WFCatalog Web Service Specification

Version 0.22
15-11-2016

Authors
Luca Trani, Reinoud Sleeman, Mathijs Koymans and the EIDA team

Purpose

To specify a web service interface for the exchange of waveform metadata\(^1\), including QC. The specification defines service name, query parameters and expected results. Also, it contains the supported data quality metrics and their descriptions.

Service Characteristics

Versioning

The service is versioned according the following three digit (x.y.z) pattern:
\( \text{SpecMajor}\.\text{SpecMinor}\.\text{Implementation} \)

where the fields have the following meaning:

\( \text{SpecMajor} \): The major specification version, all implementations sharing this SpecMajor value will be backwards compatible with all prior releases. Values are integers starting at 1.

\( \text{SpecMinor} \): The minor specification version, incremented when optional parameters or behaviour is added to the previous specification but backwards compatibility is maintained with the previous major versions, i.e. all 1.# service versions will be compatible with version 1.0. Values are integers starting at 0.

\( \text{Implementation} \): The implementation version, an integer identifier specific to the data centre implementation. Useful to track service updates for bug fixes, etc. but with no implication on conformance to the specification.

Together the \text{SpecMajor} and \text{SpecMinor} versions imply a minimum expected behaviour of a given service. This versioning scheme allows clients to expect specific behaviour based on the SpecMajor version, while allowing the extension of the service with optional parameters while maintaining backwards compatibility. Each version number is service specific, there is no

---

\(^1\) See appendix II
implication that SpecMajor version numbers across services are related.

**Calling pattern**
The service will be invoked using a subset of REST and HTTP methods. In particular HTTP GET and HTTP POST methods are supported.

**Service path and port**
The following base URI pattern is to be used at each data centre implementing the service:

```
<site>/<relativepath>/wfcatalog/<majorversion>/
```

where majorversion is an integer value specifying the major specification version supported by the service.
A site is the domain name of the data centre hosting the web service. For instance, the base URI for version 1 of the service running at ORFEUS would be: www.orfeus-eu.org/ws/wfcatalog/1

**The service should be available on TCP/IP port 80**

**Service methods**
The service must support the following methods:

- **query** – to submit a data request
- **version** – to request the full service version (*SpecMajor.SpecMinor.Implementation*)
- **application.wadl** – to request a WADL for the interface

**Minimum functionality**
Implementations of this service interface should support all methods specified as required. Additionally, interfaces should conform to the calling patterns and expected results identified in this document to be considered conform with the specification. The service definition includes required and optional parameters; an implementation must support the required parameters to be considered conform. The optional parameters supported by any given implementation should be specified in the WADL returned by the service.

**Service responses**

**No data selected**
If a properly formatted request is submitted but would result in no data being returned, the service will return a HTTP status 204 (No Content).

**Result set limitations**
Limitations on the amount of information returned for any given request may be imposed independently for the service by each data centre. If a client submits a request that would result in a data set beyond the service limit the service should return an **HTTP status 413** (Request Entity Too Large)

**Error messages**
All errors reported to the client, either HTTP 4xx or 5xx status codes, should include an error
message transmitted as MIME type text/plain using the following pattern:

Error <CODE>: <SIMPLE ERROR DESCRIPTION>
<MORE DETAILED ERROR DESCRIPTION>
Usage details are available from <SERVICE DOCUMENTATION URI>
Request: <SUBMITTED URL>
Request Submitted: <UTC DATE TIME>
Service version: <3-LEVEL VERSION>

WADL
The WADL documents returned by the service should follow these guidelines:

- All public parameters supported by an interface shall be documented in the WADL
- Parameters supported by the interface shall be documented using the long version of the parameter name and not the abbreviated version.
- Parameters not included in the specification (data centre specific extensions) shall be documented with a type and a short description

HTTP Status codes
The following table includes a list of common status codes returned by the web service.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Successful request, results follow</td>
</tr>
<tr>
<td>204</td>
<td>Request was properly formatted and submitted but no data matches the selection</td>
</tr>
<tr>
<td>400</td>
<td>Bad request due to improper specification, unrecognised parameter, parameter value out of range, etc.</td>
</tr>
<tr>
<td>413</td>
<td>Request would result in too much data being returned or the request itself is too large. Returned error message should include the service limitations in the detailed description. Service limits should be documented in the WADL</td>
</tr>
<tr>
<td>500</td>
<td>Internal server error</td>
</tr>
<tr>
<td>503</td>
<td>Service temporarily unavailable, used in maintenance mode</td>
</tr>
</tbody>
</table>

