**THE INSTITUTE OF NAVIGATION**

8551 Rixlew Lane, Ste. 360

Manassas, VA 20109-3701

PHN: (703) 366-2723/FAX: (703) 366-2724

[WWW.ION.ORG](http://WWW.ION.ORG)

**Global Navigation Satellite Systems**

**Software Defined Radio**

**Sampled Data**

**Metadata Standard**

Revision 0.2

ION GNSS SDR Standard Working Group

**Abstract**

The GNSS SDR Metadata Standard defines parameters and schema to express the contents of SDR sampled data files. The standard is designed to promote the interoperability of GNSS SDR data collection systems and processors. The standard includes a formal XML schema definition (XSD). A compliant open source C++ applications programming interface (API) is also officially supported to promote ease of integration into existing SDR systems.

**Table of Contents**

[1 Introduction 6](#_Toc489615282)

[2 Scope 6](#_Toc489615284)

[3 Metadata Format 6](#_Toc489615285)

[4 SDR Data Collection Topologies 6](#_Toc489615286)

[4.1 Single Band, Single Stream, Single File / Multiple Files 7](#_Toc489615287)

[4.2 Multi-Band, Single Stream, Single File 7](#_Toc489615288)

[4.3 Multi Stream, Single File 8](#_Toc489615289)

[4.4 Multi Stream, Single File (with Additional Data) 8](#_Toc489615290)

[4.5 Temporal Splitting of Files 8](#_Toc489615291)

[4.6 Spatial Splitting of Files 8](#_Toc489615294)

[4.7 Spatial-Temporal Splitting of Files 9](#_Toc489615295)

[5 Metadata File Naming and Association Mechanisms 9](#_Toc489615296)

[6 Domain Model 9](#_Toc489615297)

[Architecture 10](#_Toc489615298)

[6.1 10](#_Toc489615299)

[6.2 Core classes 11](#_Toc489615301)

[6.2.1 Session object 11](#_Toc489615302)

[6.2.2 System object 12](#_Toc489615303)

[6.2.3 Cluster object 12](#_Toc489615304)

[6.2.4 Source Object 13](#_Toc489615305)

[6.2.5 Band object 14](#_Toc489615306)

[6.2.6 Stream object 14](#_Toc489615307)

[6.2.7 Lump object 17](#_Toc489615309)

[6.2.8 Chunk object 18](#_Toc489615311)

[6.2.9 Block object 19](#_Toc489615313)

[6.2.10 Lane object 20](#_Toc489615314)

[6.2.11 File object 21](#_Toc489615315)

[6.2.12 FileSet Object 22](#_Toc489615316)

[6.3 Foundation Classes 22](#_Toc489615317)

[6.3.1 URI 22](#_Toc489615318)

[6.3.2 DateTime 22](#_Toc489615319)

[6.3.3 Frequency 22](#_Toc489615320)

[6.3.4 Duration 22](#_Toc489615323)

[6.3.5 Position 23](#_Toc489615324)

[6.3.6 Origin 23](#_Toc489615325)

[6.3.7 Orientation 24](#_Toc489615327)

[Encoding Functions 25](#_Toc489615328)

**List of Figures**

[Figure 1 – Fundamental GNSS SDR Data Collection Topologies 7](#_Toc489615329)

[Figure 2 – Overview of Core Metadata Classes and Generation 9](#_Toc489615330)

[Figure 3 – GNSS metadata class model (UML 2.0) 10](#_Toc489615331)

[Figure 4 – Core metadata classes specialize the base metadata element, which has a unique identifier (id), links to related artifacts (URI) and comments 10](#_Toc489615332)

[Figure 5 – Intentional Aliasing of a Multiband signal to Baseband 14](#_Toc489615333)

[Figure 6 – Illustration of Multiple Bands Present in a Stream 15](#_Toc489615334)

[Figure 7 – Illustration of a lump containing samples from N streams 17](#_Toc489615335)

[Figure 8 – Encoding schemes for a single lump within a single chunk 19](#_Toc489615336)

**List of Tables**

[Table 1 – Metadata element class attributes 11](#_Toc489615337)

[Table 2 – Definition of session attributes 11](#_Toc489615338)

[Table 3 – Definition of system attributes 12](#_Toc489615339)

[Table 4 – Definition of cluster attributes 13](#_Toc489615340)

[Table 5 – Definition of source attributes 13](#_Toc489615341)

[Table 6 – Definition of band attributes 14](#_Toc489615342)

[Table 7 – Definition of stream attributes 16](#_Toc489615343)

[Table 8 – Enumeration of stream encoding attribute 16](#_Toc489615344)

[Table 9 – Definition of lump attributes 17](#_Toc489615345)

[Table 10 – Definition of chunk attributes 18](#_Toc489615346)

[Table 11 – Definition of block attributes 20](#_Toc489615347)

[Table 12 – Definition of lane attributes 20](#_Toc489615348)

[Table 13 – Definition of file attributes 21](#_Toc489615349)

[Table 14 – Definition of fileSet attributes 22](#_Toc489615350)

[Table 15 – Definition of position attributes 23](#_Toc489615351)

[Table 16 – Definition of origin attributes 23](#_Toc489615352)

[Table 17 – Definition of orientation attributes 24](#_Toc489615353)

[Table 18 – Encoding of 2-bit samples 25](#_Toc489615354)

[Table 19 – Encoding of 3-bit samples 25](#_Toc489615355)

[Table 20 – Encoding of 4-bit samples 25](#_Toc489615356)

[Table 21 – Encoding of 5-bit samples 26](#_Toc489615357)

**List of acronyms**

|  |  |
| --- | --- |
| GNSS | Global navigation satellite system |
| IF | Intermediate frequency |
| PC | Personal computer |
| UML | Unified model language |
| URI | Universal resource identifier |
| RF  RTC | Radio frequency  Real time clock |
| SDR | Software defined radio |
| XSD | XML schema definition |
| XML | Extensible markup language |

# Introduction

The past several years has seen a proliferation of software defined radio (SDR) data collection systems and processing platforms that are particularly designed for Global Navigation Satellite System (GNSS) receiver applications or those that support GNSS bands. For post-processing, correctly interpreting the GNSS SDR sampled datasets produced or consumed by these systems has historically been a cumbersome and error-prone process. This is because these systems necessarily produce datasets of various formats, the subtleties of which are often lost in translation when communicating between the producer and consumer of these datasets. This specification standardizes the metadata associated with GNSS SDR sampled data files.

