xSEED preamble

Version 2018-03-13

Background and context

Adopted by the FDSN in 1987, the SEED format has become the canonical format for passive source seismic (and other) data. For continuous data collection, archiving and delivery it has become common to handle the time series and metadata separately. This document contains a specification for a next generation version format for the time series portion, known as miniSEED.

The key issues motivating a change to miniSEED 2.4 are limitations with identifiers for a) deployments with a very large number of nodes and b) new instruments and other data source types, e.g. synthetics. Changing the fundamental identifiers requires changes to key fields in miniSEED that render it incompatible with the current release. Such a small, but disruptive change affords the opportunity to address a number of historical issues and create new capability to address future needs.

Overview of significant changes between miniSEED 2.4 and this specification

- Convert usage of FDSN time series codes (network, station, location, channel) to a variable length Uniform Resource Name (URN) for enhanced flexibility and to allow nearly unlimited future re-definition.
  - Expansion of the individual network, station and location codes to a maximum of 8 characters.
  - The definition of a time series identifier URN and the individual codes, including an expansion of the codes is documented in a separate document.
- Incorporate critical details previously in blockettes (actual sample rate, encoding, microseconds) into the fixed section of the data header.
- Increase sample rate/period representation to a 64-bit floating point value.
- Increase start time resolution to nanoseconds.
- Specify fixed byte order (little endian) for the binary portions of the headers and define a byte order for each data encoding.
- Drop legacy data encodings and reserve their values so they are not used again in the future.
- Add a format version.
- Add a data publication version.
- Add CRC field for validating integrity of record.
- Add a “mass position offscale” flag
- Replace the blockette structure with flexible extra header construct:
  - Specify a reserved set of extra headers defined by the FDSN, provide schema for validation
  - Previous flags and blockette contents defined in reserved extra headers
  - Allow arbitrary headers to be included in a record
- Remove the restriction on record length to be powers of 2, allow variable length.
- General compression encodings for fundamental sample types and opaque data

  Encoding 50: 32-bit integers, general compressor (To Be Determined)
  Encoding 51: 32-bit IEEE floats, general compressor (To Be Determined)
  Encoding 52: 64-bit IEEE floats (doubles), general compressor (To Be Determined)
  Encoding 100: Opaque data

Near complete preservation of miniSEED 2.4 data. Information that is not retained is limited to: clock model specification per timing exception (current specification only allows a single clock model specification per record), Blockettes 400 (Beam) & 405 (Beam Delay) and Blockette 2000 (Opaque Data).

xSEED specification (in draft)

Version 2018-03-13

Table of Contents

Section 1: Overview

Purpose

The International Federation of Digital Seismograph Networks (FDSN)[1] defines miniSEED as a data format for digital time series and related information. The primary intended uses are data collection, archiving and exchange of seismological data. Support is also included for time series data for other geophysical-related measurements such as pressure, temperature, tilt, etc. In addition to the time series, storage of related state-of-health and parameters documenting the state of the recording system is supported. The FDSN metadata counterpart of miniSEED is StationXML [2] which is used to describe characteristics needed to interpret the data such as location, instrument response, etc.

Background

The Standard for the Exchange of Earthquake Data (SEED)[3] was adopted by the FDSN in the 1980s and has served as the dominant standard for seismological data archiving and exchange for decades. This specification defines a format to store both time series and a rich set of related metadata. In 1992, changes in the SEED format were adopted to officially support "data only" SEED, known as miniSEED. This specification is an evolution of miniSEED and is an extension on what was specified in SEED version 2.4.

General structure

The fundamental unit of the format is a data record. A time series is commonly stored and exchanged as a sequence of these records. Each record is independently usable even when presented in a sequence. There are data encodings for integers, floats, text or compressed data samples. To limit problems with timing system drift & resolution and practical issues of subsetting & resource limitation for readers of the data, typical record lengths for raw data generation and archiving are recommended to be in the range of 256 and 4096 bytes.

---

[1] The International Federation of Digital Seismograph Networks (FDSN): http://www.fdsn.org/
[2] FDSN StationXML metadata schema: http://www.fdsn.org/xml/station/
[3] FDSN SEED format: http://www.fdsn.org/publications/

Section 2: Record layout and fields

An record is composed of a header followed by a time series data payload. The byte order of binary fields in the header must be least significant byte first (little endian).

