White Paper: The Future of mini-SEED

Version 1.2, July 26, 2017

Reinoud Sleeman and participants[1] of the WG2 MSEED workshop (March 22-23, 2017)

Introduction

The Standard for the Exchange of Earthquake Data (SEED) is an international standard format for the exchange of digital seismological data. The SEED format is designed for equally spaced time-series data - digital data representing measurements at one point in space and at equal intervals of time, accompanied by relevant metadata. SEED was designed

[1] Reinoud Sleeman (ODC), Philipp Kaestli (ETH) Tim Ahern (IRIS), Luca Trani (ODC), Angelo Strollo (GFZ), Chad Trabant (IRIS), David Easton (Nanometrics), Peter Danecek (INGV), Seiji Tsuboi (JAMSTEC), Lion Kircher (ETH), Philip Crotwell (USC), Claudio Satriano (IPGP), Fabien Engels (RESIF), Alexandru Marmureanu (NIEP), Edelvays Spassov (Kinemetrics), Arie van Wettum (Utrecht UNi), Andres Heinloo (GFZ), John Clinton (ETH), Mathijs Koymans (ODC).

for use by the earthquake research community, primarily for the exchange of unprocessed earth motion data, and has become the ***de facto*** standard data format for the global seismological community for acquisition, archival and exchange of waveform data. The success of SEED, in particular the light version, called mini-SEED, is also reflected by the fact that many recording systems have adopted mini-SEED as a format to record and transmit seismic waveform data, and that other communities now archive their data in this format.

The origin of SEED dates back to 1987 when the FDSN, shortly after its establishment, discussed and reviewed a number of formats, of which SEED (then a new format proposed by the USGS) was adopted as a draft standard that evolved into its official release in 1988 as version 2.0. After using SEED in practice a number of major changes were proposed to make SEED more efficient and comprehensive. Version 2.1 (1990) allowed for a more generic way to describe instrumental response information and enabled efficient indexing to the waveform data. Version 2.2 (1991) addressed the common requirement to separate header data and waveform data and split the format into so-called dataless SEED and mini-SEED. In 1992 the 2-character FDSN network code was introduced in SEED version 2.3. The release of version 2.4 in 2004 introduced, amongst others, the data quality indicator. Since then the format has not been modified, and its use has dramatically increased.

While mini-SEED has not changed, there has been a major change recently introduced for the metadata describing the seismic station. With the development of Extensible Markup Language (XML) formats, the FDSN defined an XML schema (StationXML; http://www.fdsn.org/xml/station/) to represent the metadata in SEED2.4 as close as possible. The XML schema allows flexibility to extend the metadata schema and to evolve in time and add additional information. StationXML has become the FDSN ***de facto*** standard today in the exchange of seismic station metadata. Most likely changes in the StationXML may have impacts on the miniSEED as well and vice versa. It is essential that StationXML and mini-SEED retain an ability to remain synchronized as necessary.

The mini-SEED format is adopted throughout the global seismological community (e.g. manufacturers, network operators, data centers, researchers) and is fully integrated into the current seismological infrastructure in many data centers and monitoring networks, as well as by the research community. The fact that mini-SEED has become so successful can be seen as a tribute to the original developers. Additionally, since the SEED format has been mostly static over decades it could settle deeply into our infrastructure and make seismological data exchange and availability a success. An additional reality is that mini-SEED has become crucial in low-latency, real-time communication, and this functionality should not be removed lightly in any future version although the needs of portions of seismology can not be met with a minimum record size of 512 bytes due to the time it takes to fill a packet. One approach that might prove viable for those that wish to retain mini-SEED as a streaming method for data transfer is to have a straightforward mapping of the current version of mini-SEED to a new version adopted by the FDSN allowing the current version of mini-SEED to continue being used in real-time communication purposes.

Despite the success of the current mini-SEED format and the strong dependency of today's seismological infrastructure on mini-SEED, there is a need to explore whether and how the mini-SEED format could be modified, chiefly because of a) new challenges in the identification of data streams in already occurring in huge and complex network configurations and b) new technological developments.

In February 2016 this discussion was initiated in the WG2 mail list by IRIS, after consulting the WG2 chair. This was followed by a proposal by IRIS ('Strawman') that was discussed during EGU in 2016 (Vienna, April 17-22). Numerous e-mails were sent to the FDSN WG2 mailing list showing interest in considering a modernized mini-SEED format that would address future needs. Valuable input was received from the community but there were significant concerns related to, in particular, backwards compatibility as the current format is strongly embedded in today's seismological infrastructure. Valuable input on a variety of new characteristics needed in a new version of mini-SEED Following the support by the FDSN Chair for a meeting in late 2016 to continue the process, ORFEUS organized and hosted a FDSN WG2 (Data Formats, Data Centers and Software[2]) meeting in The Netherlands (Utrecht, 22-23 March 2017) to discuss and agree jointly on the process to design and adopt an extended or new standard, with topics:

- Motivations and vision for a new format
- Identifying current needs for a new format
- Technical Proposals for a new format
- Evaluation of impact and tentative implementation plan

This white paper is the outcome of this FDSN WG2 meeting and reflects the ongoing community discussion with the FDSN WG2. It provides (a) an overview of the motivation to explore changes to mini-SEED (Problem Definition), (b) a wish-list of new features and modifications (Requirements List), (c) an overview of proposed solutions (Analysis of Proposed Solutions) and provides (d) a proposal Timeline for Implementation, (e) an Impact Assessment and (f) Call to Action.