# Scope

Datasets containing GNSS SDR samples may also contain other information such as sensor data and data from radio frequency (RF) bands other than GNSS. For non-RF data, this specification supports bypassing this data during reading. For non-GNSS RF bands, only parameters common to GNSS bands are supported.

# Metadata Format

Extensible Markup Language (XML) is used in this standard. The XML schema is specified according to the XML Schema Definition (XSD) standard.

# SDR Data Collection Topologies

This standard is designed to support most current and future GNSS SDR sampled data file formats. These formats stem from the fundamental data collection topologies illustrated in Figure 1. This section describes these topologies.

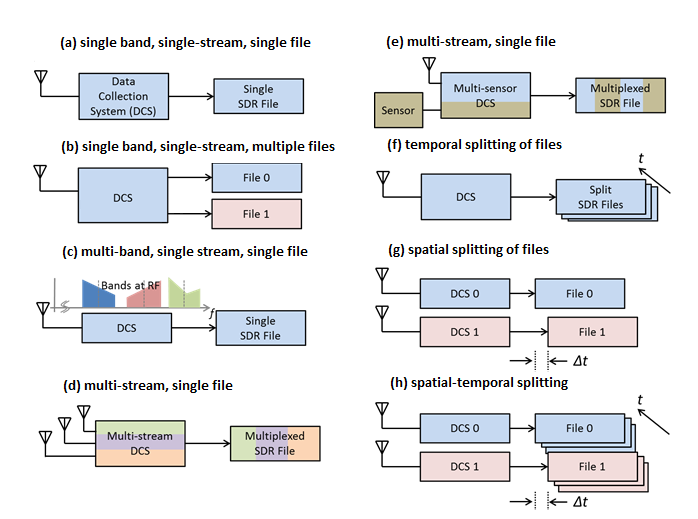


Figure – Fundamental GNSS SDR Data Collection Topologies

## Single Band, Single Stream, Single File / Multiple Files

Figure 1.a illustrates the simplest data collection topology that can exist. This is when a single contiguous region of RF spectrum (referenced henceforth as a ‘band’) is down-converted and sampled to produce a single data stream that is then written to a single data file.

For this and all subsequent topologies, the data stream may contain samples that are either real or complex valued depending on whether intermediate frequency (IF) or baseband sampling is used, respectively. These samples are packed according to a repetitive pattern. The repetitive pattern may also comprise of other information at the beginning and/or end of a fixed number of samples. This may include non-sample data such as headers and footers which, for example, may be used for data integrity check purposes. In this topology, this formatted data stream is written to one and only one file. However, some systems prefer to write the formatted data stream into several data files as shown in Figure 1.b.

## Multi-Band, Single Stream, Single File

Figure 1.c is identical to Figure 1.a in terms of how the data stream may be formed and written to disk, except the data stream contains information from more than one RF band. An example of this topology is a direct RF sampling front-end architecture that intentionally aliases multiple bands such that they appear next to each other at baseband. In this case, some bands may be spectrally inverted as a result of the digital down-conversion process.

## Multi Stream, Single File

Figure 1.d illustrates a topology where multiple sample streams are combined into a single formatted data stream and written to a single file. The formatted data stream may contain additional information as described in 4.1. Each sample stream represents a distinct time series that is independent from any and all others (i.e. independent in a mathematical time series sense, not in a statistical sense).

NOTE:

The distinction of sample stream (i.e. mathematical time series) versus data stream (i.e. formatted data bytes that are ultimately written to disk) is made above. In this standard, the term *stream* shall always imply the former. The term *data stream* shall be used specifically to refer to the latter.

In the example shown, each sample stream represents the data collected from a different antenna whose signal passes through a different RF front-end channel. This is for illustration purposes only. The standard does not assume any dependence between streams (including common sample rates or quantization).

## Multi Stream, Single File (with Additional Data)

Figure 1.e illustrates a data stream containing GNSS samples as well as data from an additional sensor. For the purpose of this standard, any data that cannot be represented as GNSS sample streams are considered unknown data. The standard defines parameters necessary to skip over unknown data bytes when decoding the data stream.

The remaining topologies (Figure 1.f - Figure 1.h) address how a data stream may be written to disk.

## Temporal Splitting of Files

The data rates of GNSS SDR streams are typically high (on the order of one to several hundred MB/sec). Hence, long-duration data collections can generate very large files that become cumbersome to manage. For this reason, the data may be written to smaller sets of files (illustrated in Figure 1.f) where the data stream continues from the end of one file to the beginning of another (possibly with some overlap to ensure data integrity). This is defined as *temporal file splitting* in this standard. The standard includes parameters that specify the order of temporally split files.

## Spatial Splitting of Files

A collection system or setup may write individual data streams or the frequency bands to multiple files (illustrated in Figure 1.g). These files may be written within the same host system (such as a personal computer (PC)) or multiple systems. This is defined as *spatial file splitting* in this standard.

NOTE:

This standard associates two or more spatially split files in a specification defined as *fileSet*.