The total length of a record is variable and is the sum of 40 (length of fixed section of header), field 11 (length of identifier), field 12 (length of extra headers), field 13 (length of payload).

Field	Field name	Type	Length	Offset	Content
1	Record header indicator (‘MS’)	CHAR	2	0	ASCII ‘MS’
2	Format version (3)	UINT8	1	2	
3	Flags	UINT8	1	3	
4	Record start time				
	Year (0 - 65535)	UINT16	2	4	
	Day-of-year (1 - 366)	UINT16	2	6	
	Hour (0 - 23)	UINT8	1	8	
	Minute (0 - 59)	UINT8	1	9	
	Second (0 - 60)	UINT8	1	10	
5	Data encoding format	UINT8	1	11	
6	Start time nanosecond (0 - 999999999)	UINT32	4	12	
7	Sample rate/period	FLOAT64	8	16	
8	Number of samples	UINT32	4	24	
9	CRC of the record	UINT32	4	28	
10	Data publication version	UINT8	1	32	
11	Length of identifier in bytes	UINT8	1	33	
12	Length of extra headers in bytes	UINT16	2	34	
13	Length of data payload in bytes	UINT32	2	36	
14	Time series identifier	CHAR	V	40	URN identifier
15	Extra header fields	CBOR	V	40 + field 11	
16	Data payload	encoded	V	40 + field 11 + field 12	

All length values are specified in bytes, which are assumed to be 8-bits in length. Data types for each field are defined as follows:

CHAR - Character data.
UINT8 - Unsigned 8-bit integer.
UINT16 - Unsigned 16-bit integer.
UINT32 - Unsigned 32-bit integer.
FLOAT64 - IEEE-754 64-bit floating point number.
CBOR - Concise Binary Object Representation as defined by RFC 7049.

Section 3: Description of fields in record header

Record header fields:

1 CHAR: Data record indicator - ASCII “MS”.

2 UINT8: Data format header version, set to **3** for this version. When a non-backwards compatible change is introduced the major version will be incremented.

3 UINT8: Flags:
[Bit 0] - Calibration signals present. *[same as SEED 2.4 FSDH, field 12, bit 0]*
[Bit 1] - Time tag is questionable. *[same as SEED 2.4 FSDH, field 14, bit 7]*
[Bit 2] - Clock locked. *[same as SEED 2.4 FSDH, field 13, bit 5]*
[Bit 3] - Reserved for future use.
[Bit 4] - Reserved for future use.
[Bit 5] - Reserved for future use.
[Bit 6] - Reserved for future use.
[Bit 7] - Reserved for future use.

4 Record start time, time of the first data sample. A representation of UTC using individual fields for year, day-of-year, hour, minute and second. A 60 second value is used to represent a time value during a positive leap second. Nanosecond resolution is specified in field 6.

5 UINT8: A code indicating the encoding format, see *Section 4: Data encoding codes* for a list of valid codes. If no data blocks are included set this value to 0.

6 UINT32: Nanosecond resolution of record start time specified in field 4.

7 FLOAT64: Sample rate encoded in 64-bit IEEE-754 floating point format. When the value is positive it represents the rate in samples per second, when it is negative it represents the sample period in seconds. Creators should use the negative value sample period notation for rates less than 1 samples per second to retain resolution. Set to 0.0 if no time series data is included in the record.

8 UINT32: Total number of data samples in the time series data payload. Set to 0 if no samples (header-only records) or unknown number of samples (e.g. for opaque payload encoding).

9 UINT32: CRC-32C (Castagnoli) value of the complete record with the 4-byte CRC field set to zeros. The CRC-32C (Castagnoli) algorithm with polynomial 0x1EDC6F41 (reversed 0x82F63B78) to be used is defined in RFC 3309, which further includes references to the relevant background material.

10 UINT8: Data publication version. Values should only be considered relative to each other for data from the same data center. Semantics may vary between data centers but generally larger values denote later and more preferred data. Recommended values: 1 for raw data, 2+ for revisions produced later, incremented for each revision. A value of 0 indicates unknown version such as when data are converted to miniSEED from another format.