The white paper will be:
- Submitted to the participants of the 2017 meeting for review (in particular the requirements list) and feedback.
- Submitted to the FDSN WG2 mailing list for review and feedback
- Presented to the FDSN community at the IASPEI Assembly in Kobe, July/August 2017 with the goal to agree upon an implementation scenario to map the way forward. A

[2] http://www.fdsn.org/message-center/topic/fdsn-wg2-data/

tentative scenario is outlined in the white paper, though of course this will depend on the particular technical proposal selected for implementation by the FDSN.

The reality now is that mini-SEED has become a standard for both real-time data streaming and data sharing as well as permanent archival of waveform data. It appears that any new format should continue to support these functions, even though they may not lead to an optimal format. For convenience, any proposed new format is called MS3 throughout the document.

Note by the Chair:
the motivations for changing mini-SEED basically are a) the running out of available identifiers for temporary and permanent networks in 5+ years and b) the manageable identification of dense, complex networks. Are large N sensor networks becoming 'standard' for the next decade(s) or are these special cases popping up now for just a relative short term? Some of the major data centers are confident that deployments of very large numbers of sensors and the support for these needs to start being supported within the FDSN to meet these needs. Is this becoming common practice for the coming years (science plans, allocated budgets etc.)? During the meeting, Tim Ahern showed a number of cases of dense networks in the US that are emerging. IRIS is planning and budgeting for these Large-N experiments now and believes that within 5 years the availability of these relatively inexpensive sensors will proliferate over the next few years. I really like to stress this question, not to hinder or slow down the discussion, but to make sure that a change with a potential big impact is lead by a structural change in monitoring network configurations in the future. Are there known plans outside the US for such large number networks ? Other FDSN members at the meeting in the Netherlands also brought up possible deployments of large numbers of inexpensive sensors such as smartphones and inexpensive strong motion sensors that support the need for the new format to support many more sensors than are currently feasible in current mini-SEED. Does the FDSN have interest in collecting, archiving and serving data from inexpensive sensors, possibly of lower quality ?

Another important question is whether the FDSN would support a disruption of the format. Here the opinions are strongly diverging. Is the FDSN convinced that a disruption of the format brings more benefits than disadvantages ? The price that must be paid for introducing an incompatible format, which solves one requirement (identification) plus a number of 'nice to have' features, seems expensive as the current format is so deeply embedded in our global seismological infrastructure. FDSN needs to take a decision here: are we willing to invest a lot in our time and manpower for this ?

It is important that a disruption caused by a new format, if accepted by the FDSN, is minimized. This can be done if a simple conversion from current mini-SEED to a new version of the format is lossless and straightforward and a utility developed that supports this. Such a capability would allow networks and data centers to continue using their current format and infrastructure until they have need to make use of the new capabilities offered by a new version of mini-SEED.

Problem Definition

Mini-SEED has been used for over two decades to permanently archive data streams from both permanent and temporary seismic networks. Currently, a typical permanent network consists of tens to hundreds of stations, some with multiple sensors, that are typically separated in distance by a few to thousands of kilometres. Current directions in temporary deployments of sensors are now in the thousands with much closer sensor positioning, sometimes sub-meter. It is clear that future deployments will consist of considerably greater numbers of sensors. The SEED standard includes rules for how streams should be identified. The identification of data streams is based on 5 codes with a fixed number of alphanumeric characters: Stationcode (up to 5), Networkcode (up to 2), Channelcode (3 characters), Locationcode (up to 2, though often left blank), and Qualitycode (1 character, often left default).

However, network configurations are becoming increasingly more complex, including much larger numbers of sensors of various types spaced at shorter distances. There are significant challenges in managing and identifying these data streams in a logical and maintainable way within the current SEED constraints. ***Example 1***: current 'large N' temporary experiments are deploying more than 1000 stations at locations spanning a small area. Our convention to assign different station codes to sites that are spatially separated by more than 1 km, allows a maximum of 'only' 100 different sensor locations (uniquely identified by the Location code) within a 3D array in 3.14 km^2. ***Example 2***: in practice, SEED has proven capable of managing data from many different sensor types, and is being used for monitoring applications extending outside seismology. However, the limited number of sensor types supported by the naming convention in SEED by the single character code in the Channelcode makes further expansion towards other sensor/instrument types very challenging. ***Example 3***: An international registry is set-up to manage unique network codes. With the proliferation of permanent and temporary seismic networks, the registry will run out of Network Codes in the near future (estimated order of 5 years).

In practice, 'best practice' approaches to overcome these hurdles are being defined by various communities. But in order to overcome the limitations of identification due to the fixed, limited identifier codes, a fundamental change in the identification of data streams seems required. Adding one or more characters to each code is a solution for current challenges but is not future proof (scaling issue). Such a solution, however, could break with the current mini-SEED format and thus leading to version incompatibility. However, after thirty years, it should not be surprising that mini-SEED is reaching its limits.

The discussion on the need for changing mini-SEED and in which direction the community should go are steered by the following questions:

- What is the scope of FDSN? Does the FDSN envision/support the need to prepare our infrastructure for large N experiments, and other more complex experiments that may go beyond seismic data ?
- Can we solve the identification limitation without major overhaul ? What are the solutions and the impacts ?
- What are the benefits and drawbacks of each solution ?
- Do we accept version incompatibility ?
- Must a new data format also support real-time, low latency data sharing, for example by useful for applications in Earthquake Early Warning ?