## Spatial-Temporal Splitting of Files

Figure 1.h illustrates the combination of spatial and temporal splitting. In this case, the fileSet parameter refers to the first of each temporally split file.

# Metadata File Naming and Association Mechanisms

The official filename extension for a metadata file is ‘.sdrx’. Use of this extension is recommended.

# Domain Model

As illustrated in Figure 2, metadata are defined in terms of 12 core classes, represented in the orange boxes. These core classes are explained in the different subsections within this chapter.

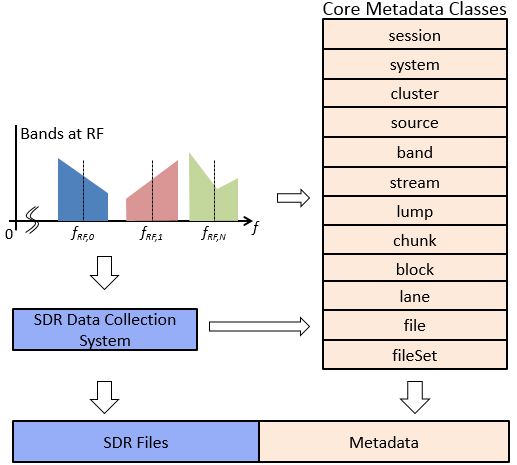


Figure – Overview of Core Metadata Classes and Generation

## Architecture

Figure 3 and Figure 4 show the relation of the different core classes. This relation is shown between the different core classes (Figure 3), and between the core classes and the main class (Figure 4), namely *metadata*.

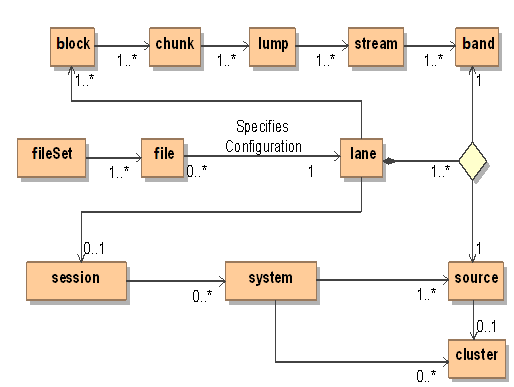


Figure – GNSS metadata class model (UML 2.0)

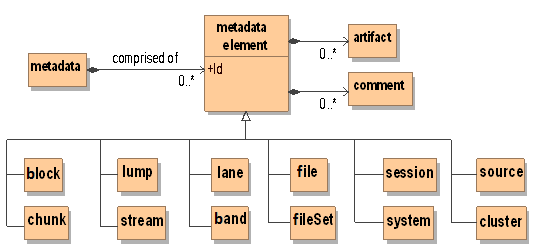


Figure – Core metadata classes specialize the base metadata element, which has a unique identifier (id), links to related artifacts (URI) and comments

All metadata objects contain the following attributes:

* **artifact**: One or more generic attributes
* **comment**: one or more comment strings
* **id**: an identification string that is used to reference a child object by the parent

Table 1describes the attributes of the metadata element class. Core metadata classes specialize the base metadata element. It encapsulates a unique identifier (id), links to related artifacts (URI) and comment strings.

Table 1 – Metadata element class attributes

| Attribute | Description | Class | Enumeration | Required | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| id | Unique identifier | string |  | Yes | “” |
| artifact | Zero or more link specifications to information pertaining to the class instance. Can be any URI formatted information | URI |  | Yes |  |
| comment | Zero or more text/html comments providing additional detail regarding the class instance. | string |  | Yes |  |

## Core classes

### Session object

A session is defined as a utilization instance of a pre-configured system for a period devoted to a particular activity.

Table 2 – Definition of session attributes

| Attribute | Description | Class | Enumeration | Required | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| toa | Time of applicability for all position and attitude parameters | dateTime1 |  | No2 |  |
| position | Platform position at TOA expressed in Geoid frame | position |  | No2 |  |
| system | The system used for this session | system |  | No |  |
| poc2 | Point of contact. Name of person or entity. | string |  | No |  |
| contact2 | POC contact information (email) | string |  | No |  |
| campaign2 | Data collection campaign | string |  | No |  |
| scenario2 | Specific scenario for this collection | string |  | No |  |

1 <https://www.w3schools.com/xml/schema_dtypes_date.asp>

2 Multiple instances of these parameters may exist. The parser shall enumerate accordingly (e.g. poc1, poc2, etc.).

### System object

A system is defined as a complete data collection apparatus. The system comprises all antennas, sensors, and other information-outputting equipment down to the disk arrays that store SDR files. The system may also include GNSS signal simulators. The standard includes geometrical parameters (position and orientation) to the extent that this information is necessary for post-processing the SDR data stream. For example, initial position and platform orientation may be needed for a dynamic scenario. The relative position and orientation of antennas and their elements with respect to the platform coordinate frame are needed for adaptive antenna signal processing.

Table 3 – Definition of system attributes

| Attribute | Description | Class | Enumeration | Required | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| source | One or more sources of sampled data. | source |  | No |  |
| cluster | Zero or more clusters of antenna sources | cluster |  | No |  |
| freqbase | Base frequency. All frequencies are specified as an integer multiple of  freqbase | frequency |  | Yes |  |
| equipment | Equipment used for this data collection | string |  | No |  |

### Cluster object

Data collection setups may contain one or more antenna units where each antenna unit may comprise one or more elements. The position and orientation of each element’s phase center and the relative delay must be known in order to perform multi-element signal processing. Hence, it is convenient to include these parameters directly as metadata. The standard defines the generic terms *cluster* and *source* to refer to an antenna unit and its elements respectively.

