013	• Clarified that IviDataGroup can and often will be the root group. • Clarified that an IviDataGroup cannot go inside another IviDataGroup • Clarified that IviTrace is always in an IviDataGroup • Removed the IviProperties section • Added ‘support required’ for several more function names under IviFunction • Added a note in the description for SIUnit regarding the use of UTF-8
0.95	December 9, 2013	• Updated IviConcatenation section • Moved circle trace example from IviImplicit to IviTrace • Added a simple example for IviImplicit • Improved description for Symbol Format Datatype example

Table 1. IVI File Format Specification Revision History

		• Improved descriptions for Timestamp and Complex datatype examples. • Updated IviFunction to include LanguageName and LanguageVersion • Cleaned up Language Names table under IviFunction • Added LastModified attribute to IviDataGroup
0.96	December 10, 2013	• Final draft for review
0.97	January 27, 2014	• Minor edits from feedback during final review as a new specification

1 Summary of Contents

The IVI File Format Specification defines a set of schemas for archiving and sharing test and measurement data using HDF5. The intent of the standard is to enable data interchange between different products from a particular vendor as well as interchange between products from different vendors.

The implementation for each schema is left to the user. However, this specification includes many examples that use the HDF5 C library. Note that the HDF5 C library is platform independent and this specification makes no assumptions about the implementing platform. Thus, this specification can be supported on any platform for which the HDF5 libraries are available.

1.1 References

This specification references concepts, data types and constant values from the HDF5 specification. Refer to the following web pages for HDF5 documentation:

- HDF5 Home Page: www.hdfgroup.org/HDF5
- HDF5 Software Documentation: www.hdfgroup.org/HDF5/doc

Many of the sections below include example output from the HDF5 utility h5dump.exe. The h5dump.exe utility can be found at the following locations:

- h5dump.exe (32-bit): http://www.hdfgroup.org/ftp/HDF5/current/bin/windows/utilities32/h5dump.exe
- h5dump.exe (64-bit): http://www.hdfgroup.org/ftp/HDF5/current/bin/windows/utilities64/h5dump.exe

1.2 HDF5 Conventions for this specification

1.2.1 HDF5 Library Version

Software that writes data in the format defined in this specification shall create output that is readable by HDF5 software release HDF5-1.8.9 or later. Software that reads data covered by this specification shall be able to read HDF5 format for HDF5-1.8.9.

Note: For Windows implementations, filenames with international characters (UTF-8 strings) are not supported by the HDF5 libraries.

1.2.2 Use of HDF5 Groups, Datasets and Attributes

1.2.2.1 Group

An HDF5 group is analogous to a file system directory. Abstractly, a group contains zero or more objects, and every object must be a member of at least one group. The root group is a special case; it may not be a member of any group.

In this specification, each schema defined below is represented by a group that has each of the attributes, datasets and groups defined by the schema.

1.2.2.2 Dataset

An HDF5 dataset is a multidimensional (rectangular) array of data elements.

In this specification, datasets are primarily used to store data. However, there may be cases where larger sets of meta-data are stored using datasets.

1.2.2.3 Attribute

An HDF5 attribute is a named data value associated with a group, dataset, or named datatype.

In this specification, attributes are used to store meta-data; that is, they store information required to interpret the well-defined schemas of this standard and to use the data itself.

1.2.3 Use of a Link versus an explicit Group or Dataset

In this specification, wherever a Group or Dataset is declared as the HDF5 Object of a schema, the implementation may substitute a Link to a Group or Dataset respectively. Thus, links are typically not defined explicitly in this specification – their application is left to the implementation.

1.2.4 Encryption

Encryption can be applied by using the filter capabilities of the HDF5 libraries. Thus, the use of filters is left to the user and can be applied as appropriate. This standard does not specify where compression (or other filters) should or should not be used.

For more information on Filters, see the HDF5 User's Guide or HDF5 Reference Manual.

1.2.5 Strings

Valid string properties are listed in the table below. All strings, including HDF5 object names (groups, datasets, etc.) may be encoded using ASCII or UTF-8 character encoding.

The datatype name 'String' is used throughout the rest of this document to refer to any valid HDF5 string as defined in the table below.

Note: Issues associated with UTF-8 encoding of object names was considered by the HDF5 standards team in the following paper: Character Encoding for Links in HDF5 Files. The H5L interface to specify the encoding for links was introduced in HDF5 1.8.0.

String Properties

String Property	Value	Description
STRSIZE	<integer length> -or- H5T_VARIABLE	A string may be fixed length or variable length
STRPAD	H5T_STR_NULLTERM	Strings are null terminated
CSET	H5T_CSET_ASCII -or- H5T_CSET_UTF8	May use ASCII or UTF-8 encoding
CTYPE	H5T_C_S1	8-bit characters

2 Schema Identification

2.1 IVI Defined Schemas

Each schema defined in this specification is identified with the attributes named “IviSchema” and “IviSchemaVersion” as defined in the following table.

Name	HDF5 Object	HDF5 Datatype	Required	Description
IviSchema	Attribute	String	Yes	Identifies the specific schema (defined below) associated with the containing HDF5 Group.
IviSchemaVersion	Attribute	String	No	Defines the version of the schema identified by the “IviSchema” attribute. The version number uses semantic versioning as defined by http://semver.org/. Default: “1.0.0”

Example

The following example shows the output for “IviSchema” and “IviSchemaVersion” attributes with values “IviExplicit” and “1.0.0” respectively.

```
ATTRIBUTE "IviSchema" {
   DATATYPE  H5T_STRING {
      STRSIZE 11;
      STRPAD H5T_STR_NULLTERM;
      CSET H5T_CSET_ASCII;
      CTYPE H5T_C_S1;
   }
   DATASPACE  SIMPLE { ( 1 ) / ( 1 ) }
   DATA {
   (0): "IviExplicit"
   }
}
ATTRIBUTE "IviSchemaVersion" {
   DATATYPE  H5T_STRING {
      STRSIZE 5;
      STRPAD H5T_STR_NULLTERM;
      CSET H5T_CSET_ASCII;
      CTYPE H5T_C_S1;
   }
   DATASPACE  SIMPLE { ( 1 ) / ( 1 ) }
   DATA {
   (0): "1.0.0"
   }
}
```

2.2 Vendor Specifc Schemas

Vendor specific extensions to this specification are supported by creating groups with the following vendor-specific identification attributes.

Note that a vendor-specific schema instance may be used anywhere within a tree with root IviDataGroup. That is, vendor-specific schemas may be added anywhere within the tree of a group of IVI data, as defined by this specification.

Schema Identification

Name	Value
IviSchema	"IviVendorSpecific"
IviSchemaVersion	"1.0.0"

Schema Members

Name	HDF5 Object	Datatype or Schema	Required	Description
IviVpp9Ident	Attribute	String	Yes	Two letter upper case abbreviation for the vendor from VPP-9: Instrument Vendor Abbreviations
SchemaDescription	Attribute	String	No	An optional description of the vendor-specific schema. The use of a UUID may be appropriate to avoid schema naming conflict within an organization.
SchemaUrl	Attribute	String	No	An optional vendor-specific URL that hosts the schema definition. The format of the schema definition is left to the vendor to define (e.g. XML, JSON, etc.)

No vendor-specific schemas are defined in this specification. This specification only defines how vendor-specific schemas are identified via the attributes defined above. Vendors may choose to publish their vendor-specific schemas, but it is not required. If the schemas are published, the optional schema definition URL is provided.

