Architecture Description - Waveform Data Fabric

Table of Contents

Purpose

This page summarizes GMS architecture descriptions related to the Waveform Data Fabric operations.

Architecture Concept/Flow

The Data Fabric provides GMS COI **ChannelSegment** query and storage operations through request-response services. The USNDC System uses similar information represented with a data model different from the GMS COI. However, GMS creates, uses, stores, and subsequently queries **ChannelSegment** objects with contents beyond those available in the USNDC System's database. Each System must access **ChannelSegment** data consistent with the other system (i.e. the Data Fabric provides GMS the ability to access **ChannelSegment** format information created by the USNDC System and provides the USNDC System the ability to access **ChannelSegment** objects written by GMS). The GMS user interface also uses the Data Fabric request-response operations to learn about new or updated **ChannelSegment** objects. The GMS user interface typically uses these operations to find new or updated raw **Waveform ChannelSegment** objects acquired by the legacy USNDC system, but also uses them to find derived **ChannelSegment** objects like **Event** beams that either the USNDC system created and stored to the USNDC database or other GMS user interface instances created and stored using the Data Fabric request-response operations.

The Data Fabric provides access to **ChannelSegment** data samples in several ways:

1. FDSN Data Select service -
    a. GMS uses this service to load raw waveform samples, which it combines with **Channel** metadata to construct raw **Waveform ChannelSegment** objects.
    b. Some Data Fabric operations construct and return **ChannelSegment** objects populated with all of their attributes except for the timeseries data samples, which they populate with claim check objects. GMS extracts the claim checks from the **ChannelSegment** objects, uses the claim checks to query the FDSN Data Select service, and then updates the **ChannelSegment** objects with the data samples returned by the Data Fabric.
2. FDSN Availability service - GMS polls this service to learn about updates to available raw waveform samples.
3. **Event** beam **Waveform ChannelSegment** query service - GMS uses this service to load **Event** beam **Waveform ChannelSegment** objects. GMS specifically queries for these objects since they do not have associations to the other processing results objects (e.g. **Event** or **SignalDetection**) that GMS loads from the Data Fabric.
4. (Future services) - Data Fabric services return non-waveform **ChannelSegment** data samples (e.g. **FkSpectra**).

The Data Fabric provides GMS COI **QcSegment** query and storage operations through request-response services. The USNDC System uses similar waveform quality control information represented with a data model different from the GMS COI. GMS creates, uses, stores, and subsequently queries **QcSegment** objects with contents beyond those available in the USNDC System's database. Each System must access waveform quality control data consistent with the other system (i.e. the Data Fabric provides GMS the ability to access waveform quality control format information created by the USNDC System and provides the USNDC System the ability to access **QcSegment** objects written by GMS). The two systems should use waveform quality control objects masking the same data samples, even if data model differences result in different representations of the masked samples (e.g. GMS may use overlapping **QcSegment** objects while the USNDC System uses non-overlapping *QCMASKSEG* records masking the same samples).

The GMS user interfaces use the Data Fabric request-response operations to load the **QcSegment** collections available for the Analyst to review. An optional *changedSinceTime* can be used to retrieve only **QcSegment** objects that have changed on or after that time, allowing the GMS user interface to learn about new or updated **QcSegment** objects stored by either the USNDC system or other GMS user interface instances. The Analyst creates new **QcSegment** objects, adjusts existing **QcSegment** objects to create new **QcSegmentVersion** objects in their version histories, and rejects existing **QcSegment** objects that do not mark actual waveform quality control problems. The GMS user interfaces store the new and updated **QcSegment** objects. When the Analyst processes raw waveforms to create derived **ChannelSegment** objects (e.g. beamed waveforms, **FkSpectra**, etc.), GMS selects applicable **QcSegment** objects, uses them to create **ProcessingMask** objects representing waveform samples excluded from processing, and then associates the **ProcessingMask** objects to the derived **ChannelSegment** to track which raw waveform samples were masked.

The Data Fabric uses a **QcSegmentBridgeDefinition** to convert between USNDC format waveform quality control objects and GMS COI format **QcSegment** objects. The Data Fabric is responsible for loading the **QcSegmentBridgeDefinition**.

⚠ **Imlementation Note**

The GMS developed data bridge concept (see References below) was to create bridged **ProcessingMask** objects using configured GMS format **ProcessingMaskDefinition** classes equivalent to the USNDC system's processing definitions describing which of its QC Masks indicate waveform quality control issues that must be removed from a waveform prior to processing. The Data Fabric may need to access the USNDC system's processing definitions to construct **ProcessingMask** objects.

COI Data Model

1. Channel Segment COI Data Model - this page describes the **ChannelSegment**, **Waveform**, **WfdiscWaveformClaimCheck**, and **FkSpectra** COI classes.
2. Waveform Quality Control COI Data Model - this page describes the **QcSegment**, **QcSegmentVersion**, and **ProcessingMask** COI classes.

Service Descriptions

Request-Response Operations

COI Operations

The provided OpenAPI file fully describes the waveform related Data Fabric operations.

⚠ **Note**

The OpenAPI file contains a schema for the data classes used by the waveform related Data Fabric operations. Use the COI Data Model (see above) and the schema together to fully understand the contents of each class and attribute included in the schema.

The following operations need to be implemented:

1. /waveform/query/event-beams-by-stations-and-timerange
   a. Finds **Event** beam **Waveform ChannelSegment** objects for the provided **Station** collection and time range.
   b. This operation has the following performance requirements:
      i. The Data Fabric shall respond to an "Event beams by Station collection and time range" query returning up to 1400 **Event** beam **Waveform ChannelSegment** objects (populated with claim checks; query based on 70 **Station** objects, beams for 20 **Event** objects and 1 **PhaseType**) in less than 3 seconds.
2. /waveform/channel-segment/store
   a. Stores the provided **ChannelSegment** objects, along with their associated *maskedBy* **ProcessingMask** objects and the derived **Channel** in each **ChannelSegment** object's **ChannelSegmentDescriptor**.
   b. This operation has the following performance requirements:
      i. The Data Fabric shall respond to a request to store up to 1000 **ChannelSegment** objects with **TimeseriesType** of Waveform with **Waveform** samples of length 5 minutes by storing the **ChannelSegment** objects within 5 seconds.
3. /waveform/qcsegment/query/channels-timerange
   a. Finds **QcSegment** objects associated to any **Channel** in the provided collection and which occur in the provided time range.
   b. A *changedSinceTime* can be optionally included to return only **QcSegment** objects that were stored on or after this time, to allow for polling.
   c. This operation has the following performance requirements:
      i. The Data Fabric shall respond to a "QcSegments by Channel collection and time range" query requesting **QcSegment** objects for up to 3000 **Channel** objects and a 1 hour time range, returning a **QcSegment** collection containing up to 1000 **QcSegment** objects and up to 1000 **QcSegmentVersion** objects, in less than 2 seconds.
4. /waveform/qcsegment/update
   a. Stores the provided **QcSegment** objects, updating a **QcSegment** object if it was previously stored.
   b. This operation has the following performance requirements:
      i. The Data Fabric shall respond to a request to store up to 100 **QcSegment** objects (each with 1 new **QcSegmentVersion**) by storing the appropriate **QcSegment** objects within 1 second.

Additional Performance Requirements

1. The Data Fabric recognizes changes to the available **ChannelSegment** objects and returns the **ChannelSegment** objects in the results of its request-response operations. These data updates have the following performance requirements:
   a. The Data Fabric's response to a query providing **ChannelSegment** objects shall include a new or updated **ChannelSegment** object if the query occurs no later than 5 seconds after data affecting the **ChanneSegment** object is stored to the USNDC database.
   b. The Data Fabric's response to a query providing **ChannelSegment** objects shall include a new or updated **ChannelSegment** object if the query occurs no later than 5 seconds after data affecting the **ChanneSegment** object is stored to the Data Fabric.

2. The Data Fabric recognizes changes to the available **QcSegment** objects and returns the **QcSegment** objects in the results of request-response operations. These data updates have the following performance requirements:
    a. The Data Fabric's response to a query providing **QcSegment** objects shall include a new or updated **QcSegment** object if the query occurs no later than 5 seconds after data affecting the **QcSegment** object is stored to the USNDC database.
    b. The Data Fabric's response to a query providing **QcSegment** objects shall include a new or updated **QcSegment** object if the query occurs no later than 5 seconds after data affecting the **QcSegment** object is stored to the Data Fabric.

Response Status Codes

The OpenAPI endpoint descriptions include response status codes and response bodies for successful responses and specific error responses. This always includes behavior for "200 OK" responses and often includes "209 Partial Success" responses. The 209 status code is a GMS specific code typically used for batch operations which succeed for some provided elements but fail for others. The OpenAPI endpoint descriptions do not include descriptions for common response codes such as 400 series client errors or 500 series server errors unless a specific behavior is expected. The Data Fabric should return these responses when appropriate.

FDSN Operations

GMS uses the Data Fabric provided FDSN web services implementations to load raw waveform samples and to determine which waveform samples the Data Fabric can provide. The following operations need to be implemented:

1. /dataselect/1/query
    a. Loads FDSN format waveforms for the provided SEED-format **Channel** identifiers and time range.
    b. In addition to the FDSN specification, the endpoint must also:
        i. Support alternate response types:
            1. Use the `accept` request header to determine the response format.
            2. Support the following response formats:
                a. `application/json`
                b. `application/msgpack`
        ii. Support alternate date-time formats:
            1. Use a custom HTTP request header (see below) to determine whether the service uses ISO-8601 or Unix Timestamp formatted date-time values.
        iii. Support requests for waveforms by their claim checks by allowing requests to include waveform identifier (e.g. wfid) collections instead of the SEED-format **Channel** identifiers (network, station, location, and channel codes).
    c. This operation has the following performance requirements:
        i. The Data Fabric shall respond to an FDSN Waveform Data Select query requesting waveforms for up to 30 **Channel** objects, each 90 minutes duration at 40Hz sampling rate, in less than 6 seconds (goal: 3 seconds).
        ii. The Data Fabric shall simultaneously handle up to 75 FDSN Waveform Data Select queries (up to 30 **Channel** objects, 90 minutes duration, 40Hz sampling rate) without exceeding the response time requirement for any of the queries.
        iii. The Data Fabric shall respond to a FDSN Waveform Data Select query for waveforms by provided identifiers requesting up to 900 **Waveform** objects, each 5 minutes duration at 40Hz sampling rate (i.e. 12,000 samples) in less than 3 seconds.

2. /availability/1/query
    a. Provides the time ranges of available waveform samples for the provided SEED-format **Channel** identifiers and time range. To support GMS use cases, this operation should:
        i. Support the `show` parameter assigned to the "latestupdate" value.
        ii. Return the latest update times for each timespan, potentially grouped in a collection with other timespans for the same SEED-format **Channel** identifier which have the same latest update time.
        iii. Support a new (i.e. an extension to the FDSN specification), optionally provided date-time parameter named `changedsince`. When provided, this endpoint limits its responses to include only timespans with "latestupdate" times on or after the provided `changedsince` value.
    b. In addition to the FDSN specification, the endpoint must also:
        i. Support alternate response types:
            1. Use the `accept` request header to determine the response format.
            2. Support the following response formats:
                a. `application/json`
                b. `application/msgpack`
        ii. Support alternate date-time formats:
            1. Use a custom HTTP request header (see below) to determine whether the service uses ISO-8601 or Unix Timestamp formatted date-time values.
    c. This operation has the following performance requirements:
        i. The Data Fabric shall respond to an FDSN Waveform Availability query requesting waveform sample availability changes since a provided date-time for up to 1500 **Channel** objects, returning up to 100 timespans, in less than 1 second.

Custom HTTP Header

A custom HTTP Header is used to notify the Data Fabric of the format of date-time and duration attributes in the request and to instruct the Data Fabric to return responses that use the same date-time and duration format. The header is named `time-format` and may have the values of ISO and EPOCH, corresponding to the ISO-8601 date and time format and the UNIX Timestamp format (i.e. date-time in epoch seconds, duration in seconds; date-time and duration both represented with floating point numbers to support fractional seconds), respectively. If no header is included, the time and date format should be ISO-8601.

References

1. Faceted Data Class Design Pattern - this page describes the faceting concept used throughout the GMS COI.
2. (Attic) Waveform Quality Control Bridges - this page describes how the GMS developed data bridge loaded and converted the legacy USDNC format records into COI format **QcSegment** objects. Since the Data Fabric now provides the data bridge, this page is provided only as a reference. It will not be updated if the **QcSegment** data model changes, the legacy USNDC format database structure changes, etc.
3. Data Fabric Bridge Conversion Parameters - this page describes the **QcSegmentBridgeDefinition** class.
4. FDSN Web Services - this page provides specifications for each FDSN service.

Change History

1. 05/2025 - update
    a. Removed the operation `/waveform/query/wfdisc-waveform-claim-check`
    b. Updated operation `/waveform/query/event-beams-by-stations-and-timerange` to use a provided **Stage** collection rather than a single **Stage**.
2. 03/2025 - update
    a. Provided basic descriptions and performance requirements for the FDSN waveform Availability and Data Select services.
    b. Updated the valid values of the `time-format` HTTP Header (replaced TIMESTAMP with EPOCH).
3. 01/2025 - update
    a. Updated the **QcSegment** query operation `/waveform/qcsegment/query/channels-timerange` to support polling.
4. 12/2024 - update
    a. Updated the **Event** beam operation to support polling, replacing the operation `/waveform/query/event-beams-by-event-hypotheses-and-stations` with `/waveform/query/event-beams-by-stations-and-timerange`
5. 10/2024 - update
    a. Described the **QcSegment** store or update operation `/waveform/qcsegment/update`
    b. Described the **ChannelSegment** store operation `/waveform/qcsegment/update`
6. 10/2024 - cleanup
    a. Removed the Data Fabric operation `/waveform/qcsegment/bridge/set-configuration`
7. 09/2024 - update
    a. "Find Event beams by EventHypothesis and Station collections" query.
8. 08/2024 - update
    a. **ChannelSegment** COI data model description.
    b. "Waveforms by WFDISC claim check" query.
9. 07/2024 - Initial release
    a. **QcSegment**, **QcSegmentVersion**, and initial **ProcessingMask** COI data model description.
    b. "QcSegments by Channel collection and time range" query.

