**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 0.1 (Initial Draft)

ION GNSS SDR Standard Working Group

**Abstract**

The GNSS SDR Metadata Standard defines parameters and schema to fully 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 fully compliant open source C++ API is also officially supported to promote ease of integration into existing SDR systems.

**Table of Contents**

[1 Introduction 3](#_Toc409938059)

[2 Scope 3](#_Toc409938060)

[3 Metadata Format 3](#_Toc409938061)

[4 SDR Data Collection Topologies 3](#_Toc409938062)

[4.1 Single Band, Single Stream, Single File 4](#_Toc409938063)

[4.2 Multi-Band, Single Stream, Single File 4](#_Toc409938064)

[4.3 Multi Stream, Single File 5](#_Toc409938065)

[4.4 Multi Stream, Single File (with Additional Data) 5](#_Toc409938066)

[4.5 Temporal Splitting of Files 5](#_Toc409938067)

[4.6 Spatial Splitting of Files 5](#_Toc409938068)

[4.7 Spatial-Temporal Splitting of Files 6](#_Toc409938069)

[5 Metadata File Naming and Association Mechanisms 6](#_Toc409938070)

[6 Domain Model 6](#_Toc409938071)

[6.1 Architecture 7](#_Toc409938072)

[6.2 Core Classes 9](#_Toc409938073)

[6.2.1 Session Object 9](#_Toc409938074)

[6.2.2 System Object 10](#_Toc409938075)

[6.2.3 Cluster Object 11](#_Toc409938076)

[6.2.4 Source Object 12](#_Toc409938077)

[6.2.5 Band Object 13](#_Toc409938078)

[6.2.6 Stream Object 14](#_Toc409938079)

[6.2.7 Lump Object 17](#_Toc409938080)

[6.2.8 Chunk Object 18](#_Toc409938081)

[6.2.9 Block Object 20](#_Toc409938082)

[6.2.10 Lane Object 21](#_Toc409938083)

[File Object 22](#_Toc409938084)

[6.2.11 FileSet Object 23](#_Toc409938085)

[6.3 Foundation Classes 24](#_Toc409938086)

[6.3.1 URI 24](#_Toc409938087)

[6.3.2 DateTime 24](#_Toc409938088)

[6.3.3 Frequency 24](#_Toc409938089)

[6.3.4 Duration 24](#_Toc409938090)

[6.3.5 Location 24](#_Toc409938091)

[6.3.6 Origin 25](#_Toc409938092)

[6.3.7 Orientation 25](#_Toc409938093)

[7 Working Group Membership 26](#_Toc409938094)

# 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 data from other sensors and data from radio frequency (RF) bands other than GNSS bands. For non-RF data, this specification includes information needed to bypass this data during reading. For non-GNSS RF bands, only parameters common to GNSS bands are supported by this standard.

# 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 (if not all) 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.