Example

The following example shows the output for a Rohde & Schwarz specific schema, with attribute name prefix 'RS'.

```
GROUP "Vendor_Specific" {
   ATTRIBUTE "IviSchema" {
      DATATYPE  H5T_STRING {
         STRSIZE 17;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      }
      DATASPACE  SCALAR
      DATA {
      (0): "IviVendorSpecific"
      }
   }
   ATTRIBUTE "IviSchemaVersion" {
      DATATYPE  H5T_STRING {
```

```
            STRSIZE 5;
            STRPAD H5T_STR_NULLTERM;
            CSET H5T_CSET_ASCII;
            CTYPE H5T_C_S1;
         }
         DATASPACE  SCALAR
         DATA {
         (0): "1.0.0"
         }
      }
      ATTRIBUTE "IviVpp9Ident" {
         DATATYPE  H5T_STRING {
            STRSIZE 2;
            STRPAD H5T_STR_NULLTERM;
            CSET H5T_CSET_ASCII;
            CTYPE H5T_C_S1;
         }
         DATASPACE  SCALAR
         DATA {
         (0): "RS"
         }
      }
      ATTRIBUTE "SchemaDescription" {
         DATATYPE  H5T_STRING {
            STRSIZE 36;
            STRPAD H5T_STR_NULLTERM;
            CSET H5T_CSET_ASCII;
            CTYPE H5T_C_S1;
         }
         DATASPACE  SCALAR
         DATA {
         (0): "f87c5e61-a965-480b-9265-eadb86abb704"
         }
      }
      ATTRIBUTE "SchemaUrl" {
         DATATYPE  H5T_STRING {
            STRSIZE 26;
            STRPAD H5T_STR_NULLTERM;
            CSET H5T_CSET_ASCII;
            CTYPE H5T_C_S1;
         }
         DATASPACE  SCALAR
         DATA {
         (0): "<URL to schema definition>"
         }
      }
   }
```

3 Schema Definitions

Following are the schemas defined by this specification.

3.1 Data Group

Description

A Data Group is the highest level grouping for a related set of data stored using the format defined in this specification. All other schemas defined herein must be included within an instance of this "IviDataGroup" schema. This can and often will be the root group.

The IVI Data Group can go anyplace within an HDF5 file – it doesn't have to go at the root level. However, it cannot go inside another IVI Data Group. For test and measurement data, the typical use case is to have one or more IVI Data Group instances in it. However, it is possible to have other information stored in the HDF5 file that is not covered by this specification.

Schema Identification

Name	Value
IviSchema	"IviDataGroup"
IviSchemaVersion	"1.0.0"

Schema Members

Name	HDF5 Object	Datatype or Schema	Required	Description
Note	Attribute	String	No	A user-defined note describing the the data group.
Contact	Attribute	String	No	The name of a contact associated with this data group.
Project	Attribute	String	No	A project name associated with this data group.
Created	Group	"IviTimestamp"	No	The date and time the data group was created.
LastModified	Group	"IviTimestamp"	No	The date and time the data was last modified

Example

Following is an example explicit data representation.

```
GROUP "Data_Group" {
    ATTRIBUTE "IviSchema" {
       DATATYPE  H5T_STRING {
          STRSIZE 12;
          STRPAD H5T_STR_NULLTERM;
          CSET H5T_CSET_ASCII;
          CTYPE H5T_C_S1;
       }
       DATASPACE  SCALAR
       DATA {
       (0): "IviDataGroup"
```

```
      }
   }
   ATTRIBUTE "IviSchemaVersion" {
      DATATYPE  H5T_STRING {
         STRSIZE 5;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      }
      DATASPACE  SCALAR
      DATA {
      (0): "1.0.0"
      }
   }
   ATTRIBUTE "Note" {
      DATATYPE  H5T_STRING {
         STRSIZE 67;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      }
      DATASPACE  SCALAR
      DATA {
      (0): "This group contains data that conforms to the IVI File Format."
      }
   }
   ATTRIBUTE "Created" {
      DATATYPE  "/Data_Group/IviTimestampType"
      DATASPACE  SCALAR
      DATA {
      (0): {
            1380671672,
            1717792167608758784
         }
      }
   }
   DATATYPE "IviTimestampType" H5T_COMPOUND {
      H5T_STD_I64LE "s";
      H5T_STD_U64LE "f";
   }
}
```

3.2 Trace

Description

The Trace is the top level data schema. It is always in the IVI Data Group. It is an aggregation of other data schemas and can be envisioned as something to be displayed or analyzed, like the trace on an oscilloscope. Traces can be multidimensional, such as the data from a joint time-frequency analysis or a DPO oscilloscope.

Traces are composed of dependent and independent data. The number of independent data sets is normally equal to the dimensionality of the dependent data sets. For example, an oscilloscope trace is one dimensional, so it should have one independent data set, the time axis. A DPO oscilloscope image has two dimensions and two independent data sets, time and voltage. By default, the ordering of the independent data sets (0,1, …) corresponds to the ordering of the indices in the dependent data. This can be changed with the IndependentMap. Each element of the IndependentMap corresponds to an independent data set (e.g. element 1 of the Independent map corresponds to Independent data set "1"). The values in the IndependentMap indicate which dimension of the Dependent data that the corresponding Independent data maps to. The dimensionality must match. A negative value means that the corresponding Independent data set is not used. Since IndependentMap is an attribute of Dependent data sets, there can be different independent data sets for each Dependent data set.

Multiple sets of dependent data use the same independent data. For example, a two channel oscilloscope would save two sets of dependent data (the voltage data) and one set of independent data (the time data).

The use of independent data is optional. Any dimension of the dependent data which does not have a corresponding independent data set is assumed to use a simple zero-based index. For example, a researcher logs 1000 data points at indeterminate times in the dependent data set with no independent data set. On read, the X values will use 0, 1, 2, … A more complex example is saving 2×2 S matrices. The user stores 1000 S matrices as a 1000×2×2 dataset – the dependent data. The independent data is a 1000 element set of frequency points. The unspecified axes are read as 2×2 S matrices. Higher order arrays are treated as row-major, since this is the default for HDF5.

Schema Identification

Name	Value
IviSchema	"IviTrace"
IviSchemaVersion	"1.0.0"

Schema Members

Name	HDF5 Object	Datatype or Schema	Required	Description
Independent	Group	Collection of <Data Schema>	No	This group contains a set of groups labeled 0..n – each of these groups contains a <Data Schema>
Dependent	Group	Collection of <Data Schema>	Yes	This group contains a set of groups labeled 0..n – each of these groups contains a <Data Schema>
IndependentMap	Attribute of specific Dependent group	1D integer array	No	Map of which Independent data sets correspond to what indices of the Dependent data set. See example below. Default is {0,1,…}
PreviewImage	Image	H5IM	No	Representative image of the trace that can be quickly displayed for rapid preview of the data
Properties	Group	Collection of "IviProperties"	No	This group contains a set of groups labeled 0..n – each of these groups contains "IviProperties" schemas

Example

Simple 1D Example – 2 Channel Oscilloscope

A two-channel oscilloscope takes a data set containing 1024 points per channel. The Trace contains two groups, Independent and Dependent. The Independent group contains an IviImplicit data set specifying the time data in group "0". The Dependent group contains two groups, "0" and "1", which contain IviExplicit data holding the channel data.

Trace
- Independent
  - 0 – IviImplicit data set containing timing information (1D array)
- Dependent
  - 0 – IviExplicit data from channel 1 of the oscilloscope (1D array)
  - 1 – IviExplicit data from channel 2 of the oscilloscope (1D array)

Simple 2D Example – Single Channel DPO

A DPO (digital phosphor oscilloscope) generates a 2D image with the x-axis being time, the y-axis voltage, and each element representing the probability that the trace crossed that voltage-time point. A typical image has width 1024 pixels and height 256 pixels. The Independent group contains two elements, "0" and "1" corresponding to the time and voltage information. The Dependent group contains one element, a 2D array containing the DPO data.