Table 1

Request Parameters
Table 2 describes the request parameters for the service

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Support</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>net[work]</td>
<td>string</td>
<td>Required</td>
<td>Select the network code. Supports lists</td>
<td>[Any]</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Required</td>
<td>Description</td>
<td>Values</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>sta[tion]</td>
<td>string</td>
<td>Required</td>
<td>Select the station code. Supports lists and wildcards [see next paragraph]</td>
<td>[Any]</td>
</tr>
<tr>
<td>loc[ation]</td>
<td>string</td>
<td>Required</td>
<td>Select the location identifier. Supports lists and wildcards [see next paragraph]</td>
<td>[Any]</td>
</tr>
<tr>
<td>cha[nnel]</td>
<td>string</td>
<td>Required</td>
<td>Select the channel code. Supports lists and wildcards [see next paragraph]</td>
<td>[Any]</td>
</tr>
<tr>
<td>start[time]</td>
<td>ISO 8601</td>
<td>Required</td>
<td>Limit to results starting on or after the specified start time</td>
<td>[Any]</td>
</tr>
<tr>
<td>end[time]</td>
<td>ISO 8601</td>
<td>Required</td>
<td>Limit to results ending on or before the specified end time</td>
<td>[Any]</td>
</tr>
<tr>
<td>format</td>
<td>string</td>
<td>Optional</td>
<td>Specifies the desired output format. Valid values: json</td>
<td>default json</td>
</tr>
<tr>
<td>include</td>
<td>string</td>
<td>Required</td>
<td>Choose the level of detail of the results. See section “Results level of detail” E.g: include=sample or include=all</td>
<td>default (^4)</td>
</tr>
<tr>
<td>granularity</td>
<td>string</td>
<td>Required</td>
<td>Define the level of granularity for metric computation. Eg: day, hour, month. Minimum supported granularity is day.</td>
<td>day</td>
</tr>
<tr>
<td>minlength /</td>
<td>float</td>
<td>Optional</td>
<td>Limit results to continuous data segments of a minimum length specified in seconds, and include information about these segments</td>
<td>[Any]</td>
</tr>
<tr>
<td>minlen</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>longestonly</td>
<td>boolean</td>
<td>Optional</td>
<td>Limit results to the longest continuous segment per channel, and include information about this segment</td>
<td>FALSE</td>
</tr>
<tr>
<td>csegments</td>
<td>boolean</td>
<td>Optional</td>
<td>Include information about continuous segments</td>
<td>FALSE</td>
</tr>
<tr>
<td>[metric_filter]</td>
<td>Metric dependent</td>
<td>Required</td>
<td>Limit the results to streams that satisfy a filter on a specific metric value. [metric_filter] is a place-holder applicable to any(^5) specific metric defined in Table 3. Multiple occurrences separated by '&amp;' can be specified. See section “Metrics parameters extensions” for a detailed explanation. E.g.: sample_max=10 or sample_max_lt=10 &amp; sample_max_gt=3 or sample_max=10 &amp; sample_mean=-5.</td>
<td>[Any]</td>
</tr>
</tbody>
</table>

Table 2

\(^2\) Rounded down to the previous granule
\(^3\) Rounded up to the next granule
\(^4\) See Table 3
\(^5\) Except: data_quality_flags, activity_flags, io_and_clock_flags
**Wildcards and list in constraint parameters**

Some constraint parameters, indicated in the Table 2, support the use of wildcards. Two types of wildcards are defined:

* - Matches 0 to many characters
? - Matches any single character

**Lists**

Some constraints parameters, indicated in the Table 2, are allowed to hold multiple values comma separated. E.g.: `include=sample_stdev,sample_rms; or channel=BHZ,BHN,BHE`

**Time parameter**

The `starttime` parameter should be interpreted as selecting any data or information at times on or later than the value specified. Similarly, the `endtime` selects any data or information at times on or prior to the value specified. The selection shall include any data intersecting with the specified time range.

**Waveform characteristics and metrics**

The following entries, defined in Table 3, shall be available and supported by the service. For detailed metrics definitions see Appendix I.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mseed header</strong></td>
<td></td>
</tr>
<tr>
<td>quality*</td>
<td>SEED quality indicator. E.g.: D, R, Q, M</td>
</tr>
<tr>
<td>sample_rate*</td>
<td>List containing unique sample rates</td>
</tr>
<tr>
<td>record_length*</td>
<td>List containing unique record lengths</td>
</tr>
<tr>
<td>encoding*</td>
<td>List containing unique encodings</td>
</tr>
<tr>
<td>num_records*</td>
<td>Number of records</td>
</tr>
<tr>
<td><strong>Sample metrics</strong></td>
<td></td>
</tr>
<tr>
<td>num_samples*</td>
<td>Number of samples</td>
</tr>
<tr>
<td>sample_max</td>
<td>Maximum value of all the samples</td>
</tr>
<tr>
<td>sample_min</td>
<td>Minimum value of all the samples</td>
</tr>
<tr>
<td>sample_stdev</td>
<td>Standard deviation of all the samples</td>
</tr>
<tr>
<td>sample_rms</td>
<td>Root mean square value of all the samples</td>
</tr>
<tr>
<td>sample_mean</td>
<td>Average value of all the samples</td>
</tr>
<tr>
<td>sample_median</td>
<td>50th percentile of all the samples</td>
</tr>
<tr>
<td>sample_lower_quartile</td>
<td>25th percentile of all the samples</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sample_upper_quartile</td>
<td>75th percentile of all the samples</td>
</tr>
<tr>
<td>num_gaps*</td>
<td>Total number of gaps</td>
</tr>
<tr>
<td>num_overlaps*</td>
<td>Total number of overlaps</td>
</tr>
<tr>
<td>max_gap*</td>
<td>Largest gap in seconds</td>
</tr>
<tr>
<td>max_overlap*</td>
<td>Largest overlap in seconds</td>
</tr>
<tr>
<td>sum_gaps*</td>
<td>Sum of all the gaps in seconds</td>
</tr>
<tr>
<td>sum_overlaps*</td>
<td>Sum of all the overlaps in seconds</td>
</tr>
<tr>
<td>percent_availability*</td>
<td>Percentage of available data</td>
</tr>
</tbody>
</table>