11 UINT8: Length, in bytes, of time series identifier in field 14.

12 UINT16: Length, in bytes, of extra headers in field 15. If no extra headers, set this value to 0.

13 UINT32: Length, in bytes, of data payload starting in field 16. If no data payload, set this value to 0.

14 CHAR: Time series identifier URN.

15 CBOR: Extra header fields of variable length encoded in Concise Binary Object Representation as defined by RFC 7049. A reserved set of headers fields is defined by the FDSN, see *Section 5: Extra header fields*. Other header fields may be present and should be defined by the organization that created them.

16 encoded: Data payload, length indicated in field 13, encoding indicated in field 5.

Section 4: Data encoding codes

Encodings in the format are identified by a code (number). These codes are assigned by the FDSN Data Exchange Working Group. A list of valid codes are as follows:

CODE	Description
0	Text
1	16-bit integer, little endian
3	32-bit integer, little endian
4	IEEE 32-bit floats, little endian
5	IEEE 64-bit floats (doubles), little endian
10	Steim-1 integer compression, big endian
11	Steim-2 integer compression, big endian
19	Steim-3 integer compression, big endian (not in common use in archives as of 2018)
53	32-bit integers, little endian, general compressor (To Be Determined, e.g. Brotli)
54	32-bit IEEE floats, little endian, general compressor (To Be Determined, e.g. Brotli)
55	64-bit IEEE floats, little endian, general compressor (To Be Determined, e.g. Brotli)
100	Opaque data - only for use in special scenarios, not intended for long term archiving

Overview and description of the Steim-1 and Steim-2 compression encodings may be found in the SEED 2.4 manual, Appendix B.

Retroactive future encodings

New data encodings may be added to the format in the future without incrementing the format version. There is no default encoding, readers must check the encoding value to determine if the encoding is supported.

Retired encoding values

The following numeric codes were used in miniSEED 2.x and should not be used for encodings defined in the future:

2 (24-bit integers)
12 (GEOSCOPE multiplexed format 24-bit integer)
13 (GEOSCOPE multiplexed format 16-bit gain ranged, 3-bit exponent)
14 (GEOSCOPE multiplexed format 16-bit gain ranged, 4-bit exponent)
15 (US National Network compression)
16 (CDSN 16-bit gain ranged)
17 (Graefenberg 16-bit gain ranged)
18 (IPG-Strasbourg 16-bit gain ranged)
30 (SRO format)
31 (HGLP format)
32 (DWWSSN gain ranged format)
33 (RSTN 16-bit gain ranged format)

Section 5: Extra header fields

The extra headers are encoded as CBOR as defined by [RFC 7049](). All extra headers are optional as far as the format is concerned.

The use of CBOR for extra headers must follow these rules:

- All extra headers are contained in an anonymous (unnamed) root map
- All entries are key-value pairs where:
  - Keys must be strings (Unicode characters encoded as UTF-8 per the CBOR specification)
  - Values can be maps, arrays, strings, numbers or any CBOR type with a JSON equivalent
- The key value of “FDSN” in the root map is reserved for values defined by the FDSN

Validation and documentation, in JSON representation

Extra headers in CBOR, with the rules specified, are always convertible to JSON using the guidelines in Section 4.1 of RFC 7049 (CBOR).

Extra headers are specified and documented in JSON Schema (http://json-schema.org/). The recommended method of validating extra headers is to convert the CBOR to JSON and validate using the published schemas.

A JSON representation is the recommended method to document the data structures of extra headers.

FDSN reserved headers

The key named “FDSN” with value of an map (JSON object) in the root container of the extra headers is reserved for definition by the FDSN. The documentation and schema of these headers are specified in JSON Schema here:

https://iris-edu.github.io/xseed-specification/ExtraHeaders/ExtraHeaders-FDSN.schema.json
(location subject to change)

Guidelines for extension

Network operators, manufacturers, data centers, users and other agencies may wish do define their own extra headers. The following guidelines should be considered, in particular for data that is expected to reside in a public repository:

- All headers defined by a group or agency should be contained in a map (JSON object) that is the value of a key in the root container with a clearly identifiable name, e.g. parallel to “FDSN”.
- Creation of a JSON Schema document describing the field(s) is strongly recommended. The schema should be submitted to the FDSN to be made publically available.
- Headers that would be generally useful should be submitted to the FDSN for consideration of being added to the reserved headers for general definition and use.