Requirements List

During the FDSN WG2 workshop a number of requirements/wishes were presented and discussed. The requirements can be divided into the following categories: new features and simplification/reorganization issues:

New features:

- Revision of the stream identification limitation to support large numbers of instruments in various configurations (i.e. large number of sensors, dense sensor spacing, different sensor types).
- New encoding types (general, opaque, fixed-point) for example to support other compression techniques (e.g. ***32-bit integers, general compressor; 32-bit IEEE floats, general compressor; 64-bit IEEE floats (doubles), general compressor; Opaque data, general compressor***).
- Adding of CRC (cyclic redundancy check) to detect accidental errors in the data by validating the integrity of a record.
- Include both a format and a data version number.
- Variable length records are required for efficiency and flexibility (e.g. real time streaming with low latency).

Simplification/reorganization:

- Move selected blockette details into fixed header.
- Remove blockette support, add optional (opaque) headers.
- Simplify/improve record start time.
- Eliminate time correction field.
- Combine and drop bit flags.
- Eliminate sequence numbers.

With general guiding principles:

- Jointly address the evolution of StationXML to optimize and complement the new format (e.g. see the initiative by ETH to start a discussion on the Proposal of a major revision of StationXML).
- Continue to support both realtime, low latency streaming as well as permanent archival.
- Keep critical recording details attached to the data samples (e.g. actual sample rate, byte order, record length, data encoding, microseconds).
- Ensure that current, important mini-SEED information can be mapped into the new format.
- Address needed and desired enhancements in addition to inefficient and unnecessary or inconvenient historical artifacts.
    - Allow for unorganized instruments (no central registry, no hierarchical organization, self-identification of data streams).
    - Allow for different time scales of instrument/network operation.
    - Allow for different sampling rates (<< 1 Hz to order of MHz).
    - Allow for non-raw, derived data (e.g. processed data, quality parameters, metadata versioning, synthetic data).
    - Acceptance of version incompatibility should be used to fix current. inconveniences (e.g. removal of non-used bits).
- Do not introduce ‘solutions’ that may become new limitations in the future (be future proof). When designing a new data format, it is however not possible to foresee all future uses. Therefore extensibility is a very important feature of a data format.
- Turn best practices and conventions in identification into well documented standards.
- Provide a clear migration path moving forward from SEED 2.4, that includes support for both versions for some time yet encourages teams to migrate to the new format.

Analysis of Proposed Solutions

A number of different solutions have been proposed at the FDSN WG2 meeting in the Netherlands, varying in complexity and extent of disruption to the community. Two of the solutions were heavily discussed on the FDSN WG2 mailing list in advance of the meeting: the IRIS ‘Strawman’, using a major re-organisation of the mini-SEED headers, and Option 2 (described in the following paragraph), a lighter approach that uses the inherent feature of SEED to extend the format by its blockettes. However, due to resistance within the community the ‘Strawman’ proposal was not supported further. After the workshop in Utrecht, IRIS reconsidered a refined ‘Strawman’ proposal which is described below as Option

1. Option 3 and 4 were developed during the meeting, and a final solution, Option 5, which includes hierarchical structures inside containers, has been proposed in the scientific literature as a more general solution for describing seismological datasets. During the workshop in Utrecht it was decided to open a github repository to further evaluate the newly proposed tentative solutions based on brainstorming as well as allow for additional proposals/comments from the wider WG2.

Note from the Chair:
One option that was not discussed is to not change mini-SEED at all and assign one (temporary) network code to the large N sensor network and a unique station code to each sensor (more than 10x10^6 possibilities). This however would not solve the running out of network codes in the near future. Also, the need to extend instrumental codes to allow new types of sensors would not be solved.

Each option is described below, with technical details in the corresponding appendix. Note that the ranking of options is arbitrary and does not reflect any order of preference.

Option 1 (proposed by IRIS):

IRIS proposed a strawman format in April 2016. Several organizations commented on this proposal, many of which helped to improve the concept. However, the original strawman proposal was rejected due to lack of community support. After the discussions at the FDSN WG2 meeting in Utrecht the 2016 IRIS Strawman was reconsidered by IRIS and combined with the feedback received, the "stream identifier" concept discussed at the Feb. 2017 meeting, and refinements identified by the technical evaluation group participating in the discussion in the github repository, resulting in this new proposal which was submitted after the Utrecht workshop. This proposal has resulted in two documents that contain a nearly complete specification with only a few issues not fully addressed, to be found at:
https://github.com/iris-edu/mseed3-evaluation/wiki/Simple-Header:-Strawman,-plus-feedback,-plus-URN-identifiers

Option 2 (proposed by ORFEUS; contact Reinoud Sleeman):

Summary

This option proposes forward and backward compatibility of the current format and uses the existing extension mechanism available in mini-SEED by means of blockettes. It introduces a new blockette, 1002, that enables the implementation of new features (e.g. as listed in the requirements list) that is in line with the existing extension mechanism.

Format extensibility provides forward compatibility (old versions of software can read new versions of the format, albeit ignoring some features) and facilitates backward compatibility (new versions of software can read old versions of the format). We can distinguish first-class and second-class extensions:

- ***First-class*** extensions are "equal" to core features. Example: adding new tags and attributes to an XML schema.
- ***Second-class*** extensions are inferior to core features. Example: using proposed optional (opaque) headers in mini-SEED.

A mini-SEED record consists of a fixed header, a linked list of blockettes and waveform samples. Perhaps putting too much information into the fixed header was a design flaw of mini-SEED; some, or even most of this information could have been implemented as blockettes. A blockette is, however, an excellent ***first class*** extension mechanism. This mechanism can be used to implement an improved identification of data streams in mini-SEED with (limited) forward and fully backward compatible.

<u>Expanding the stream identification</u>:
Instead of fixed-length NSLC, we suggest a list of 4 variable length fields separated by a special character (tilde). Additional fields (4+), a numeric "type of stream ID" or a textual prefix like "nslc:" (default) could be allowed for future purposes.