**MSEED header metrics**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timing_quality_mean</td>
<td>Average of the timing quality percentage value</td>
</tr>
<tr>
<td>timing_quality_median</td>
<td>50th percentile of all timing quality percentage value</td>
</tr>
<tr>
<td>timing_quality_lower_quartile</td>
<td>25th percentile of all timing quality percentage value</td>
</tr>
<tr>
<td>timing_quality_upper_quartile</td>
<td>75th percentile of all timing quality percentage value</td>
</tr>
<tr>
<td>timing_quality_max</td>
<td>Maximum of all timing quality percentage value</td>
</tr>
<tr>
<td>timing_quality_min</td>
<td>Minimum of all timing quality percentage value</td>
</tr>
<tr>
<td>timing_correction</td>
<td>Percentage of data for which field 16 in the record header is non-zero</td>
</tr>
<tr>
<td>data_quality_flags</td>
<td>Data quality flags percentages. It works only with include. E.g.: include=data_quality_flags will return all the percentages for each data quality flag.</td>
</tr>
<tr>
<td>amplifier_saturation</td>
<td>Percentage of data for which the bit 0 in the Data Quality Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>digitizer_clipping</td>
<td>Percentage of data for which the bit 1 in the Data Quality Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>spikes</td>
<td>Percentage of data for which the bit 2 in the Data Quality Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>glitches</td>
<td>Percentage of data for which the bit 3 in the Data Quality Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>missing_padded_data</td>
<td>Percentage of data for which the bit 4 in the Data Quality Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>telemetry_sync_error</td>
<td>Percentage of data for which the bit 5 in the Data Quality Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>digital_filter_charging</td>
<td>Percentage of data for which the bit 6 in the Data Quality Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>suspect_time_tag</td>
<td>Percentage of data for which the bit 7 in the Data Quality Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>Activity</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>activity_flags</td>
<td>Activity flags percentages. It works only with <code>include</code>. E.g.: <code>include=activity_flags</code> will return all the percentages for each activity flag.</td>
</tr>
<tr>
<td>calibration_signal</td>
<td>Percentage of data for which the bit 0 in the Activity Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>time_correction_applied</td>
<td>Percentage of data for which the bit 1 in the Activity Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>event_begin</td>
<td>Percentage of data for which the bit 2 in the Activity Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>event_end</td>
<td>Percentage of data for which the bit 3 in the Activity Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>positive_leap</td>
<td>Percentage of data for which the bit 4 in the Activity Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>negative_leap</td>
<td>Percentage of data for which the bit 5 in the Activity Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>event_in_progress</td>
<td>Percentage of data for which the bit 6 in the Activity Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>io_and_clock_flags</td>
<td>I/O and clock flags percentages. It works only with <code>include</code>. E.g.: <code>include=io_and_clock_flags</code> will return all the percentages for each I/O and Clock flag.</td>
</tr>
<tr>
<td>station_volume</td>
<td>Percentage of data for which the bit 0 in the I/O and Clock Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>long_record_read</td>
<td>Percentage of data for which the bit 1 in the I/O and Clock Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>short_record_read</td>
<td>Percentage of data for which the bit 2 in the I/O and Clock Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>start_time_series</td>
<td>Percentage of data for which the bit 3 in the I/O and Clock Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>end_time_series</td>
<td>Percentage of data for which the bit 4 in the I/O and Clock Flag byte is set to ‘1’</td>
</tr>
<tr>
<td>clock_locked</td>
<td>Percentage of data for which the bit 5 in the I/O and Clock Flag byte is set to ‘1’</td>
</tr>
</tbody>
</table>

Table 3

* These are returned by default

**Results level of detail**
The service provides a number of metrics and characteristics by default. Additional features can be requested using the parameter include.

<table>
<thead>
<tr>
<th>Level name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>Includes all the default entries</td>
</tr>
<tr>
<td>sample</td>
<td>Includes default + all sample metrics</td>
</tr>
<tr>
<td>header</td>
<td>Includes default + all MSEED header metrics</td>
</tr>
<tr>
<td>all</td>
<td>Includes all metrics</td>
</tr>
</tbody>
</table>

**Metrics parameters extensions**

The query URL could contain `value=\textit{X}` to retrieve only metrics with that value, or `value\_gt=\textit{X}` for metrics that exceed \textit{X}.

The available extensions for parameter names are:

- \_\texttt{eq} - Equal (the default if no extension is used).
- \_\texttt{ne} - Not equal.
- \_\texttt{gt} - Greater than.
- \_\texttt{ge} - Greater than or equal.
- \_\texttt{lt} - Less than.
- \_\texttt{le} - Less than or equal.

This extension mechanism enables selection of parameters with values between an interval specified by the user. E.g.: `sample\_max\_ge=100&sample\_max\_le=500` would return all the samples with max value belonging to \([100,500]\).

**Behaviour of the service**

The \texttt{version} method shall return the implementation version as simple text string using the MIME type \texttt{text/plain}.

The \texttt{application.wadl} method shall return a WADL conform description of the service using MIME type \texttt{application/xml}.

The \texttt{query} method shall return the results fulfilling a user query in the proper format according to the \texttt{format} parameter. The default MIME type is \texttt{application/json}.

The default format of the returned payload represents a \textit{waveform metadata} object according to the JSON schema provided in Appendix II.

Queries can be submitted using either HTTP GET or POST methods.

For the HTTP GET method, the parameters are specified as \texttt{key=value} separated by “\&” and may not be specified more than once.

For the HTTP POST method, the parameter must be submitted as part of a POST body. The

---

6 See Table 3
parameters network, station, location, channel, starttime, endtime may be repeated as many times as necessary, all other parameters should be specified as key=value pairs on separate lines following this pattern:

```
parameter1=value
parameter2=value
NET STA LOC CHA STARTTIME ENDTIME
NET STA LOC CHA STARTTIME ENDTIME
NET STA LOC CHA STARTTIME ENDTIME
```

All rules for parameters apply equally whether specified using GET or POST methods. However when using POST no blank spaces are allowed as value, use “--” instead.