A cluster is defined as a grouping of sources. A coordinate frame is associated with a cluster. The origin and orientation of this frame is specified with respect to the platform coordinate frame.

Table 4 – Definition of cluster attributes

| Attribute | Description | Class | Enumeration | Required | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| id | Unique identifier | string |  | No |  |
| position | Origin of cluster reference frame w.r.t. platform coordinate frame | position |  | No |  |
| orientation | Orientation of cluster frame w.r.t. platform frame | orientation |  | No |  |
| vendor | Vendor name | string |  | No |  |
| model | Model number | string |  | No |  |
| serial | Serial number | string |  | No |  |

### Source Object

A source is defined as the originator of an electrical signal. A coordinate frame is associated with a cluster. The origin and rotation of this frame is specified with respect to the platform coordinate frame.

Table 5 – Definition of source attributes

| Attribute | Description | Class | Enumeration | Required | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| idcluster | cluster that this source belongs to | string |  | No |  |
| type | Electrical type of this source | enumerator | “UndefinedType”,  "Patch", "Dipole", "Helical", "Quadrifilar", "Simulator", “Other” | No | “UndefinedType” |
| polarization | Element polarization | enumerator | “UndefinedType”,  "RHCP", "LHCP", "Linear", "Horizontal", "Vertical" | No | “UndefinedType” |
| origin | Origin with respect to cluster | position |  | No |  |
| orientation | Orientation of normal vector to this source plane w.r.t. cluster | orientation |  | No |  |

### Band object

A band is defined as a span of RF spectrum. Each band is received from a single source and converted to a sample stream by a signal processor that is typically referred to as an RF front-end. This analog signal represented by the band experiences the following changes as it passes through this mixed-signal processing chain:

* The RF center frequency, FRF, is translated to FIF
* The spectrum may become inverted such that the frequency FRF+dF is translated to FIF-dF, where dF is a frequency offset from FRF.
* The sampled representation of the band is delayed with respect to the signal incident at the phase center of the source (i.e. antenna element). This delay may vary with time, and is hence defined at the system time of applicability, toa.
* An approximate double-sided half power bandwidth can be specified for the stream representation of the band.

The above are specified in terms of band attributes.

Table 6 – Definition of band attributes

| Attribute | Description | Class | Enumeration | Required | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| centerfreq | Center frequency of band incident at source | frequency |  | Yes |  |
| translatedfreq | Translated center frequency of band | frequency |  | Yes |  |
| inverted | Binary flag indicating spectral inversion | boolean | “true”, ”false” | No | ”false” |
| delaybias | Delay of band measured from source to sampled stream, specified at toa. | duration |  | No | 0 |
| bandwidth1 | Approximate double-sided half power bandwidth | frequency |  | No |  |

1 Bandwidth is measured by processing the sample stream. For streams containing multiple bands, it is recommended that other bands be muted to measure a given bandwidth.

### Stream object

A frequency-translated signal may contain more than one band. For example, in a direct RF sampling front-end, the sample rate may be chosen such that multiple passbands are intentionally aliased to fall adjacent to one another in the spectrum of the sampled signal. This is illustrated in Figure 5.

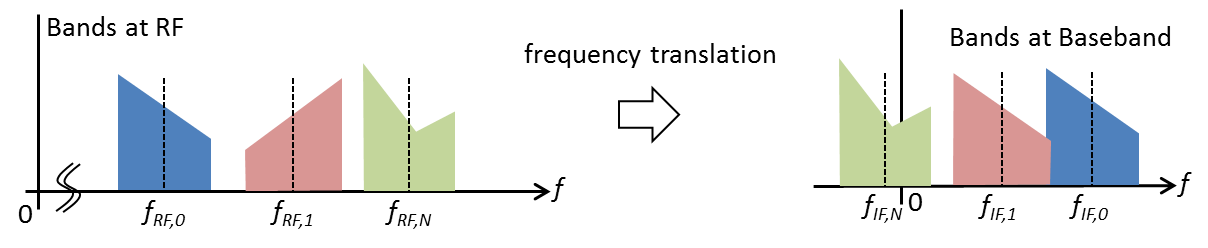


Figure – Intentional Aliasing of a Multiband signal to Baseband

Figure 6 illustrates the conceptual representation of the digitization of a signal containing multiple bands. The output of this process is a sampled representation of the multi-band signal referred to as a sample stream.

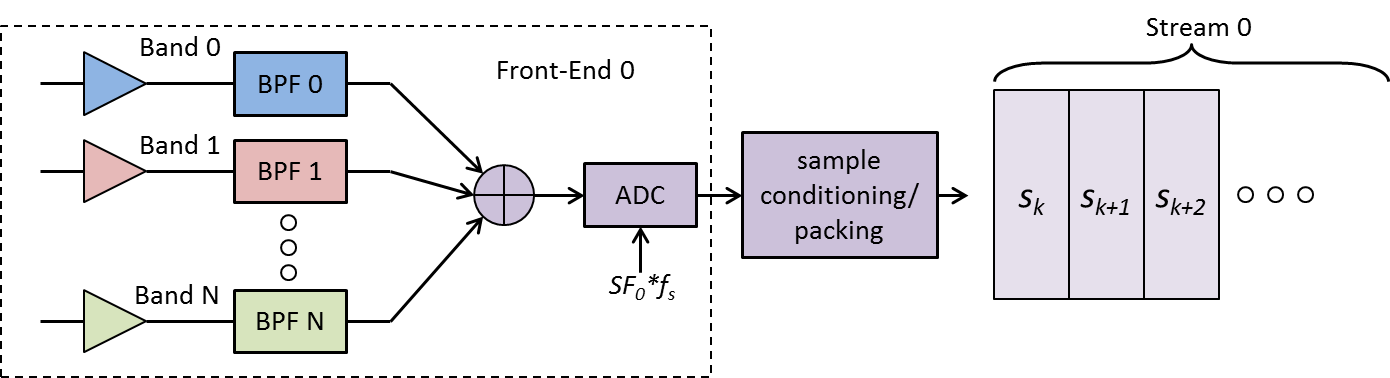


Figure – Illustration of Multiple Bands Present in a Stream

A (sample) stream is defined as a discrete-time discrete-amplitude series that is the sampled representation of a combination of one or more bands.