Channel Segment COI Data Model

Table of Contents

List of Figures

List of Tables

Data Model

Channel Segment

The figure below shows the **ChannelSegment** classes that represent the time-series data that is acquired and processed by GMS. The **ChannelSegment** class provides basic information about the data (*startTime*, *endTime*, associated **Channel**, etc.), but does not include the actual data samples. The data itself is contained in concrete specializations of the abstract **BaseTimeseries** class. Note that each **ChannelSegment** includes a non-empty **BaseTimeseries** collection, allowing for the possibility that a long **ChannelSegment** may have gaps or sample rate changes within it. The actual data samples are split into the collection such that each object contains a continuous interval of valid data with the same sample rate (i.e. a **BaseTimeseries** object is not intended to contain gaps or represent variable sample rate data).

The specializations of the **BaseTimeseries** class that may be contained in **ChannelSegment** instances include the following:

1. **Timeseries** - an abstract base class extending **BaseTimeseries** with attributes included in all of the **BaseTimeseries** specializations which include actual data samples.
    a. A **Waveform** is a one-dimensional **Timeseries** representing a geophysical measurement collected from a **Channel**, or a derived quantity created by processing data from one or more **Channels** (e.g., beams, filters, etc.).
    b. An **FkSpectra** is a multi-dimensional collection of power values over time derived from **Waveform** data from a **Channel** collection. The FK (frequency-wavenumber) transform converts a **Waveform** collection from three or more **Channel** objects into a time-by-slowness

(north-south and east-west) array of coherency values, over a given two-dimensional range of slowness values, and filtered over a given frequency band. This is the primary way that array stations can accurately measure the azimuth and slowness of an arrival of energy (i.e., a **SignalDetectionHypothesis**).

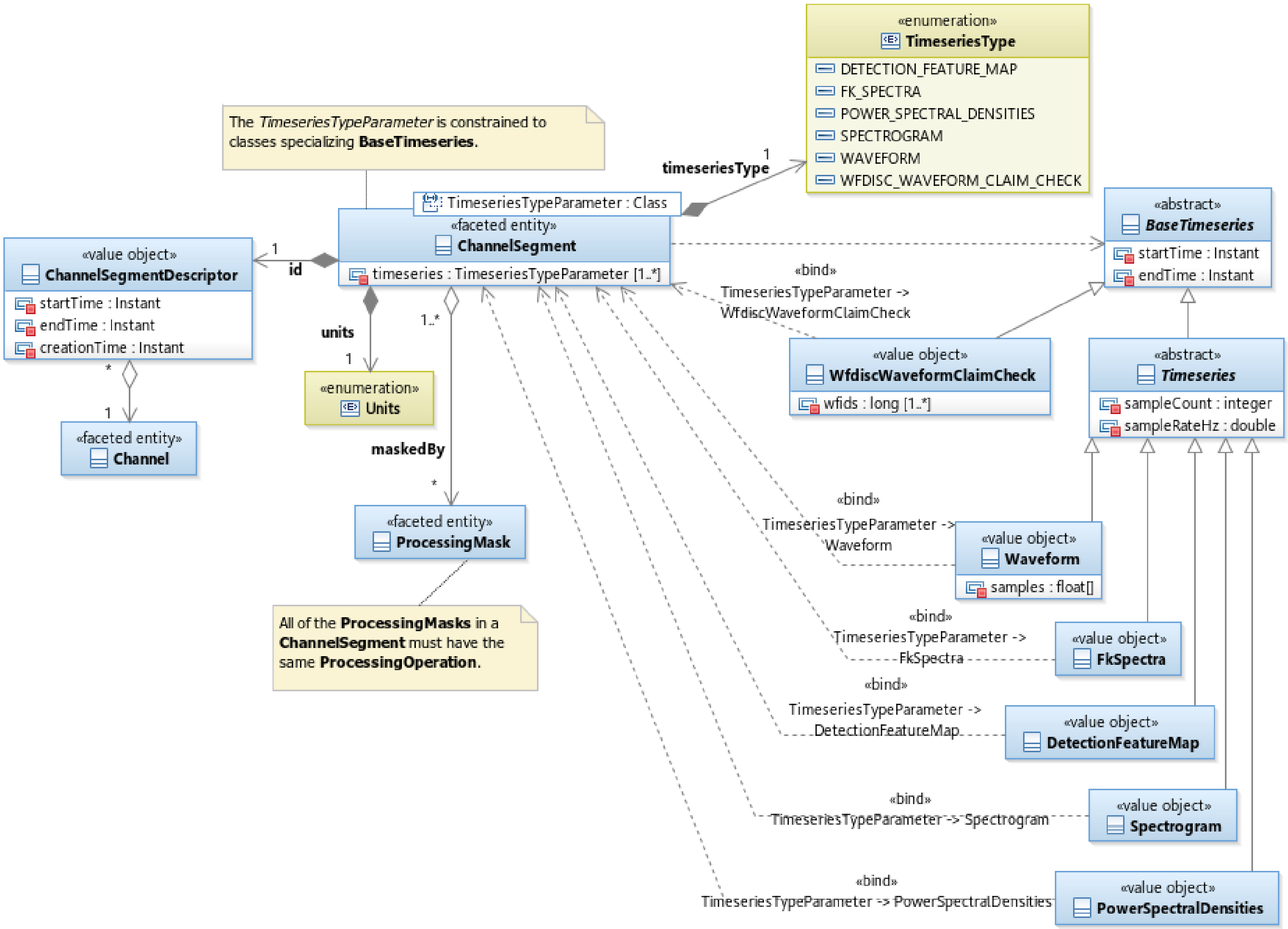
c. A **PowerSpectralDensities** is a **Timeseries** with each sample representing a power spectral density derived from a segment of **Wavefor** data produced by a **Channel**.
d. A **Spectrogram** is a **Timeseries** with each sample representing a power spectrum derived from a segment of **Waveform** data produced by a **Channel**.
e. A **DetectionFeatureMap** is a generalized matrix of values for a specific feature over time as produced by processing a **Channel** collection from a **Station**. The matrix contains a feature vector calculated for each point in time based on the processing of one or more **Channel** objects from the **Station**. The feature vector is a set of values indexed by secondary independent variables (e.g., frequency). For example, the Progressive Multi-Channel Correlation (PMCC) processing algorithm used for infrasound signal detection produces a **DetectionFeatureMap**, consisting of coherency, azimuth, and apparent velocity as a function of time and frequency.

2. **WfdiscWaveformClaimCheck** - contains identifying information that can be used to load a **Waveform** object from a data store providing WFDISC data in the Center for Seismic Studies 3.0 (CSS 3.0) format (nominally; the actual implementation may vary from the standard).

ChannelSegment is faceted and has several possible instantiations:

1. References contain only a populated *id* (all of the other **ChannelSegment** attributes are unpopulated).
2. Populated objects contain values for every attribute.

Figure 1: **ChannelSegment** class structure



ChannelSegment has the following attributes:

> **Note**
>
> This table's *Populated* column refers to whether each attribute is optional or always populated in a populated **ChannelSegment** object. Other attribute populations are possible since **ChannelSegment** is a faceted class and can represent a reference or a populated object. See the Channel Segment overview above for details.

Table 1: ChannelSegment

Attribute	DataType	Units	Range	Populated	Default Facet Population	Description
id	**ChannelSegementDescriptor**	N/A	N/A	Always	N/A	The **ChannelSegment** object's unique identifier.
maskedBy	**ProcessingMask** collection	N/A	N/A	Always	**ProcessingMask** objects: references	Contains all of the **ProcessingMask** objects affecting this derived **ChannelSegment**. Each **ProcessingMask** indicates raw **ChannelSegment** samples that were masked in the processing that eventually created this **ChannelSegment**. The **ProcessingMask** objects provide provenance about how this derived **ChannelSegment** was created. While a **ProcessingMask** may only mask raw waveform samples, all subsequent downstream processing is affected by the masked samples. Each of these **ProcessingMask** objects must have the same **ProcessingOperation** literal assigned to their *processingOperation* attribute. Always empty for raw **ChannelSegment** objects. Possibly empty for derived **ChannelSegment** objects.
timeseries	**TimeseriesTypeParameter**[1..*] (A collection of one of the concrete specializations of the **BaseTimeseries** class)	N/A	N/A	Always	N/A	A collection of objects containing the actual data samples. Each object is of the same type, which must be one of the concrete **TimeseriesBase** specializations. The concrete type corresponds to this **ChannelSegment** object's *timeseriesType* attribute value. The objects in this collection are sorted by time and cannot overlap in time. Each object contains a continuous series of data samples with a consistent sample rate.
timeseriesType	**TimeseriesType**	N/A	N/A	Always	N/A	A **TimeseriesType** literal indicating the concrete type of the objects in this **ChannelSegment** object's *timeseries* collection.
units	**Units**	N/A	N/A	Always	N/A	A **Units** literal describing the units of the *timeseries* data samples. Must match the **Units** in the **ChannelSegmentDescriptor** object's associated **Channel**.

ChannelSegmentDescriptor includes the identifying information for a **ChannelSegment**. It has the following attributes:

Table 2: ChannelSegmentDescriptor

Attribute	DataType	Units	Range	Populated	Default Facet Population	Description
channel	**Channel**	N/A	N/A	Always	Version reference	The **Channel** producing the *timeseries* data samples.
creationTime	(ISO-8601 date and time)	Varies / handled by ISO-8601.	>= end*Time*	Always	N/A	The time when the **ChannelSegment** was created. Since the data available for a **Channel** within a time range may change due to newly acquired data filling in gaps, *creationTime* is needed to accurately recreate a **ChannelSegment** (i.e. two **ChannelSegment** objects produced by the same **Channel** for the same time range but with different *creationTime* values may have different values for their *timeseries* samples).
endTime	(ISO-8601 date and time)	Varies / handled by ISO-8601.	>= *startTime*	Always	N/A	An inclusive time marking the end of the **ChannelSegment**. It is the same as the *endTime* of the latest element in the **ChannelSegment** *timeseries* collection.
startTime	(ISO-8601 date and time)	Varies / handled by ISO-8601.	N/A	Always	N/A	An inclusive time marking the beginning of the **ChannelSegment**. It is the same as the *startTime* of the earliest element in the **ChannelSegment** *timeseries* collection.

TimeseriesType defines the concrete type of a **ChannelSegment** object's **Timeseries** collection. **TimeseriesType** has the literals listed in the following table. **ChannelSegment** objects with a particular **TimeseriesType** include a *timeseries* collection containing objects of the corresponding **Timeseries** specialization (e.g. a **ChannelSegment** object with a **TimeseriesType** of WAVEFORM contains **Waveform** objects in its *timeseries* collection).

Table 3: TimeseriesType

Literal
DETECTION_FEATURE_MAP
FK_SPECTRA
POWER_SPECTRAL_DENSITIES

SPECTROGRAM
WAVEFORM
WFDISC_WAVEFORM_CLAIM_CHECK

BaseTimeseries is the abstract base class specialized by each of the concrete timeseries classes. It contains attributes shared by all of the specializations. **BaseTimeseries** has the following attributes:

Table 4: BaseTimeseries

Attribute	DataType	Units	Range	Populated	Description
endTime	Instant	N/A	>= *startTime*	Always	The inclusive ending time of the data samples. This is the same as the sample time of the last sample.
startTime	Instant	N/A	N/A	Always	The inclusive starting time of the data samples. This is the same as the sample time of the first sample.

Timeseries is an abstract base class extending **BaseTimeseries** with attributes shared by all of the **BaseTimeseries** specializations which include actual data samples. Because **Timeseries** objects represent a continuous collection of data samples (i.e. without gaps) at a constant sampling rate, **Timeseries** objects generally do not need to contain an array of sample time values. Instead, the first sample occurs at *startTime* (defined in **BaseTimeseries**) and each subsequent sample occurs 1/*sampleRateHz* later, with the last sample occurring at *endTime* (defined in **BaseTimeseries**). **Timeseries** has the following attributes:

Table 5: Timeseries

Attribute	DataType	Units	Range	Populated	Description
sampleCount	integer	N/A	N/A	Always	The number of samples in this **Timeseries**.
sampleRateHz	double	Hz	> *0Hz*	Always	The sample rate (in samples per second) for all of the samples in this **Timeseries**.

Waveform

The **Waveform** class includes a one-dimensional array of data samples for the sampling times indicated by the **Timeseries** and **BaseTimeseries** object. Alternatively, the waveform data can instead be represented as a claim check containing identifying information that can be used to load the waveform samples, independent of the other **ChannelSegment** contents, from a data store.

> ⚠ **Implementation Note**
>
> The GMS COI uses the Faceted Data Class Design Pattern to avoid populating large data objects, such as waveform data samples, when they are not needed. Since claim checks solve a similar problem, these approaches have overlapping responsibilities within the **ChannelSegment** data model. GMS includes waveform claim checks to ease integration with the Data Fabric.