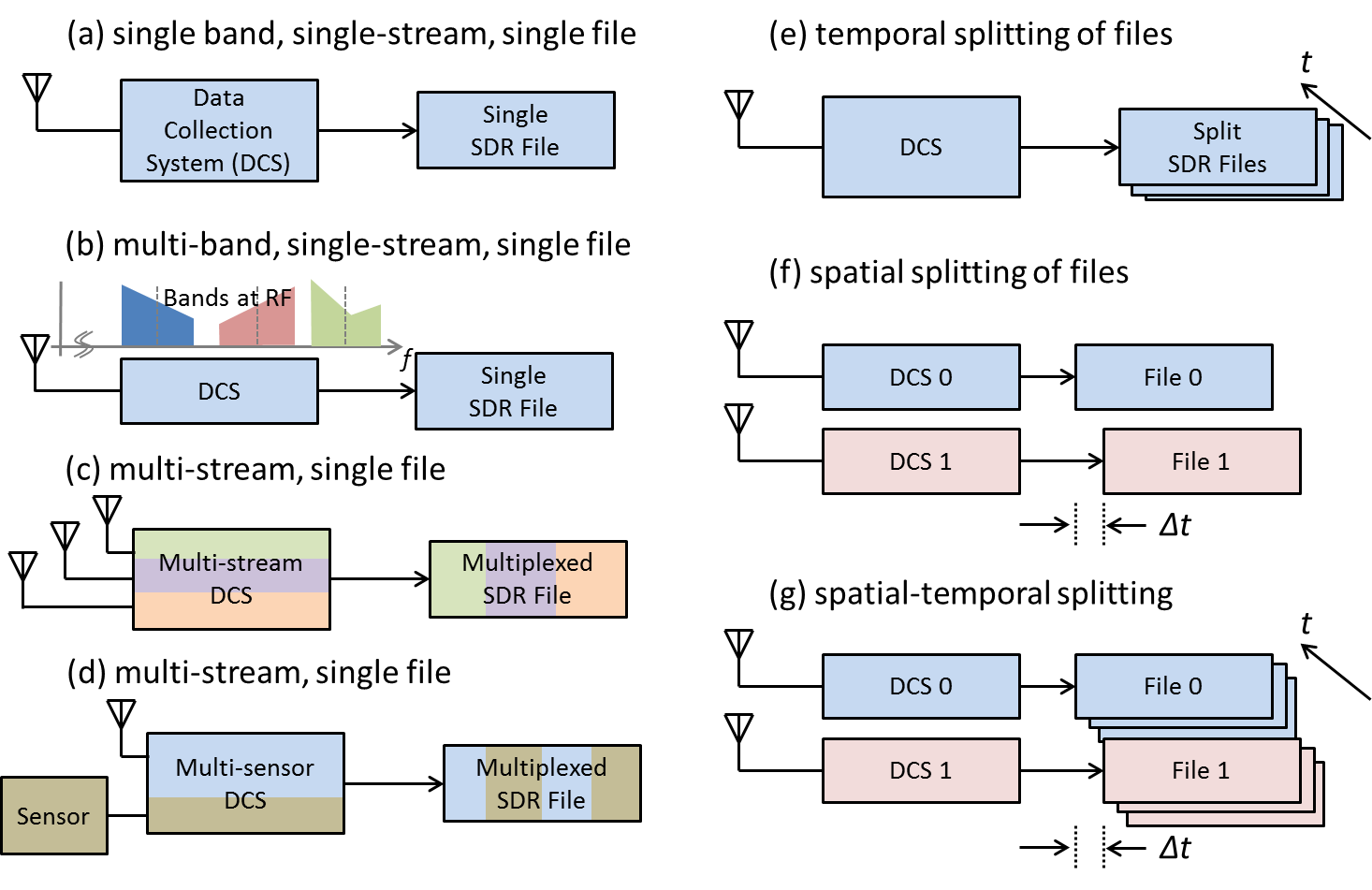


Figure – Fundamental GNSS SDR Data Collection Topologies

## Single Band, Single Stream, Single File

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 block of samples. This may include non-sample data such as headers and footers which may be used for data integrity check purposes. In this topology, this formatted data stream is written to one and only one file.

## Multi-Band, Single Stream, Single File

Figure 1.b 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 to fall 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.c 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 (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 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.d 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 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 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.

NOTE:

A metadata file typically exists for each data file. Optionally, all information for a multi-file set may be contained within one metadata file. For the former case, the first metadata file of a set must contain or make reference to the complete set of metadata parameters and subsequent files may contain only those that change from file to file.

## Spatial Splitting of Files

A collection system or setup may write individual data streams to multiple files. 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.g 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.

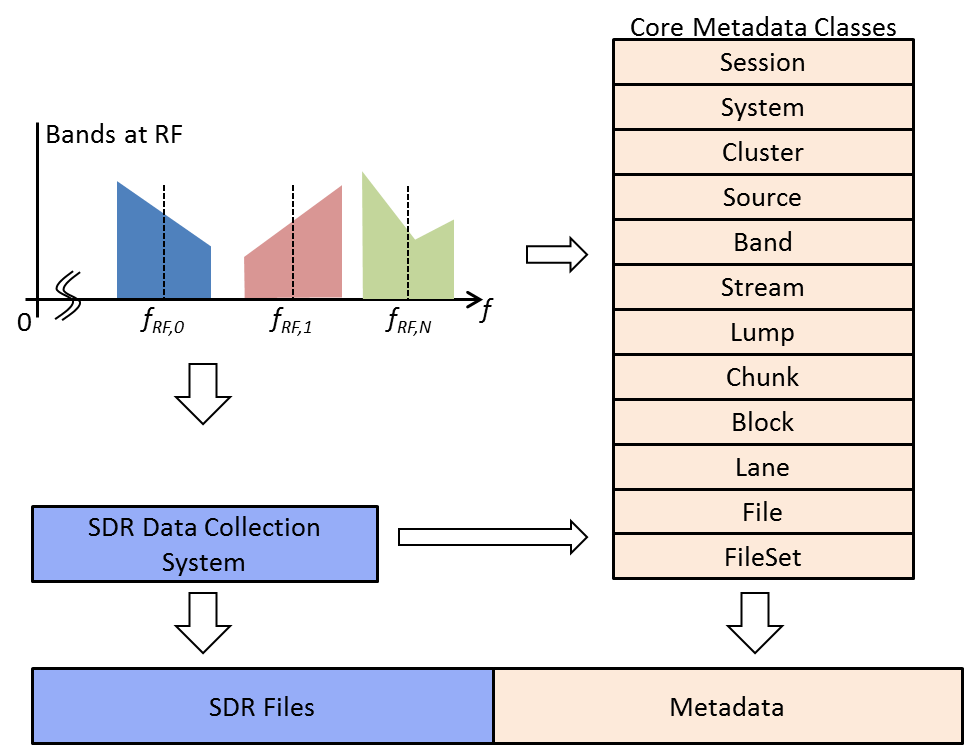


Figure – Overview of Core Metadata Classes and Generation

## Architecture

Figure 3 shows the UML 2.0 class model for the GNSS metadata structure.