A stream has the following properties:

* The stream contains the sampled representation of one or more bands.
* An stream is sampled at a given sample rate. This sample rate may be different to other streams in the system. The sample rate of a stream is specified as an integer multiple (ratefactor) of the system base sample rate (freqbase). As such, freqbase should represent the highest common integer factor of the sample rates of all streams.
* Sample values may be real or complex depending on whether IF sampling or baseband sampling is used, respectively. Some or all the numerical values expressed in the stream may be inverted.
* Each sample value is represented by one or more bits which may be encoded using various established schemes. The value quantization should reflect the number of bits required to express all quantization levels, being rounded up when the number of quantization levels is not a power of two (i.e. three-level quantization requires two bits).
* The value packedbits represents the total number of bits occupied by the collection of samples contained in a chunk (the chunk is a segment of data packed in one of the unsigned integer standards, a more accurate description of the chunk is given in section 6.2.8) in the stream where:

packedbits ≥ ratefactor × quantization,

for real data, and:

packedbits ≥ 2 × ratefactor × quantization,

for complex data.

* When inequality holds, the alignment of the quantized samples with respect to the packed samples must be known in order to interpret the sample values correctly.

The above are specified in terms of stream attributes.

Table 7 – Definition of stream attributes

| Attribute | Description | | Class | | Enumeration | Required | | | Default (if not specified) |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| band1 | One or more bands present in this stream | | band | |  | Yes | | |  |
| ratefactor | | Sample rate factor | uint16\_t | |  | | Yes |  | |
| quantization | | Sample quantization (bits) | uint8\_t |  | | Yes | | |  |
| packedbits | | Packed representation (bits) | uint8\_t |  | | Yes | | |  |
| alignment | Sample alignment | | enumerator | | "Left","Right", “Undefined” | Yes | | |  |
| shift | Shift direction | | Enumerator | | "Left","Right", “Undefined” | Yes | | |  |
| format | Sample representation | | enumerator | | "IF","IFn","IQ","IQn","InQ", InQn","QI","QIn","QnI",  "QnIn"  (where `n’ signifies inversion) | Yes | | |  |
| encoding | Numeric encoding scheme | | enumerator | | See Table 8 | Yes | | |  |

1 Multiple instances of these parameters may exist. The parser shall enumerate accordingly



Table 8 – Enumeration of stream encoding attribute

|  |  |
| --- | --- |
| XML String | Description |
| “SIGN” | Sign |
| “OB” | Offset-Binary |
| “SM” | Sign-Magnitude |
| “TC” | Two's Compliment |
| “OG” | Offset-Gray Code |
| “OBA” | Offset-Binary Adjusted |
| “SMA” | Sign-Magnitude Adjusted |
| “TCA” | Two's Compliment Adjusted |
| “OGA” | Offset-Gray Code Adjusted |

### Lump object

Samples from two or more sample streams may be time multiplexed to form a single data stream that is ultimately written to disk (after additional formatting is applied, as described later in this document). This standard assumes that all samples belonging to a finite interval of time are packed into a contiguous grouping of bits, known as a lump.

A lump is defined as the ordered containment of all samples occurring within an interval ts=1/fs. As more than one sample from each stream may exist within a given lump, the variable *shift* indicates which sample is chronologically first. When *shift* is set to “Left” the samples located at the most significant bits are the earliest, and when it is set to “Right” the samples located at the least significant bits are the earliest.

Figure 7 illustrates a lump containing all samples from N samplestreams.