Waveform has the following attributes:

Table 6: Waveform

Attribute	DataType	Units	Range	Populated	Description
samples	float[]	Defined by the **ChannelSegment** attribute *units*	N/A	Always	The data samples for the sample times defined by the time range, sample rate, and sample count attributes defined in the **BaseTimeseries** and **Timeseries** classes: 1. The first and last samples have sample times indicated by the **BaseTimeseries** *startTime* and *endTime* attributes. 2. The *samples* array is of the same length as the **Timeseries** *sampleCount* attribute value. 3. Each sample is separated from adjacent samples by the same duration. This is represented as a sampling rate in the **Timeseries** *sampleRateHz* attribute.

WfdiscWaveformClaimCheck represents a claim check used to query waveform data samples from a CSS 3.0 or similarly structured database. A **WfdiscWaveformClaimCheck** corresponds to a single **Waveform** object that can be constructed using the identified WFDISC records, limiting the loaded samples to the time range as defined by the **WfdiscWaveformClaimCheck** in its the **BaseTimeseries** attributes *startTime* and *endTime*.

WfdiscWaveformClaimCheck has the following attributes:

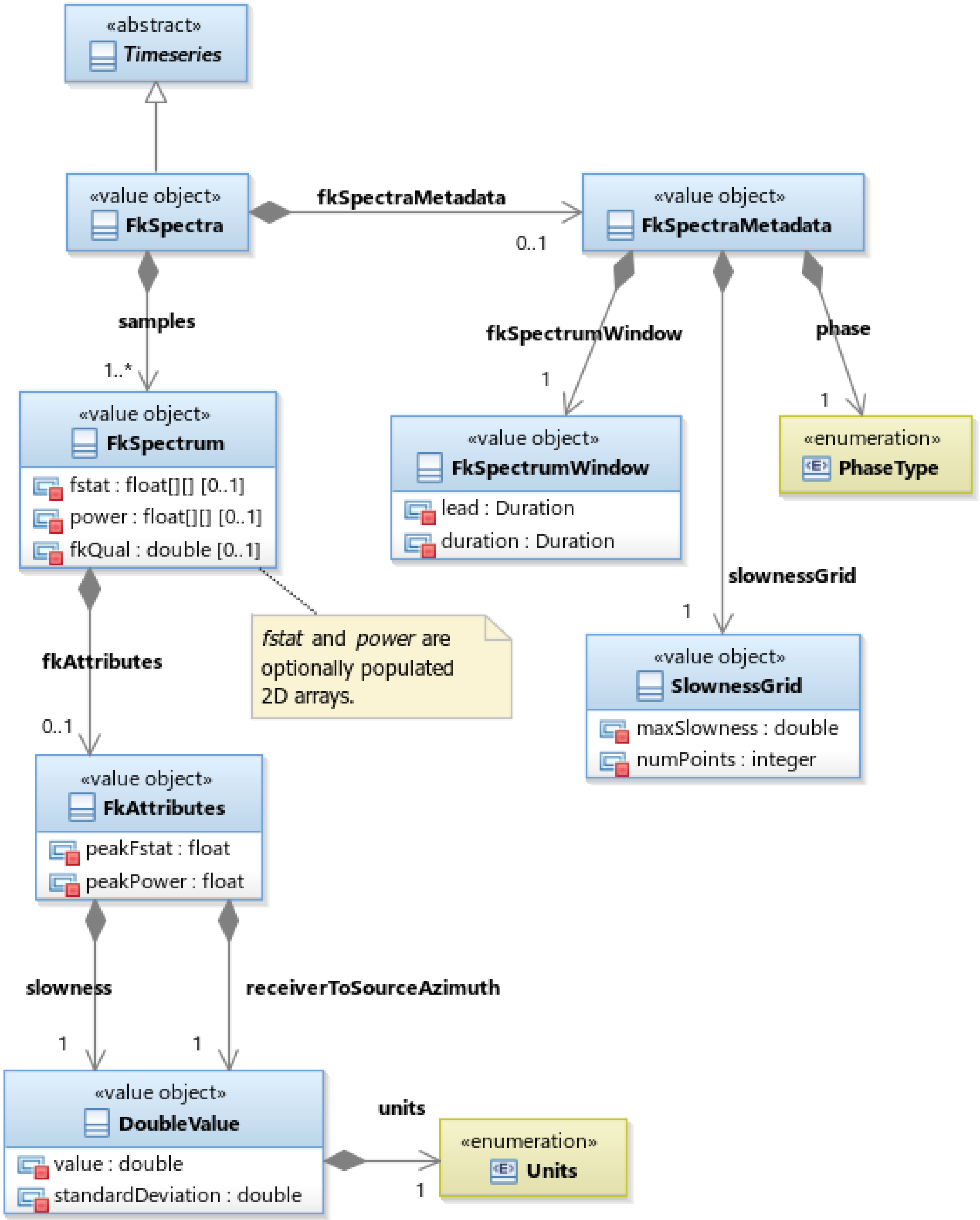
Table 7: WfdiscWaveformClaimCheck

Attribute	DataType	Units	Range	Populated	Description
wfids	long[1..*]	N/A	N/A	Always	A non-empty collection of WFDISC record identifiers for the records containing a **ChannelSegment** object's **Waveform** data samples. The **BaseTimeseries** *startTime* and *endTime* attributes further limit the **Waveform** samples times within the **Waveform** object retrievable using this claim check. The Center for Seismic Studies 3.0 Database Schema defines the nominal WFDISC table format. The table's structure may vary in actual implementations.

FkSpectra

The **FkSpectra** class is a type of **Timeseries** where each sample is an **FkSpectrum**. The *startTime* is the reference time of the first **FkSpectrum** in the series of **FkSpectrum** and the *endtime* is the reference time of the last **FkSpectrum**. The *sampleRateHz* is the number of **FkSpectrum** per second and the *sampleCount* is the total number of **FkSpectrum**. A **FkSpectrum** optionally includes a two-dimensional array *power* containing the power values for different x, y slowness coordinates and optionally includes a two-dimensional array *fstat* containing the fstat values for different x, y slowness coordinates. The **FkSpectrum** also optionally contains *fkQual* (a measure of how clear the peak power is) and *fkAttributes*. **FkAttributes** contains information about the peak in an **FkSpectrum** object's 2-dimensional *power* and *fstat* arrays; it has *peakFstat,* the maximum *fstat* value*,* and *slowness* and *sourceToReceiverAzimuth* which are the slowness and angle, relative to the **Channel** producing the **FkSpectra**, to reach the maximum power in the *power* array. **FkSpectra** also optionally contains **FkSpectraMetadata**. **FkSpectraMetadata** has information which is also contained in **FkSpectraDefinition**; it is used for convenience in finding information about parameters that determined how the **FkSpectrum** was created. The *slownessGrid* in **FkSpectraMetadata** has *maxSlowness*, the maximum slowness from the origin, and *numPoints*, the number of slowness increments from [-*maxSlowness* to + *maxSlowness*]; these parameters determine the slowness coordinates of the *fstat* and *power* arrays in **FkSpectrum**. The *phase* is the **PhaseType** of the signal that was used in calculating **FkSpectrum**. The *fkSpectrumWindow* contains *duration*, which is the duration of the time window that a single **FkSpectrum** is calculated over and *lead*, which is the lead before the reference time at which the **FkSpectrum** starts.

Figure 2: **FkSpectra** class structure



FKSpectra has the following attributes:

Table 8: FkSpectra

Attribute Name	Data Type	Units	Range	Populated	Description
fkSpectraMetadata	**FkSpectraMetadata**	N/A	N/A	Optional	Limited parameters describing the contents of the **FkSpectrum** and how they were created. Because the **FkSpectraDefinition** also includes this information, this attribute is optional and may be unpopulated in contexts where the **FkSpectraDefinition** is also available.
samples	**FkSpectrum** collection	N/A	N/A	Always	The time ordered collection of **FkSpectrum** for the **FkSpectra**. The first **FkSpectrum** has sample time equal to the **FkSpectra** *startTime* (see **BaseTimeseries**).

The **FkSpectraMetadata** class contains information describing the contents and calculation of the **FkSpectrum** collection. The **FkSpectraDefinition** also includes this information, along with other parameters more completely describing the FK calculation. **FkSpectraMetadata** is a convenience class to quickly see what parameters were used in the **FkSpectrum** calculation.

FkSpectraMetadata has the following attributes:

Table 9: FkSpectraMetadata

Attribute Name	Data Type	Units	Range	Populated	Description
fkSpectrumWindow	**FkSpectrumWindow**	N/A	N/A	Always	Defines the interval of **Waveform** data used to compute the **FkSpectrum**, relative to a reference time (i.e. the **FkSpectrum** object's sample time).
phase	**PhaseType**	N/A	N/A	Always	The expected phase of a signal in the **Waveform** collection used to compute the **FkSpectra**.
slownessGrid	**SlownessGrid**	N/A	N/A	Always	The parameters that determine the size of the *power* and *fstat* arrays.

This class contains the information that determines the window start and end times for the **Waveform** samples used to calculate an **FkSpectrum**. **FKSpectrumWindow** has the following attributes:

Table 10: FkSpectrumWindow

Attribute Name	Data Type	Units	Range	Populated	Description
duration	Duration (ISO-8601 time duration)	Varies / handled by ISO-8601. Will be a unit of elapsed time (e.g. seconds)	>0.0s	Always	The duration of the time window of **Waveform** data used to create an **FkSpectrum**.
lead	Duration (ISO-8601 time duration)	Varies / handled by ISO-8601. Will be a unit of elapsed time (e.g. seconds)	>=0.0s	Always	The lead before the reference time used to create an **FkSpectrum**.

This class contains the parameters that determine the grid of *fstat* and *power* values in an **FkSpectrum**. **SlownessGrid** has the following attributes:

Table 11: SlownessGrid

Attribute Name	Data Type	Units	Range	Populated	Description
maxSlowness	double	seconds /degrees	>0.0 sec /deg	Always	The maximum slowness from the origin (centered at (0,0)). The **FkSpectrum** *fstat* and *power* arrays are square and this value is 1/2 the length of that square's edges.
numPoints	integer	N/A	>= 1, odd	Always	The number of points (increments) between -maxSlowness and maxSlowness. This value must be odd because it includes the origin point.

FkSpectrum contains the results, including the *power* and *fstat* arrays, of an **FkSpectrum** calculation. **FkSpectrum** has the following attributes:

Table 12: FkSpectrum

Attribute Name	Data Type	Units	Range	Populated	Description
fkAttributes	**FkAttributes**	N/A	N/A	Optional	Describes the peak in the *fstat* and *power* arrays.
fkQual	double	N/A	N/A	Optional	A measure of the quality of the **FkPowerSpectrum**, which is a measure of the peak's clarity.
fstat	float[][]	N/A	N/A	Optional	The 2-D f-statistic array calculated by the FK analysis.
power	float[][]	dB	N/A	Optional	The 2-D power array calculated by the FK analysis.

FkAttributes holds the *peakFstat* value and the *receiverToSourceAzimuth* and *slowness* of the peak power value. These values may be determined by an algorithm or selected by an Analyst. **FkAttributes** has the following attributes:

Table 13: FkAttributes

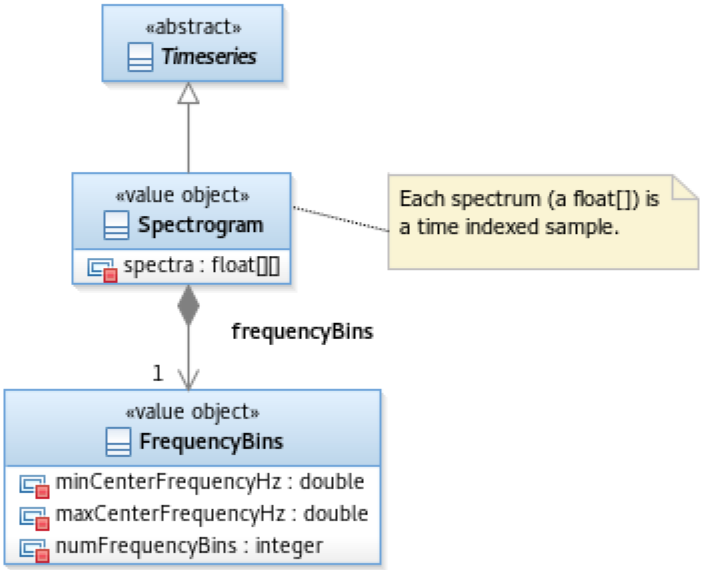
Attribute Name	Data Type	Units	Range	Populated	Description
peakFstat	float	N/A	N/A	Always	The maximum value in the 2-D array of the f-statistic.
peakPower	float	N/A	N/A	Always	The maximum value in the 2-D array of the power.
recieverToSourceAzimuth	**Double Value**	Defined in the **DoubleValue**; expected to be degrees	0<= *azimuth*<=360 or NaN if directly below **Station**	Always	The *value* of the **DoubleValue** is the angle to the peak power in the power array measured clockwise from north, relative to the location of the derived **Channel** producing the **FkSpectra**. The *standardDeviation* of the **DoubleValue** is the azimuth uncertainty. If there is no azimuth uncertainty provided, the *standardDeviation* will be an empty optional.
slowness	**Double Value**	Defined in the **DoubleValue**; expected to be seconds /degree	*-2*maxSlowness<=* *slowness* *<=2*maxSlowness* Note: the **Slowness Grid** and **FkSpectra Definition** class es define *maxSlowness*.	Always	The *value* of the **DoubleValue** is the slowness from the origin to the peak power. The *standardDeviation* of the **DoubleValue** is the slowness uncertainty. When the slowness uncertainty is unknown, the *standardDeviation* will be an empty optional.