Trace
- Independent
  - 0 – IviImplicit data set containing timing information (1024 element 1D array)
  - 1 – IviImplicit data set containing voltage information (256 element 1D array)
- Dependent
  - 0 – IviExplicit data set containing DPO data (1024×256 element 2D array)

Using Multiple Independent Data Sets with One Dependent Data Set (IndependentMap use)

Normally, the number of Independent data sets equals the number of dimensions of the Dependendent data sets, and the mapping of Independent to Dependent axes is implicit (e.g. Independent data set "0" maps to Dependent data set axis 0). However, in practice, this is not always true. Consider a voltage signal being measured with a DPO scope using two bias points. 256 points are taken at the first bias, 768 at the second. In this case, there are three Independent data sets (time, voltage and bias) and one Dependent (DPO image). To accurately set which Independent data set corresponds to which Dependent axis, the IndependentMap attribute is added to the Dependent data set. It is an array of signed integers which specifies which axis each Independent data set corresponds to.

Trace
- Independent
  - 0 – IviImplicit data set containing timing information (1024 element 1D array)
  - 1 – IviConcatentation data set containing bias information (1024 element 1D array)
    - 0 – IviImplicit data set containing first bias (256 element 1D array)
    - 1 – IviImplicit data set containing second bias (768 element 1D array)
  - 2 – IviImplicit data set containing voltage information (256 element 1D array)
- Dependendent
  - 0 – IviExplicit data set containing DPO data (1024×256 element 2D array)
    - IndependentMap – HDF5 attribute, 1D array {0,0,1}

Multiple Mixed Segments with Multiple Independent Data Sets

An engineer takes three oscilloscope data sets, one with a constant bias A, one with bias A, then B, and one with bias B. The engineer saves this as a single trace. The bias and time are saved as independent data sets, the voltage waveform from the oscilloscope as dependent data with an IndependentMap.

Trace
- Independent
  - 0 - IviConcatenation
    - 0 – IviRange data set containing time values from 0µs to 10µs in 500ps increments for 20 points
    - 1 – IviRange data set containing time values from 0µs to 20µs in 500ps increments for 40 points
    - 2 – IviRange data set – points to same data as item "0", 20 points
  - 1 – IviConcatenation
    - 0 – IviImplicit data set containing 40 points of bias A
    - 1 – IviImplicit data set containing 40 points of bias B
- Dependent
  - 0 – IviImplicit data – 80 voltage points from oscilloscope
    - IndependentMap – HDF5 Attribute, 1D Array, {0,0}

Example of 2 Different Ways of Representing the Same Data

Let's take a simple example that contains two independent variables and represent it in two different ways. Suppose we have a variable Dep that represents a sampled time-domain voltage waveform over two different bias points. For each bias point, the sampled waveform contains 3 sample points (at times t=0, 1, and 2 ns). The two bias points are 3V and 5V.

Here is one way of representing the data:

Trace (name = Dep)
- Independent
  - 0 – IviExplicit data set containing bias values 3V and 5V
  - 1 – IviExplicit data set containing time points 0ns, 1ns, and 2ns
- Dependent
  - 0 – IviExplicit data set with dimensions (2x3) containing the sampled voltage waveforms.Notice that in the above example we don't have an IndependentMap property because by default it is (0,1).

Here is a different way of representing the same thing:

Trace (name = Dep)
- Independent
  - 0 – IviExplicit data set (vector of length 6) containing bias values (3, 3, 3, 5, 5, 5) (in volts)
  - 1 – IviExplicit data set (vector of length 6) containing time points (0, 1, 2, 0, 1, 2) (in ns)
- Dependent
  - 0 – IviExplicit data set (vector of length 6) containing the sampled voltage waveforms.
  - IndependentMap – HDF5 Attribute, 1D Array, {0, 0}

The only difference between the above two representations is the way you index into the data.

Example of 2-port S Parameters over a Range of Frequencies

Suppose we want to represent 2-port S Parameters, which is represented by a 2x2 matrix, swept over 101 frequencies (100 MHz through 1 GHz in steps of 10 MHz).

Here is a straightforward way of representing this:

Trace
- Independent
  - 0 – IviRange data set containing values from 100 MHz to 1 GHz for 101 points
- Dependent
  - 0 – IviExplicit data set with dimensions (101x2x2) containing the S-parameters for each frequency.

Example of a circle

Following is an example for using implicit data to represent a circle. The independent data is a sine wave with points at every degree (step in domain is $\pi/180$). The dependent data is a cosine (90° phase, third coefficient)

```
Circle
    IviSchema: IviTrace
    IviSchemaVersion: 1.0.0
    Independent:
        0:
            IviSchema: IviImplicit
            IviSchemaVersion: 1.0.0
            Function:
                IviSchema: IviFunction
                IviSchemaVersion: 1.0.0
                Function: Sine
                Coeff: {1,1,0,0}
            Domain:
                IviSchema: IviRange
```

```
            IviSchemaVersion: 1.0.0
            Start: 0
            Count: 360
            Step: 0.017453293
Dependent
    0:
        IviSchema: IviImplicit
        IviSchemaVersion: 1.0.0
        Function:
            IviSchema: IviFunction
            IviSchemaVersion: 1.0.0
            Function: Sine
            Coeff: {1,1,90,0}
        Domain: link to Independent Domain
```

3.3 Data Schemas

This section defines schemas that are used to respresent sets of data. Collectively, these schemas are referred to as "Data Schemas".

Wherever <Data Schema> is referenced, any of the schemas in this section may be used.

3.3.1 Explicit Data

Description

The Explicit Data schema represents explicit data. Each data element is explicitly defined in the dataset named 'Data' (as defined below). Fixed point data can be used by specifying an integer data type and setting the Scaling factor appropriately.

For performance reasons, a fixed sized dataset may be allocated to avoid space allocation for each chunk written. If this is done, the entire dataset may not be used. In this case, the Count attribute is used to specify the extent of the valid data. Count will have the same dimensionality as Data. If Data is one dimension, Count can be either a scalar or a single element, 1D array.

In some cases invalid data may be embedded in a larger data set. For example, a digitizer may have invalid data if the input value is larger than the digitizer range or an experiment may have momentary bad values during periods of change to the input parameters. In these cases, the user may specify the indices of these data points with the Invalid dataset. Invalid always has the same dimensionality as Data. The first array index is the index to the element of the array. The remaining dimensions are necessary for fully specifying the invalid point. For example a 2D data array has three invalid points at indices {6,4}, {1,3} and {5,5}. The Invalid array would be a 2D array with value {{6,4}, {1,3}, {5,5}}.

Schema Identification

Name	Value
IviSchema	"IviExplicit"
IviSchemaVersion	"1.0.0"

Schema Members

Name	HDF5 Object	Datatype or Schema	Required	Description
Data	Dataset	<Any HDF5 type>	Yes	The set of explicit data elements
Unit	Group	"IviUnit"	No	The unit for each data element, represented by an instance of the IviUnit schema. If not specified, the default is a dimensionless unit 1.
Scaling	Group	"IviFunction"	No	Scaling to be applied to each element of Data.
Timestamp	Group	"IviTimestamp"	No	Time at which the first point of data was taken
Count	Attribute	hsize_t or array of hsize_t	No	Number of valid points in the dataset.
Invalid	Dataset	array of hsize_t	No	Indices (0 based) for invalid data elements.