**Examples**

QUERY = www.orfeus-eu.org/ws/wfcatalog/1/query?network=NL&station=HGN&cha=BHZ&start=2001-01-02&end=2001-01-03&csegments=true&include=all

```json
[
  {
    "miniseed_header_percentages": {
      "data_quality_flags": {
        "suspect_time_tag": 0,
        "spikes": 0,
        "amplifier_saturation": 0,
        "glitches": 0,
        "telemetry_sync_error": 0,
        "missing_padded_data": 0,
        "digital_filter_charging": 0,
        "digitizer_clipping": 0
      },
      "activity_flags": {
        "event_in_progress": 0,
        "positive_leap": 0,
        "negative_leap": 0,
        "calibration_signal": 0,
        "event_begin": 0,
        "event_end": 0,
        "time_correction_applied": 0
      },
      "io_and_clock_flags": {
        "short_record_read": 0,
        "station_volume": 0,
        "long_record_read": 0,
        "end_time_series": 0,
        "start_time_series": 0,
        "clock_locked": 0
      },
      "timing_correction": 0,
      "timing_quality_mean": null,
      "timing_quality_median": null,
      "timing_quality_min": null,
      "timing_quality_max": null,
      "timing_quality_lower_quartile": null,
      "timing_quality_upper_quartile": null
    }
  }
]```
"sample_stdev": 805.2504836461063,
"sample_mean": 192.0957620724346
]}

QUERY = www.orfeus-eu.org/ws/wfcatalog/1/query?network=NL&station=HGN&cha=BHZ&start=2001-01-02&end=2001-01-03&include=sample

[
{
  "sample_rms": 755.5320646257303,
  "sample_rate": [40],
  "waveform_format": "miniSEED",
  "sum_overlaps": 0,
  "quality": "D",
  "num_overlaps": 0,
  "sum_gaps": 79862.40000009537,
  "num_gaps": 2,
  "num_records": 120,
  "max_overlap": null,
  "sample_stdev": 746.0162835391818,
  "location": "",
  "channel": "BHZ",
  "encoding": ["STEIM1"],
  "station": "HGN",
  "num_samples": 261504,
  "sample_upper_quartile": 576,
  "record_len": [4096],
  "sample_median": 90,
  "start_time": "2001-01-02T00:00:00.000Z",
  "percent_availability": 7.566666666556287,
  "sample_lower_quartile": -355,
  "network": "NL",
  "end_time": "2001-01-03T00:00:00.000Z",
  "max_gap": 52214.40000009537,
  "sample_mean": 119.53411802496329,
  "sample_max": 3454,
  "sample_min": -3641,
  "version": "1.0.0",
  "producer": {
    "name": "Orfeus Data Center - KNMI/ODC",
    "agent": "ObsPy",
    "created": "2016-05-02T13:50:24.966Z"
  }
}

QUERY = www.orfeus-eu.org/ws/wfcatalog/1/query?network=NL&station=HGN&cha=BHZ&start=2001-01-02&end=2001-01-03&include=header

[
{
  "miniseed_header_percentages": {
    "data_quality_flags": {
      "suspect_time_tag": 0,
      "spikes": 0,
      "amplifier_saturation": 0,
      "glitches": 0,
      "telemetry_sync_error": 0,
      "suspect_time_tag": 0,
      "spikes": 0,
      "amplifier_saturation": 0,
      "glitches": 0,
      "telemetry_sync_error": 0,}]]