Spectrogram

The **Spectrogram** class is a type of **Timeseries** where each sample is a 1-D array of power values for the frequency bins defined by its **FrequencyBins** (i.e. a spectrum). A **Spectrogram** represents the spectrum **Timeseries** using a spectrum array (i.e. a 2-D array). Within a **Spectrogram**, each spectrum's sample time is the start time of the **Waveform** time window used to compute the spectrum. A **Spectrogram** object's *startTime* is therefore the start time of the **Waveform** window used to create the first spectrum via a Fourier transform, and its *endTime* is the start time of the **Waveform** window used to create its last spectrum. A **Spectrogram** object's *sampleRateHz* is its number of spectra per second. A **Spectrogram** object's *sampleCount* is the total number of spectra it contains.

A **Spectrogram** object's **FrequencyBins** describes the frequency values associated with each entry in each spectrum. **FrequencyBins** attributes *minCenterFrequencyHz* and *maxCenterFrequencyHz* define the center frequency of the frequency bins represented by the spectrum's first and last elements. **FrequencyBins** attribute *numFrequencyBins* defines the number of frequency bins in each spectrum, which is equivalent the the length of each spectrum's 1-D power array. Each frequency bin has the same width. The first frequency bin includes frequencies centered on *minCenterFrequencyHz* (i.e. [*minCenterFrequencyHz* - width/2, *minCenterFrequencyHz* + width/2]), and each subsequent bin is centered on the frequency equal to the previous bin's center frequency plus the frequency bin width.

Figure 3: **Spectrogram** class


«abstract»
Timeseries
«value object»
Spectrogram
spectra : float[][]
Each spectrum (a float[]) is a time indexed sample.
frequencyBins
1
«value object»
FrequencyBins
minCenterFrequencyHz : double
maxCenterFrequencyHz : double
numFrequencyBins : integer

Spectrogram has the following attributes:

Table 14: **Spectrogram**

Attribute	DataType	Units	Range	Populated	Description
frequencyBins	**FrequencyBins**	N/A	N/A	Always	Describes the frequency bins corresponding to the entries in each of this object's spectra.

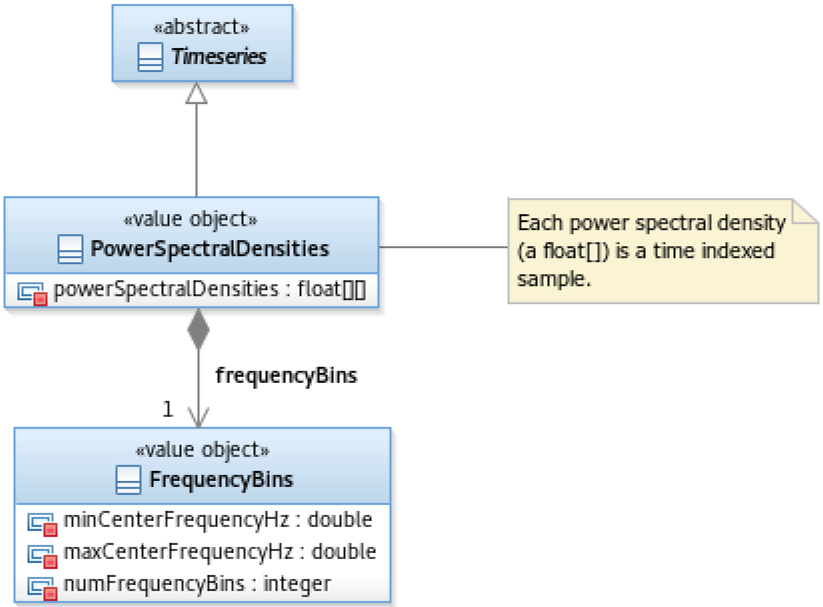
spectra	float[][]	Units of power (e.g. MICROPASCALS_SQUARED_PER_SECOND , NANOMETERS_SQUARED_PER_SECOND, PASCALS_SQUARED_PER_SECOND).	N/A	Always	The **Spectrogram Timeseries** samples. Each sample is a spectrum represented by a 1-D array of power values ordered by increasing center frequency, so a sample series is a spectrum array (i.e. an array of power value arrays) ordered by increasing sample time. Each spectrum element corresponds to a power value for a frequency bin. This object's *frequencyBins* describes the frequency bins.

PowerSpectralDensities

The **PowerSpectralDensities** class is a type of **Timeseries** where each sample is a power spectral density (i.e. a 1-D array of power values) for the frequency bins defined by its **FrequencyBins**. **PowerSpectralDensities** represents its **Timeseries** using an power spectral density array (i.e. a 2-D array), though a **PowerSpectralDensities** object will typically include a single sample representing the power spectral density for a particular time range of **Waveform ChannelSegment** samples. Within **PowerSpectralDensities**, each sample's time is the start time of the **Waveform** time window used to compute the corresponding power spectral density. A **PowerSpectralDensities** object's *startTime* is therefore the start time of the **Waveform** window used to create its first power spectral density, and its *endTime* is the start time of the **Waveform** window used to create its last power spectral density. In the typical case of a **PowerSpectralDensities** object including a single sample, the object's *startTime* and *endTime* are equal. A **PowerSpectralDensities** object's *sampleRateHz* is its number of power spectral densities per second. A **PowerSpectralDensities** object's *sampleCount* is the total number of power spectral densities it contains.

A **PowerSpectralDensities** object's **FrequencyBins** describes the frequency values associated with each entry in each sample (i.e. each entry in each power spectral density). **FrequencyBins** attributes *minCenterFrequencyHz* and *maxCenterFrequencyHz* define the center frequency of the frequency bins represented by the power spectral density's first and last elements. **FrequencyBins** attribute *numFrequencyBins* defines the number of frequency bins in each power spectral density, which is equivalent the the length of their 1-D power arrays. Each frequency bin has the same width. The first frequency bin includes frequencies centered on *minCenterFrequencyHz* (i.e. [*minCenterFrequencyHz* - width/2, *minCenterFrequencyHz* + width/2]), and each subsequent bin is centered on the frequency equal to the previous bin's center frequency plus the frequency bin width.

Figure 4: **PowerSpectralDensities** class


«abstract»
Timeseries
«value object»
PowerSpectralDensities
powerSpectralDensities : float[][]
Each power spectral density (a float[]) is a time indexed sample.
frequencyBins
1
«value object»
FrequencyBins
minCenterFrequencyHz : double
maxCenterFrequencyHz : double
numFrequencyBins : integer

PowerSpectralDensities has the following attributes:

Table 15: **PowerSpectralDensities**

Attribute	DataType	Units	Range	Populated	Description
powerSpectralDensities	float[][]	Units of power (e.g. MICROPASCALS_SQUARED_PER_SECOND, NANOMETERS_SQUARED_PER_SECOND, PASCALS_SQUARED_PER_SECOND).	N/A	Always	An array of power spectral densities. Each power spectral density is represented as a 1-D array; each entry in each power spectral density corresponds to a frequency bin as defined by *frequencyBins.*
frequencyBins	**FrequencyBins**	N/A	N/A	Always	Describes the frequency bins corresponding to the entries in each of this object's power spectral densities.

Detection Feature Map

Future Work

The **DetectionFeatureMap** COI data model will be described when GMS begins using **DetectionFeatureMap** data.

Common

Frequency Bins

FrequencyBins defines equal width frequency bins with center frequencies between its *minCenterFrequencyHz* and *maxCenterFrequencyHz.* **Frequency Bins** has the following attributes:

Table 16: **FrequencyBins**

Attribute	DataType	Units	Range	Populated	Description
maxCenterFrequencyHz	double	Hz	>0 Hz	Always	The highest center frequency (inclusive).
minCenterFrequencyHz	double	Hz	>=0 Hz	Always	The lowest center frequency (inclusive).

numFrequencyBins	Integer	N/A	>0	Always	The number of frequency bins.
					Each frequency bin has the same width, defined as follows:
					`binWidthHz = (maxCenterFrequencyHz - minCenterFrequencyHz) / (numFrequencyBins - 1)`
					which results in frequency bins with the following center frequencies:
					`[minCenterFrequencyHz,`
					`minCenterFrequencyHz + binWidthHz,`
					`minCenterFrequencyHz + (2 * binWidthHz),`
					`...,`
					`maxCenterFrequencyHz = minCenterFrequencyHz + ((numFrequencyBins - 1) * binWidthHz)]`
					For example, if
					minCenterFrequencyHz = 0.0Hz, *minCenterFrequencyHz* = 19.375Hz, and *numFrequencyBins* = 32,
					then *binWidthHz = (19.375Hz / 31 bins = .625Hz/bin)*
					and *binCenterFrequenciesHz* = [0.0Hz, 0.625Hz, 1.250Hz, ..., 19.375Hz].

Notes

1. None.

Open Issues

1. Similar to **Waveform**, the Data Fabric's **FkSpectra** or **DetectionFeatureMap** data persistence approaches may require GMS to use claim checks to access the data samples.

Change History

1. PI32 Updates
    a. 07/2025 - Updated **Waveform**, **FkSpectra**, **Spectrogram**, and **PowerSpectralDensities** classes to represent data samples with floats instead of doubles.
2. PI31 Updates
    a. 04/2025 - Added the **Timeseries** specializations **Spectrogram** and **PowerSpectralDensities**.
3. PI30 Updates
    a. 01/2025
        i. **FkSpectrum** arrays *fstat* and *power* now optional.
4. PI29 Updates
    a. 09/2024
        i. Removing **ChannelSegment** attribute *missingInputChannels*.
    b. 08/2024
        i. Restructured to include only COI data model contents, removing the Repository component descriptions.
        ii. Extended **Timeseries** data model to support **Waveform** claim checks.
5. PI24 Updates

    a. 07/2023 - Added FkSpectra and associated classes
6. PI23 Updates
    a. 04/2023 - Described the *findEventBeamsByEventHypothesesAndStations(...)* operation to the **WaveformRepository** interface.
7. PI13
    a. 10/2020 - Initial architecture description.

TODO

1. None.

Waveform Quality Control COI Data Model

Table of Contents

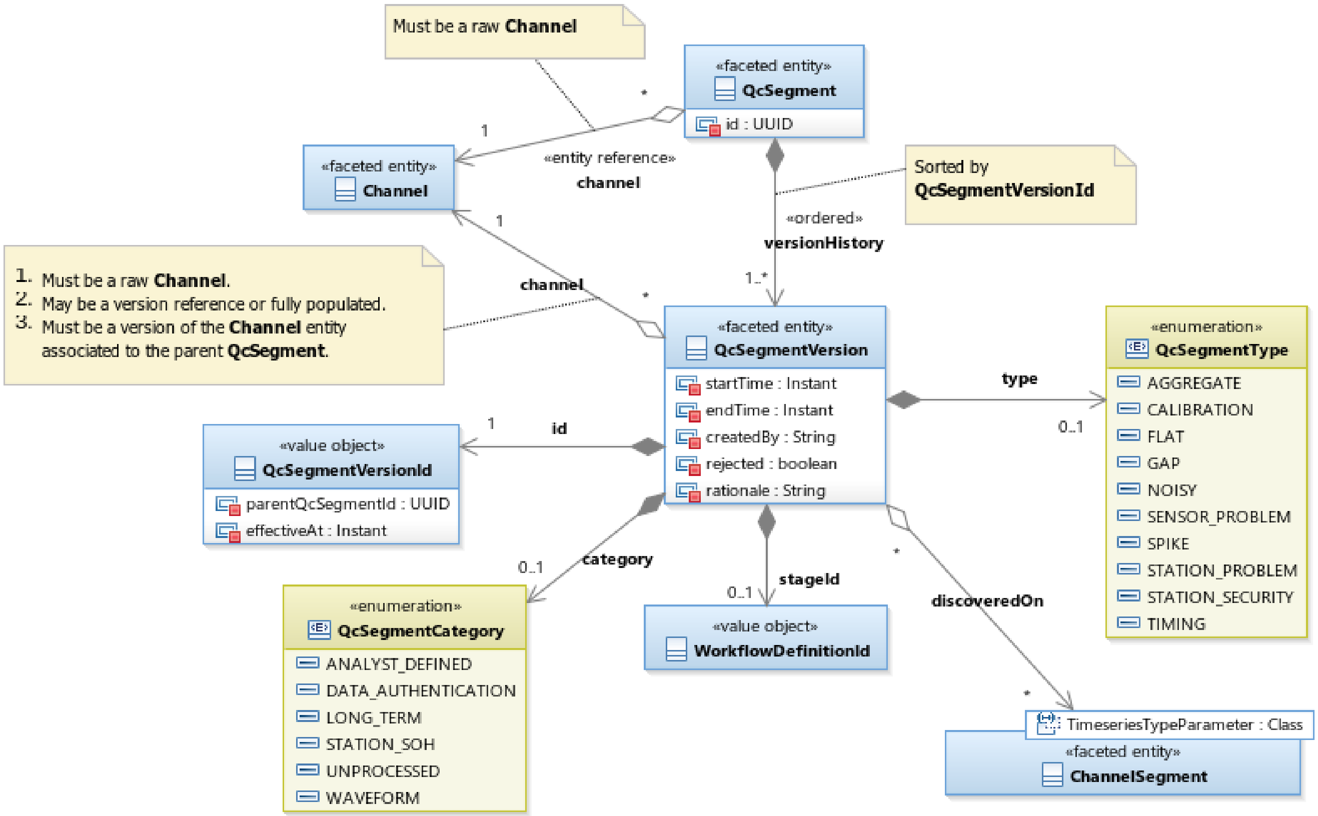
List of Figures

List of Tables

Data Model

QC Segment

Figure 1: **QcSegment** and **QcSegmentVersion** class structure



A **QcSegment** represents an interval of raw waveform data containing a data quality control issue (e.g. missing data, spike, etc.). Analysts and automatic processing may modify **QcSegment** objects, such as by adjusting the time window, so the **QcSegment** class represents a **QcSegment** entity's time history with each **QcSegmentVersion** representing a particular version of the **QcSegment**. The latest **QcSegmentVersion** is the current version (i.e. the **QcSegmentVersion** with the latest *effectiveAt* in its **QcSegmentVersionId**). Each **QcSegmentVersion** may be associated to the *discoveredOn* **ChannelSegment** collection corresponding to the raw waveforms analyzed or processed to find the quality control problem. A **ChannelSegment** may have multiple quality control issues, including multiple issues within the same time frame, so several **QcSegmentVersion** objects may be associated to the same **ChannelSegment**. **QcSegmentVersion** attributes *startTime* and *endTime* refer to the time in the waveform data containing the data quality control issue.