Example

Following is an example explicit data representation.

```
GROUP "Explicit_Data" {
   ATTRIBUTE "IviSchema" {
      DATATYPE  H5T_STRING {
         STRSIZE 11;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      }
      DATASPACE  SCALAR
      DATA {
      (0): "IviExplicit"
      }
   }
   ATTRIBUTE "IviSchemaVersion" {
      DATATYPE  H5T_STRING {
         STRSIZE 5;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      }
      DATASPACE  SCALAR
      DATA {
      (0): "1.0.0"
      }
   }
   DATASET "Data" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  SIMPLE { ( 1, 20 ) / ( 1, 20 ) }
      DATA {
      (0,0): 1000, 1010, 1020, 1030, 1040, 1050, 1060, 1070, 1080, 1090,
      (0,10): 1100, 1110, 1120, 1130, 1140, 1150, 1160, 1170, 1180, 1190
      }
   }
   GROUP "Unit" {
      ATTRIBUTE "IviSchema" {
         DATATYPE  H5T_STRING {
            STRSIZE 7;
            STRPAD H5T_STR_NULLTERM;
            CSET H5T_CSET_ASCII;
            CTYPE H5T_C_S1;
         }
         DATASPACE  SCALAR
```

```
            DATA {
            (0): "IviUnit"
            }
         }
         ATTRIBUTE "IviSchemaVersion" {
            DATATYPE  H5T_STRING {
               STRSIZE 5;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            }
            DATASPACE  SCALAR
            DATA {
            (0): "1.0.0"
            }
         }
         ATTRIBUTE "SIUnit" {
            DATATYPE  H5T_STRING {
               STRSIZE 2;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            }
            DATASPACE  SCALAR
            DATA {
            (0): "Hz"
            }
         }
      }
   }
```

3.3.2 Implicit Data

Description

The Implicit Data schema represents a set of data defined by a function with an associated domain. Either Domain or Count must be specified. If both Domain and Count are specified, Count is ignored.

Schema Identification

Name	Value
IviSchema	"IviImplicit"
IviSchemaVersion	"1.0.0"

Schema Members

Name	HDF5 Object	Datatype or Schema	Required	Description
Function	Group	"IviFunction"	Yes	See IviFunction definition
Domain	Dataset or Group	Dataset or <Data Schema>	No	The domain for the Function defined above. The default is 0, 1, 2, …
Count	Attribute	hsize_t	No	Specifies the number of points. Only used if Domain isn't present.
Unit	Group	"IviUnit"	No	The unit for each data element, represented by an instance of the IviUnit schema. The default is unitless.

Name	HDF5 Object	Datatype or Schema	Required	Description
Scaling	Group	"IviFunction"	No	Scaling to be applied to each element of Data. Fixed point data can be used by specifying an integer data type for the Domain and setting the Scaling factor appropriately.
Timestamp	Group	"IviTimestamp"	No	Time at which the first point of data was taken

Example

Here is an example where we represent the polynomial f(x) = 3 + 5x, for x = 0, 1, 2, …, 10.

```
Line
    IviSchema: IviImplicit
    IviSchemaVersion: 1.0.0
    Function:
        IviSchema: IviFunction
        IviSchemaVersion: 1.0.0
        Function: Polynomial
        Coeff: {3,5}
    Domain:
        IviSchema: IviRange
        IviSchemaVersion: 1.0.0
        Start: 0
        Count: 11
        Step: 1
```

3.3.3 Range

Description

The IviRange schema is used to define a range of equidistant points. The range is defined by attributes defining the first value, the number of points and the step between two adjacent points.

Schema Identification

Name	Value
IviSchema	"IviRange"
IviSchemaVersion	"1.0.0"

Schema Members

Name	HDF5 Object	Datatype or Schema	Required	Description
Start	Attribute	Numeric	Yes	The Start attribute defines the first data point of a range definition.
Count	Attribute	Numeric	Yes	The number of data points the range consists of. The attribute is a positive scalar value.

Name	HDF5 Object	Datatype or Schema	Required	Description
Step	Attribute	Numeric	No	The interval between two data points. The sign of the range is may be positive or negative. This attribute can be omitted. Default = 1.0
Unit	Group	"IviUnit"	No	Unit definition of the range. Default is unitless.

Example

Following is an example implicit data representation:

```
GROUP "Range_Data" {
   ATTRIBUTE "IviSchema" {
      DATATYPE  H5T_STRING {
         STRSIZE 8;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      }
      DATASPACE  SCALAR
      DATA {
      (0): "IviRange"
      }
   }
   ATTRIBUTE "IviSchemaVersion" {
      DATATYPE  H5T_STRING {
         STRSIZE 5;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      }
      DATASPACE  SCALAR
      DATA {
      (0): "1.0.0"
      }
   }
   ATTRIBUTE "Start" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  SCALAR
      DATA {
      (0): 0
      }
   }
   ATTRIBUTE "Count" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  SCALAR
      DATA {
      (0): 256
      }
   }
   ATTRIBUTE "Step" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  SCALAR
      DATA {
      (0): 1
      }
   }
}
```

3.3.4 Concatenation

Description

The IviConcatenation schema allows multiple sets of data (which we'll call the referred data) to be represented as one larger set of data that contains the concatenation of all the referred data. It provides a compact way to concatenate data together without explicitly having to allocate a larger new dataset.

The writer is responsible for ensuring each element can be concatenated (e.g. the dimensions are compatible) and the reader should validate this as well.

Schema Identification

Name	Value
IviSchema	"IviConcatenation"
IviSchemaVersion	"1.0.0"

Schema Members

Name	HDF5 Object	Datatype or Schema	Required	Description
0	Group or Dataset	Dataset or <Data Schema>	Yes	First element in the concatenation list must be named the integer zero, represented as a string
1	Group or Dataset	Dataset or <Data Schema>	No	Each successive element is the next integer increment represented as a string
…				Any number of elements

Example

MyData represents an array of length 90 where the first 40 elements are 1 … 40 and the 41st element is 1 and 42nd element is 2 … and the 90th element is 50.

```
MyData
    IviSchema: IviConcatenation
    IviSchemaVersion: 1.0.0
        0:
            IviSchema: IviRange
            IviSchemaVersion: 1.0.0
            Start: 1
            Count: 40
            Step: 1
        1:
            IviSchema: IviRange
            IviSchemaVersion: 1.0.0
            Start: 1
            Count: 50
            Step: 1
```

3.3.5 Digital Data

Description

The Digital Data schema defines how to represent data and metadata for a digital bitstream. Various forms of digital bitstreams can be represented by this format.

Schema Identification

Name	Value
IviSchema	"IviDigital"
IviSchemaVersion	"1.0.0"

Schema Members

Name	HDF5 Object	Datatype or Schema	Required	Description
Data	Dataset	Any unsigned integer type, or a type derived from an integer type (with sign set to H5T_SGN_NONE) or any bitfield type, or a type derived from a bitfield type	Yes	The digital data Integer type is recommended so that the n-bit filter can be used, if desired. See section 5.6.1 of the HDF5 User's Guide for information on the n-bit filter.
BytesPerSymbol	Attribute	Unsigned integer	Yes	The number of bytes in a single digital "word" or "symbol".
SymbolFormat	Attribute	A scalar or array of BitField datatype (defined in a later section)	Yes	An array of Symbol Format datatype (defined below) describing each field in a single symbol.

3.3.5.1 Symbol Format Datatype

This datatype is used in the 'SymbolFormat' attribute of the 'IviDigital' schema. The datatype describes a field for the digital data stored in the 'Data' member of the 'IviDigital' schema. If there is more than one field for the digital data, then the 'SymbolFormat' attribute will be an array of elements with this datatype.