```json
{
  "missing_padded_data": 0,
  "digital_filter_charging": 0,
  "digitizer_clipping": 0,
  "activity_flags": {
    "event_in_progress": 0,
    "positive_leap": 0,
    "negative_leap": 0,
    "calibration_signal": 0,
    "event_begin": 0,
    "event_end": 0,
    "time_correction_applied": 0,
  },
  "io_and_clock_flags": {
    "short_record_read": 0,
    "station_volume": 0,
    "long_record_read": 0,
    "end_time_series": 0,
    "start_time_series": 0,
    "clock_locked": 0,
  },
  "timing_correction": 0,
  "timing_quality_mean": null,
  "timing_quality_median": null,
  "timing_quality_min": null,
  "timing_quality_max": null,
  "timing_quality_lower_quartile": null,
  "timing_quality_upper_quartile": null,
  "sample_rate": [40],
  "waveform_format": "miniSEED",
  "sum_overlaps": 0,
  "quality": "D",
  "num_overlaps": 0,
  "num_records": 120,
  "sum_gaps": 79862.40000009537,
  "num_gaps": 2,
  "max_overlap": null,
  "location": "",
  "channel": "BHZ",
  "encoding": ["STEIM1"],
  "station": "HGN",
  "num_samples": 261504,
  "record_len": [4096],
  "start_time": "2001-01-02T00:00:00.000Z",
  "percent_availability": 7.566666666556287,
  "network": "NL",
  "end_time": "2001-01-03T00:00:00.000Z",
  "max_gap": 52214.40000009537,
  "version": "1.0.0",
  "producer": {
    "name": "Orfeus Data Center - KNMI/ODC",
    "agent": "ObsPy",
    "created": "2016-05-02T13:51:58.662Z"
  }
}
```
Appendix I: Quality metrics definition

The data quality metrics supported by this version of the WFCatalog API are compliant to the proposal for standardisation for FDSN v2.0 by Reinoud Sleeman. The proposal is included below.

Proposal for FDSN standardization of waveform quality metrics

v2.0 - 21 October 2016

Knowledge of quality of seismic waveform data and related metadata is essential for any scientific analysis and interpretation of the data. Automated processes to calculate data quality parameters are required (a) to handle huge amounts of data, (b) to enable data centers to automatically monitor changes or variations in data quality over different time scales and (c) to provide services to the research community to search for and harvest the best quality data based on these parameters.

Currently, parallel developments are on-going (e.g. IRIS DMC, ORFEUS EIDA) to calculate data quality parameters and use these in different services. This document proposes the standardization of a number of basic metrics of which most are common in both systems.

Quality parameters are calculated for a time windowed time series. In the following a time series is considered to belong to a data stream uniquely identified by a SEED network code, stations code, channel code, location code and data header/quality indicator (D|R|Q|M).

In this proposal two types of metrics are defined: those based on series of sample values and those based on the SEED data record headers.

Waveform data archived in SEED files typically do not start and/or end at pre-defined, fixed times, e.g. midnight or 00:00:00.0, and no standard exists within FDSN to do this. Therefore, calculating sample value metrics in a file that loosely fits a pre-defined, fixed time window does not yield appropriate results. The approach in this proposal therefore is to define the sample value metrics in a defined time window of length 24 hours (starting at 00:00:00.0) independent of the archive structure. This proposal first provides the definitions to relate the sample value metrics to the data, independent of the archive format. Similarly, data records do not start or end exactly on pre-defined, fixed times. In this proposal, header based metrics use the first record that (partially) fits the time window and ignores the last record that (partially) fits the time window (see definitions below).

Definitions

Data (dis)continuity, gap and overlap

A sample value at time t represents a continuous signal within time window [t, t+Δt), in which Δt is the sample interval between two adjacent samples as defined by the sample rate factor
and the sample rate multiplier in the data record (header, fields 10 and 11) that contains the sample at time $t$. When the actual sample rate in blockette 100 is available this value should be used instead.

A continuous (discrete) time series is defined as a time series in which the time interval between two adjacent samples (a) is constant ($\Delta t$) or (b) does not differ from this constant by more than a certain time tolerance ($\varepsilon$). The tolerance value is defined here as 50% of the sampling rate.

**Continuity condition between two samples:**

$$\Delta t - \varepsilon \leq t_{i+1} - t_i \leq \Delta t + \varepsilon.$$

A discontinuity thus occurs when (a) the time interval between two adjacent samples (with corresponding times $t_i$ and $t_{i+1}$) exceeds the sample rate interval ($\Delta t$) by more than the (sampling rate dependent) time tolerance $\varepsilon$: $|t_{i+1} - (t_i + \Delta t)| > \varepsilon$ or (b) when the sampling rate between 2 samples changes with respect to the previous 2 samples by more than the tolerance value (50% of the sampling rate).

A gap is a positive valued discontinuity, an overlap is a negative valued discontinuity.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Expression</th>
<th>Gap length</th>
<th>Overlap length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gap condition</td>
<td>$t_{i+1} - t_i &gt; \Delta t + \varepsilon$</td>
<td>$t_{i+1} - t_i - \Delta t$</td>
<td></td>
</tr>
<tr>
<td>Overlap condition</td>
<td>$t_{i+1} - t_i &lt; \Delta t - \varepsilon$</td>
<td>$t_i + \Delta t - t_{i+1}$</td>
<td></td>
</tr>
</tbody>
</table>

Within a time window $[T_1, T_2)$ a time series may have a start time $t_1$ (defined by the first sample in $[T_1, T_2)$) later than $T_1$ or an end time $t_N$ (defined by the last sample) before $T_2$. In cases that $T_1$ or $T_2$ intercept a gap in the continuous (discrete) time series a start gap or end gap is defined within $[T_1, T_2)$. When the actual sample rate in blockette 100 is available in time window $[T1, T2)$ the mode of these values in these blockettes should be used as sample rate $\Delta t$. 

![Diagram of time series continuity and discontinuity conditions](image-url)
Data in SEED format is stored in records. Each SEED record contains a continuous time series. Gaps or overlaps in data may occur between records.

The **start time** $R_s$ of a mini-SEED record is the time $t_1$ of the first sample in the record (including time correction if applicable).