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 1 describes 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 | Type | Enumeration | Optional | 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. | Comment |  | 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 | Type | Enumeration | Optional | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| TOA | Time of applicability for all position and attitude parameters | DateTime1 |  | Yes2 |  |
| POSITION | Platform position at TOA expressed in Geoid frame | Position |  | Yes2 |  |
| ORIENTATION | Orientation of the platform at TOA with respect to the local-level frame | Attitude |  | Yes2 |  |
| SYSTEM | The system used for this session | System |  | Yes |  |
| POC3 | Point of contact. Name of person or entity. | string |  | Yes |  |
| CONTACT3 | POC contact information (email) | string |  | Yes |  |
| CAMPAIGN3 | Data collection campaign | string |  | Yes |  |
| SCENARIO3 | Specific scenario for this collection | string |  | Yes |  |

NOTES:

1 <http://www.w3schools.com/schema/schema_dtypes_date.asp>

2 TOA, Position and Attitude may be back-annotated into metadata file following post processing.

3 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 (location and orientation) to the extent that this information is necessary for post-processing 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 | Type | Enumeration | Optional | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| Source | One or more sources of sampled data. | Source |  |  |  |
| Cluster | Zero of more clusters of antenna sources | Cluster |  |  |  |
| FREQBASE | Base frequency. All frequencies are specified as an integer multiple of FREQBASE | Frequency |  | Required |  |
| EQUIPMENT | Equipment used for this data collection | String |  | Yes |  |

### 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 for each element must be known in order to perform multi-element signal processing. The antenna element geometry with respect to its reference frame may be supplied by the manufacturer. 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 | Type | Enumeration | Optional | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| ID | Unique identifier | string |  | Yes |  |
| LOCATION | Origin of cluster reference frame w.r.t. platform coordinate frame | Position |  | Yes |  |
| ORIENTATION | Orientation of cluster frame w.r.t. platform frame | Orientation |  | Yes |  |
| VENDOR | Vendor name | string |  | Yes |  |
| MODEL | Model number | string |  | Yes |  |
| SERIAL | Serial number | string |  | Yes |  |

### 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 | Type | Enumeration | Optional | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| IDCLUSTER | cluster that this source belongs to | string |  | Yes |  |
| TYPE | Electrical type of this source | string | PATCH,  DIPOLE,  HELICAL, QUADRIFILAR,  SIMULATOR ,  OTHER, UNKNOWN | Yes | UNKNOWN |
| POLARIZATION | Element polarization | string | RHCP, LHCP, LINEAR, HORI, VERT, UNKNOWN | Yes | UNKNOWN |
| ORIGIN | Origin with respect to platform | Position |  | Yes |  |
| ORIENTATION | Orientation of normal vector to this source plane w.r.t. platform | Orientation |  | Yes3 |  |

### Band Object

A Band is defined as a finite 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 | Type | Enumeration | Optional | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| CENTERFREQ | Center frequency of band incident at source | Frequency |  | Required |  |
| TRANSLATEDFREQ | Translated center frequency of band | Frequency |  | Required |  |
| INVERTED | Binary flag indicating spectral inversion | Boolean | 0, 1 | Yes | 0 |
| DELAY | Delay of band measured from source to sampled stream, specified at TOA. | Duration |  | Yes | 0 |
| BANDWIDTH1 | Approximate double-sided half power bandwidth |  |  | Yes |  |

NOTES:

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 is chosen such that multiple passbands are intentionally aliased to fall adjacent to each other in the sampled signal spectrum. This is illustrated in Figure 5.

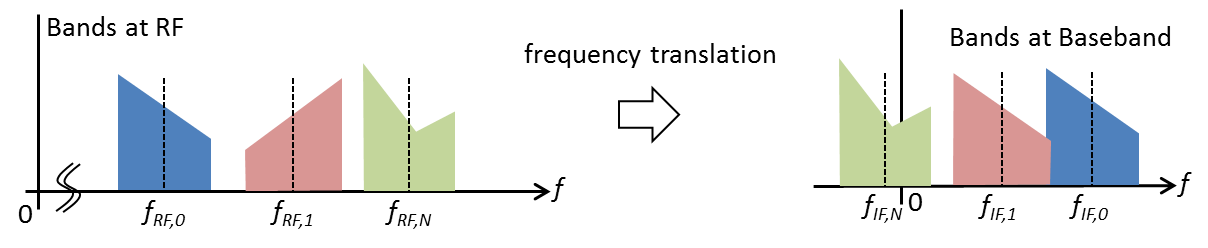


Figure - Intentional Aliasing of a Multiband signal to Baseband

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.