H5T_COMPOUND Datatype

Field Name	Datatype	Required	Description
Name	String	Yes	The name of the bit field.
FirstBit	Unsigned integer	Yes	The location of the first bit of the bit field.
LastBit	Unsigned integer	Yes	The location of the last bit of the bit field. If the bit field is only one bit wide, FirstBit and LastBit will have the same value.
BitsPerSample	Unsigned integer	No	The numbers of bits associated with each digital sample. Default: 1 (bit per sample)
LogicStates	String	No	The specific logic states. Note that IEEE 1164 defines nine logic states. Default: "01"

Example

Following is an example 'SymbolFormat' attribute using the Symbol Format compound datatype defined above.

```
ATTRIBUTE "SymbolFormat" {
   DATATYPE  H5T_COMPOUND {
      H5T_STRING {
         STRSIZE H5T_VARIABLE;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      } "Name";
      H5T_STD_U16LE "FirstBit";
      H5T_STD_U16LE "LastBit";
      H5T_STD_U16LE "BitsPerSample";
      H5T_STRING {
         STRSIZE H5T_VARIABLE;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      } "LogicStates";
   }
   DATASPACE  SIMPLE { ( 1, 2 ) / ( 1, 2 ) }
   DATA {
   (0,0): {
         "Data",
         0,
         5,
         2,
         "01Z"
      },
   (0,1): {
         "Pad",
         6,
         7,
         1,
         "0"
      }
   }
}
```

3.4 Supplemental Schemas

3.4.1 Function

Description

The IviFunction schema is used within the fileformat for scaling data, custom units conversion and implicit data. It basically defines a mathematical operation that generates data. The data that is generated may be based on coefficients only (recommended) or based on formula expressions with variables. There is a list of predefined functions that reader and writer shall support. The known functions are listed in Table 2.

Schema Identification

Name	Value
IviSchema	"IviFunction"
IviSchemaVersion	"1.0.0"

Schema Members

Name	HDF5 Object	Datatype or Schema	Required	Description
Function	Attribute	String	Yes	Function name or expression
Coeff	Attribute	Array of Numeric	Yes	A one-dimensional array of numeric coefficients for the specified function. For example polynomial coefficients are stored here in case of the "Polynomial" Function.
Expression	Attribute	String	No	An expression for function "Arbitrary".
LanguageName	Attribute	String	No	The name of the language used for the 'Expression' attribute when the Function is 'Arbitrary'. This attribute is required if the function is 'Arbitrary'
LanguageVersion	Attribute	String	No	The version of the language specified by 'LanguageName'. This attribute is required if the function is 'Arbitrary'

Function Names

The following table defines the set of functions supported by this schema. The 'Function Name' column defines the string to use in the 'Function' attribute defined in the schema members table above.

Note that not all functions must be supported by all IVI File Format readers. See the 'Support Required' column.

Table 2 Overview on the supported functions by IviFunction schema

Function Name	Description	Support Required
Arbitrary	Natural function syntax with variables. The syntax is specified in the language section.	
Constant	f(x) := a0 generates a constant value.	Yes
Exponential	This function produces an exponential ramp.	Yes
ExponentialRepetitive	This function produces a repetitive exponential ramp.	
HalfSine	This function produces a half cycle sine wave between two points.	
Haversine	This function produces a haversine wave – a sine wave with the base on the x-axes.	
Linear	f(x) := a0 + a1*x generate linear values by offset and slope.	Yes
Logarithmic	This function produces a logarithmic ramp.	Yes
LogarithmicModulo	This function produces a repetitive logarithmic ramp.	
NoiseBernoulli	Generates a pseudorandom pattern of ones and zeroes.	

Function Name	Description	Support Required
NoiseBinomial	Generates a binomial distributed pseudorandom pattern.	
NoiseGamma	Generates a Gamma distributed pseudorandom pattern.	
NoiseGaussian	Generates a Gaussian-distributed pseudorandom pattern.	
NoisePeriodic	Generates an array of periodic random noise.	
NoisePoisson	Generates a Poisson distributed pseudorandom pattern.	
NoiseUniformWhite	Generates a pseudorandom uniform white noise pattern.	
Polynomial	f(x) := a0 + a1*x + a2*x^2 + …	Yes
PulseGaussian	This function produces a single, Gaussian profile pulse.	
PulseImpulse	This function produces an impulse.	
PulseLorenz	This function produces a single, lorenzian profile pulse.	
PulseSquare	This function produces a single flat-topped pulse.	
Ramp	This function produces a line.	Yes
Sawtooth	This function produces a sawtooth wave.	Yes
Sinc	This function produces a sinc waveform.	
Sine	This function produces a sine wave.	Yes
Square	This function produces a square vave.	Yes
Stairstep	This function produces a stairstep waveform.	
SweepEllipse	This function produces a swept repetitive elliptical segment wave.	
SweepSawtooth	This function produces a swept sawtooth wave.	
SweepSine	This function produces a swept sine wave.	
SweepSquare	This function produces a swept square wave.	
SweepTriangle	This function produces a swept triangle wave.	
Trapezoid	This function produces a trapezoid segment.	
Triangle	This function produces a triangle wave.	Yes

Language Names

The following table defines well-known language names used in Test and Measurement for the optional ‘LanguageName’ attribute.

Language Name	Description
C	C
python	Python script
perl	Perl script
matlab	MathWorks MATLAB®
lvmath	National Instruments LabVIEW® math VI format
vb	Microsoft Visual Basic®

Examples

Following is an example for a linear function with coefficients {1000, 10}:

```
GROUP "Function" {
   ATTRIBUTE "IviSchema" {
      DATATYPE  H5T_STRING {
         STRSIZE 11;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      }
      DATASPACE  SCALAR
      DATA {
      (0): "IviFunction"
      }
   }
   ATTRIBUTE "IviSchemaVersion" {
      DATATYPE  H5T_STRING {
         STRSIZE 5;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      }
      DATASPACE  SCALAR
      DATA {
      (0): "1.0.0"
      }
   }
   ATTRIBUTE "Function" {
      DATATYPE  H5T_STRING {
         STRSIZE 6;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      }
      DATASPACE  SCALAR
      DATA {
      (0): "Linear"
      }
   }
   ATTRIBUTE "Coeff" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  SIMPLE { ( 1, 2 ) / ( 1, 2 ) }
      DATA {
      (0,0): 1000, 10
      }
   }
}
```

3.4.2 Unit

The unit definition in the IVI file format strictly follows the SI definitions and guidelines as standardized by NIST and BIPM.

[1] BIPM Bureau International des Poids et Mesures (SI maintenance agency)
http://www.bipm.org/en/si/
[2] NIST Special Publication 330, 2008 Edition, The International System of Units (SI)
http://physics.nist.gov/Pubs/SP330/sp330.pdf
[3] NIST Special Publication 811, 2008 Edition, Guide for the Use of the International System of Units (SI)
http://physics.nist.gov/cuu/pdf/sp811.pdf

Allowed unit symbols are:

- **SI base units** [2, table 1]
  - m, kg, s, A, K, mol, cd
- **SI coherent derived units** [2, table 3 and table 4]
  Derived units are expressed algebraically in terms of base units or other derived units. Dimensionless units are defined in this table as well (rad, sr).
  - rad, sr, Hz, N, Pa, J, W, C, V, F, Ω, S, Wb, T, H, °C, lm, lx, Bq, Gy, Sv, kat
  - Dimensional products of unit symbols are allowed (SI derived units).
- **Units accepted for use with SI** [2, table 6]
  Certain units are widely in use and are accepted for use with the SI.
  - min, h, d, °, ‘, ‘’, ha, L, T, Np, B, dB
- **Non-SI units accepted for use with SI, whose values are obtained experimentically** [2, table 7]
  These are: electrovolt, astronomical unit, unified atomic mass unit, Dalton
  - eV, ua, u, Da
- **Natural and atomic units (used by special interest groups only)** [2, table 7]
  In some cases (basic science) values are expressed in terms of fundamental constants. Those units are accepted by NIST, but not CIPM. The physical quantities are, see Table 8 [2].
  - c_0, $\hbar$, m_e, $\hbar/(m_e c_0^2)$, e, a_0, E_h, $\hbar /E_h$
- **Other non-SI units accepted (allowed but not recommended)** [2, table 8]
  These units are defined in Table 9 [2].
  - Å, b, bar, mmHg, Ci, R, rad, rem

Unit symbols not allowed are:

- **CGS Units**
  - erg, dyn, P, St, Gs, G, Oe, Mx, sb, ph, Gal
- **Not accepted by SI**
  - Fermi, metric carat, Torr, atm, kgf, µ, $cal_{th,}$ xu, st, γ, λ
- **Custom units**
  - Not allowed within the SIUnit definition.
  - Use of scaling for raw data to SI recommended
  - Conversion factors see [3]

Allowed **prefixes** are defined in [2, table 5]:

- da, h, k, M, G, T, P, E, Z, Y
- d, c, m, µ, n, p, f, a, z, y

Note that units are case sensitive.