Multiple JSON Schema documents are easily combined for use in validating extra headers that may contain headers defined in multiple schema documents.

Appendix A: Example extra headers

An example of the reserved FDSN headers defined by the FDSN is provided below in JSON representation. In this example, all fields are illustrated in common usage.

https://iris-edu.github.io/xseed-specification/ExtraHeaders/Example-ExtraHeaders-FDSN.json
(location subject to change)

Example reserved headers in JSON representation:


```json
{
    "FDSN": {
        "Time": {
            "ClockLocked": false,
            "TagQuestionable": false,
            "Quality": 100,
            "Correction": 1.234,
            "LeapSecond": 1,
            "Exception": [
                {
                    "Time": "2012-01-13T18:46:36Z",
                    "VCOCorrection": 50.7812,
                    "ReceptionQuality": 80,
                    "Count": 23,
                    "Type": "Valid Timemark",
                    "ClockStatus": "SNR=48,51,51,50,50,48,46,48,48,45,45"
                },
                {
                    "Time": "2012-01-13T18:48:12.7654Z",
                    "VCOCorrection": 44.1313,
                    "ReceptionQuality": 55,
                    "Count": 19690,
                    "Type": "Missing timemarks",
                    "ClockStatus": "SNR=50,48,46,48,48,45,45"
                }
            ]
        },
        "Event": {
            "Begin": true,
            "End": true,
            "InProgress": true,
            "Detection": [
                {
                    "Type": "GENERIC",
                    "SignalAmplitude": 80,
                    "SignalPeriod": 0.4,
                    "BackgroundEstimate": 18,
                    "OnsetTime": "2012-01-13T18:28:06.185000Z",
                    "Detector": "Dalek STA/LTA"
                },
                {
                    "Type": "MURDOCK",
                    "SignalAmplitude": 80,
                    "SignalPeriod": 0.4,
```

```
                "BackgroundEstimate": 18,
                "Wave": "DILATATION",
                "Units": "COUNTS",
                "OnsetTime": "2012-01-13T18:28:06.185000Z",
                "MEDSNR": [1, 3, 2, 1, 4, 0],
                "MEDLookback": 2,
                "MEDPickAlgorithm": 0,
                "Detector": "Z_SPWWSS"
            }
        ]
    },
    "Calibration": {
        "InProgress": true,
        "Sequence": [
            {
                "Type": "Step",
                "BeginTime": "2012-01-13T18:28:06.185000Z",
                "Steps": 12,
                "StepFirstPulsePositive": true,
                "StepAlternateSign": true,
                "Trigger": "AUTOMATIC",
                "Continued": false,
                "Amplitude": 1345,
                "InputUnits": "COUNTS",
                "Duration": 603.456,
                "SinePeriod": 5.0,
                "StepBetween": 500.0,
                "InputChannel": "CAL",
                "ReferenceAmplitude": 45.8,
                "Coupling": "RESISTIVE",
                "Rolloff": "Description of rolloff",
            },
            {
                "Type": "Step",
                "EndTime": "2012-01-13T18:28:16.185000Z",
            },
            {
                "Type": "Sine",
                "BeginTime": "2012-01-13T18:28:06.185000Z",
                "EndTime": "2012-01-13T18:28:06.185000Z",
                "Trigger": "MANUAL",
                "Continued": true,
                "Amplitude": 1345,
                "InputUnits": "COUNTS",
                "AmplitudeRange": "PEAKTOPEAK",
                "SinePeriod": 5.0,
                "InputChannel": "CAL",
                "ReferenceAmplitude": 45.8,
                "Coupling": "RESISTIVE",
                "Rolloff": "Description of rolloff",
            },
            {
                "Type": "PseudoRandom",
                "BeginTime": "2012-01-13T18:28:06.185000Z",
                "EndTime": "2012-01-13T18:28:06.185000Z",
```

```
                "Trigger": "MANUAL",
                "Amplitude": 0.0001,
                "InputUnits": "M/S",
                "Duration": 300,
                "InputChannel": "CAL",
                "ReferenceAmplitude": 45.8,
                "Coupling": "CAPACITIVE",
                "Rolloff": "Very randomly",
                "Noise": "White"
            },
            {
                "Type": "Generic",
                "BeginTime": "2012-01-13T18:28:06.185000Z",
                "EndTime": "2012-01-13T18:28:06.185000Z",
                "Trigger": "MANUAL",
                "Amplitude": 1345,
                "Duration": 100,
            }
        ]
    },
    "Flags": {
        "MassPositionOffscale": true,
        "AmplifierSaturation": true,
        "DigitizerClipping": true,
        "Spikes": true,
        "Glitches": true,
        "FilterCharging": true,
        "StationVolumeParityError": true,
        "LongRecordRead": true,
        "ShortRecordRead": true,
        "StartOfTimeSeries": true,
        "EndOfTimeSeries": true,
        "MissingData": true,
        "TelemetrySyncError": true,
    },
    "Logger": {
        "Model": "DM24",
        "Serial": "A4567"
    },
    "Sensor": {
        "Model": "T240",
        "Serial": "123123"
    },
    "Clock": {
        "Model": "P273T11N16",
        "Serial": "24A00000"
    },
    "DataQuality": "D",
    "Sequence": 123456
  }
}
```