QcSegmentVersion contains a *category* and *type* describing the type of quality control problem. These values help GMS determine which **QcSegmentVersion** objects should become **ProcessingMask** objects (see below) indicating that waveform samples have been excluded for specific data processing operations (e.g. beamforming, rotation). The **QcSegmentVersion** objects within a **QcSegment** may have different *category* or *type* values (e.g. an **Analyst** may adjust the *type* of an ANALYST_DEFINED **QcSegment**). **QcSegmentVersion** attribute *rationale* describes why the waveform samples were marked as containing the particular data quality issue.

QcSegment is faceted and has two possible instantiations:

1. An id-only instantiation has a populated *id* but the other attributes are not populated.
2. A full instantiation contains values for every attribute.

QcSegmentVersion is faceted and has two possible instantiations:

1. An id-only instantiation has a populated *id* but the other attributes are not populated.
2. A full instantiation contains values for every attribute.

QcSegment has the following attributes:

⚠ **Note**

This table's *Populated* column refers to whether each attribute is optional or always populated in a populated **QcSegment** entity object. Other attribute populations are possible since **QcSegment** is a faceted class and can represent an entity reference or a populated entity version. See the QC Segment COI data model overview description above for details.

Table 1: **QcSegment**

Attribute	Data Type	Units	Range	Populated	Default Facet Population	Description
id	UUID	N/A	N/A	Always	N/A	The **QcSegment** object's unique identifier.
channel	**Channel**	N/A	N/A	Always	Entity reference	The raw **Channel** with waveform samples containing the quality control issue. This attribute must be populated as a **Channel** entity reference.
versionHistory	**QcSegmentVersion**[1..*]	N/A	N/A	Always	The current **QcSegmentVersion** object (i.e. the last entry in the collection) is populated according to the **QcSegmentVersion** default faceted population described in the **QcSegmentVersion** table below. All other **QcSegmentVersion** objects in the collection are id-only instances (entity references).	An ordered collection of **QcSegmentVersion** objects representing the time history of this **QcSegment**. The collection is in ascending order based on the **QcSegmentVersionId** *effectiveAt* times.

QcSegmentVersion has the following attributes:

Note

This table's *Populated* column refers to whether each attribute is optional or always populated in a populated **QcSegmentVersion** entity object. Other attribute populations are possible since **QcSegmentVersion** is a faceted class and can represent an entity reference or a populated entity version. See the QC Segment COI data model overview description above for details.

Table 2: **QcSegmentVersion**

Attribute	Data Type	Units	Range	Populated	Default Facet Population	Description
id	**QcSegmentVersionId**	N/A	N/A	Always	N/A	A unique identifier for the **QcSegmentVersion** combining the parent **QcSegment's** *id* with an additional *effectiveAt* time identifying the **QcSegmentVersion** within the **QcSegment**.
channel	**Channel**	N/A	N/A	Always	Version reference	A raw **Channel** with waveform samples containing the quality control issue represented by this **QcSegmentVersion**. This attribute must be populated with a **Channel** version reference or a populated **Channel** object. The **Channel** must be a version of the same **Channel** associated to the parent **QcSegment** *channel* attribute. This **QcSegmentVersion** object's *startTime* and *endTime* must occur within the **Channel** version's effective time range indicated by **Channel** *effectiveAt* and *effectiveUntil*.
category	**QcSegmentCategory**[0..1]	N/A	N/A	Optional. Rejected **QcSegmentVersions** do not have a **QcSegmentCategory**.	N/A	A broad description of the quality control issue marked by this **QcSegmentVersion**.
type	**QcSegmentType**[0..1]	N/A	N/A	Optional. Some **QcSegmentCategory** literals do not have any corresponding **QcSegmentTypes**. A rejected **QcSegmentVersion** has an unpopulated **QcSegmentType**.	N/A	A specific description of the quality control issue marked by this **QcSegmentVersion**. A **QcSegmentVersion** object's **QcSegmentCategory** determines the possible **QcSegmentType** literals.
startTime	Instant (ISO-8601 Date and Time)	Varies / handled by ISO-8601	N/A	Always	N/A	The time (inclusive) of the first data sample containing the quality control issue.

endTime	Instant (ISO-8601 Date and Time)	Varies / handled by ISO-8601	N/A	Always	N/A	The time (inclusive) of the last data sample containing the quality control issue.
createdBy	String	N/A	N/A	Always	N/A	The name of the **Analyst** or automatic process which created this **QcSegment Version**.
rejected	boolean	N/A	N/A	Always	N/A	Indicates whether this **QcSegment** has been rejected. A rejected **QcSegment** does not affect downstream processing and may not become a **ProcessingMask** (see below). Only the final **QcSegmentVersion** in a **QcSegment** may be rejected.
rationale	String	N/A	N/A	Always	N/A	A description of why this **QcSegmentVersion** was created, or why a previous **QcSegmentVersion** was modified or rejected.
stageId	**WorkflowDefinitionId**[0..1]	N/A	N/A	Optional. Populated when known.	N/A	This **QcSegmentVersion** was created in this automatic or interactive workflow **Stage**.
discoveredOn	**ChannelSegment**[*]	N/A	N/A	Optional.	Entity reference	A **ChannelSegment** collection analyzed or processed to find this quality control issue. *discoveredOn* is a collection to allow flexibility in cases where a quality control issue spans multiple **ChannelSegment** objects (e.g. a **QcSegmentVersion** with a FLAT **QcSegmentType** may occur over several **ChannelSegment** objects). These **ChannelSegment** objects must be produced by a raw **Channel**. Whether *discoveredOn* is populated for a new **QcSegmentVersion** is left to the component creating that **QcSegmentVersion**. The collection may be empty for quality control issues determined without waveform processing (e.g. because the issue corresponds to an **AcquiredChannelEnvironmentIssue**).

QcSegmentVersionId has the following attributes:

Table 3: **QcSegmentVersionId**

Attribute	Data Type	Units	Range	Populated	Description
parentQcSegmentId	UUID	N/A	N/A	Always	The *id* of the **QcSegment** entity this **QcSegmentVersion** is part of.
effectiveAt	Instant (ISO-8601 Date and Time)	Varies / handled by ISO-8601	N/A	Always	The date and time when this **QcSegmentVersion** was created. All of the **QcSegmentVersions** within a **QcSegment** form a time history. This field distinguishes when each version was created and also determines which is the current version.

QcSegmentCategory has the literals listed in the following table. Each **QcSegmentCategory** limits the available **QcSegmentTypes** the **QcSegmentVersion** may contain in its *type* attribute, which the table also lists.

Table 4: **QcSegmentCategory**

Literal	Description	Possible QcSegmentTypes
ANALYST_DEFINED	Quality control issue marked by an **Analyst**. An ANALYST_DEFINED **QcSegment** may be of any **QcSegmentType** or may not have a **QcSegmentType**.	May be populated with any **QcSegmentType**. May also be N/A (*type* attribute populated as an empty optional)
DATA_AUTHENTICATION	Quality control issue related to an authentication problem between an external station and GMS.	N/A (*type* attribute populated as an empty optional)
LONG_TERM	Quality control issue indicating a standing issue with the waveform samples produced by a **Channel**.	N/A (*type* attribute populated as an empty optional)
STATION_SOH	Quality control issue indicated in the metadata acquired along with the raw waveform samples produced by a **Channel**.	Must be populated with one of: CALIBRATION, NOISY, SENSOR_PROBLEM, STATION_PROBLEM, STATION_SECURITY, TIMING
UNPROCESSED	Waveform samples have not been processed for quality control issues.	N/A (*type* attribute populated as an empty optional)

WAVEFORM	Quality control issue within waveform samples. Found by automatic processing.	Must be populated with one of: AGGREGATE, FLAT, GAP, NOISY, SPIKE

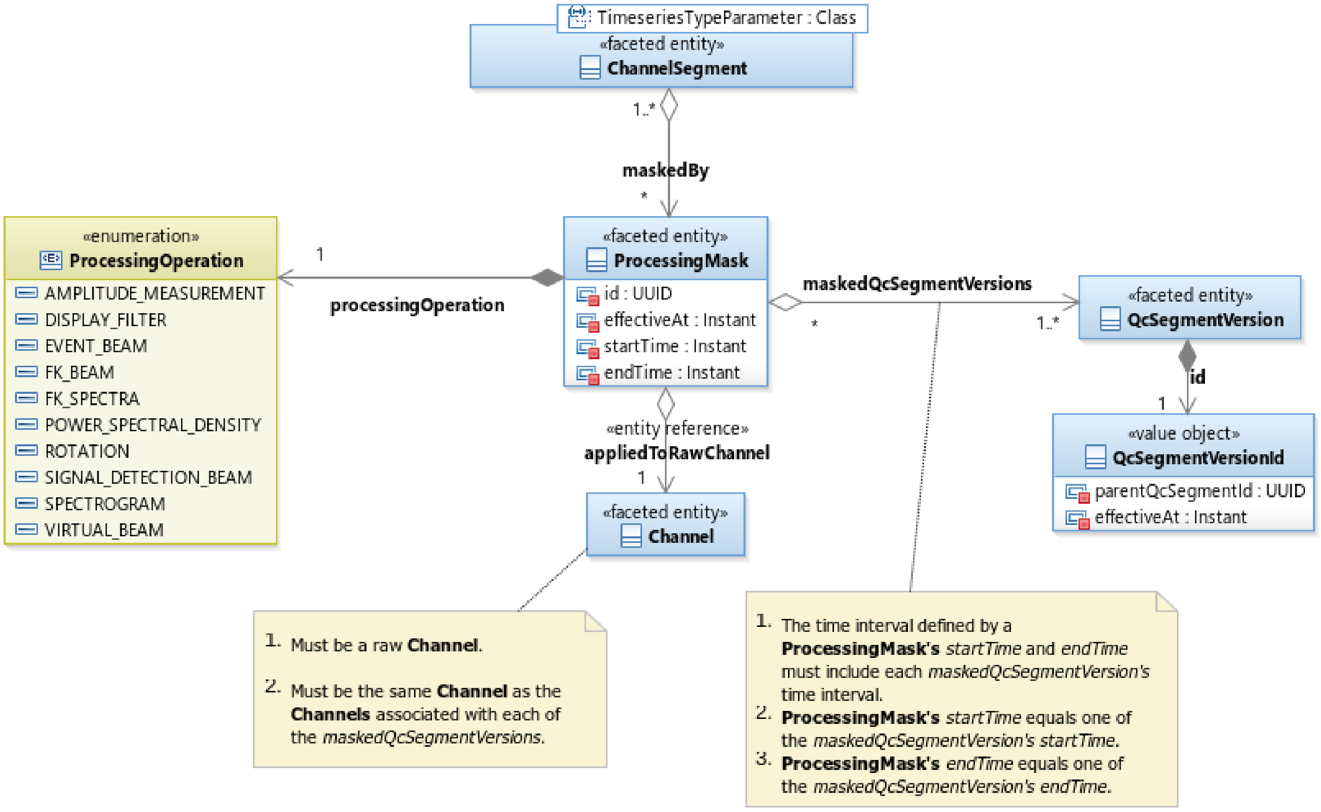
QcSegmentType has the follow literals:

Table 5: **QcSegmentType**

Literal	Description
AGGREGATE	Quality control issue which is the combination of several other quality control issues.
CALIBRATION	Waveform samples contain an instrument calibration signal.
FLAT	Waveform samples contain the same amplitude value.
GAP	Missing waveform samples.
NOISY	Interval of noisy waveform samples not attributed to normal background noise.
SENSOR_PROBLEM	Quality control issue limited to a single sensor (e.g. clipped, digitizer problem, etc.)
SPIKE	Short duration amplitude spike.
STATION_PROBLEM	Quality control issue affecting an entire station (e.g. power failure, value door opened, etc.)
STATION_SECURITY	Security related quality control issue affecting an entire station (e.g. authentication seal broken, etc.)
TIMING	Time values in the waveform samples may be incorrect.

Processing Mask

Figure 2: **ProcessingMask** class structure



1. Must be a raw **Channel**.
2. Must be the same **Channel** as the **Channels** associated with each of the *maskedQcSegmentVersions*.

1. The time interval defined by a **ProcessingMask's** *startTime* and *endTime* must include each *maskedQcSegmentVersion's* time interval.
2. **ProcessingMask's** *startTime* equals one of the *maskedQcSegmentVersion's startTime*.
3. **ProcessingMask's** *endTime* equals one of the *maskedQcSegmentVersion's endTime*.