The **end time** $R_e$ of a record is defined by the end of the time interval represented by the last sample ($N$) in the record (including time correction if applicable): $t_N + \Delta t$

The time interval of a record containing $N$ samples is thus defined as: $[R_s, R_e) = [t_1, t_N + \Delta t)$.

The **length** of a record is $R_e - R_s = t_N - t_1 + \Delta t$
Metrics calculation in time window

Two types of metrics are defined, one type based on samples/values, and one type based on mini-SEED data record header.

Sample based metrics are calculated within a time window \([T_1, T_2)\). Time \(T_1\) is included, time \(T_2\) is excluded.

Samples at \(t=T_1\) are included in the sample based metrics calculations, samples at \(t=T_2\) are excluded. In other words, all samples with time \(t\) for which \(T_1 \leq t < T_2\) are included.
Therefore:

Data records for which \( R_e \leq T_1 \) are not used in the sample based metrics calculations.

Data records for which \( R_s \geq T_2 \) are not used in the sample based metrics calculations.

Data records for which \( T_1 < R_e < T_2 \) are (partially) used in the sample based metrics calculations. In these records the samples are used with time \( t \) for which \( t \geq T_1 \).

Data records for which \( T_1 \leq R_e < T_2 \) are (partially) used in the sample based metrics calculations. In these records the samples are used with time \( t \) for which \( t < T_2 \).

Record based metrics are calculated within a time window \([T_1, T_2)\). Time \( T_1 \) is included, time \( T_2 \) is excluded.

Data records for which \( R_e \leq T_1 \) are not used in the record based metrics calculations.

Data records for which \( R_e \geq T_2 \) are not used in the record based metrics calculations.
Metrics based on sample values

The following metrics are calculated from all sample values within time window \([T_1, T_2]\).

- **sample_mean**
  
  Average value of all samples \((x_1, ..., x_N)\).
  
  \[
  mean = \frac{1}{N} \sum_{i=1}^{N} x_i
  \]

- **sample_max**
  
  Maximum value of all samples \((x_1, ..., x_N)\).

- **sample_min**
  
  Minimum value of all samples \((x_1, ..., x_N)\).

- **sample_median**
  
  Median value of all samples \((x_1, ..., x_N)\). The middle value of the sorted samples. The 50-th percentile of all samples \((x_1, ..., x_N)\).

- **sample_stdev**
  
  The standard deviation (or RMS variance) of all samples \((x_1, ..., x_N)\).
  
  \[
  stddev = \sqrt{\frac{\sum_{i=1}^{N} (x_i - mean)^2}{N}}
  \]

- **percent_availability**
  
  Data availability is the percentage of data available in a time window \([T_1, T_2]\). It is the length of the time window minus the sum of all gaps in this time window, relative to the length of the time window \([T_1, T_2]\).
  
  \[
  percent\_availability = 100 \times \frac{(T_2 - T_1) - \text{sum}\_gaps}{T_2 - T_1}
  \]
**Metrics based on mini-SEED data record header**

The following metrics are extracted from header flags in data records fitting time window \([T_1, T_2]\).

A flag in the record header of a SEED record applies to all samples in the data record.

- **amplifier_saturation**
  
  Number of records fitting time window \([T_1, T_2]\) for which bit 0 in the Data Quality Flag byte is set to '1'.

- **digitizer_clipping**
  
  Number of records fitting time window \([T_1, T_2]\) for which bit 1 in the Data Quality Flag byte is set to '1'.

- **spikes**
  
  Number of records fitting time window \([T_1, T_2]\) for which bit 2 in the Data Quality Flag byte is set to '1'.

- **glitches**
  
  Number of records fitting time window \([T_1, T_2]\) for which bit 3 in the Data Quality Flag byte is set to '1'.

- **missing_padded_data**
  
  Number of records fitting time window \([T_1, T_2]\) for which bit 4 in the Data Quality Flag byte is set to '1'.

- **telemetry_sync_error**
  
  Number of records fitting time window \([T_1, T_2]\) for which bit 5 in the Data Quality Flag byte is set to '1'.

- **digital_filter_charging**
  
  Number of records fitting time window \([T_1, T_2]\) for which bit 6 in the Data Quality Flag byte is set to '1'.

- **suspect_time_tag**
  
  Number of records fitting time window \([T_1, T_2]\) for which bit 7 in the Data Quality Flag byte is set to '1'.

- **calibration_signal**
  
  Number of records fitting time window \([T_1, T_2]\) for which bit 0 in the Activity Flag byte is set to '1'.

- **event_begin**
Number of records fitting time window \([T_1, T_2]\) for which bit 2 in the Activity Flag byte is set to '1'.

- **event_end**

Number of records fitting time window \([T_1, T_2]\) for which bit 3 in the Activity Flag byte is set to '1'.

- **event_in_progress**

Number of records fitting time window \([T_1, T_2]\) for which bit 6 in the Activity Flag byte is set to '1'.

- **clock_locked**

Number of records fitting time window \([T_1, T_2]\) for which bit 5 in the I/O & Clock Flag byte is set to '1'.

- **timing_correction**

Number of records fitting time window \([T_1, T_2]\) for which field 16 ("Time correction") in the record header is non-zero.

- **timing_quality**

Average of the timing quality percentage value stored in miniSEED blockettes 1001. Value is NULL if not present in the data records.

**num_gaps**

The number of gaps in time interval \([T_1, T_2]\).

**num_overlaps**

The number of overlaps in time interval \([T_1, T_2]\).

**max_gap**

Largest gap in seconds in time interval \([T_1, T_2]\).

**max_overlap**

Largest overlap in seconds in time interval \([T_1, T_2]\).
Appendix II: Waveform Metadata (WFMetadata) JSON Schema

{
  "id": "wfmetadata-schema-uri",
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Waveform Metadata Json Schema",
  "description": "This schema represents Waveform Metadata characterising traces with their QC",
  "type": "object",

  "properties": {
    "wfmetadata_id": {
      "description": "Unique identifier of the metadata document. This can be a DOI, a Handle or any other type of PID",
      "$ref": "#/definitions/stringLiteral",
      "format": "uri"
    },
    "producer": {
      "description": "The producer of this document. For instance it can be: a software, a person or an organization",
      "type": "object",
      "additionalProperties": {
        "$ref": "#/definitions/agent"
      }
    },
    "waveform_type": {
      "description": "Describes the type of waveform. E.g.: seismic, infrasound",
      "$ref": "#/definitions/stringLiteral",
      "additionalProperties": false
    },
    "waveform_format": {
      "description": "Describes the waveform format. E.g.: miniSEED",
      "type": {
        "enum": ["miniSEED"]
      },