Appendix B: Mapping from miniSEED 2.4

The following listing specifies the mapping of all fields in miniSEED 2.4 to this specification. In this listing the following abbreviations are used:

- EH = Extra Header (in this specification)
- TSID = Time Series Identifier (in this specification)
- FSDH = Fixed Section Data Header (in either specifications)

Fixed Section Data Header (FSDH)

2.4 field	Description	This specification
1	Sequence number	EH: FDSN.Sequence
2	Data header/quality indicator	EH: FDSN.DataQuality
3	Reserved byte	[no mapping]
4	Station identifier code	Incorporated into TSID, FSDH field 14
5	Location identifier	Incorporated into TSID, FSDH field 14
6	Channel identifier	Incorporated into TSID, FSDH field 14
7	Network code	Incorporated into TSID, FSDH field 14
8	Record start time	FSDH field 4
9	Number of samples	FSDH field 8
10	Sample rate factor	Incorporated into FSDH field 7
11	Sample rate multiplier	Incorporated into FSDH field 7
12	Activity flags, bits: 0 = calibration signals present 1 = time correction applied 2 = begining of event 3 = end of event 4 = positive leap second included 5 = negative leap second included 6 = event in progress	Extra headers: 0 = EH FDSN.Calibration.InProgress 1 = [no mapping] 2 = FDSN.Event.Begin 3 = FDSN.Event.End 4 = FDSN.Time.LeapSecond 5 = FDSN.Time.LeapSecond 6 = FDSN.Event.InProgress
13	I/O flags, bits: 0 = Station volume parity error 1 = Long record read 2 = Short record read 3 = Start of time series 4 = End of time series 5 = Clock locked	Extra headers: 0 = FDSN.Flags.StationVolumeParityError 1 = FDSN.Flags.LongRecordRead 2 = FDSN.Flags.ShortRecordRead 3 = FDSN.Flags.StartOfTimeSeries 4 = FDSN.Flags.EndOfTimeSeries 5 = FDSN.Time.ClockLocked
14	Data quality flags, bits: 0 = Amplifier saturation detected 1 = Digitizer clipping detected 2 = Spikes detected 3 = Glitches detected	Extra headers: 0 = FDSN.Flags.AmplifierSaturation 1 = FDSN.Flags.DigitizerClipping 2 = FDSN.Flags.Spikes 3 = FDSN.Flags.Glitches

	4 = Missing/padded data present 5 = Telemetry synchronization error 6 = Digital filter may be charging 7 = Time tag questionable	4 = FDSN.Flags.MissingData 5 = FDSN.Flags.TelemetrySyncError 6 = FDSN.Flags.FilterCharging 7 = FDSN.Time.TagQuestionable
15	Number of blockettes that follow	[no mapping]
16	Time correction	EH FDSN.Time.Correction
17	Beginning of data	[no mapping]
18	First blockette	[no mapping]

Blockette 100 (Sample Rate)

2.4 field	Description	This specification
3	Actual sample rate	FSDH field 7
4	Flags (to be defined)	[no mapping]
5	Reserved byte	[no mapping]

Blockette 200 (Generic Event Detection)