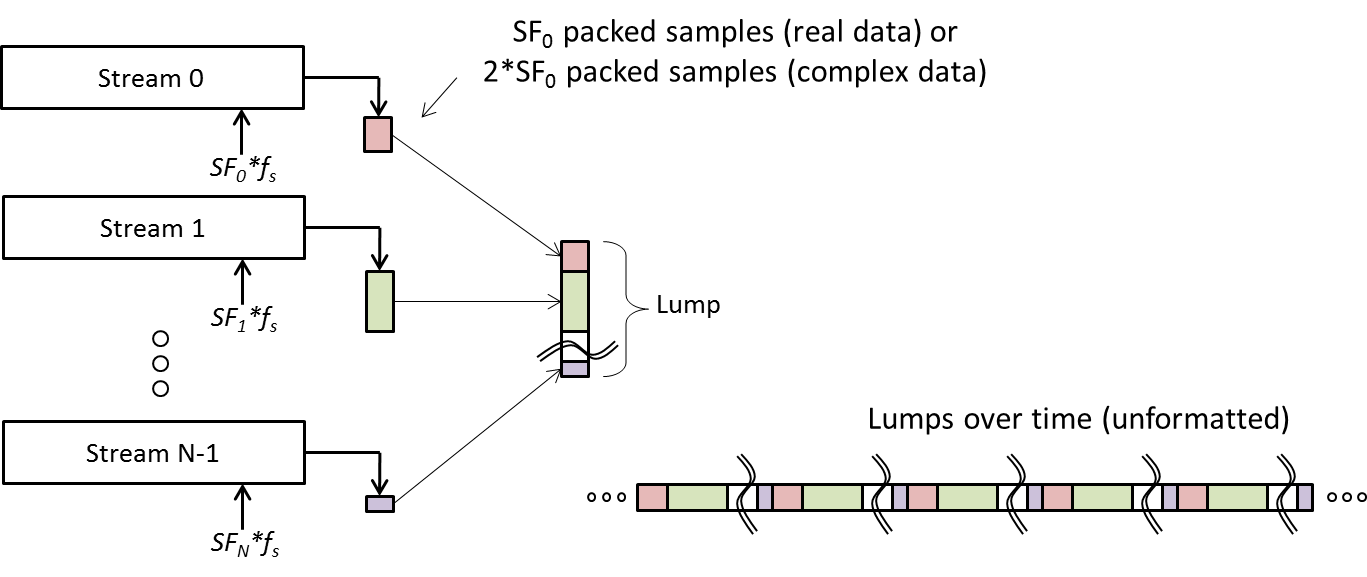


Figure – Illustration of a lump containing samples from N streams

Table 9 – Definition of lump attributes

| Attribute | | Description | | Class | Enumeration | | Required | Default (if not specified) | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| stream | | One or more Streams present in this lump (ordered). | | stream |  | | No |  | |
| shift | Shift direction. | | enumerator | | | “Left”, “Right” | No | |  |

### Chunk object

The packing scheme of samples in a data stream must be known to correctly decode them. For example, consider 32 1-bit real samples packed into two uint16\_t words represented in little-endian format. Due to the little-endian representation, these samples will be decoded incorrectly if read back as a single uint32\_t word and shifted out. Further, some systems pack samples from left to right within a word whereas others perform the opposite.

This standard defines a metadata parameter known as a chunk that together with stream and lump parameters unambiguously describes how samples shall be decoded from a data stream.

A chunk is defined as a segment of data consisting of one or more lumps that have been packed using one of four standard unsigned integer data types (uint8, uint16, uint32, or uint64). This provides a means of describing the occupied memory in a manner that can be natively manipulated by a processor, using standard memory structures (char, int, array).

Table 10 – Definition of chunk attributes

| Attribute | Description | Class | Enumeration | Required | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| lump | One or more lumps | lump |  | Yes |  |
| sizeword | The size, in bytes, of the fundamental integer data-type (word) that shall be read. | uint8\_t | 1, 2, 4, 8 (Corresponds to uint8\_t, uint16\_t, uint32\_t and uint64\_t) | Yes |  |
| countwords | Total number of words to be read in order to read/decode this chunk | uint8\_t |  | Yes |  |
| endian | Endianness of words stored in chunk | enumerator | "Big","Little", “Undefined”‘ | No | “Little” |
| padding | Padding applied during encoding | Enumerator | "None","Head","Tail" | No | “None" |
| wordshift | Shift direction | Enumerator | “Left”, “Right” | yes |  |

Figure 8 illustrates four different schemes where a single 7-bit lump may be encoded within a chunk. The number of bits of information contained within a lump (and hence the number of bits to discard while decoding a chunk – shown as whitespace) is determined implicitly by parsing the referenced lump and stream parameters.

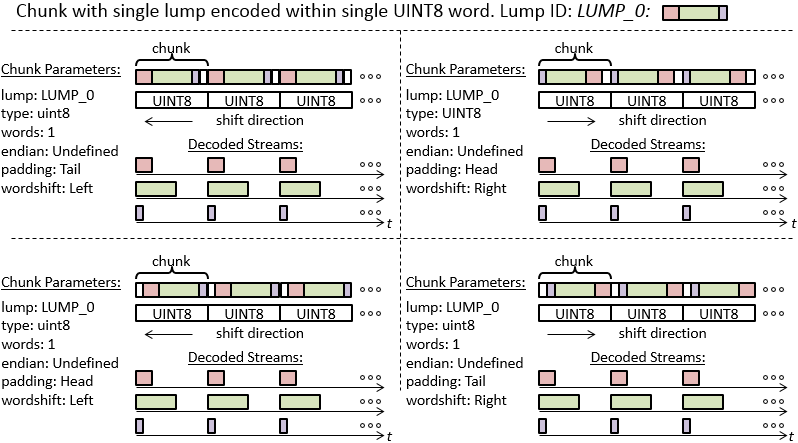


Figure – Encoding schemes for a single lump within a single chunk

### Block object

A data stream may contain other undefined bytes of information. This standard includes parameters necessary to skip over these bytes while decoding sample streams. This information is contained within a metadata object referred to as a block.

A block has the following properties:

* A block is comprised of a finite integer number of chunks greater than zero
* Chunks within a block are sequential and contiguous
* A block may begin with a data segment of arbitrary size (integer number of bytes) known as a *header*.
* A block may end with a data segment of arbitrary size (integer number of bytes) known as a *footer*.
* A block may contain data integrity features that are implemented within the header and/or footer segments.
* The block data structure shall remain constant for the entire data collection session (i.e. block format shall not change dynamically).

A block is defined as a data segment comprised of one or more chunks, where the chunk data appears contiguously anywhere within said segment.

Table 11 – Definition of block attributes

| Attribute | Description | Class | Enumeration | | Required | | Default (if not specified) |
| --- | --- | --- | --- | --- | --- | --- | --- |
| chunk | One or more chunks | chunk |  | | Yes | |  |
| cycles | For the ordered chunk pattern described in the attribute chunk, the integer number of cycles that this pattern repeats within a block | uint32\_t |  | | Yes | |  |
| sizeheader | Integer number of bytes to skip in order to access first byte of chunk data | uint32\_t |  | No | | 0 | |
| sizefooter | Integer number of bytes to skip in order to access first byte of next block | uint32\_t |  | No | | 0 | |

### Lane object

A lane is defined as a conduit that transports data comprised of one or more types of blocks. The contents of one or more lanes are written to disk to produce files. However, the standard does not assume that this writing is synchronized to the start of a block within a lane.