      "additionalProperties": false
    },
    "version": {
      "description": "The version of the metadata document. E.g.: 1.0.0",
      "type": {
        "enum": ["1.0.0"]
      },
      "additionalProperties": true
    },
    "start_time": {
      "description": "Start time of the window used for metric computation for this entry in UTC",
      "$ref": "#/definitions/timeLiteral"
    },
    "end_time": {
  "}}
"description": "End time of the window used for metric computation for this entry in UTC",
"$ref": "#/definitions/timeLiteral"
},

"network": {
  "description": "Network code",
  "$ref": "#/definitions/stringLiteral",
  "pattern": "[A-Z0-9]{1,6}"
},

"station": {
  "description": "Station code",
  "$ref": "#/definitions/stringLiteral",
  "pattern": "[A-Z0-9]{1,5}"
},

"channel": {
  "description": "Channel code",
  "$ref": "#/definitions/stringLiteral",
  "pattern": "[A-Z0-9]{3}"
},

"location": {
  "description": "Location code",
  "$ref": "#/definitions/stringLiteral",
  "pattern": "[A-Z0-9]{0,2}"
},

"quality": {
  "description": "SEED quality indicator. This is SEED format specific",
  "type": {
    "enum": [
      "D",
      "R",
      "Q",
      "M"
    ],
    "additionalProperties": false
  },
  "sample_rate": {
    "description": "Array of unique sample rates",
    "type": "array",
    "minItems": 1,
    "items": {
      "$ref": "#/definitions/positiveDouble"
    },
    "uniqueItems": true,
    "additionalProperties": false
  },
  "num_samples": {
    "description": "Number of data samples",
    "$ref": "#/definitions/positiveInteger"
  },
  "encoding": {
    "description": "Array of unique encodings. E.g. in SEED: int32, int64, float32, float64, opaque (for log channels).",
    "type": "array",
    "minItems": 1,
    "items": {
      "$ref": "#/definitions/stringLiteral"
    }
  }
}
"uniqueItems": true,
"additionalProperties": false
},
"num_records": {
"description": "Number of records. This is SEED format specific",
"$ref": "#/definitions/positiveInteger"
},
"record_length": {
"description": "Array of unique record lengths. This is SEED format specific",
"type": "array",
"minItems": 1,
"items": {
"$ref": "#/definitions/positiveInteger"
},
"uniqueItems": true,
"additionalProperties": false
},
"num_gaps": {
"description": "Number of data gaps",
"$ref": "#/definitions/positiveInteger"
},
"max_gap": {
"description": "Duration of the largest gap in seconds",
"$ref": "#/definitions/positiveDouble"
},
"num_overlaps": {
"description": "Number of data overlaps",
"$ref": "#/definitions/positiveInteger"
},
"max_overlap": {
"description": "Duration of the largest overlap in seconds",
"$ref": "#/definitions/positiveDouble"
},
"sum_gaps": {
"description": "Total duration of gaps in seconds",
"$ref": "#/definitions/positiveDouble"
},
"sum_overlaps": {
"description": "Total duration of overlaps in seconds",
"$ref": "#/definitions/positiveDouble"
},
"sample_max": {
"description": "Maximum sample value",
"$ref": "#/definitions/numberLiteral"
},
"sample_min": {
"description": "Minimum sample value",
"$ref": "#/definitions/numberLiteral"
},
"sample_mean": {
"description": "Mean of the sample values",
"$ref": "#/definitions/numberLiteral"
},
"sample_rms": {
"description": "Rms of the sample values",
"$ref": "#/definitions/positiveDouble"
{"sample_lower_quartile": {
  "description": "Lower quartile of the sample values",
  "$ref": "/definitions/numberLiteral"
},
"sample_upper_quartile": {
  "description": "Upper quartile of the sample values",
  "$ref": "/definitions/numberLiteral"
},
"sample_median": {
  "description": "50th percentile of the sample values",
  "$ref": "/definitions/numberLiteral"
},
"sample_stddev": {
  "description": "Standard deviation of the sample values",
  "$ref": "/definitions/positiveDouble"
},
"miniseed_header_percentages": {
  "type": "object",
  "properties": {
    "timing_correction": {
      "description": "Percentage of data for which field 16 in the record header is non-zero",
      "$ref": "/definitions/numberLiteral"
    },
    "timing_quality_mean": {
      "description": "Mean of the timing quality percentage values stored in mSEED blockettes 1001",
      "$ref": "/definitions/numberLiteral"
    },
    "timing_quality_min": {
      "description": "Minimum of the timing quality percentage values stored in mSEED blockettes 1001",
      "$ref": "/definitions/numberLiteral"
    },
    "timing_quality_max": {
      "description": "Maximum of the timing quality percentage values stored in mSEED blockettes 1001",
      "$ref": "/definitions/numberLiteral"
    },
    "timing_quality_median": {
      "description": "The 50th percentile of all timing quality percentage values stored in mSEED blockettes 1001",
      "$ref": "/definitions/numberLiteral"
    },
    "timing_quality_lower_quartile": {
      "description": "The 25th percentile of all timing quality percentage values stored in mSEED blockettes 1001",
      "$ref": "/definitions/numberLiteral"
    },
    "timing_quality_upper_quartile": {
      "description": "The 75th percentile of all timing quality percentage values stored in mSEED blockettes 1001",
      "$ref": "/definitions/numberLiteral"
    }
  }
},
"data_quality_flags": {
  "type": "object",
  "properties": {
    "..."
"amplifier_saturation": {
  "description": "Percentage of data for which bit 0 in the DQ flag is set to 1",
  "$ref": "/definitions/positivePercentage"
},
"digitizer_clipping": {
  "description": "Percentage of data for which bit 1 in the DQ flag is set to 1",
  "$ref": "/definitions/positivePercentage"
},
"spikes": {
  "description": "Percentage of data for which bit 2 in the DQ flag is set to 1",
  "$ref": "/definitions/positivePercentage"
},
"glitches": {
  "description": "Percentage of data for which bit 3 in the DQ flag is set to 1",
  "$ref": "/definitions/positivePercentage"
},
"missing_padded_data": {
  "description": "Percentage of data for which bit 4 in the DQ flag is set to 1",
  "$ref": "/definitions/positivePercentage"
},
"telemetry_sync_error": {
  "description": "Percentage of data for which bit 5 in the DQ flag is set to 1",
  "$ref": "/definitions/positivePercentage"
},
"digital_filter_charging": {
  "description": "Percentage of data for which bit 6 in the DQ flag is set to 1",
  "$ref": "/definitions/positivePercentage"
},
"suspect_time_tag": {
  "description": "Percentage of data for which bit 7 in the DQ flag is set to 1",
  "$ref": "/definitions/positivePercentage"
}

"required": [
  "amplifier_saturation",
  "digitizer_clipping",
  "spikes",
  "glitches",
  "missing_padded_data",