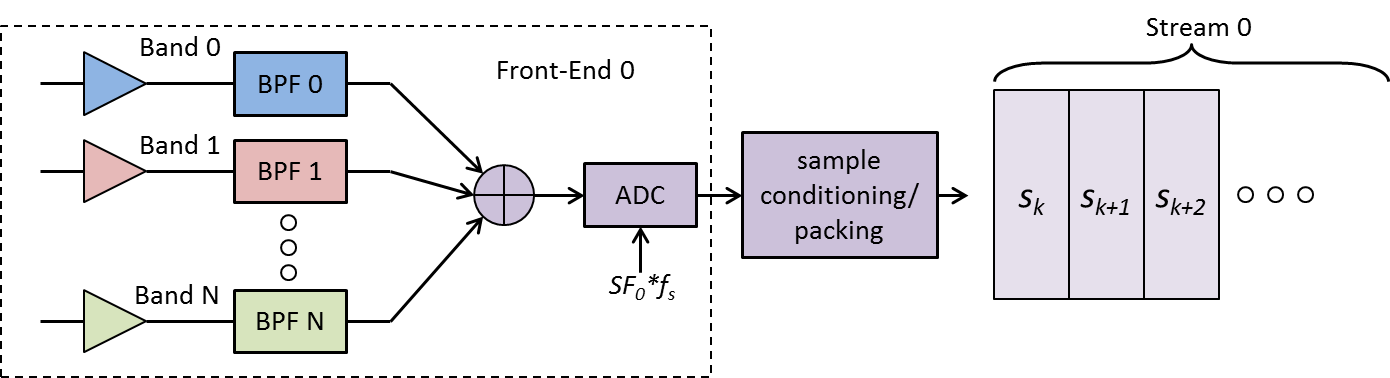


Figure - Illustration of Multiple Bands Present in a Stream

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.
* A 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 of the System base sample rate (FREQBASE).
* Sample values may be real or complex depending on whether IF sampling or baseband sampling is used, respectively. Some or all of the numerical values expressed in the stream may be inverted.
* Each sample value is represented by one or more bits, known as quantization. These values may be encoded using various established schemes.
* A sample of QUANTIZATION bits may be expressed by a value of PACKEDBITS in the Stream where PACKEDBITS ≥ QUANTIZATION.
* When PACKEDBITS > QUANTIZATION, the ALIGNMENT of the quantized sample with respect to the packed sample must be known in order to interpret the sample value correctly. This interpretation also depends on the ENCODING used.

The above are specified in terms of Stream Attributes.

Table 7 – Definition of Stream Attributes

| Attribute | Description | Type | Enumeration | Optional | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| BAND1 | One or more Bands present in this stream | Band |  | Required |  |
| RATEFACTOR | Sample rate factor | UINT16 |  | Required |  |
| QUANTIZATION | Sample quantization (bits) | UINT8 |  | Required |  |
| PACKEDBITS | Packed representation (bits) | UINT8 |  | Required |  |
| ALIGNMENT | Sample alignment | Alignment | ‘L’ – left aligned  ‘R’ – right aligned  ‘N’ – N/A | Required |  |
| FORMAT | Sample representation | SampleFormat | IF, IF’, IQ, IQ’, I’Q, I’Q’, QI, QI’, Q’I, Q’I’  (where ‘ signifies inversion) | Required |  |
| ENCODING | Numeric encoding scheme | string | SIGN – sign bit  SM – sign-magnitude  INT – twos complement  BIN – offset binary  FP – floating point | Required |  |

NOTES:

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

Table - Sample Encoding Schemes

| QUANTIZATION | | Encoding | Set | Range Min | Range Max |
| --- | --- | --- | --- | --- | --- |
| 1 | sign | | {-1, +1} | -1 | +1 |
| 2 | sign-magnitude | | {-3, -1, +1, +3} | -3 | +3 |
| signed integer | | {-2, -1, 0, 1} | -2 | +1 |
| offset binary | | {-2, -1, 0, 1} | -2 | +1 |
| 4 | sign-magnitude | | {-8, -7,…,-1,+1,…, +8} | -8 | +8 |
| signed integer | | {-8, …, 0…, +7} | -8 | +7 |
| offset binary | | {-8, …, 0…, +7} | -8 | +7 |
| 8 | sign-magnitude | | {-128, -127,…, +127, +128} | -127 | +128 |
| signed integer | | {-128, …, 0…, 127} | -128 | +127 |
| offset binary | | {-128, …, 0…, 127} | -128 | +127 |
| 16 | sign-magnitude | | {-215,… ,-1,+1,…, +215} | -215 | +215 |
| signed integer | | {-215, …, 0…, 215-1} | -215 | +215-1 |
| offset binary | | {-215, …, 0…, 215-1} | -215 | +215-1 |
| floating point | | IEEE 754-2008, FP16 | | |
| 32 | sign-magnitude | | {-231,… ,-1,+1,…, +231} | -231 | +231 |
| signed integer | | {-231, …, 0…, 231-1} | -231 | +231-1 |
| offset binary | | {-231, …, 0…, 231-1} | -231 | +231-1 |
| floating point | | IEEE 754-2008, FP32 | | |
| 64 | sign-magnitude | | {-263,… ,-1,+1,…, +263} | -263 | +263 |
| signed integer | | {-263, …, 0…, 263-1} | -263 | +263-1 |
| offset binary | | {-263, …, 0…, 263-1} | -263 | +263-1 |
| floating point | | IEEE 754-2008, FP64 | | |

### 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.

Figure 7 illustrates a *Lump* containing all samples from *N* *Sample Streams*.