Table 12 – Definition of lane attributes

| Attribute | Description | Class | Enumeration | Required | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| block | One or types of blocks in this lane (in order) | block |  | Yes |  |
| bandsrc | Associates predefined bands with sources | string |  | Yes |  |
| session | Session information for this lane | session |  | Yes |  |
| system | System information for this lane | system |  | Yes |  |

### File object

A file is defined as the ordered collection of bytes retrieved from a single lane over a finite interval of time and stored in a digital media device.

When a lane is written to a file, it may or may not be synchronized to the start of a block. For this reason there may be a byte offset from the beginning of the file to the first byte of the first block. This offset may be different for each file.

The creation time of the file may be tagged as metadata. This time is typically obtained from the system RTC.

Table 13 – Definition of file attributes

| Attribute | Description | Class | Enumeration | Required | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| url | Unique identifier for the file (path/filename)? | URI |  | Yes |  |
| timestamp | Time the file was generated | dateTime |  | No |  |
| offset | Byte offset to start of first Block | uint32\_t |  | No | 0 |
| lane | Identifies which lane the data came from | lane |  | Yes |  |
| previous | Name of previous file (for temporally split files) | URI |  | No |  |
| next | Name of next file (for temporally split files) | URI |  | No |  |
| owner | String specifying owner of this file | string |  | No |  |
| copyright | Copyright information | string |  | No |  |

### FileSet Object

For spatially and spatial-temporally split files, the file set must be identified. This is done by the FileSet parameters that identify the *first set of files*. All other information can be obtained by parsing the metadata of those files.

Table 14 – Definition of fileSet attributes

| Attribute | Description | Class | Enumeration | Requires | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| file | Names of files comprising the file set | URI |  | Yes, for spatial or spatial-temporal |  |

## Foundation Classes

The domain object model foundation classes define basic types used by the core metadata elements.

### URI

A Universal Resource Identifier (URI) defines a unique path (e.g. URL) for locating an associated resource. The URI type is used to enable specification in a XML compatible format.

### DateTime

The dateTime string specifies the day of the year and the time in standard XML format. See <https://www.w3schools.com/xml/schema_dtypes_date.asp>

An example of the representation of this type of parameter is shown below, as a definition of the time of applicability (toa):

<toa>2014-12-30T22:38:54.905999999Z</toa>

In the example is possible to check the day of applicability (30th of December of 2014) and the hour of applicability (22:38:54.905999999).

### Frequency

Specifies frequency. Units can be Hz, kHz, MHz, or GHz. Format can be double or a ratio of the form ‘xxxx.yyyy’ where frequency = xxxx/yyyy where xxxx and yyyy are signed and unsigned 32-bit integers, respectively.

An example of the representation of this type of parameter is shown below, as a definition of the centerfreq parameter:

<centerfreq format=”Hz”>1227600000e+000</centerfreq>

In the example is possible to see that the center frequency of the received signal is 1227600000Hz, or 1,2276GHz.

### Duration

Used for specifying an interval of time. Units include ns, us, ms, sec. Format is double. An example of the representation of this parameter type is shown:

<delaybias format=”sec”>0.0000000000000000e+000</delaybias>

### Position

The position attribute is used to specify the location of the platform with respect to the Geoid. For a dynamic scenario, this is typically the initial location.

Table 15 – Definition of position attributes

| Attribute | Description | Class | Enumeration | Required | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| datum | Datum used for the Geoid | string | “WGS-84” | No | “WGS-84” |
| lat | The latitude coordinate of the position | double |  | Yes |  |
| lon | The longitude coordinate of the position | double |  | Yes |  |
| height | The height coordinate of the position | double |  | Yes |  |

An example of the position is shown:

<position lat=”48.17154012” lon=”11.80868949” height=”576.860”/>

### Origin

Represents the origin of a child reference frame with respect to the parent reference frame.

Table 16 – Definition of origin attributes

| Attribute | Description | Class | Enumeration | Required | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| lat | The latitude coordinate of the origin | double |  | Yes |  |
| lon | The longitude coordinate of the origin | double |  | Yes |  |
| height | The height coordinate of the origin | double |  | Yes |  |

### 

### Orientation

Orientation defines a rotation from a parent coordinate frame to a child frame (i.e. this frame). By default, the rotation is specified in terms of a [3 x 1] set of Euler angles. Other forms are supported by enumerating the class attribute (if it exists).

Table 17 – Definition of orientation attributes

| Attribute | Description | Class | Enumeration | Required | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| type | Type of rotation matrix used. | enumeration | “Euler” | No | “Euler” |
| value | Rotation values, in degrees (0,360) | double (x3) |  | Yes |  |



Appendix:

## Encoding Functions

Below are examples of each of the sample encoding schemes which can be specified in the Stream attributed `encoding’ for a selection of bit widths including 2, 3, 4, and 5-bit digitization. The first column, entitled `Binary’ represents the binary data packed in the stream, MSB left, while the remaining columns represent the physical amplitude of the sample.