<u>New encoding types</u>:
In mini-SEED 2.4, the encoding type of waveform samples is a numerical code in blockette 1000. Types 1..5 (general), 10..18 (FDSN networks) and 30..33 (other networks) are defined. New encoding types can be added to this list. Obviously, old software would not be able to decode such data.

<u>Variable length records:</u>
The feature of variable length records is the only requirement that cannot be implemented using a blockette. On the other hand, it can be argued whether variable length records justify an incompatible change of format, or are reasonable at all. For data archives, assuming sorted records with fixed length it is very efficient (~two seeks) to find the start position of a time window, while this is not the case in case of variable length records;

In the case of real-time data transfer the use of small (variable length) records would help to reduce latency, but the large record header would waste bandwidth and thus affect latency. A better alternative could be the progressive transfer of (fixed length) records, which is currently not possible with mini-SEED due to fields in the fixed header that cannot be computed before complete record is available.

Pros and cons

- Using "blockette 1002" it is possible to implement all proposed new features, except the variable length records, without breaking compatibility with existing software (huge cost saving).

- There is a lot of software that does not require the new features and may not have to be modified at all; the remaining software can be upgraded in much slower pace.
- Unfortunately some space would be wasted due to duplicated and unused (or rarely used) header fields, but the benefits outweigh the drawbacks.
- The risks (eg., using wrong network code because of outdated software) are negligible.
- Software to write and read mini-SEED must be extended to use blockette 1002.
- The simplification/reorganization issues are not touched as to guarantee format compatibility but must be subject for clearly defining how to use these.

Further details in Appendix B

Technical details of this proposed solution of the introduction of blockette 1002 are provided in Appendix B. If this option is selected it enables the implementation of new features, except for the variable record length.

Option 3 (proposed by Andres Heinloo, GFZ):

Summary

If extension of mini-SEED with additional blockette(s) is not sufficient and a complete new format is required, we must keep in mind that one purpose of mini-SEED is long-term data archival and the data must remain unambiguously interpretable after decades (if not centuries). Obviously, the format has to be extensible, but extensions must be well defined and well documented, as is the case with the current mini-SEED blockettes. Extending the format with, for example, ad hoc key-value pairs (e.g. JSON data) could be useful for data transmission and short-term storage, but not for long-term archival. Also, it would be wise to not depend on complex external formats, which may become obsolete sooner than mini-SEED itself.

This option advocates to reuse and extend the concept of blockettes. A problem with mini-SEED is that a lot of information was added to the fixed header that later became obsolete and thus wasted space in a record. In a new format, we might keep the fixed header very minimal and put almost all information into blockettes or “chunks”. In this case, a version number might not be needed as new revisions of the format can add new chunk types and advise against using obsolete ones. To ensure future proof and correct interpretation of the chunk types a centralized registry is required.

The extensions need to be registered in order to be interpretable. There would be the following ways to add an extension:

- Add the extension to the FDSN standard.
- Register the extension with one of the organizations, such as IRIS or EIDA.

- Register the extension with one of the listed manufacturers.
- Use opaque chunk (126) for simple ad hoc key-value pairs
- Generate an UUID (https://en.wikipedia.org/wiki/Universally_unique_identifier) and use generic chunk (127)

Pros and Cons

- The concept of blockettes is re-used to ensure future proof flexibility
- A central registry for the blockettes, or data chunks, ensures correct interpretation
- This option would completely break with the existing format

Further details in Appendix C

Appendix C illustrates the technical description and an example of this concept to re-use the concept of blockettes in a new format. Additional details are available also on the github repository (https://github.com/iris-edu/mseed3-evaluation/wiki/Chunks).

Option 4 (proposed by Philipp Kästli, ETH):

Summary

This proposal aims to provide a minimalistic, multipurpose data format for evenly sampled single attribute time series with the maximal flexibility to support:

- sampling intervals ranging from nanoseconds to years; entire time series can cover billions of years
- organization in records consisting of fixed size fields in header and footer, and variable size fields for stream id and data. Record size ranging from ~64 bytes to 4 GBytes
- variable & extensible stream identification scheme
- agnostic to data content (observable type (real or simulated), processing history)

The proposal fully separates data from metadata (provenance, instrumentation, response, data quality and processing history), and connect these via a flexible, globally unique URI identification in the data stream. It is insensitive to reformatting (encoding, compression, record size) allowing support for different purposes (real-time transfer, archiving, ...).

The records would consists of a header, data and a footer:

[**header**]: all items ordered

Format version & type	fixed size
High resolution time stamp	fixed size
Sampling rate	fixed size
Record size	fixed size

Data type	fixed size
Compression type	fixed size
Data start offset	fixed size
Stream identifier	variable

(any URI, mseed legacy bridge e.g. with: sncl:net~sta~loc~chan~version)

[**data**]

DATA size: record size – fixed header size - offset – footer size

[**footer**]

Number of samples	fixed size
CRC or other integrity	fixed size

Other "stuff" can be stored with a different record type, with stuff added in DATA (extension strategy for covering legacy mseed timing and data quality flags).