A **ProcessingMask** marks an interval of raw waveform data that has been removed for a particular **ProcessingOperation**, e.g. a spiked segment of data on one channel of an array might be left out of beamforming. The **ProcessingOperation** literals typically represent the overall purpose of a chain of processing steps. For example, a **ProcessingMask** created to mask raw waveform samples for an amplitude measurement **ProcessingOperation** is created to mask the waveform samples at the beginning of a processing chain that may include additional steps such as filtering, beaming, and amplitude measurement. Each **ProcessingMask** is associated to the raw **Channel** producing the waveform data samples with the quality control issue. Only quality control issues represented by **QcSegment** objects may contribute to **ProcessingMask** objects. The *maskedQcSegmentVersions* in each **ProcessingMask** refer to the **QcSegmentVersion** objects leading to its creation. To avoid alternating short intervals of masked and unmasked waveform samples, a single, longer **ProcessingMask** may be associated with a sequence of shorter **QcSegmentVersion** objects. The **ProcessingMask** object's *startTime* and *endTime* mark the overall masked interval. This interval must include the time intervals in each of the *maskedQcSegmentVersions*. Further, the **ProcessingMask** object's *startTime* is equal to the *startTime* of one of the *maskedQcSegmentVersions* and the **ProcessingMask** object's *endTime* is equal to the *endTime* of one of the *maskedQcSegmentVersions.* Since each **ProcessingMask** is for a particular **ProcessingOperation**, it is possible for a **QcSegmentVersion** to be associated with several **ProcessingMask** objects with different **ProcessingOperation** literals (e.g. a **QcSegmentVersion** may indicate waveform samples removed from processing for both signal detection and amplitude measurement).

A derived **ChannelSegment** object may be aggregate any number of **ProcessingMask** objects. The **ProcessingMask** objects provide provenance about how the derived **ChannelSegment** was created. See Channel Segment COI Data Model for details.

ProcessingMask is faceted and has two possible instantiations:

1. An id-only instantiation has a populated *id* but the other attributes are not populated.
2. A full instantiation contains values for every attribute.

ProcessingMask has the following attributes:

⚠ **Note**

This table's *Populated* column refers to whether each attribute is optional or always populated in a populated **ProcessingMask** entity object. Other attribute populations are possible since **ProcessingMask** is a faceted class and can represent an entity reference or a populated entity version. See the Processing Mask COI data model overview description above for details.

Table 6: **ProcessingMask**

Attribute	Data Type	Units	Range	Populated	Default Facet Population	Description
id	UUID	N/A	N/A	Always	N/A	The **ProcessingMask** object's unique identifier.
effectiveAt	Instant (ISO-8601 Date and Time)	Varies / handled by ISO-8601	N/A	Always	N/A	The time when this **ProcessingMask** was created and began masking samples from the *appliedToRawChannel* **Channel** for calculations indicated by the *processingOperation*.
startTime	Instant (ISO-8601 Date and Time)	Varies / handled by ISO-8601	N/A	Always	N/A	The time (inclusive) of the first data sample masked by this **ProcessingMask**.
endTime	Instant (ISO-8601 Date and Time)	Varies / handled by ISO-8601	N/A	Always	N/A	The time (inclusive) of the last data sample masked by this **ProcessingMask**.
processingOperation	**ProcessingOperation**	N/A	N/A	Always	N/A	The **ProcessingOperation** which needs the masked samples removed from the raw waveform prior to processing.
appliedToRawChannel	**Channel**	N/A	N/A	Always	Entity reference (no other populations allowed)	The raw **Channel** producing the masked waveform samples. Must be the same as the **Channel** associated with each **QcSegmentVersion** in the *maskedQcSegmentVersions* collection.
maskedQcSegmentVersions	**QcSegmentVersion**[1..*]	N/A	N/A	Always	Reference	The **QcSegmentVersion** objects included in this **ProcessingMask**.

ProcessingOperation has the following literals:

Table 7: **ProcessingOperation**

Literal	Description
AMPLITUDE_MEASUREMENT	The masked waveform will contribute to a waveform used for amplitude measurement.
DISPLAY_FILTER	The masked waveform will be filtered for display to an Analyst.
EVENT_BEAM	The masked waveform will contribute to a beamed waveform that is steered to an **EventHypothesis** location.

FK_BEAM	The masked waveform will contribute to a beamed waveform that is steered to an azimuth and slowness selected from an **FkSpectra's** peak.
FK_SPECTRA	The masked waveform will contribute to an **FkSpectra**.
POWER_SPECTRAL_DENSITY	The masked waveform will contribute to a **Spectrogram**.
ROTATION	The masked waveform will contribute to a rotated waveform.
SIGNAL_DETECTION_BEAM	The masked waveform will contribute to a beamed waveform used for signal detection.
SPECTROGRAM	The masked waveform will contribute to a spectrogram.
VIRTUAL_BEAM	The masked waveform will contribute to a beamed waveform that is steered to a potential event location.

Notes

1. None.

Change History

1. PI-31 Updates
    a. 04/2025 - Added **ProcessingOperation** literal POWER_SPECTRAL_DENSITY.
2. PI-28 - Data Fabric Updates
    a. 07/2024
        i. Removed repository class and operation descriptions.
        ii. Version 1.0.0 released.
3. PI-22
    a. 12/2022 - Added the *store(QcSegment) : StoreQcSegmentResponse* operation to **QcSegmentRepository**.
4. PI-21
    a. 07/2022 - Initial guidance.

References

1. See the Station Definition COI Data Model
2. See the Faceted Data Class Design Pattern.

Open Issues

1. None.

(Attic) Waveform Quality Control Bridges

Table of Contents

List of Figures

List of Tables

Overview

In a GMS deployment using bridged **QcSegments** and **ProcessingMasks**, the **WaveformManager** provides request-response access to the **WaveformAccessor**, which is backed by the **QcSegmentRepositoryBridged** and **ProcessingMaskRepositoryBridged**.

QcSegmentRepositoryBridged provides the **QcSegmentRepository** interface using a legacy USNDC database. It implements the Data Bridge architecture. The GMS COI **QcSegment** data model is mismatched with the legacy data model. Important differences include the legacy data model does not track version history or overlapping masked samples. Additionally, the legacy database records equivalent to **QcSegment** COI objects, *QCMASKSEG* records, use a primary key which may change when then masked data changes. Together, these issues complicate conversions between **QcSegment** objects and legacy database records. To simplify the implementation, **QcSegmentRepositoryBridged** ensures the *QCMASKSEG* records stored in the legacy database mask only the samples also masked by the latest **QcSegmentVersion** object of each bridged **QcSegment** (read or written). Instead of attempting to exactly reconstruct **QcSegment** objects from *QCMASKSEG* records, **QcSegmentRepositoryBridged** uses the **BridgedQcSegmentCache** to store the bridged **QcSegment** and **QcSegmentVersion** objects. This approach allows **QcSegmentRepositoryBridged** to respond to queries with the expected **QcSegment** objects, including in repeated queries, write-read cycles, and queries from multiple clients. It also allows subsequent automatic processing in the legacy system to mask the same samples as GMS processing.

The legacy database does not contain tables with equivalent information to the **ProcessingMask** COI class. **ProcessingMaskRepositoryBridged** implements the **ProcessingMaskRepository** interface by using **QcSegmentRepositoryBridged** and **ProcessingMaskDefinition** objects to create **ProcessingMasks** from bridged **QcSegments**. The **ProcessingMaskRepositoryBridged** and GMS interactive analysis use the same **ProcessingMaskDefinition** objects, so this approach is sufficient to bridge **ProcessingMasks** equivalent to those used by the legacy system's automatic processing, as well as retrieving the **ProcessingMasks** created by the GMS user interface components for association to the derived **ChannelSegment** objects created by GMS Analysts.

⚠ **Implementation Note**

1. The **QcSegmentRepositoryBridged** implementation may store bridged **QcSegment** objects in GMS specific extension tables to the NDCP3 schema instead of in the **BridgedQcSegmentCache**. Using a **BridgedQcSegmentCache** avoids delivery, installation, and maintenance complexity associated with augmenting the legacy system databases with GMS specific extension tables. It comes at the cost of additional **QcSegmentRepositoryBridged** implementation complexity to query from and write to the cache (though this may be less complicated than using NDCP3 extension tables), manage the cache's contents and eviction policy, and maintain parity between the cache's contents and the legacy database's contents. However, if **QcSegmentRepositoryBridged** uses the **BridgedQcSegmentCache** then separate GMS deployments may have different views of bridged **QcSegment** objects. This is because the **BridgedQcSegmentCache** tracks **QcSegment** version history and is the component which allows **QcSegmentRepositoryBridged** to consistently provide **QcSegment** objects (e.g. for repeated queries; in write-read scenarios, especially when intermediate automatic processing occurs), despite differences between the legacy and GMS COI data models. The implementation team should have a design discussion with the Architecture team if they choose an implementation approach different from the description in this document.

2. Until GMS reads late arriving **QcSegments**, two Analysts may have inconsistent **QcSegment** objects if one Analyst loads a time interval before legacy system automatic waveform quality control processing executes to update the results in the time interval, and another Analyst loads the time interval after the automatic processing.
    a. For example, when one Analyst saves a **QcSegment**, another Analyst may receive an update message containing that **QcSegment** and find it overlaps a different **QcSegment** they already have loaded, but the first Analyst is unaware the second **QcSegment** exists because automatic processing created it after the first Analyst loaded the time interval.
    b. For example, when one Analyst saves a **QcSegment**, another Analyst may receive an update message containing that **QcSegment** and find it contains multiple **QcSegmentVersion** objects the first Analyst has not seen (e.g. one version that automatic processing created after the Analyst loaded the time interval and one version the other Analyst created).
    c. For example, one Analyst may save a **QcSegment** and find their new **QcSegmentVersion** is in conflict with a newer **QcSegmentVersion** bridged by a different Analyst and cached in **BridgedQcSegmentCache**.

Components

QC Segment Components

QC Segment Repository Bridged

QcSegmentRepositoryBridged is a legacy data bridge component responsible for providing access to bridged **QcSegment** and **QcSegmentVersion** objects.

QcSegmentRepositoryBridged implements the **QcSegmentRepository** interface's query operations by using **QcSegmentDatabaseConnector** to query records from a legacy USNDC database and using **QcSegmentConverter** to convert those legacy records into equivalent **QcSegment** COI objects. **QcSegmentRepositoryBridged** implements the **QcSegmentRepository** interface's storage operations by using **QcSegmentConverter** to convert **QcSegment** COI objects into equivalent legacy records and using **QcSegmentDatabaseConnector** to store those records into a legacy USNDC database.

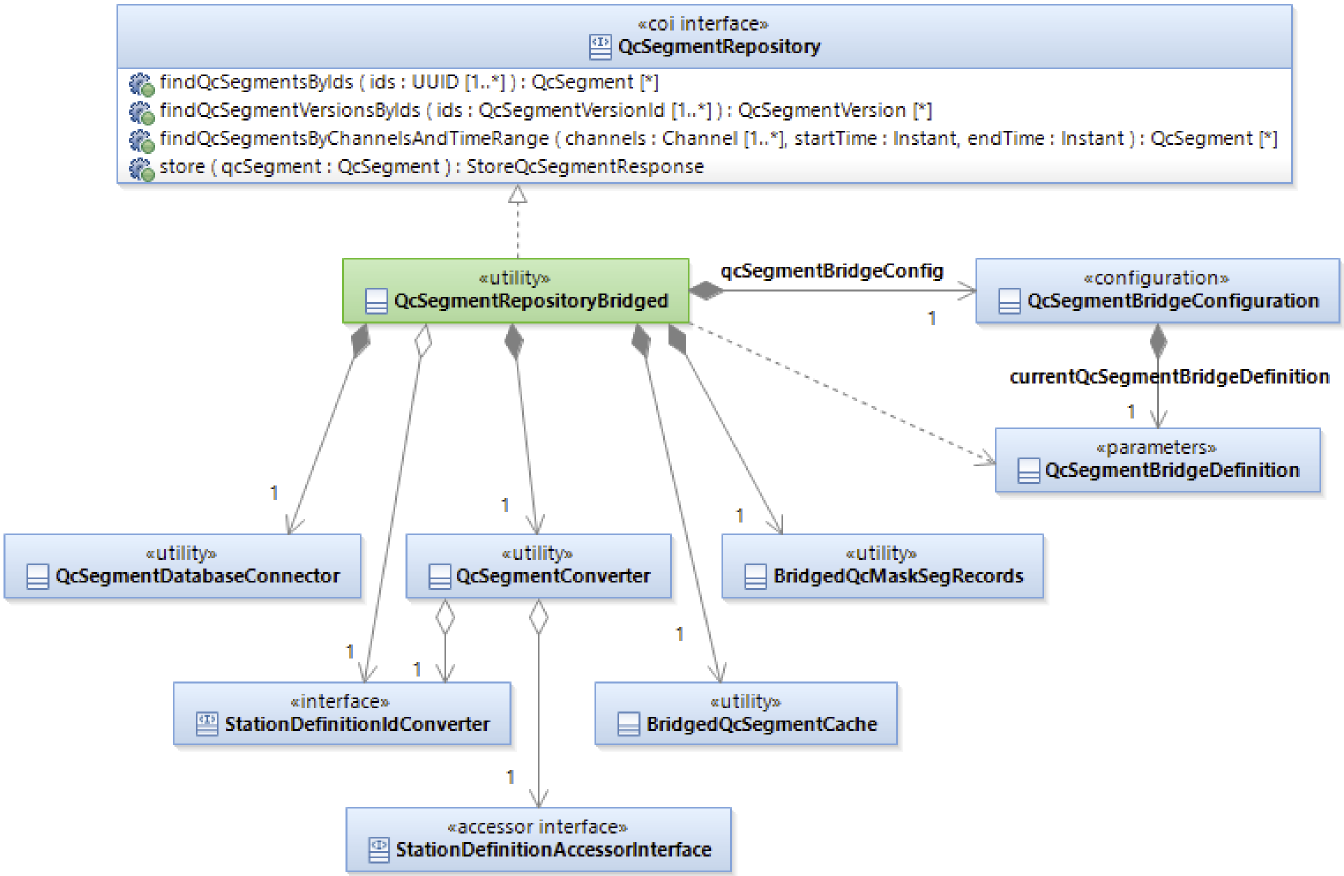

«coi interface»
QcSegmentRepository
findQcSegmentsByIds (ids : UUID [1..*]) : QcSegment [*]
findQcSegmentVersionsByIds (ids : QcSegmentVersionId [1..*]) : QcSegmentVersion [*]
findQcSegmentsByChannelsAndTimeRange (channels : Channel [1..*], startTime : Instant, endTime : Instant) : QcSegment [*]
store (qcSegment : QcSegment) : StoreQcSegmentResponse
«utility»
QcSegmentRepositoryBridged
qcSegmentBridgeConfig
1
«configuration»
QcSegmentBridgeConfiguration
currentQcSegmentBridgeDefinition
1
«parameters»
QcSegmentBridgeDefinition
1
1
1
«utility»
QcSegmentDatabaseConnector
«utility»
QcSegmentConverter
«utility»
BridgedQcMaskSegRecords
1
1
1
«interface»
StationDefinitionIdConverter
«utility»
BridgedQcSegmentCache
1
«accessor interface»
StationDefinitionAccessorInterface

*Figure 1: **QcSegmentRepositoryBridged** static structure*

QcSegmentRepositoryBridged implements **QcSegmentRepository** operations using the following components:

1. **QcSegmentBridgeConfiguration** - provides **QcSegmentRepositoryBridged** access to the **QcSegmentBridgeDefinition** and database connection parameters.
2. **QcSegmentDatabaseConnector** - implements queries against the legacy USNDC database and stores records to the USNDC database. The **QcSegmentDatabaseConnector** is only used by **QcSegmentRepositoryBridged**.
3. **QcSegmentConverter** - converts between legacy USNDC database format records and COI format **QcSegment** objects, in both directions. **QcSegmentConverter** is only used by **QcSegmentRepositoryBridged**.
4. **BridgedQcMaskSegRecords** - tracks which *QCMASKSEG* records **QcSegmentRepositoryBridged** has bridged (either by reading or writing).
5. **StationDefinitionIdConverter** - to find the legacy database (*sta, chan*) name pairs corresponding to COI **Channel** entity names.
6. **BridgedQcSegmentCache** - contains the **QcSegment** and **QcSegmentVersion** objects that **QcSegmentRepositoryBridged** previously bridged (either by reading or writing).