The SI allows **algebraic operations** with units [2, chapter 5.3.1]. The unit symbol is treated as a mathematical entity. Both a numeric value and unit may be treated by the ordinary rules of algebra. The operations allowed are listed below.

Custom units are not supported. In case the data is represented in custom units, use a scaling function to obtain the numerical value with the SI unit definition. There are an incredible number of custom units. Official scaling factors are supplied in [3].

Schema Identification

Name	Value
`IviSchema`	`"IviUnit"`
`IviSchemaVersion`	`"1.0.0"`

Schema Members

Name	HDF5 Object	Datatype or Schema	Required	Description
SIUnit	Attribute	String	Yes	The SI base or derived unit using units symbol representation as defined in The International System of Units (SI). The unit is case sensitive. UTF-8 strings are required to fully support the SI standard, but are not required if the SI string can be encoded as ASCII. In case no SI unit is given, the default is unitless. In case no SI unit is available (for example $), specify "Undefined" and use DisplayUnit. Scaling defined with explicit or implicit data may be used to achieve a custom unit conversion (in case the data is not stored in SI).
DisplayUnit	Attribute	String	No	The display unit may be used to express additional information not given by the SIUnit attribute. The use of DisplayUnit attribute is useful in case of unitless or for localization or a unit that is not SI
DisplayScale	Group	"IviFunction"	No	The display unit might simply be a translation, so the scale may not be required.

Examples for SI Unit

- **SI base units and coherent derived units**
  m, kg, s, m/s, 1, kg/m^3
- **Coherent derived units in the SI with special names**
  rad, Hz, Pa, Ω, °C, S, F, V, C, W, J
- **Examples of coherent derived units**
  N/m, J/K, V/m, $W/(m^2\,sr)$
- **Examples of prefixes**
  cm^3, μs^{-1}, V/cm, cm^{-1}

- **Dimensionless units**
  rad, sr, 1
- **Non SI units accepted**
  min, h, d, °, L
- **Non SI units accepted in terms of fundamental constants**
  eV, Da, u, ua
- **Other non SI units**
  bar, mmHg, Å
- **Logarithmic units**
  log(mV), dB(mV)

Mathematics with units

The units are stored as unicoded strings. Mathematical operations allowed in unit strings are shown in the next table.

Operation	Notation
Brackets	( )
Multiplication	*
Division	/
Power	^
Logarithmic	log([<SI unit>])
Decibel	dB([<numeric_value>] [<SI unit>]) where <numeric_value> and <SI unit> default to 1. dB() and dBm are not allowed.

To specify a reference for the dB-based unit, use parentheses after dB and specify the base unit inside the parentheses. For example, for dBm use dB(mW) . Refer to the SI specifications for scaling factor. An optional numeric may be included inside the parentheses with the base unit.

Implementation notes

The SCPI or IEEE 488.2 definition of units differ from the SI definitions. The main difference is, that unit symbols and unit scales are not case sensitive or have different short forms:

Examples	**SCPI, IEEE 488.2**	**SI**
Absorbed dose	GY	Gy
Amount of substance	MOL	mol
Electric conductance	SIE	S
Prefix exa	EX	E
Prefix mega	MA	M
Special unit symbol	MHZ	MHz
Special unit symbol	MOHM	MΩ

4 General Datatype Definitions

This section defines datatypes that are general in nature and don't apply to any particular IVI schema

4.1 Timestamp

Description

Timestamp is represented by a 128bit compound data type with signed 64 bits for seconds and unsigned 64 bits for fractional seconds. The epoch is 0 h, 1 January 1900 UTC (as defined in the IETF spec for NTP-4). It includes a 64-bit signed seconds field spanning 584 billion years and a 64-bit fraction field resolving .05attosecond (i.e., 0.5e-18). The sign of the seconds field shall be interpreted as the sign for the entire representation.

The container (dataset/attribute) needs to specify if timestamps are relative or absolute, and if it is relative, it should specify the epoch.

H5T_COMPOUND Datatype Fields

Field Name	Datatype	Description
s	Signed 64-bit integer	Represents the whole number of seconds since the epoch.
f	Unsigned 64-bit integer	Represents the fractional number of seconds since the epoch.

Example

Following is an example Timestamp attribute.

```
ATTRIBUTE "Timestamp" {
   DATATYPE  "/IviTimestampType"
   DATASPACE  SCALAR
   DATA {
   (0): {
         1370894136,
         9223372036854775808
      }
   }
}
DATATYPE "IviTimestampType" H5T_COMPOUND {
   H5T_STD_I64LE "s";
   H5T_STD_U64LE "f";
}
```

4.2 Complex

Description

Complex data, such as '1.2+3.4j' is represented by a compound data type (H5T_COMPOUND) with real and imaginary values of the same datatype. As defined in the table below, both members must have the same numeric atomic datatype class. The name of the first member is "r" and represents the real part. The name of the second member is "i" and represents the imaginary part.

H5T_COMPOUND Datatype Fields

Field Name	Datatype	Description
r	Numeric	Represents the real part of the complex data.
i	Numeric (same as r)	Represents the imaginary part of the complex data.

Example

Following is an example dataset using the Complex datatype defined above.

```
DATASET "Dataset" {
   DATATYPE  H5T_COMPOUND {
      H5T_IEEE_F64LE "r";
      H5T_IEEE_F64LE "i";
   }
   DATASPACE  SIMPLE { ( 1, 2 ) / ( 1, 2 ) }
   DATA {
   (0,0): {
         1,
         1.5
      },
   (0,1): {
         2,
         2.5
      }
   }
}
```

5 Usage Recommendations

5.1 Avoid stripping out information

When reading and then again writing an existing file, avoid stripping out any information currently in the file. For example, vendor-specific information may not be important for another vendor's process, but should be left intact.

5.2 Versioning

5.2.1 Backwards Compatibility

When newer versions of software read in old file formats, any required fields missing due to additions to a schema will be handled by the software providing default values to the client asking for the data. Note: any sort of breaking change to a schema will be handled in the specification by incrementing the major version number for the schema. In the backwards compatibility case, the reading software only needs to handle schemas that existed when it was written.

5.2.2 Forwards Compatibility

When older versions of software read in a newer file format, there are two primary issues to deal with. The first issue is when there are new fields which exist in schemas that existed in the older version of the file format. In this case, the additional fields can be ignored and would be indicated by a minor version number change for the schema. If a new required field is added to a schema, this would be indicated by a major version number change for the schema.

5.3 Minimalist Data Persistence Strategy

Many of the schema elements defined in this specification are not required to be present in the HDF5 file. This is desirable for minimizing the file size.

In cases where elements are not "required", data retrieval code must be able to handle the absence of that particular element. Some non-required schema elements will have a default value defined. If the value is defined, higher level APIs will be expected to supply the default value as if the schema member were present in the file.

Higher level APIs will also be expected to provide a mechanism to determine if any value is present in the underlying HDF5 file. This applies to non-required elements that don't provide a default value as well as those that do provide a default value.

6 Implicit Functions

6.1 Arbitrary

This function produces an arbitrary formula. The format for the formula is stored in the **parser** and **parser_format** attributes. The formula itself is stored in the **func_coef** array. Each element of the array is interpreted as a single ASCII character, so the array represents an ASCII string which defines the formula. All coefficients must be in the formula specification.

6.2 DC

This function produces a single line of slope 0.