The characteristics of the proposed data format are
- record size: about 64 bytes to 4 GBytes
- volume size: 1 record to unlimited
- time series payload: 1 byte up to about 4 GBytes per record
- overhead: starting from ~64 bytes per record (54 bytes of fixed headers)
- record size can be variable, fixed in storage size, or fixed in time interval

Pros and Cons

Pros: the proposal is extremely flexible for future changes / scalability, supporting
- large number of sensors/instruments
- unorganized instrument (no central registry required [however may be implied by stream ID], no hierarchical organization, self-identification of data streams possible, managerial and spatial information in metadata)
- dynamic network configuration (both in time and place)
- wide range of sampling rates sample rates
  - For different waves in different media (e.g. geochemical sensors)
  - For different spatial scales (e.g. globe: 10k km, < 1 Hz; crust. 25-500 km, 100-250 Hz; in-situ injection experiment, 50m, 20 kHz; rock lab, 5 cm, 20 MHz)
- non-raw data possible (metadata external to the time series format)
  - Defined quality control
  - Versioning metadata
  - Time window operations (PGA…)
  - Multi-channel processing (stacking, cross correlation traces etc.
  - Synthetic data
- new content types
- Generic:

    - Low latency streaming / processing
    - Low archiving overhead
    - Fast seek (e.g. fixed block size)
    - Long lifetime
    - no (implicit) software dependency
    - no transfer protocol dependency
    - Easy to find and read the actual time series samples
- A valid and identifiable time series may be generated ad hoc, without much metadata present. Metadata may be amended/maintained without requiring to change the data records.

Cons:

- the method is a complete re-write, requiring changes everywhere. It would be challenging to convert existing archives / metadata to this format.
- The lack of clear header information beyond the stream identifier can lead to lost / orphaned / unidentified data in data centers where the identifier information and the metadata is not properly handled. This makes it challenging to implement in certain environments, and may hinder ultimate adoption of the method

Possible implementation plan

The proposal documents a completely new format. While it is compatible with current station XML, it would be fully leveraged only by a next generation station XML (as the current metadata format does not allow for unorganized and self-identified streams). As the format is technically not backward compatible (current mseed is not a special case of the new format, and no flavour of the new format is valid mseed), It would be a long term strategy to implement such a solution, with temporal overlap of maintaining both formats for legacy software.

Further details in Appendix D

Technical details of this proposed solution are provided in Appendix D.

Option 5 ASDF (proposed by Lion Krischer, ETH):

An interesting recent proposal that can serve as an inspiration for design of next generation of mini-SEED. Can be viewed as a future solution to replace and extend full SEED (i.e. combine next generation mini-Seed with StationXML or other inventory format with event or extended site information).

ASDF, (Adaptable Seismic Data Format; https://doi.org/10.1093/gji/ggw319), is a modern and practical data format for all branches of seismology and beyond. ASDF is designed to resolve key issues:

- **Efficiency,** mainly in terms of data operations (processing, analysis) in scientific workflows. More efficient and better performing data processing and analysis tools are badly needed.
- **Data organization** - different types of data are needed for a variety of tasks. This results in ad hoc data organization and formats that are hard to maintain, integrate, reproduce, and exchange
- **Data exchange** of complex data sets.
- **Reproducibility is a** critical aspect for science. Often not existent to do and should be strongly encouraged.
- **Mining, visualization and understanding of data:** As data volumes grow, more complex, new techniques to query and visualize large data sets are needed.

ASDF, at its most basic level, organizes its data in a hierarchical structure inside a container—in a simplified manner a container can be pictured as a file system within a file. The contents are roughly arranged in four sections, as follows.

- Details about seismic events of any kind (earthquakes, mine blasts, rock falls, etc.) are stored in a QuakeML document.
- Seismic waveforms are sorted in one group per seismic station together with meta information in the form of a StationXML document. Each waveform is stored as an HDF5 array.
- Arbitrary data that cannot be understood as a seismic waveform is stored in the auxiliary data section.
- Data history (provenance) is kept as a number of SEIS-PROV documents (an extension to W3C PROV).

Large parts of the ASDF definition are independent of the employed container format. An advantage of this approach is a certain resilience to technological changes as major pieces of ASDF can in theory be adapted to other container formats. Nonetheless, the container format has to be fixed to not severely affect interoperability and ease of data exchange. We evaluated a number of possibilities and chose HDF5 (Hierarchical Data Format version 5).

Container
• HDF5
• The de-facto standard for binary array data
• Tools and libraries for essentially every language
• Built-in data compression, check-summing
• Parallel I/O via MPI
• Journaling should be in the next HDF5 version

Waveform data:
• Sorts waveforms at a per-station granularity
• One StationXML file per station

• Arbitrary number of arbitrary big waveforms per station
• Links to event information and provenance

Auxiliary data
• Anything that is not a seismic waveform
• Always a data array plus any number of key/value pairs as meta information
• Arbitrary nesting
• Files/Cross Correlations/Receiver Functions/…
• No hard definition but link to provenance

Provenance:
• SEIS-PROV, based on W3C PROV

Note that ASDF is not presented here to change mini-SEED but as a flexible container format to incorporate our above requirements. This thus may include an extended/modified mini-SEED format.

Proposed Implementation Timelines

- Define the new target format (based on the Requirements List once reviewed by the FDSN WG2 and testing outcomes of selected prototype implementations).
- Establish a working group to work out and fully document details of the format.
- Establish a timeline that is realistic to move to the new format
- Inform the community about the new format and the timeline
- Develop tools and systems (for users and data centers) before beginning the conversion of the mini-SEED
- Establish a target time for the process to be completed (until then centers can continue using current formats)
- Finish within N years (N is <5 years but TBD), then stop support of current mini-SEED

Next Steps:

Mid - July 2017	Email: White Paper distributed to WG II Request comments in time for FDSN WG II meeting
2 Aug 2017	FDSN WG II meeting at IASPEI, Kobe. Discussion of white paper; propose the path forward to FDSN plenary

4 Aug 2017	FDSN Plenary meeting at IASPEI, Kobe. Vote on path forward

A proposal for the Path Forward:

Sep-Dec 2017	Review of the Requirements list. Proposed Implementation Timelines and general comments to the white paper.
Jan-Jun 2018	Setup of Technical Working Group (4-5 p) to implement draft realisations of each proposal. Provide technical report evaluating performance based on agreed metrics. (Alternative proposals still can be added to the public github repository). Note: a start of a technical discussion and evaluation already started (https://github.com/iris-edu/mseed3-evaluation).
Jul-Dec 2018	WGII community discuss and agree preferred proposal.
2019	Technical Working Group work on the definition of the selected new format, first draft in Summer 2019. Includes both descriptions of the new data format and any extensions required to be compatible with stationXML / provenance. DC managers Working Group to provide an updated implementation plan and impact assessment on the preferred proposal.
2019/2020	- further rounds of feature requests - work on defining stationXML3 compliant with MS3 - open discussion on new labels for SNCL / naming standards (not conventions) - funded work on new documentation for MS3
2020/2022	- development of software to use MS3 / convert between mini-SEED2.4 and MS3 - assignment of first longer network codes / - support of MS3 from dataloggers, streaming softwares, processing softwares, dissemination tools (eg fdsnws_dataselect) (supporting both versions)
2023	- IRIS start standard use of MS3

Impact Assessment

The mini-SEED format is fully integrated into our current seismological infrastructure and is therefore of high importance in many data centers and monitoring systems, as well as for the research community. Decades of (both technical and financial) support to SEED by the seismological community and many years of experience and 'best practices' in using SEED by all stakeholders have made SEED a common 'language' being used throughout the global seismological community and beyond. Therefore the introduction of a new format, or a significant change of the existing format, may have a big impact on local, regional and global seismological infrastructures, real-time hazard monitoring systems, data center operations and services for the research communities. Coping with such a significant change may put high demands on resources (manpower, finances) even before adapting our current systems to a new format. In order to successfully introduce a new format, or a format change, all risks of such an introduction must be discovered, identified and properly managed by individual stakeholders.

At the same time we need to acknowledge that most community software relies on a few key libraries/tools, so the community infrastructure, other than things that are one-of in nature can be largely addressed in the key analysis tools such MatLab, SAC, ObsPy to name a few. For homegrown software changes needs to be made.

In order to attempt discovering and identifying the risks of the introduction of a new format we divide the seismological infrastructure into a number of stakeholder groups.

<u>Funding agencies</u>
The introduction of a new format will require funding to change and adopt the current system. Resources must be estimated and timely allocated by all stakeholders to support and implement any change.

<u>Manufacturers</u>
The chain of data production starts here, therefore any FDSN decision on the next generation data format for seismological data must be supported by the equipment producing it. Manufacturers must be involved in the design process to investigate whether any new format is technically feasible and can be implemented according to any FDSN standard within an agreed timeline.

<u>Software developers</u> (e.g. acquisition protocols & analysis (SeisComP, Antelope, ...), community tools (ObsPy, ...), ...). Format changes can have a large impact on today's, existing and broadly used software tools and packages (real-time protocol, operational analysis systems, manuals ...). When the target format is technically not backward compatible (current mini-SEED not being a special case of the new format, and no flavour of the new format is valid mini-SEED), it must be a long term strategy to implement the target format, with temporal overlap of maintaining both formats for legacy software. A comprehensive inventory of affected software must be made.

Seismic observatories (e.g. national organizations and networks, regional and global cooperation, …). Often daily work at observatories depends on a mixture of commercial, Open-source and in-house developed software and scripts that are connected and configured based on years of experience and hands-on practice. A change of data format may have a huge impact in hundreds of seismic observatories that rely on mini-SEED. The timely implementation and testing of the new format in existing community tools, together with a comprehensive manual for the new format and a communication plan are crucial for quick acceptance throughout the community.

Data Centers (e.g. IRIS-DMC, ORFEUS-EIDA, …). Data centers usually have complex structures to simultaneously acquire, process, archive and serve data through a variety of services and tools, and can be organized as a single data archive center or being part of a federated structure. A format change will have an impact on every pipeline in the operations and require additional resources for adopting a new format and maintaining both 'old' and 'new' formats operational simultaneously for some time.

Research community (e.g. universities, …). Possibly this community is less affected by a disruptive change as long as the services to collect data are compatible and software tools to read data are available.

Appendix A - Technical details option 1

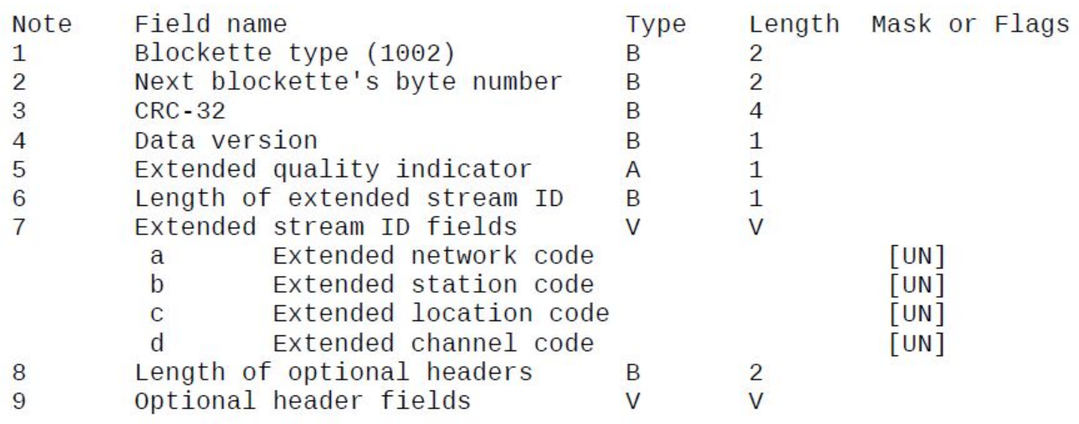
Two documents describing the technical details of the renewed strawman proposal are to be found here:

https://github.com/iris-edu/mseed3-evaluation/wiki/Simple-Header:-Strawman,-plus-feedback,-plus-URN-identifiers