Startup and Configuration

On startup, **QcSegmentRepositoryBridged** constructs a **QcSegmentBridgeConfiguration** object using a **Configuration** utility (see Configuration Framework). **QcSegmentBridgeConfiguration** uses the **Configuration** utility to resolve **ConfigurationRuleSets** into instances of **QcSegmentBridgeDefinition** and locally caches the currently valid definition to avoid repeated configuration resolution. Any **QcSegmentRepositoryBridged** operation needing access to **QcSegmentBridgeDefinition** retrieves it from **QcSegmentBridgeConfiguration**. **QcSegmentRepositoryBridged** does not cache **QcSegmentBridgeDefinition** instances outside of **QcSegmentBridgeConfiguration**. Figure 2 shows the structure of the **EventBridgeConfiguration** and **EventBridgeDefinition** classes.

On startup, **QcSegmentRepositoryBridged** constructs a single **QcSegmentDatabaseConnector** which is connected to the GLOBAL account in a legacy format database. **QcSegmentRepositoryBridged** uses **QcSegmentBridgeConfiguration** to determine the database connection parameters.

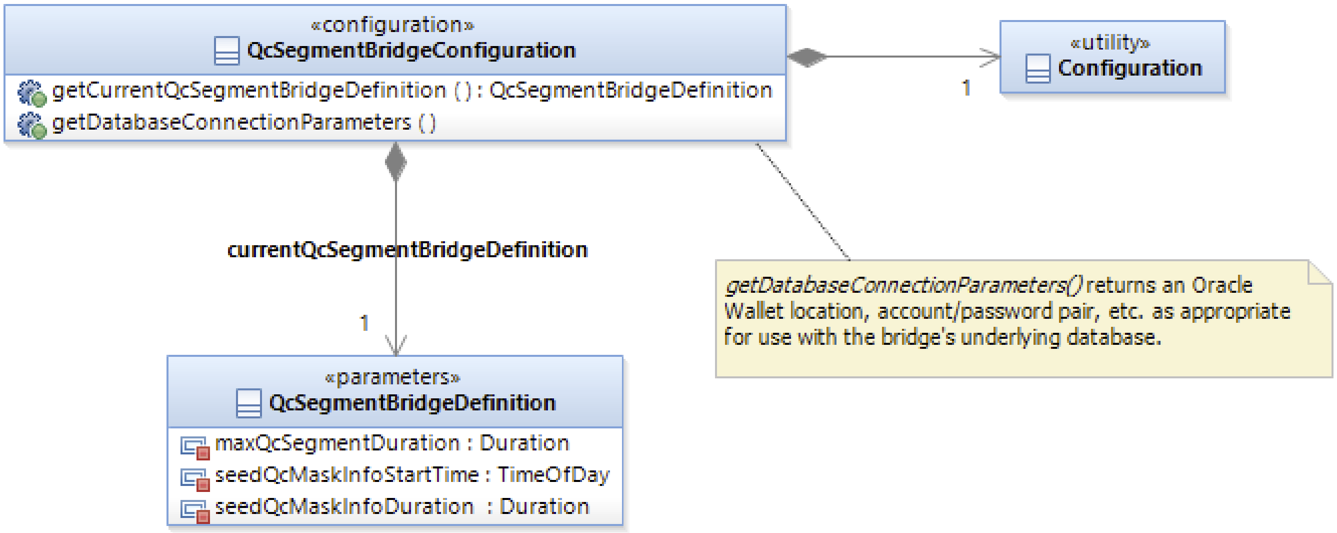

«configuration»
QcSegmentBridgeConfiguration
getCurrentQcSegmentBridgeDefinition ( ) : QcSegmentBridgeDefinition
getDatabaseConnectionParameters ( )
«utility»
Configuration
1
currentQcSegmentBridgeDefinition
1
getDatabaseConnectionParameters() returns an Oracle Wallet location, account/password pair, etc. as appropriate for use with the bridge's underlying database.
«parameters»
QcSegmentBridgeDefinition
maxQcSegmentDuration : Duration
seedQcMaskInfoStartTime : TimeOfDay
seedQcMaskInfoDuration : Duration

*Figure 2: **QcSegmentBridgeConfiguration** and **QcSegmentBridgeDefinition** static structure*

QcSegmentBridgeDefinition contains the following parameters:

1. *maxQcSegmentDuration : Duration* - the maximum duration of bridged **QcSegmentVersion** objects. **QcSegmentConverter** splits bridged *QCMASKSEG* records longer than this duration into multiple **QcSegment** objects.
2. *seedQcMaskInfoStartTime : TimeOfDay* - the time of day, without a date, **QcSegmentRepositoryBridged** uses to determine how to assign the *time* attribute when it generates seed *QCMASKINFO* records (see the *store(QcSegment)* operation).
3. *seedQcMaskInfoDuration: Duration* - the duration the **QcSegmentRepositoryBridged** combines with *seedQcMaskInfoStartTime* to assign the *endtime* attribute when it generates seed *QCMASKINFO* records (see the *store(QcSegment)* operation).

Bridging QC Segments

QcSegmentRepositoryBridged uses the following legacy USNDC database tables. The tables contain information **QcSegmentRepositoryBridged** needs to create **QcSegment** COI objects from legacy records, or to store **QcSegment** object information into legacy records:

1. *QCMASKINFO* (read and written): each record indicates the number of *QCMASKSEG* records that exist for a particular **Channel** and time range. Each record has a unique *qcmaskid*.
2. *QCMASKSEG* (read and written): each record represents a single QC mask segment and indicates the mask type as well as the range of masked samples. Each record has a unique (*qcmaskid, startsample*) pair. *qcmaskid* links the *QCMASKSEG* record back to a *QCMASKINFO* record.
3. *WFDISC* (read only): each record contains a **Channel's** actual waveform sample rate for a time interval. **QcSegmentRepositoryBridged** uses these sample rates to create new *QCMASKINFO* records.

QcSegmentConverter creates **QcSegment** COI objects from *QCMASKINFO* records, *QCMASKSEG* records, and existing **QcSegment** objects. **QcSegmentConverter** creates *QCMASKINFO* and *QCMASKSEG* records from **QcSegment** and **QcSegmentVersion** objects, existing *QCMASKINFO* and *QCMASKSEG* records, and existing *WFDISC* records.

Operation Implementations

QcSegmentRepositoryBridged implements the **QcSegmentRepository** operations as follows:

1. *findQcSegmentsByIds(ids : UUID[*]) : QcSegment[*]*
    a. Creates an empty output collection of **QcSegment** objects.
    b. For each **QcSegment** *id* provided in the query predicate:
        i. Uses the **BridgedQcSegmentCache** to load the **QcSegment** entity with the *id*.
        ii. Uses the **BridgedQcSegmentCache** to load the latest **QcSegmentVersion** of the **QcSegment** entity with the *id*.
        iii. Adds the **QcSegment** to the output collection of **QcSegment** objects.
    c. Returns the output collection of **QcSegment** objects.

    ⚠ **Note**

    This operation is not used in the architecture flows related to loading bridged **QcSegments**. This operation is indirectly called by the **WaveformFacetingUtility** to load **QcSegment** objects. It is also a general purpose query that may be used during research, testing, etc.

2. *findQcSegmentVersionsByIds(ids : QcSegmentVersionId[*]) : QcSegmentVersion[*]*
    a. Creates an empty output collection of **QcSegmentVersion** objects.
    b. For each **QcSegmentVersionId** provided in the query predicate:
        i. Uses the **BridgedQcSegmentCache** to load the **QcSegmentVersion** with the **QcSegmentVersionId**.
        ii. Adds the **QcSegmentVersion** to the output collection of **QcSegmentVersion** objects.
    c. Returns the output collection of **QcSegmentVersion** objects.

> **Note**
>
> **WaveformFacetingUtility** indirectly calls this operation to load **QcSegmentVersion** objects. It does this when the **Interactive AnalysisStateManager** loads **QcSegments** with populated instances of each **QcSegmentVersion** and when the **WaveformManager** creates **QcSegmentSavedEvent** objects. **WaveformFacetingUtility** may also call this operation when it is called by other components. This is is also a general purpose query that may be used during research, testing, etc.

3. *findQcSegmentsByChannelsAndTimeRange(channels : Channel[*], startTime : Instant, endTime : Instant) : QcSegment[*]*
    a. Creates an empty output collection of **QcSegment** objects.
    b. For each **Channel** provided in the query predicate:
        i. Uses **StationDefinitionIdConverter** to find the legacy database *(sta, chan)* pair corresponding to the **Channel** object's entity identifier.
        ii. Loads the legacy *QCMASKINFO* and *QCMASKSEG* records matching the query predicate:
            1. Uses the **QcSegmentDatabaseConnector** to query for the *QCMASKINFO* records matching the legacy *(sta, chan)* and which occur in the provided time range.
            2. Uses the **QcSegmentDatabaseConnector** to query for the *QCMASKSEG* records with *qcmaskid* values matching those in the *QCMASKINFO* records found in the previous step.
                a. Does not load any *QCMASKSEG* records already in the **BridgedQcMaskSegRecords**.
            3. Updates the **BridgedQcMaskSegRecords** to include the *QCMASKSEG* records.
        iii. Uses the **BridgedQcSegmentCache** to load the collection of **QcSegment** objects which have a latest **QcSegmentVersion** object matching the query predicate, along with each **QcSegment** object's latest **QcSegmentVersion**.
        iv. Uses the **QcSegmentConverter** to update the loaded collection of **QcSegment** objects to reflect the samples masked by the loaded collection of *QCMASKSEG* records (see Legacy to COI Conversion).
        v. Caches the updated collection of **QcSegment** objects in the **BridgedQcSegmentCache**.
        vi. Adds the updated collection of **QcSegment** objects to the output collection of **QcSegment** objects.
    c. Returns the output collection of **QcSegment** objects.
    d. (Future) Once GMS loads late arriving data, consider implementing this operation to only read from the **BridgedQcSegmentCache**. This avoids having to publish **QcSegmentChangedEvent** or **QcSegmentCreatedEvent** from this operation and allows all Analysts to have consistent views of the legacy database.