2.4 field	Description	This specification
3	Signal amplitude	EH: FDSN.Event.Detection.SignalAmplitude
4	Signal period	EH: FDSN.Event.Detection.SignalPeriod
5	Background estimate	EH: FDSN.Event.Detection.BackgroundEstimate
6	Event detection flags, bits: 0 = if set: dilatation wave, otherwise compression wave 1 = if set: units are after deconvolution, otherwise counts 2 = if set: wave is undetermined	Extra headers: 0 = FDSN.Event.Detection.Wave 1 = FDSN.Event.Detection.Units 2 = FDSN.Event.Detection.Wave existence
7	Reserved byte	[no mapping]
8	Signal onset time	EH: FDSN.Event.Detection.OnsetTime
9	Detector name	EH: FDSN.Event.Detection.Detector

Blockette 201 (Murdock Event Detection)

2.4 field	Description	This specification
3	Signal amplitude	EH: FDSN.Event.Detection.SignalAmplitude

4	Signal period	EH: FDSN.Event.Detection.SignalPeriod
5	Background estimate	EH: FDSN.Event.Detection.BackgroundEstimate
6	Event detection flags, bits: 0 = if set: dilatation wave, otherwise compression wave	Extra headers: 0 = FDSN.Event.Detection.Wave
7	Reserved byte	[no mapping]
8	Signal onset time	EH: FDSN.Event.Detection.OnsetTime
9	Signal-to-noise ratio values	EH: FDSN.Event.Detection.MEDSNR (array)
10	Lookback value	EH: FDSN.Event.Detection.MEDLookback
11	Pick algorithm	EH: FDSN.Event.Detection.MEDPickAlgorithm
12	Detector name	EH: FDSN.Event.Detection.Detector

Blockette 300 (Step Calibration)

2.4 field	Description	This specification
3	Beginning of calibration time	EH: FDSN.Calibration.Sequence.Begintime
4	Number of step calibrations	EH: FDSN.Calibration.Sequence.Steps
5	Calibration flags, bits: 0 = if set, first pulse positive 1 = if set, calibration alt. sign 2 = if set, calibration was automatic, otherwise manual 3 = if set, calibration continued from previous record	Extra Headers: 0 = FDSN.Calibration.Sequence.StepFirstPulsePositive 1 = FDSN.Calibration.Sequence.StepAlternateSign 2 = FDSN.Calibration.Sequence.Trigger 3 = FDSN.Calibration.Sequence.Continued
6	Duration of step	EH: FDSN.Calibration.Sequence.Duration
7	Time between calibration steps	EH: FDSN.Calibration.Sequence.StepBetween
8	Amplitude of calibration signal	EH: FDSN.Calibration.Sequence.Amplitude
9	Channel with calibration signal	EH: FDSN.Calibration.Sequence.InputChannel
10	Reserved byte	[no mapping]
11	Reference amplitude	EH: FDSN.Calibration.Sequence.ReferenceAmplitude
12	Coupling of calibration signal	EH: FDSN.Calibration.Sequence.Coupling
13	Rolloff characteristics for filters	EH: FDSN.Calibration.Sequence.Rolloff

Blockette 310 (Sine Calibration)

2.4 field	Description	This specification

3	Beginning of calibration time	EH: FDSN.Calibration.Sequence.Begintime
4	Reserved byte	[no mapping]
5	Calibration flags, bits: 2 = if set, calibration was automatic, otherwise manual 3 = if set, calibration continued from previous record 4 = if set, peak-to-peak amplitude 5 = if set, zero-to-peak amplitude 6 = if set, RMS amplitude	Extra Headers: 2 = FDSN.Calibration.Sequence.Trigger 3 = FDSN.Calibration.Sequence.Continued 4 = FDSN.Calibration.Sequence.AmplitudeRange 5 = FDSN.Calibration.Sequence.AmplitudeRange 6 = FDSN.Calibration.Sequence.AmplitudeRange
6	Calibration duration	EH: FDSN.Calibration.Sequence.Duration
7	Period of signal	EH: FDSN.Calibration.Sequence.Period
8	Amplitude of signal	EH: FDSN.Calibration.Sequence.Amplitude
9	Channel with calibration signal	EH: FDSN.Calibration.Sequence.InputChannel
10	Reserved byte	[no mapping]
11	Reference amplitude	EH: FDSN.Calibration.Sequence.ReferenceAmplitude
12	Coupling of calibration signal	EH: FDSN.Calibration.Sequence.Coupling
13	Rolloff characteristics for filters	EH: FDSN.Calibration.Sequence.Rolloff