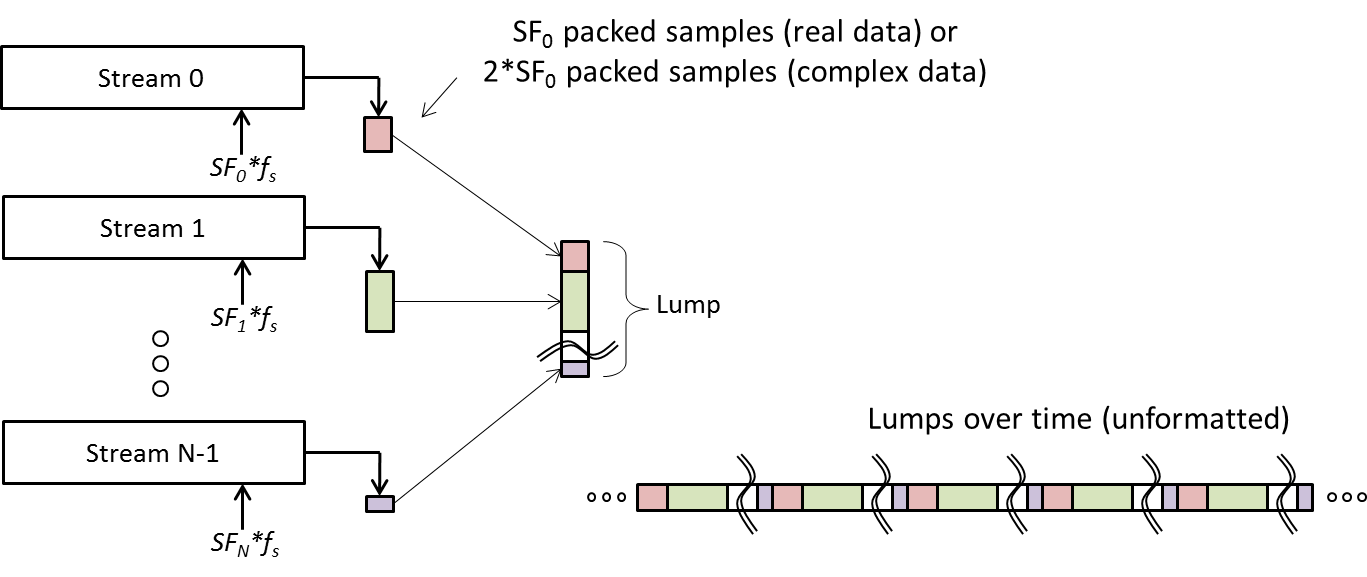


Figure - Illustration of a lump Containing Samples from N Streams

Table 9 – Definition of Lump Attributes

| Attribute | Description | Type | Enumeration | Optional | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| STREAM | One or more Streams present in this lump(ordered). | Stream |  | Required |  |

### 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 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 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 completely and 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.

Table 10 – Definition of Chunk Attributes

| Attribute | Description | Type | Enumeration | Optional | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| LUMP | One or more lumps (specified in order) present in this chunk. | Lump |  | Required |  |
| SIZEWORD | Bytes of unsigned integer datatype that data shall be read as | UINT8 | 1, 2, 4, 8 (Corresponds to UINT8, UINT16, UINT32 and UINT64) | Required |  |
| COUNTWORDS | Total number of words to be read in order to read/decode this chunk | UINT8 |  | Required |  |
| ENDIAN | Endianness of words stored in chunk | Endian | ‘L’ – Little  ‘B’ – Big  ‘N’ – not applicable | yes | ‘N’ |
| PADDING | Padding applied during encoding | Padding | ‘H’ – head padding  ‘T’ – Tail padding  ‘N’ – No padding | yes | ‘N’ |
| SHIFT | Word shift direction | WordShift | ‘L’ – Left shift  ‘R’ – Right shift | Required |  |

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.