Table 18 – Encoding of 2-bit samples

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Binary** | **OB** | **OBA** | **SM** | **SMA** | **TC** | **TCA** | **OG** | **OGA** |
| 00 | -2 | -3 | 0 | 1 | 0 | 1 | -2 | -3 |
| 01 | -1 | -1 | 1 | 3 | 1 | 3 | -1 | -1 |
| 10 | 0 | 1 | 0 | -1 | -2 | -3 | 1 | 3 |
| 11 | 1 | 3 | -1 | -3 | -1 | -1 | 0 | 1 |

Table 19 – Encoding of 3-bit samples

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Binary** | **OB** | **OBA** | **SM** | **SMA** | **TC** | **TCA** | **OG** | **OGA** |
| 000 | -4 | -7 | 0 | 1 | 0 | 1 | -4 | -7 |
| 001 | -3 | -5 | 1 | 3 | 1 | 3 | -3 | -5 |
| 010 | -2 | -3 | 2 | 5 | 2 | 5 | -1 | -1 |
| 011 | -1 | -1 | 3 | 7 | 3 | 7 | -2 | -3 |
| 100 | 0 | 1 | 0 | -1 | -4 | -7 | 3 | 7 |
| 101 | 1 | 3 | -1 | -3 | -3 | -5 | 2 | 5 |
| 110 | 2 | 5 | -2 | -5 | -2 | -3 | 0 | 1 |
| 111 | 3 | 7 | -3 | -7 | -1 | -1 | 1 | 3 |

Table 20 – Encoding of 4-bit samples

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Binary** | **OB** | **OBA** | **SM** | **SMA** | **TC** | **TCA** | **OG** | **OGA** |
| 0000 | -8 | -15 | 0 | 1 | 0 | 1 | -8 | -15 |
| 0001 | -7 | -13 | 1 | 3 | 1 | 3 | -7 | -13 |
| 0010 | -6 | -11 | 2 | 5 | 2 | 5 | -5 | -9 |
| 0011 | -5 | -9 | 3 | 7 | 3 | 7 | -6 | -11 |
| 0100 | -4 | -7 | 4 | 9 | 4 | 9 | -1 | -1 |
| 0101 | -3 | -5 | 5 | 11 | 5 | 11 | -2 | -3 |
| 0110 | -2 | -3 | 6 | 13 | 6 | 13 | -4 | -7 |
| 0111 | -1 | -1 | 7 | 15 | 7 | 15 | -3 | -5 |
| 1000 | 0 | 1 | 0 | -1 | -8 | -15 | 7 | 15 |
| 1001 | 1 | 3 | -1 | -3 | -7 | -13 | 6 | 13 |
| 1010 | 2 | 5 | -2 | -5 | -6 | -11 | 4 | 9 |
| 1011 | 3 | 7 | -3 | -7 | -5 | -9 | 5 | 11 |
| 1100 | 4 | 9 | -4 | -9 | -4 | -7 | 0 | 1 |
| 1101 | 5 | 11 | -5 | -11 | -3 | -5 | 1 | 3 |
| 1110 | 6 | 13 | -6 | -13 | -2 | -3 | 3 | 7 |
| 1111 | 7 | 15 | -7 | -15 | -1 | -1 | 2 | 5 |

Table 21 – Encoding of 5-bit samples

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Binary** | **OB** | **OBA** | **SM** | **SMA** | **TC** | **TCA** | **OG** | **OGA** |
| 00000 | -16 | -31 | 0 | 1 | 0 | 1 | -16 | -31 |
| 00001 | -15 | -29 | 1 | 3 | 1 | 3 | -15 | -29 |
| 00010 | -14 | -27 | 2 | 5 | 2 | 5 | -13 | -25 |
| 00011 | -13 | -25 | 3 | 7 | 3 | 7 | -14 | -27 |
| 00100 | -12 | -23 | 4 | 9 | 4 | 9 | -9 | -17 |
| 00101 | -11 | -21 | 5 | 11 | 5 | 11 | -10 | -19 |
| 00110 | -10 | -19 | 6 | 13 | 6 | 13 | -12 | -23 |
| 00111 | -9 | -17 | 7 | 15 | 7 | 15 | -11 | -21 |
| 01000 | -8 | -15 | 8 | 17 | 8 | 17 | -1 | -1 |
| 01001 | -7 | -13 | 9 | 19 | 9 | 19 | -2 | -3 |
| 01010 | -6 | -11 | 10 | 21 | 10 | 21 | -4 | -7 |
| 01011 | -5 | -9 | 11 | 23 | 11 | 23 | -3 | -5 |
| 01100 | -4 | -7 | 12 | 25 | 12 | 25 | -8 | -15 |
| 01101 | -3 | -5 | 13 | 27 | 13 | 27 | -7 | -13 |
| 01110 | -2 | -3 | 14 | 29 | 14 | 29 | -5 | -9 |
| 01111 | -1 | -1 | 15 | 31 | 15 | 31 | -6 | -11 |
| 10000 | 0 | 1 | 0 | -1 | -16 | -31 | 15 | 31 |
| 10001 | 1 | 3 | -1 | -3 | -15 | -29 | 14 | 29 |
| 10010 | 2 | 5 | -2 | -5 | -14 | -27 | 12 | 25 |
| 10011 | 3 | 7 | -3 | -7 | -13 | -25 | 13 | 27 |
| 10100 | 4 | 9 | -4 | -9 | -12 | -23 | 8 | 17 |
| 10101 | 5 | 11 | -5 | -11 | -11 | -21 | 9 | 19 |
| 10110 | 6 | 13 | -6 | -13 | -10 | -19 | 11 | 23 |
| 10111 | 7 | 15 | -7 | -15 | -9 | -17 | 10 | 21 |
| 11000 | 8 | 17 | -8 | -17 | -8 | -15 | 0 | 1 |
| 11001 | 9 | 19 | -9 | -19 | -7 | -13 | 1 | 3 |
| 11010 | 10 | 21 | -10 | -21 | -6 | -11 | 3 | 7 |
| 11011 | 11 | 23 | -11 | -23 | -5 | -9 | 2 | 5 |
| 11100 | 12 | 25 | -12 | -25 | -4 | -7 | 7 | 15 |
| 11101 | 13 | 27 | -13 | -27 | -3 | -5 | 6 | 13 |
| 11110 | 14 | 29 | -14 | -29 | -2 | -3 | 4 | 9 |