Variables	Name	Description
a0	a_0	level

6.3 Ellipse

This function produces a repetitive half-elliptical waveform. The formula is

$$y = off + a\sqrt{1 - \frac{(\mathrm{mod}(360 \cdot f \cdot x + \varphi, 360) - 180)^2}{180^2}}$$

where mod(…) is the modulo function.

Variables	Name	Description
a0	f	Frequency
a1	a	Peak-to-peak amplitude
a2	φ	Phase in degrees
a3	off	Offset

6.4 Exponential

This function produces an exponential ramp. The formula is

$$y = a_2 e^{a_0(x - a_1)} + a_3 .$$

Variables	Name	Description
a0	a_0	Multiplier
a1	a_1	X offset
a2	a_2	Amplitude
a3	a_3	Y offset

6.5 Exponential Repetitive

This function produces a repetitive exponential ramp. The formula is

$$f = cycles / \ell$$

$$a = \frac{A_e - A_s}{e^{1/(f\tau)} - 1}$$

$$c = A_s - a$$

$$y = a \cdot e^{\mathrm{mod}\left(x + \frac{\varphi}{360f}, \frac{1}{f}\right)/\tau} + c$$

where mod(…) is the modulo function and ℓ is the length of the axis.

Variables	Name	Description
a0	cycles	Number of cycles
a1	A_s	Start amplitude of each cycle
a2	A_e	End amplitude of each cycle
a3	φ	Phase in degrees
a4	τ	Time constant

6.6 HalfSineConnector

This function produces a half cycle sine wave between two points. The formula is

$$y = \left(\frac{y_1 - y_0}{2}\right)\left(1 - \cos\left(\frac{\pi}{x_1 - x_0}(x - x_0)\right)\right) + y_0.$$

Variables	Name	Description
a0	x_0	Start X coordinate
a1	y_0	Start Y coordinate
a2	x_1	End X coordinate
a3	y_1	End Y coordinate

6.7 Haversine

This function produces a haversine wave – a sine wave with the base on the X-axis. The formula is

$$y = \frac{a}{2} \cdot \left(1 - \cos\left[2\pi\left(f \cdot x - \frac{\varphi}{360}\right)\right]\right).$$

Variables	Name	Description
a0	f	Frequency
a1	a	Peak-to-peak amplitude.
a2	φ	Phase in degrees

6.8 Log

This function produces a logarithmic ramp. The formula is

$$y = a_1 \ln(x - a_0) + a_2.$$

Variables	Name	Description
a0	a_0	X offset
a1	a_1	Amplitude
a2	a_2	Y offset

6.9 Log Repetitive

This function produces a repetitive logarithmic ramp. The formula is

$$f = cycles / \ell$$

$$a = \frac{A_e - A_s}{\ln(1/f + a_4) - \ln(a_4)}$$

$$c = A_s - a\ln(a_4)$$

$$y = a\ln\left[\mathrm{mod}\left(x + \frac{\varphi}{360f}, \frac{1}{f}\right) + a_4\right] + c$$

where mod(…) is the modulo function and ℓ is the length of the axis.

Variables	Name	Description
a0	cycles	Number of cycles
a1	A_s	Start amplitude of each cycle
a2	A_e	End amplitude of each cycle
a3	φ	Phase in degrees
a4	a_4	Constant

6.10 Noise

This group of functions produce a waveform of noise. The type of noise is determined by the first parameter (a_0). Shared parameters are shown in the **Common Parameters** list. Other parameters are shown under the individual noise types.

6.10.1 Common Parameters

Variables	Name	Description
a0	seed	Seed. This will be truncated to a 32-bit signed integer

6.10.2 Noise - Bernoulli

Generates a pseudorandom pattern of ones and zeroes. Each element of the output pattern is computed by flipping a weighted coin. The weight is the ones probability input. If ones probability is 0.7, then each element of the output pattern has a 70% chance of being one and 30% chance of being zero.

Variables	Name	Description
a1	onesprob	Ones probability is the probability of a given element of Bernoulli noise being true (1).

6.10.3 Noise - Binomial

Generates a binomially distributed pseudorandom pattern whose values are the number of occurrences of an event given the probability of that event occurring and the number of trials.

Variables	Name	Description
a1	trials	Trials is the number of trials performed for each element of binomial noise.
a2	trialprob	Trial probability is the probability that a given trial is true (1).

6.10.4 Noise - Gamma

Generates a pseudorandom pattern of values which are the waiting times to the **order** number event of a unit mean Poisson process

Variables	Name	Description
a1	order	Order is the event number of the unit mean Poisson process. It will be truncated to a signed 32 bit integer.

6.10.5 Noise - Gaussian White

Generates a Gaussian-distributed, pseudorandom pattern whose statistical profile is (μ,sigma) = (0,s), where s is the absolute value of the specified **stdev**

Variables	Name	Description
a1	stdev	Standard deviation of the noise

6.10.6 Noise - Periodic Random

Generates an array of periodic random noise

Variables	Name	Description
a1	specamp	Spectral amplitude of each frequency component of the noise

6.10.7 Noise - Poisson

Generates a pseudorandom sequence of values which are the number of discrete events occurring in a given interval (**mean**) of a unit rate Poisson process.

Variables	Name	Description
a1	mean	Mean is the interval of a unit rate Poisson process.

6.10.8 Noise - Uniform White

A uniformly distributed, pseudorandom pattern whose values are in the range [-a:a], where a is the absolute value of **amplitude**

Variables	Name	Description
a1	amplitude	Amplitude of the noise. The expected standard deviation of the noise is $\text{amplitude}/\sqrt{3}$

6.11 Polynomial

This function generates a polynomial curve. The formula is

$$y = \sum_{i=0}^{n} a_i x^i$$

Variables	Name	Description
a*i*	a_i	Coefficients for the polynomial equation. The number of coefficients determines the order of the equation.

6.12 Pulse - Gaussian

This function produces a single, gaussian profile pulse. The formula is

$$y = a \cdot e^{-\left(\frac{x-d}{\sigma}\right)^2} + off$$

Variables	Name	Description
a0	a	Amplitude of the pulse
a1	d	Delay to center of pulse
a2	σ	Standard deviation of pulse profile
a3	off	Base offset of the pulse

6.13 Pulse - Impulse

This function produces an impulse. The formula is

$$y = \begin{cases} off & \text{if } x \neq d \\ a + off & \text{if } x = d \end{cases}$$

Variables	Name	Description
a0	a	Amplitude of the impulse
a1	d	Delay to impulse
a2	off	Base offset of the impulse

6.14 Pulse - Lorenz

This function produces a single, lorenzian profile pulse. The formula is

$$y = \frac{a}{1 + \left(\frac{2(x-d)}{w}\right)^2} + off$$

Variables	Name	Description
a0	a	Amplitude of the pulse
a1	d	Delay to center of pulse
a2	w	Width of pulse profile at half maximum
a3	off	Base offset of the pulse

6.15 Pulse - Square

This function produces a single flat-topped pulse. The formula is

$$y = \begin{cases} a + off & \text{if } d \le x < (d + w) \\ off & \text{elsewhere} \end{cases}$$

Variables	Name	Description
a0	a	Amplitude of the pulse
a1	d	Pulse delay
a2	w	Pulse width
a3	off	Base offset of the pulse

6.16 Ramp

This function produces a line. The formula is

$$y = \frac{y_1 - y_0}{\ell} x + y_0$$

where ℓ is the length of the axis.

Variables	Name	Description
a0	y_0	Start y value
a1	y_1	End y value

6.17 Sawtooth

This function produces a sawtooth wave. The formula is

$$y = a\left(\frac{\text{mod}(360 \cdot f \cdot x - \varphi, 360)}{180} - 1\right) + off$$

where mod(x,y) is x modulo y. Note that the initial value of the function is –a.