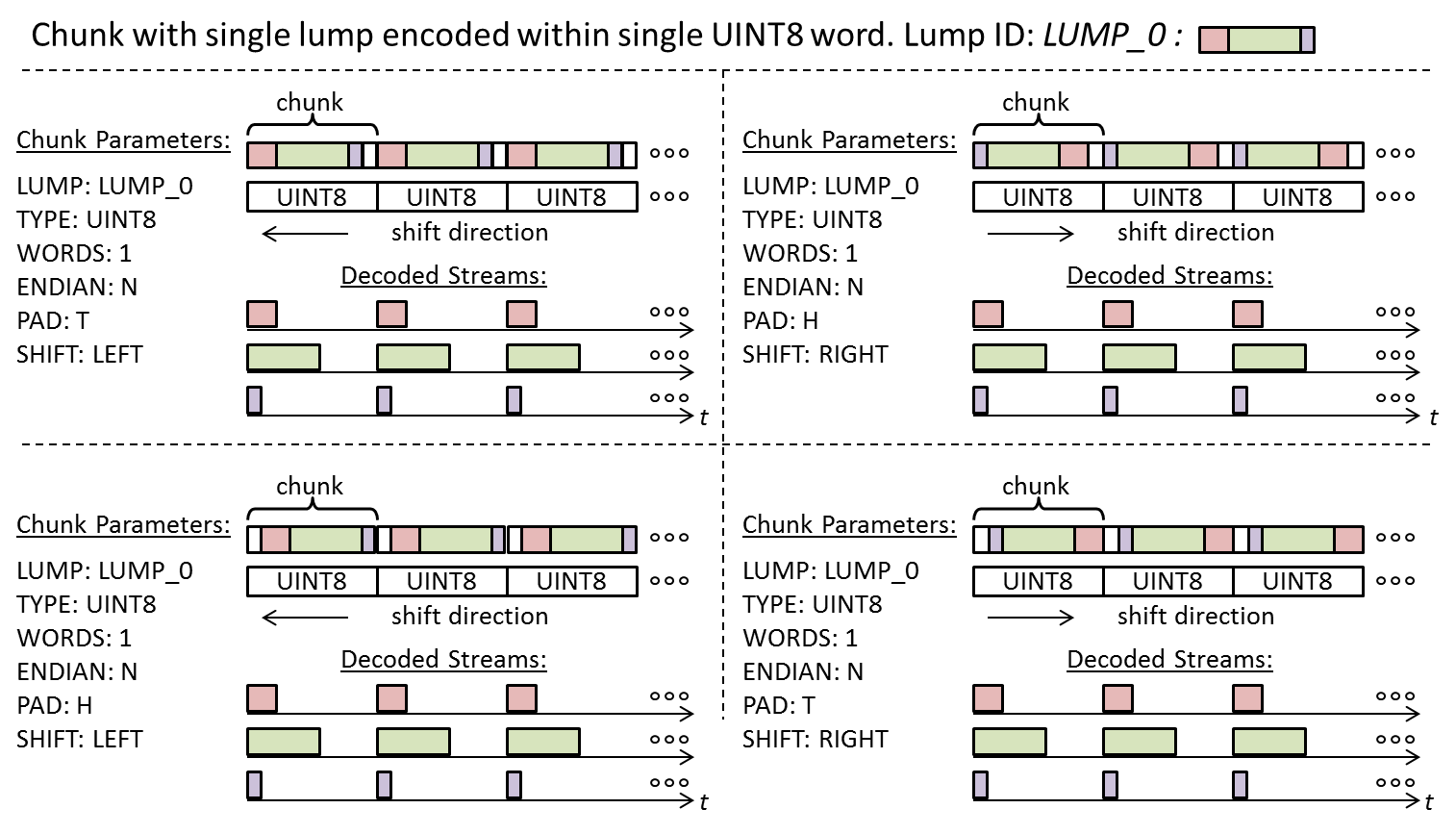


Figure - Encoding Schemes for a single Lump within a single Chunk

Figure 9 illustrates two schemes where ten lumps are encoded within a chunk comprised of 9 UINT16 words. In the first case, the UINT16 words are written to disk in big-endian format. The bytes are swapped in the second case since the words are written in little-endian format.