Blockette 320 (Pseudo-random Calibration)

2.4 field	Description	This specification
3	Beginning of calibration time	EH: FDSN.Calibration.Sequence.Begintime
4	Reserved byte	[no mapping]
5	Calibration flags, bits: 2 = if set, calibration was automatic, otherwise manual 3 = if set, calibration continued from previous record 4 = if set, random amplitudes	Extra Headers: 2 = FDSN.Calibration.Sequence.Trigger 3 = FDSN.Calibration.Sequence.Continued 4 = FDSN.Calibration.Sequence.AmplitudeRange
6	Calibration duration	EH: FDSN.Calibration.Sequence.Duration
7	Peak-to-peak amplitude of steps	EH: FDSN.Calibration.Sequence.Amplitude
8	Channel with calibration signal	EH: FDSN.Calibration.Sequence.InputChannel
9	Reserved byte	[no mapping]
10	Reference amplitude	EH: FDSN.Calibration.Sequence.ReferenceAmplitude
11	Coupling of calibration signal	EH: FDSN.Calibration.Sequence.Coupling
12	Rolloff characteristics for filters	EH: FDSN.Calibration.Sequence.Rolloff

13	Noise type	EH: FDSN.Calibration.Sequence.Noise

Blockette 390 (Generic Calibration)

2.4 field	Description	This specification
3	Beginning of calibration time	EH: FDSN.Calibration.Sequence.Begintime
4	Reserved byte	[no mapping]
5	Calibration flags, bits: 2 = if set, calibration was automatic, otherwise manual 3 = if set, calibration continued from previous record	Extra Headers: 2 = FDSN.Calibration.Sequence.Trigger 3 = FDSN.Calibration.Sequence.Continued
6	Calibration duration	EH: FDSN.Calibration.Sequence.Duration
7	Amplitude of signal	EH: FDSN.Calibration.Sequence.Amplitude
8	Channel with calibration signal	EH: FDSN.Calibration.Sequence.InputChannel
9	Reserved byte	[no mapping]

Blockette 395 (Calibration Abort)

2.4 field	Description	This specification
3	End of calibration time	EH: FDSN.Calibration.Sequence.Endtime
4	Reserved byte	[no mapping]

Blockettes 400 (Beam), 405 (Beam Delay)

No mapping for these blockettes.

Blockette 500 (Timing)

2.4 field	Description	This specification
3	VCO correction	EH: FDSN.Time.Exception.VCOCorrection
4	Time of exception	EH: FDSN.Time.Exception.Time
5	Microsecond offset	[no mapping, included in record start time]
6	Reception quality	EH: FDSN.Time.Exception.ReceptionQuality
7	Exception count	EH: FDSN.Time.Exception.Count

8	Exception type	EH: FDSN.Time.Exception.Type
9	Clock model	EH: FDSN.Clock.Model
10	Clock status	EH: FDSN.Time.Exception.ClockStatus

Blockette 1000 (Data Only SEED)

2.4 field	Description	This specification
3	Encoding format	FSDH field 5
4	Word order	[no mapping, no longer needed]
5	Data record length	[no mapping, no longer needed]
6	Reserved byte	[no mapping]

Blockette 1001 (Data Extension)

2.4 field	Description	This specification
3	Timing quality	EH: FDSN.Time.Quality
4	Microsecond offset	[no mapping, included in record start time]
5	Reserved byte	[no mapping]
6	Frame count	[no mapping]

Blockette 2000 (Variable Length Opaque)

No mapping for this blockette. The opaque data encoding may be used to specify an opaque payload for nearly equivalent functionality.

miniSEED 2.4 information loss during conversion to this specification

The following defined information in miniSEED 2.4 cannot be represented in this specification:

- Clock model specification per timing exception, current specification only allows a single clock model specification per record.
- Blockettes 400 (Beam) & 405 (Beam Delay)
- Blockette 2000 (Opaque Data)