Variables	Name	Description
a0	f	Frequency
a1	a	Amplitude. Peak to peak value is twice the amplitude.
a2	φ	Phase in degrees
a3	off	Offset

6.18 Sinc

This function produces a sinc waveform. The formula is

$$y = a \cdot \text{sinc}\left(\frac{x - x_0}{w}\right) + off$$

where sinc is unnormalized where sinc(X) = sin(X)/(X) and X is in radians.

Variables	Name	Description
a0	x_0	X value at the peak of the function
a1	a	Amplitude
a2	off	Phase in degrees
a3	w	Width of the function

6.19 Sine

This function produces a sine wave. The formula is

$$y = a \cdot \sin\left[2\pi\left(f \cdot x - \frac{\varphi}{360}\right)\right] + off$$

Variables	Name	Description
a0	f	Frequency
a1	a	Amplitude. Peak to peak value is twice the amplitude.
a2	φ	Phase in degrees
a3	off	Offset

6.20 Square

This function produces a square wave. The formula is

$$y = \begin{cases} a + off & \text{if } 0 \le \dfrac{\mathrm{mod}(360 \cdot f \cdot x - \varphi, 360)}{360} < \dfrac{dc}{100} \\ -a + off & \text{elsewhere} \end{cases}$$

where mod(x,y) is x modulo y.

Variables	Name	Description
a0	f	Frequency
a1	a	Amplitude. Peak to peak value is twice the amplitude.
a2	φ	Phase in degrees
a3	off	Offset
a4	dc	Duty cycle in percent

6.21 Stairstep

This function produces a stairstep waveform. The formula is

$$y = a \cdot \mathrm{floor}(f \cdot x + \varphi / 360) + off$$

where floor(x) is x rounded down to the nearest integer.

Variables	Name	Description
a0	f	Frequency of the steps
a1	a	Amplitude change with each step.
a2	φ	Phase in degrees
a3	off	Offset of the function start point

6.22 Sweep

This is a group of functions that sweep the frequency of a repetitive function over a specified interval. Three types of sweeps are defined – linear, exponential, and logarithmic. The sweep type determines how the frequency changes with x-value through the waveform. The initial and final x-values (x_0 and x_1) can be queried or derived from the axis attributes **start** and **increment** and the trace attribute **n_points**. The dependence of frequency on x value is given as follows:

6.22.1.1 Linear Sweep

$$f = f_0 + \left(\frac{f_1 - f_0}{x_1 - x_0}\right)(x - x_0)$$

6.22.1.2 Exponential Sweep

$$f = f_0 e^{\left[\left(\frac{\ln f_0 - \ln f_1}{x_0 - x_1}\right)(x - x_0)\right]}$$

6.22.1.3 Logarithmic Sweep

$$f(x) = \ln\left[\left(\frac{e^{f_1} - e^{f_0}}{x_1 - x_0}\right)(x - x_0) + e^{f_0}\right]$$

To generate the actual sweep, the fx term in the repetitive function must be replaced by $\varphi_e = \int_{x_0}^{x_v} f(x)dx$, where f(x) is given by one of the functions above. The formulae for φ_e:

6.22.1.4 Linear Effective Phase

$$\varphi_e(x) = f_0(x - x_0) + \frac{1}{2}\left(\frac{f_1 - f_0}{x_1 - x_0}\right)(x - x_0)^2$$

6.22.1.5 Exponential Effective Phase

$$\varphi_e(x) = \frac{f_0(x_1 - x_0)}{\ln(f_1) - \ln(f_0)} e^{\left[\left(\frac{\ln(f_1) - \ln(f_0)}{x_1 - x_0}\right)(x - x_0)\right]} - \frac{f_0(x_1 - x_0)}{\ln(f_1) - \ln(f_0)}$$

6.22.1.6 Logarthmic Effective Phase

$$a = \frac{e^{f_1} - e^{f_0}}{x_1 - x_0}$$

$$b = e^{f_0}$$

$$c(x) = a(x - x_0) + b$$

$$\varphi_e(x) = \frac{1}{a}[c(x)\ln(c(x)) - c(x)] - \frac{1}{a}[b\ln(b) - b]$$

6.22.2 Common Parameters

Variables	Name	Description
a0	type	Type of sweep. An enum with the following possibilities: 0 - linear 1 - exponential 2 - logarithmic
a1	f0	Start frequency
a2	f1	End frequency
a3	a	Amplitude. This may be peak to peak or half peak-to-peak. Consult the repetitive function of the same name for details.
a4	φ	Phase in degrees
a5	off	Offset

6.22.3 29 - Sweep – Ellipse

This function produces a swept repetitive elliptical segment wave. The formula is

$$y = off + a\sqrt{1 - \frac{(\mathrm{mod}(360 \cdot \varphi_e(x) + \varphi, 360) - 180)^2}{180^2}}$$

where mod(x,y) is x modulo y.

6.22.4 30 - Sweep – Sawtooth

This function produces a swept sawtooth wave. The formula is

$$y = a\left(\frac{\mathrm{mod}(360 \cdot \varphi_e(x) - \varphi, 360)}{180} - 1\right) + off$$

where mod(x,y) is x modulo y.

6.22.5 31 - Sweep – Sine

This function produces a swept sine wave. The formula is

$$y = a \cdot \sin\left[2\pi\left(\varphi_e(x) - \frac{\varphi}{360}\right)\right] + off$$

6.22.6 32 - Sweep – Square

This function produces a swept square wave. The formula is

$$y = \begin{cases} a + off & \text{if } 0 \le \dfrac{\mathrm{mod}(360 \cdot \varphi_e(x) - \varphi, 360)}{360} < \dfrac{dc}{100} \\ -a + off & \text{elsewhere} \end{cases}$$

where mod(x,y) is x modulo y.

Variables	Name	Description
a6	dc	Duty cycle in percent

6.22.7 33 - Sweep – Triangle

This function produces a swept triangle wave. The formula is

$$p = \mathrm{mod}(360 \cdot \varphi_e(x) + \varphi - 90, 360)$$

$$y = \begin{cases} a\left(\dfrac{p}{90} - 3\right) + off & \text{if } 0 \le p < 180 \\ a\left(1 - \dfrac{p}{90}\right) + off & \text{if } 180 \le p < 360 \end{cases}$$

where mod(x,y) is x modulo y.

6.23 Trapezoid

This function produces a trapezoid segment. The formula is

$$y = \begin{cases} \left(\dfrac{y_1 - y_0}{x_1 - x_0}\right)x + \left(\dfrac{x_1 y_0 - x_0 y_1}{x_1 - x_0}\right) & \text{if } x_0 \le x < x_1 \\ y_1 & \text{if } x_1 \le x < x_2 \\ \left(\dfrac{y_0 - y_1}{x_3 - x_2}\right)x + \left(\dfrac{x_3 y_1 - x_2 y_0}{x_3 - x_2}\right) & \text{if } x_2 \le x < x_3 \end{cases}$$

where y_0 = offset and y_1 = offset + h.

Variables	Name	Description
a0	x_0	Left most point
a1	x_1	Right most point of the initial ramp
a2	x_2	Left most point of the final ramp
a3	x_3	Right most point
a4	h	Height
a5	off	Offset

6.24 Triangle

Description: This function produces a triangle wave. The formula is

$$p = \mathrm{mod}(360 \cdot f \cdot x + \varphi - 90, 360)$$

$$y = \begin{cases} a\left(\frac{p}{90} - 3\right) + off & \text{if } 0 \le p < 180 \\ a\left(1 - \frac{p}{90}\right) + off & \text{if } 180 \le p < 360 \end{cases}$$

where p is the phase modulo 360˚. The 90˚ phase shift in the formula for p reduces the cases for y from three to two.

Variables	Name	Description
a0	f	Frequency
a1	a	Amplitude. Peak to peak value is twice the amplitude.
a2	φ	Phase in degrees
a3	off	Offset