Appendix B - Technical details option 2

Blockette 1002

The proposed solution is the introduction of blockette 1002 that enables the implementation of new features (see above), except for the variable record length:



Note	Field name	Type	Length	Mask or Flags
1	Blockette type (1002)	B	2	
2	Next blockette's byte number	B	2	
3	CRC-32	B	4	
4	Data version	B	1	
5	Extended quality indicator	A	1	
6	Length of extended stream ID	B	1	
7	Extended stream ID fields	V	V	
	a Extended network code			[UN]
	b Extended station code			[UN]
	c Extended location code			[UN]
	d Extended channel code			[UN]
8	Length of optional headers	B	2	
9	Optional header fields	V	V	

Notes:
(3) CRC is to be calculated over the entire record with the CRC bytes assumed to be zero for the purposes of calculation
(7) Each field is a variable length ASCII string, terminated by the tilde character (ASCII 126). Each code must either be equal to the respective code in the fixed header or the code in fixed header must be replaced by a special (reserved) value.
(9) Optional, ad-hoc header fields may be added if necessary. Each optional header field is a variable length string, terminated by the tilde character.

Appendix C - Technical details option 3

In this proposal to reuse and extend the concept of blockettes in a new format a record comprises a (minimal) header, followed by pieces of information, including data samples, called “chunks” or blockettes. The fixed header is very minimal and almost all information is stored into the “chunks”.

```
Record
        Minimal header
        N Chunks
                type
                length
                data
```

The minimal header only contains a magic string to identify the data format, and maybe the record length. Example:

Minimal header = NDF4096

Then a series of chunks follow, for example:

```
        // Different types of channel ID supported
        Chunk type = ID_NSLC
                GE~WLF~~BHZ
        // Time; can be absolute and/or relative
        Chunk type = ABS_TIME
                2017/03/23 8:00
        // Different encodings of waveform data supported
        Chunk type = STEIM2
                data samples
        // Also different types of non-waveform data can be embedded
        Chunk type = OPAQUE
                KEY1=VALUE1~KEY2=VALUE2
        Chunk type EVENT_FLAGS
                ...
        Chunk type DATA_QUALITY
                ...
        Chunk type = TIMING_QUALITY
                ...
        // All mini-SEED 2.x blockettes can be included
        Chunk type = MSEED2_BLK
                ...
        Chunk type = PROV
                ...
        Chunk type = CRC
                ....
```

Chunk type = SHA2
....

Variable length integer encoding might be used for chunk type and length, such that commonly used chunks take less space, but the number of different chunk types is still unlimited. See Python example at https://gist.github.com/andres-h/d6edcc9ebd16c1d30191fe1c4434f610

Example allocation of chunk types and pre-defined id's:

- 0...999999 reserved for organizations
  - 0...99999 reserved for FDSN standard
  - 0...127 essential chunks (1-byte ID)
    - 0: special purpose
    - 100: timing quality
    - 126: opaque (tilde-delimited strings)
    - 127: generic (UUID-based)
  - 128..16383 important chunks (2-byte ID)
    - 1000...2000 mini-SEED 2.x blockettes (deprecated)
    - 1000: blockette 1000
    - 1001: blockette 1001
    - 1100: blockette 100
    - 1200: blockette 200
  - 100000..199999 reserved for IRIS extensions
  - 200000..299999 reserved for EIDA extensions
- 1000000...1999999 reserved for manufacturer extensions, e.g.:
  - 1000000...1009999 Quanterra
  - 1010000...1019999 Nanometrics
  - 1020000...1029999 Guralp
  - 1030000...1039999 Kinemetrics
  - etc.

Please consult the following pages for some more information:
https://github.com/iris-edu/mseed3-evaluation/issues/14#issuecomment-314811414
https://github.com/iris-edu/mseed3-evaluation/wiki/Chunks

Appendix D - Technical details option 4

Design goal:

A minimalistic, multipurpose data format that is:

- adequate for equally sampled time series of sampling intervals from nano-seconds to years;
- flexible with regards to sampling rate, record size, identification scheme – thus flexible in its application
- agnostic of data content (observable, processing history or simulation context)
- separating data from metadata (provenance, instrumentation, response, data quality and processing history) by a flexible globally unique URI identification of a data stream
- insensitive to reformatting (encoding, compression, record size) for different purpose (real-time transfer, archiving, ...)

Characteristics of the proposed data format:

- record size: about 64 bytes to 4 GBytes
- volume size: 1 record to unlimited
- time series payload: 1 byte up to about 4 GBytes per record
- overhead: starting from ~64 Bytes per record (54 bytes fixed header)
- record size can be variable, fixed in storage size, or fixed in time interval.