  "telemetry_sync_error",
  "digital_filter_charging",
  "suspect_time_tag"
],

"activity_flags": {
  "type": "object",
  "properties": {
    "calibration_signal": {
      "description": "Percentage of data for

which bit 0 in the Activity flag is set to 1",
"$ref": "/definitions/positivePercentage"
},
"time_correction_applied": {
"description": "Percentage of data for
which bit 1 in the Activity flag is set to 1",
"$ref": "/definitions/positivePercentage"
},
"event_begin": {
"description": "Percentage of data for
which bit 2 in the Activity flag is set to 1",
"$ref": "/definitions/positivePercentage"
},
"event_end": {
"description": "Percentage of data for
which bit 3 in the Activity flag is set to 1",
"$ref": "/definitions/positivePercentage"
},
"positive_leap": {
"description": "Percentage of data for
which bit 4 in the Activity flag is set to 1",
"$ref": "/definitions/positivePercentage"
},
"negative_leap": {
"description": "Percentage of data for
which bit 5 in the Activity flag is set to 1",
"$ref": "/definitions/positivePercentage"
},
"event_in_progress": {
"description": "Percentage of records for
which bit 6 in the Activity flag is set to 1",
"$ref": "/definitions/positivePercentage"
}
},
"required": [
"calibration_signal",
"time_correction_applied",
"event_begin",
"event_end",
"positive_leap",
"negative_leap",
"event_in_progress"
]
},

"io_and_clock_flags": {
"type": "object",
"properties": {
"station_volume": {
"description": "Percentage of records for
which bit 0 in the I/O and Clock flag is set to 1",
"$ref": "/definitions/positivePercentage"
},
"long_record_read": {
"description": "Percentage of records for
which bit 1 in the I/O and Clock flag is set to 1",
"$ref": "/definitions/positivePercentage"
"short_record_read": {  
  "description": "Percentage of records for which bit 2 in the I/O and Clock flag is set to 1",
  "$ref": "#/definitions/positivePercentage"
},
"start_time_series": {  
  "description": "Percentage of records for which bit 3 in the I/O and Clock flag is set to 1",
  "$ref": "#/definitions/positivePercentage"
},
"end_time_series": {  
  "description": "Percentage of records for which bit 4 in the I/O and Clock flag is set to 1",
  "$ref": "#/definitions/positivePercentage"
},
"clock_locked": {  
  "description": "Percentage of records for which bit 5 in the I/O & Clock flag is set to 1",
  "$ref": "#/definitions/positivePercentage"
}
}
"properties": {
  "sample_rate": {
    "description": "Sample rate in the continuous segment",
    "$ref": "#/definitions/positiveDouble"
  },
  "sample_min": {
    "description": "Minimum of samples in a continuous segment",
    "$ref": "#/definitions/numberLiteral"
  },
  "sample_max": {
    "description": "Maximum of samples in a continuous segment",
    "$ref": "#/definitions/numberLiteral"
  },
  "sample_mean": {
    "description": "Mean of the sample values in a continuous segment",
    "$ref": "#/definitions/numberLiteral"
  },
  "sample_rms": {
    "description": "Rms of the sample values in a continuous segment",
    "$ref": "#/definitions/positiveDouble"
  },
  "sample_lower_quartile": {
    "description": "Lower quartile of the sample values in a continuous segment",
    "$ref": "#/definitions/numberLiteral"
  },
  "sample_upper_quartile": {
    "description": "Upper quartile of the sample values in a continuous segment",
    "$ref": "#/definitions/numberLiteral"
  },
  "sample_median": {
    "description": "50th percentile of the sample values in a continuous segment",
    "$ref": "#/definitions/numberLiteral"
  },
  "start_time": {
    "description": "Time of the first sample of the segment in UTC",
    "$ref": "#/definitions/timeLiteral"
  },
  "end_time": {
    "description": "Time of the last sample of the segment in UTC",
    "$ref": "#/definitions/timeLiteral"
  },
  "num_samples": {
    "description": "Number of data samples in the continuous segment",
    "$ref": "#/definitions/positiveInteger"
  }
}
"description": "Standard deviation of samples in the continuous segment",
"$ref": "#/definitions/positiveDouble"
},
"segment_length": {
  "description": "Length in seconds of the specific continuous segment",
  "$ref": "#/definitions/numberLiteral"
}
},
"required": [
  "sample_rate",
  "start_time",
  "end_time",
  "num_samples",
  "segment_length"
]
}

"definitions": {
  "agent": {
    "$ref": "#/definitions/entity"
  },
  "typedLiteral": {
    "type": "object",
    "properties": {
      "$": {
        "type": "string"
      },
      "type": {
        "type": "string",
        "format": "uri"
      },
      "lang": {
        "type": "string"
      }
    },
    "required": ["$"],
    "additionalProperties": false
  },
  "stringLiteral": {
    "type": "string",
    "additionalProperties": false
  },
  "numberLiteral": {
    "oneOf": [
      {"type": "number"},
      {"type": "null"}
    ],
    "additionalProperties": false
  },
  "booleanLiteral": {
    "type": "boolean",
    "additionalProperties": false
  },
  "timeLiteral": {
"type": "string",
"format": "date-time",
"additionalProperties": false
},
"literalArray": {
  "type": "array",
  "minItems": 1,
  "items": {
    "anyOf": [
      {"$ref": "/definitions/stringLiteral"},
      {"$ref": "/definitions/numberLiteral"},
      {"$ref": "/definitions/booleanLiteral"},
      {"$ref": "/definitions/typedLiteral"},
      {"$ref": "/definitions/literalArray"}
    ]
  }
},
"attributeValues": {
  "anyOf": [
    {"$ref": "/definitions/stringLiteral"},
    {"$ref": "/definitions/numberLiteral"},
    {"$ref": "/definitions/timeLiteral"},
    {"$ref": "/definitions/booleanLiteral"},
    {"$ref": "/definitions/typedLiteral"},
    {"$ref": "/definitions/literalArray"}
  ]
},
"entity": {
  "title": "entity",
  "additionalProperties": {
    "$ref": "/definitions/attributeValues"
  }
},
"positiveInteger": {
  "oneOf": [
    {"type": "integer",
     "minimum": 0,
     "exclusiveMinimum": false},
    {"type": "null"}
  ],
  "additionalProperties": false
},
"positiveDouble": {
  "oneOf": [
    {"type": "number",
     "minimum": 0,
     "exclusiveMinimum": false},
    {"type": "null"}
  ]}
"additionalProperties": false
},
"positivePercentage": {
"oneOf": [
{"type": "number",
"minimum": 0,
"maximum": 100,
"exclusiveMinimum": false,
"exclusiveMaximum": false},
{"type": "null" }
],
"additionalProperties": false
},
"required": [

"producer",
"version",
"waveform_format",
"start_time",
"end_time",
"network",
"station",
"channel",
"location",
"sample_rate",
"max_gap",
"max_overlap",
"percent_availability",
"num_samples",
"num_gaps",
"num_overlaps",
"sum_gaps",
"sum_overlaps"
]
}