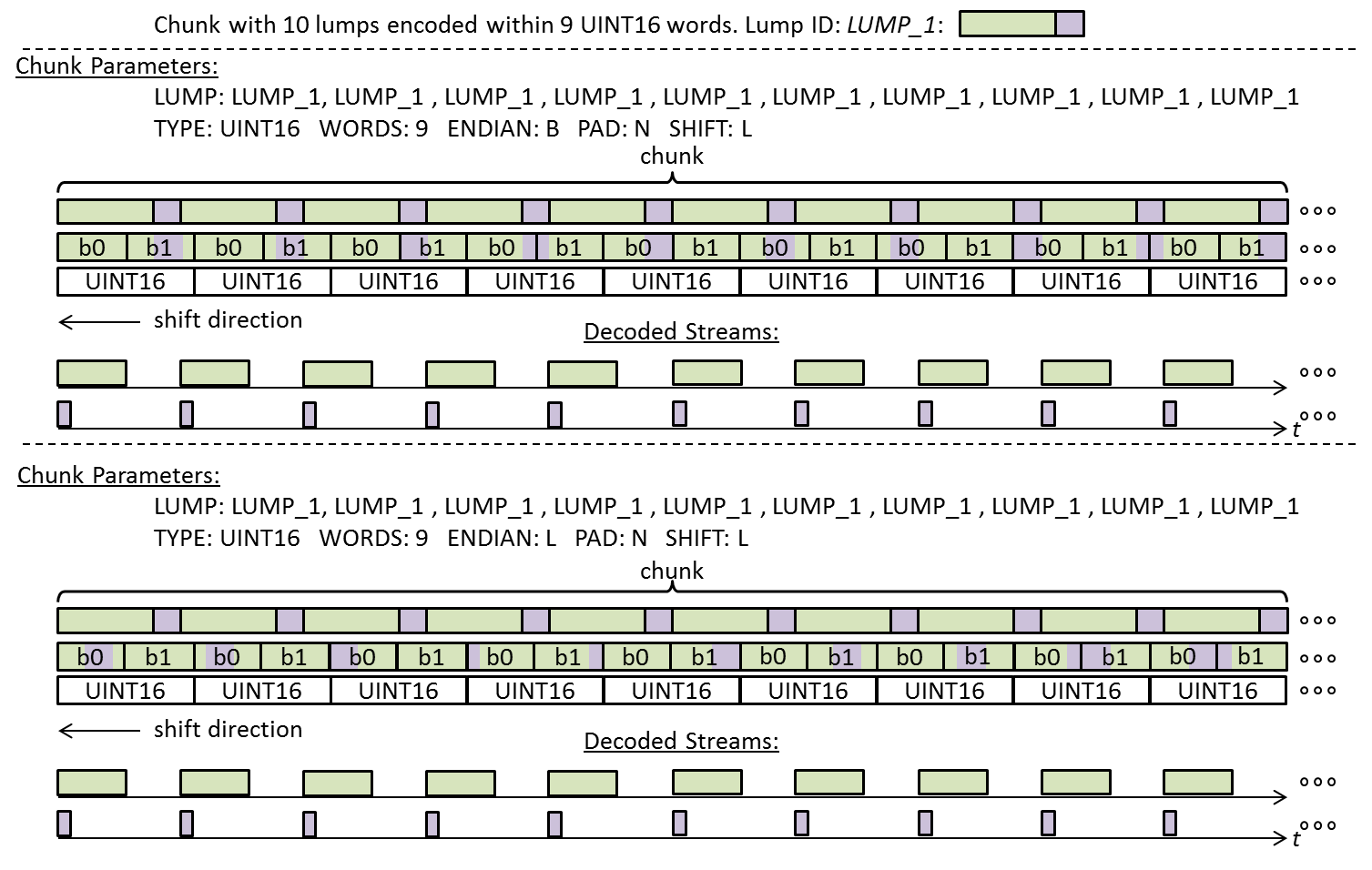


Figure - Encoding Schemes for a Chunk containing 10 Lumps

### 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 | Type | Enumeration | Optional | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| CHUNK | One or more Chunks present in this Block in order. | Chunk |  | Required |  |
| CYCLES | For the ordered Chunk pattern described in CHUNK, the integer number of cycles that this pattern repeats within a Block | UINT32 |  | Required |  |
| SIZEHEADER | Integer number of bytes to skip in order to access first byte of chunk data | UINT32 |  | yes | 0 |
| SIZEFOOTER | Integer number of bytes to skip in order to access first byte of next block | UINT32 |  | yes | 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 | Type | Enumeration | Optional | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| BLOCK | One or types of blocks in this lane (in order) | Block |  |  |  |
| BANDSRC | Associates predefined bands with sources | BandSource |  | Required?? |  |
| SESSION | Session information for this Lane | Session |  | Required |  |
| SYSTEM | System information for this Lane | System |  | Required |  |

### 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 | Type | Enumeration | Optional | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| URL | Unique identifier for the file (path/filename)? | anyURI |  | yes |  |
| TIMESTAMP | Time the file was generated | dateTime |  | Yes |  |
| OFFSET | Byte offset to start of first Block | UINT32 |  | Yes | 0 |
| LANE | Identifies which lane the data came from | Lane |  | required |  |
| PREVIOUS | Name of previous file (for temporally split files) | anyURI |  | Yes |  |
| NEXT | Name of next file (for temporally split files) | anyURI |  | Yes |  |
| OWNER | String specifying owner of this file | string |  | Yes |  |
| COPYRIGHT | Copyright information | string |  | Yes |  |

### 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.

| Attribute | Description | Type | Enumeration | Optional | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| FILE | Names of files comprising the file set | anyURI |  | Required 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

DateTime string specified in standard XML format. See <http://www.w3schools.com/schema/schema_dtypes_date.asp> for details

### 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.

### Duration

Used for specifying an interval of time. Units include ns, us, ms, sec. Format is double.

### Location

The location 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.

| Attribute | Description | Type | Enumeration | Optional | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| DATUM | Datum used for the Geoid | string | ‘WGS-84’ | Yes | ‘WGS-84’ |
| TYPE | Type of representation used to specify position | string | ‘LLH’, ‘ECEF’ | yes | ‘LLH’ |
| Value | Coordinate values | Double (3 x 1 vector) |  |  |  |

### Origin

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

| Attribute | Description | Type | Enumeration | Optional | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| TYPE | Type of representation used to specify origin | string | ‘XYZ’, ‘RTP’ | yes | ‘XYZ’ |
| Value | Coordinate values | Double (3 x 1 vector) |  |  |  |

### 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 3] direction cosine matrix. Other forms are supported by enumerating the TYPE attribute (if it exists).