Proposed record structure:

Length in bytes	position	Content	Definition	Notes
- fixed header section -				
2	0	Some type of record format identifier	Fixed value, e.g. 01	1)
4	2	Record size (m)	4 bytes unsigned integer Allows for records up to 4G	2)

14	6	Bytes / content 4 (signed long) / Year (e.g., 1987) 2 / Day of Year (Jan 1 is 1) 1 / Hours of day (0—23) 1 / Minutes of hour (0—59) 1 / Seconds of minute (0—59, 60 for leap seconds) 5 / Multiples of .000‘000‘000‘001 seconds	Timestamp of the first sample	3)
8	20	IEEE double precision floating-point value	Positive values: Samples per second Negative values: sampling period in seconds	4)
2	28	Unsigned Integer	Data type	5)
2	30	Unsigned Integer	Offset in bytes of the beginning of data (= length of variable header section	
		- variable header section -		
n	32	String, representing an RFC 3986 URI	Identifier of the data stream.	6)
		- data section -		
m – n - 54	32+n		Time series data, according to data type	

- fixed footer section -				
6	M – 22	Unsigned 6 bytes integer	Number of samples	7)
16		128 bit checksum	MD5 checksum, calculated over bytes[0...M-17]	8)

1) Will allow (future) data only files with different record type (different header structure).
2) Absolute record length in byte (compared to an index of two in mini-SEED) preferred: thus, no record padding is required, and given the data format allows linear reading, this may be transferred incrementally, and interpreted without the number of samples available yet.
3) Note that with sampling rates of >> 10 MHz e.g. in rock sample analysis, time resolution should be higher than 10^{-9} sec.
   - Note that the current timestamp representation allows for representation of time series of ~ ~4 billions of Years; allowing for long-term modelling in many fields of sciences, however excluding some application cases of astrophysics.
   - À discuter: in such a traditional, absolute, calendar-based time format, time is not fully continuous (leap seconds). Using an epoch (e.g. seconds before since 1. 1. 1601 or 1. 1. 1970) & relative time would avoid this problem, making it more easier to process the data (but more complex to write it, if the clock, such as GPS, provides an absolute time stamp)
4) Rationale for sign: allows to express a wide range of sampling rates typically without rounding.
   Rationale for precision: dual precision value makes sure that at the maximum record size, uncertainty in the sampling rate does not disallow to assess whether or not two records are exactly adjacent.
5) Note: the list of data types defined in SEED (SEED manual Appendix G) may be insufficient; especially higher number of significant bits for uncompressed data, and other compression schemes required. This topic requires input from the community and experts, however, it is widely independent from the decision on the data format structure, and the migration from mini-SEED to a generic time series format.
6) The namespace of the URI is defined by its schema prefix (doi, http, ...) ; for legacy seismic data streams with legacy identification of ~~IRIS/ISC~~ FDSN registered network, station, and seed stream ID, one may use the following stream ID convention:
   sncl:net~sta~loc~chan[~version[#localExtension]].
   This permissive identification scheme allows virtually everybody to issue his own stream IDs (e.g. DOIs referring to simulated data streams in the context of a specific research or publication), or new identification standards to be introduced later without compromising the storage format of the time series.
7) Note: with maximum record size, this allows for a maximum compression rate of a factor of roughly 2^{16} without padding. If the number of samples is omitted (000000), it can/must be derived from data, assuming no padding.
8) ***À discuter***: Alternative solutions like sha2/256bit or sha2/512bit would introduce better data integrity control, but more computational and storage overhead. However, the lifetime of a hashing standard being capable to ***fully*** ensure data integrity is probably lower than the intended lifetime of the format.

Considerations on miniseed features not covered in the new format:

- ***Data quality:***
  D(unknown) / R(raw) / Q(uality controlled)/M (data center modified) of classical miniseed is not supported. As these 4 classes are not defined, it is considered virtually useless in a context of data exchange
- ***Activity flags:***
  Bit 0 and 2 to 6 in standard seed refer to a point in time, or a time interval within the record, without defining this/these time(s). As a result, information is loose, and re-encoding of the data with changing record size is irreversible and includes information loss.
  Bit 1 applies to the start of the record, but this information cannot be consistently maintained over changes of the record size (as the reference point of time may not be explicitly flagged in the new record).
- ***I/O and clock flags:***
  Bit 0-2 information is available from context in the new format.
  Bit 3 and 4 are not supported any more.
  Bit 5 (clock lock) refers to the current record start, which, after re-coding, may not be referenced any more. Thus, the information is not stable with reference to re-encoding. It should be kept within the metadata.
  Much of this information on data treatment, quality, and state of health is more consistently represented in metadata frameworks such as the one of IRIS Mustang or EIDA WFcatalog.
- ***Binary representation:***
  Encoding format and word order are not described within the data files; they should be fixed by convention.

Implications for the development of a seismic station/response metadata format

For the time being, a conventional URI with net/sta/loc/chan information guarantees the linking between miniseed-NG and both StationXML and Full & Dataless SEED. Having an URI as a generic stream identifier in the next generation waveform format sets the boundary condition for the next generation metadata format to use the same ID type for stream identification. Having the same type of generic ID in both formats will allow to leverage both for usage also with short-term deployments, simulated data streams and other applications outside of the domain of classical network seismology.

An extension for adding more legacy information

If there is a strong requirement for maintaining legacy miniseed header information (data quality, activity, IO, and clock flags, and maybe more opaque or legacy data such as seed blokettes) for classical seismological applications of the new time series format, an additional record type may be designed with the following definition:

Length in bytes	position	Content	Definition	Notes
- fixed header section -				

2	0	Some type of record format identifier	Fixed value, e.g. 02	1)
4	2	Record size (m)	4 bytes unsigned integer Allows for records up to 4G	2)
14	6		Reference timestamp	3)
8	20	IEEE double precision	Duration in seconds	
1	28	Byte	D/R/Q/M miniseed quality flag	
3	29	3 bytes	Activity, quality, and clock flags, according to seed standard	
- variable opaque content section -				
m-32	32		Opaque data, e.g. seed volumes	

When converting from mini-SEED, time intervals of subsequent mini-SEED records with identical flags may be collapsed. The resulting record describes an interval, and does not allow for record realignment (and corresponding data losses) any more.