Table 14 – Definition of Rotation Attributes

| Attribute | Description | Type | Enumeration | Optional | Default (if not specified) |
| --- | --- | --- | --- | --- | --- |
| TYPE | Type of rotation matrix used. | Enumeration | DCM | Yes | DCM |
| Matrix | Rotation matrix values | Double | 3 x 3 matrix |  |  |

# Working Group Membership

|  |  |  |
| --- | --- | --- |
|  | **NAME** | **COMPANY/AFFILIATION** |
| 1 | AKOS, Dennis M. | University of Colorado |
| 2 | AL-MASYABI, Walid | Raytheon |
| 3 | ARRIBAS, Javier | Centre Tecnològic de Telecomunicacions de Catalunya (CTTC) |
| 4 | BAVARO, Michele | One Talent GNSS |
| 5 | BELABBAS, Boubeker | DLR |
| 6 | BHATTI, Jahshan | University of Texas at Austin |
| 7 | BRAASCH, Michael | Ohio University |
| 8 | CHANSARKAR, Mangesh | CSR |
| 9 | CHEN, Xin | Shanghai Jiao Tong University |
| 10 | COSGROVE, Mathew | Northrop Grumman NSD |
| 11 | CRAMPTON, Paul G. | Spirent Federal Systems |
| 12 | CURRAN, James | Joint Research Center, Italy |
| 13 | DOVIS, Fabio | Politecnico di Torino |
| 14 | FAVENZA, Alfredo | ISTITUTO SUPERIORE MARIO BOELLA (ISMB) |
| 15 | FERNÁNDEZ HERNÁNDEZ, Ignacio | Galileo Supervisory Agency |
| 16 | FERNÁNDEZ-PRADES, Carles | Centre Tecnològic de Telecomunicacions de Catalunya (CTTC) |
| 17 | GAVRILOV, Artyom | GNSS-SDR |
| 18 | GLENNON, Eamonn | University of New South Wales |
| 19 | GOODRICH, Brian | NavCom |
| 20 | GUNAWARDENA, Sanjeev | Air Force Institute of Technology |
| 21 | HODO, David | Integrated Solutions for Systems, Inc |
| 22 | KALYANARAMAN, Sai K. | Rockwell Collins Inc. |
| 23 | KOU, Yanhong | Beihang University |
| 24 | KUBO, Nobuaki | Tokyo University of Marine Science and Technology |
| 25 | LANGER, Markus | Karlsruhe Institute of Technology (KIT) |
| 26 | LEDVINA, Brent | Coherent Navigation |
| 27 | LITTLE, Jon C. | Applied Research Laboratories of the University of Texas at Austin |
| 28 | LOHAN, Elena-Simona | Tampere University of Technology |
| 29 | LÓPEZ-ALMANSA, José María | GMV |
| 30 | LOPEZ-RISUEÑO, Gustavo | European Space Agency |
| 31 | MACDONALD, John C | AFRL Sensors Directorate |
| 32 | MATHEWS, Michael B. | Loctronix |
| 33 | MORTON, Yu (Jade) | Colorado State University |
| 34 | O’BRIEN, Andrew J. | Ohio State University |
| 35 | PANY, Thomas | Ifen GmbH |
| 36 | PARSONS, Bryan Masamitsu | Applied Research Laboratories of the University of Texas at Austin |
| 37 | PELOSI, Lou | Cast Navigation |
| 38 | PETOVELLO, Mark | University of Calgary |
| 39 | PINCHIN, James | University of Nottingham |
| 40 | PSIAKI, Mark | Cornell University |
| 41 | RIEDL, Bernhard | IFEN Gmbh |
| 42 | RUDRA, Angsuman | D-TA Systems |
| 43 | RÜGAMER, Alexander | Fraunhofer IIS |
| 44 | SAHMOUDI, Mohamed | University of Toulouse |
| 45 | SCHIPPER, Brian | Honeywell |
| 46 | SCHLEPPE, John B. | NovAtel |
| 47 | SCOTT, Logan | LS Consulting |
| 48 | SHIVARAMAIAH, Nagaraj | GNSS Labs |
| 49 | SOLOVIEV, Andrey | Qunav |
| 50 | STAHL, Manuel | Fraunhofer IIS |
| 51 | SUZUKI, Taro |  |
| 52 | TKATCH, Alex | Rohde & Schwarz USA Inc. |
| 53 | UNWIN, Martin | SSTL |
| 54 | VINANDE, Eric | AFRL Sensors Directorate |
| 55 | WARD, Phillip, W. | Navward GPS Consulting |
| 56 | WESSON, Kyle | Zeta Associates |
| 57 | WON, Jong-Hoon | ISTA at University FAF Munich |
| 58 | YANG, Ning | Draper Laboratory |
| 59 | YAO, Zheng | Tsinghua University |
| 60 | YU, Jim | Trimble |
| 61 | YU-HSUAN, Chen | Stanford University |
| 62 | ZHU, Zhen | East Carolina University |