4. *store(QcSegment) : StoreQcSegmentResponse*
    a. Determines if the provided **QcSegment** is in conflict with the currently stored version of the **QcSegment** entity.
        i. Uses the **BridgedQcSegmentCache** to load the cached instance of the provided **QcSegment** entity.
        ii.
            1. There is no conflict if the **BridgedQcSegmentCache** does not contain the **QcSegment**.
    b. If there is a conflict:
        i. Uses the **BridgedQcSegmentCache** to load all the **QcSegmentVersion** objects in the **QcSegment** object's history (**QcSegmentRepositoryBridged** includes these objects in the **StoreQcSegmentResponse** it returns from this operation; see below for details).
    c. If there is no conflict:
        i. Extracts the latest **QcSegmentVersion** from the provided **QcSegment**.
        ii. Provides the COI **Channel** entity associated to the **QcSegment** to the **StationDefinitionIdConverter** to find the corresponding legacy database station and channel names (i.e., *sta* and *chan*).
        iii. Uses the **QcSegmentDatabaseConnector** to load the existing *QCMASKINFO* and *QCMASKSEG* records that it may need to update. The **QcSegmentDatabaseConnector**:
            1. Loads any existing *QCMASKINFO* records associated to the (*sta, chan)* with time intervals overlapping the **QcSegmentVersion** object's time interval.
            2. Loads any existing *QCMASKSEG* records associated to those *QCMASKINFO* records which have time intervals overlapping the **QcSegmentVersion** object's time interval.
        iv. If the **QcSegmentVersion** object's time interval extends beyond the combined time interval of the *QCMASKINFO* records, then **QcSegmentRepositoryBridged** needs to create one or more new *QCMASKINFO* records. To prepare for this, **QcSegmentRepositoryBridged**:
            1. Loads a seed *QCMASKINFO* record for the **QcSegmentConverter** to use to create new *QCMASKINFO* records:
                a. If the **QcSegmentDatabaseConnector** loaded one or more existing *QCMASKINFO* records in the previous step, then **QcSegmentRepositoryBridged** uses the loaded *QCMASKINFO* record with the latest time interval as the seed record.
                b. Otherwise, **QcSegmentRepositoryBridged** uses **QcSegmentDatabaseConnector** to load the *QCMASKINFO* record associated to the (*sta, chan)* pair with the latest time interval that is before the **QcSegmentVersion** object's time interval, and uses it as the seed record.
                c. When **QcSegmentDatabaseConnector** does not find a *QCMASKINFO* record in either of the previous steps, then **QcSegmentRepositoryBridged** creates a new seed *QCMASKINFO* record as follows (see COI to Legacy Conversion for attribute data types and descriptions):

***QCMASKINFO* attribute**	**How to assign the seed record's attribute values**
qcmaskid	Set to 0.
sta	i. Uses the **StationDefinitionIdConverter** to find the legacy database station and channel names (i.e., *sta* and *chan*) corresponding to the COI **Channel** entity associated to the **QcSegmentVersion**. ii. Assigns the *sta* attribute to that *sta* value.

chan	i. Uses the **StationDefinitionIdConverter** to find the legacy database station and channel names (i.e., *sta* and *chan*) corresponding to the COI **Channel** entity associated to the **QcSegmentVersion**. ii. Assigns the *chan* attribute to that *chan* value.
time	i. Creates an **Instant** object (i.e., a date and time value) by combining the current date with the current **QcSegmentBridgeDefinition** object's *seedQcMaskInfoStartTime*. ii. Assigns the *time* attribute to that **Instant**.
endtime	Assigns to *time* plus the current **QcSegmentBridgeDefinition** object's *seedQcMaskInfoDuration*.
samprate	Sets to 1.0 **Note** The **StationDefinitionIdConverter** uses *WFDISC* records to find the correct sample rate for each new *QCMASKINFO* record.
nseg	Sets to 0.
qcdefid	i. Legacy database constraints require each *QCMASKINFO* record to be associated to a *QCMASKDEF* record describing the waveform quality control processing parameters that automatic processing used while creating the *QCMASKINFO* record's associated *QCMASKSEG* records. This constraint requires associations from GMS generated seed *QCMASKINFO* records to *QCMASKDEF* records. ii. The processing parameters described in *QCMASKDEF* records do not apply to Analyst created or modified *QCMASKSEG* records, so the contents of the *QCMASKDEF* records associated to GMS created *QCMASKINFO* records are irrelevant. Since the attributes in *QCMASKDEF* records do not have N/A values, GMS creates a *QCMASKDEF* record with dummy values for every attribute (e.g., 0.0, -1, etc.). i. GMS uses the legacy database's *get_next_id()* function to generate the *qcdefid* identifier. iii. **QcSegmentRepositoryBridged** creates and stores a single dummy *QCMASKDEF* record. It associates every *QCMASKINFO* record it creates from a GMS generated seed *QCMASKINFO* record to this *QCMASKDEF* record. iv. To assign the *qcdefid* attribute value, **QcSegmentRepositoryBridged** either: i. Uses the *qcdefid* of the existing dummy *QCMASKDEF* record. This may require **QcSegmentRepositoryBridged** to use the **QcSegmentDatabaseConnector** to find the dummy record in the legacy database. ii. Creates the dummy *QCMASKDEF* record, uses **QcSegmentDatabaseConnector** to store the record, and uses that record's *qcdefid*.
auth	Sets to the string: "GMSDataBridge"
lddate	Set to the current time.

Table 1: Creating a seed QCMASKINFO record

2. Determines the actual sample rates for the **Channel** versions associated to the **QcSegmentVersion** object during the **QcSegmentVersion** object's time interval:
    - a. Uses **QcSegmentDatabaseConnector** to load the *WFDISC* records for the *(sta, chan)* which overlap the **QcSegmentVersion** object's time interval.
    - b. Extracts the *time, endtime,* and *samprate* attributes from each *WFDISC* record.
        - i. **QcSegmentRepositoryBridged** combines adjacent time intervals with the same *samprate*.

v. Uses the **BridgedQcSegmentCache** to load the collection of **QcSegment** objects which are associated to the same **Channel** entity and have a latest **QcSegmentVersion** object overlapping the provided **QcSegmentVersion** object, along with each **QcSegment** object's latest **QcSegmentVersion**.

vi. Uses **QcSegmentConverter** to create new or updated *QCMASKINFO* and *QCMASKSEG* records from the **QcSegmentVersion** object, the loaded *QCMASKINFO* and *QCMASKSEG* records, the seed *QCMASKINFO* object, the sample rates for the **QcSegmentVersion** object's associated **Channel**, and the loaded **QcSegment** objects.

vii. Uses **QcSegmentDatabaseConnector** to store the *QCMASKINFO* and *QCMASKSEG* records.

viii. Updates the **BridgedQcMaskSegRecords** to include the *QCMASKSEG* records.

ix. Caches the provided **QcSegment** object in the **BridgedQcSegmentCache**.

d. **QcSegmentRepositoryBridged** returns to its caller a **StoreQcSegmentResponse** object including:

- i. A **StoreStatus** literal indicating whether it successfully stored the **QcSegment**, did not store it due to a conflict, or did not store it due to some other error.
- ii. If it did not store the **QcSegment** due to a conflict: the stored instance of the **QcSegment**, including every **QcSegmentVersion** object in its history.

QC Segment Database Connector

QcSegmentDatabaseConnector implements the query operations that **QcSegmentRepositoryBridged** needs to realize the **QcSegmentRepository** semantics described above. **QcSegmentDatabaseConnector** implements the operations by directly querying the USNDC database. The specific **QcSegmentDatabaseConnector** operations are left as a development decision. Since the **QcSegmentRepositoryBridged** encapsulates the **QcSegmentDatabaseConnector**, implementations have flexibility in defining both the operations declared by the **QcSegmentDatabaseConnector** and which data classes the operations use (e.g. the data classes might correspond to legacy database records or they might be custom classes containing exactly the attributes that **QcSegmentConverter** needs to create a COI object).

QC Segment Converter

QcSegmentConverter is responsible for creating **QcSegment** COI objects using the legacy database records loaded by **QcSegmentDatabaseConnector**, and for using **QcSegment** COI objects to create legacy database records for **QcSegmentDatabaseConnector** to write to the legacy database. Mismatches in the COI and legacy data models prevent direct conversions between the two models, so the **QcSegmentConverter** does not attempt to create *QCMASKSEG* records directly corresponding to **QcSegmentVersion** objects (most importantly, the legacy data model does not track version history or overlapping masked samples). Instead, the **QcSegmentConverter** follows this principle: The stored *QCMASKSEG* records must mask only the samples masked by the latest **QcSegmentVersion** object of each stored **QcSegment**.

During conversions, the **QcSegmentConverter** uses components from several other subdomains:

1. **StationDefinitionIdConverter** to find the COI **Channel** entity names corresponding to legacy database (*sta, chan*) pairs, and vice-versa.
2. **StationDefinitionAccessor** to lookup COI **Channel** versions.

Legacy to COI Conversion

QcSegmentConverter uses legacy database records to create and update **QcSegment** objects. **QcSegmentConverter** has two possibilities for each *QCMASKSEG* record:

1. **QcSegmentRepositoryBridged** has not yet bridged *QCMASKSEG* records for the *QCMASKSEG* record's time interval into **QcSegment** objects, and has not yet written **QcSegment** objects into *QCMASKSEG* records for the *QCMASKSEG* record's time interval.
    - a. This occurs when the **BridgedQcSegmentCache** does not contain any **QcSegment** objects with latest **QcSegmentVersion** objects overlapping the *QCMASKSEG* record's time interval.
    - b. In this case, **QcSegmentConverter** creates one or more **QcSegment** objects for each *QCMASKSEG* record.
        - i. **QcSegmentConverter** splits a *QCMASKSEG* record into multiple **QcSegment** objects if the samples masked by a *QCMASKSEG* record correspond to a duration longer than a configured maximum duration (see **QcSegmentBridgeConfiguration**).
        - ii. **QcSegmentConverter** creates a single **QcSegmentVersion** for each **QcSegment** it creates.
        - iii. When **QcSegmentConverter** splits a *QCMASKSEG* record into multiple **QcSegment** objects, it creates **QcSegments** with **QcSegmentVersions** of equal durations.
    - c. For example, if the configured maximum duration is 2 hours and a *QCMASKSEG* record has a duration of 3 hours, then the **QcSegmentConverter** creates two **QcSegment** objects. Each **QcSegment** has a single **QcSegmentVersion** with a 1.5 hour duration.

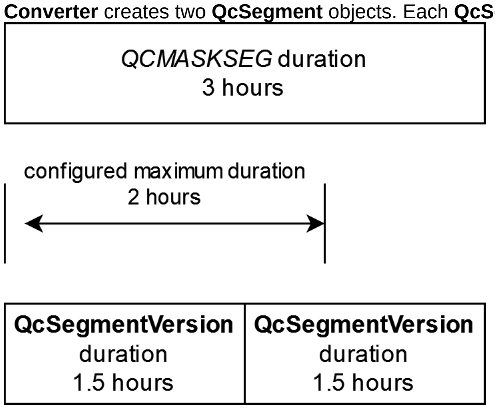


Figure 3: Example QCMASKSEG record bridged into multiple ***QcSegment*** *objects*

2. **QcSegmentRepositoryBridged** has previously bridged *QCMASKSEG* records for the *QCMASKSEG* record's time interval into **QcSegment** objects, or has written **QcSegment** objects into *QCMASKSEG* records for the *QCMASKSEG* record's time interval.
    - a. This occurs when the **BridgedQcSegmentCache** contains a **QcSegment** object with a latest **QcSegmentVersion** object overlapping the *QCMASKSEG* record's time interval.
    - b. In this case, the **QcSegmentConverter** updates the collection of **QcSegment** objects to reflect the samples masked by the collection of *QCMASKSEG* records:
        - i. Uses the **BridgedQcMaskSegRecords** to determine if it previously read or wrote each *QCMASKSEG* record. Discards any such *QCMASKSEG* records since their information is reflected in the loaded **QcSegments**.
        - ii. Finds differences between the samples masked in the collection of *QCMASKSEG* records and the samples masked by the latest **QcSegmentVersion** of each **QcSegment** in the loaded collection of **QcSegment** objects:
            1. Finds samples masked in the collection of *QCMASKSEG* records that are not masked in the collection of **QcSegment** objects.
            2. Finds samples not masked in the collection of *QCMASKSEG* records that are masked in the collection of **QcSegment** objects.
            3. Finds samples masked in the collection of *QCMASKSEG* records and in the collection of **QcSegment** objects, but with different types.
        - iii. Determines how to update the collection of loaded **QcSegment** objects to mask the same samples as the collection of *QCMASKSEG* records. This may include:
            1. Updating existing **QcSegment** objects with new **QcSegmentVersion** objects which mask either fewer samples or additional samples than the previous **QcSegmentVersion** objects.

2. Creating new **QcSegment** objects.

> **Note**
>
> a. Only update an existing **QcSegment** with a new **QcSegmentVersion** if its latest **QcSegmentVersion** was created by automatic processing (i.e., its *category* is not ANALYST_DEFINED) and the latest **QcSegmentVersion** is not rejected.
> b. Due to mismatches in the COI and legacy data models, it is not possible for the collection of records to represent all the information in the collection of **QcSegment** objects. There may be many possible ways to update the collection of **QcSegment** objects to mask the same samples as the collection of *QCMASKSEG* records. The implementation may decide how to update the collection of **QcSegment** objects (e.g., the implementation may prefer to create a new **QcSegment** object which overlaps an existing **QcSegment** object when a *QCMASKSEG* record has a different type than the existing **QcSegment** object).

iv. Updates the collection of **QcSegment** objects to mask the same samples as the collection of *QCMASKSEG* records.

The tables below describe how **QcSegmentConverter** uses legacy database record attributes to construct new **QcSegment** and **QcSegmentVersion** objects.

QcSegment attribute	How QcSegmentConverter assigns this attribute's value from a *QCMASKSEG* record
id	Randomly generates a new *id*.
channel	1. Finds the *QCMASKINFO* record associated to the *QCMASKSEG* record. 2. Uses the **StationDefinitionIdConverter** to find the COI **Channel** entity corresponding to the *QCMASKINFO* record's *sta* and *chan* attributes.
versionHistory	Follows the details described below to create a **QcSegmentVersion** object.

*Table 2: Creating **QcSegment** objects from legacy database records*

<table>
<tr><th>QcSegmentVersion attribute</th><th>How QcSegmentConverter assigns this attribute's value from a <i>QCMASKSEG</i> record</th></tr>
<tr><td><i>id</i></td><td><table>
<tr><th>QcSegmentVersionId attribute</th><th>How QcSegmentConverter assigns this attribute's value from a <i>QCMASKSEG</i> record</th></tr>
<tr><td><i>parentQcSegmentId</i></td><td>Assigns to the parent <b>QcSegment</b> entity object's <i>id</i>.</td></tr>
<tr><td><i>effectiveAt</i></td><td>Assigns to <i>QCMASKSEG lddate</i></td></tr>
</table></td></tr>
<tr><td><i>channels</i></td><td>1. Finds the <i>QCMASKINFO</i> record associated to the <i>QCMASKSEG</i> record.
2. Uses <b>StationDefinitionIdConverter</b> to find the COI <b>Channel</b> entity corresponding to the <i>QCMASKINFO</i> record's <i>sta</i> and <i>chan</i> attributes.
3. Uses <b>StationDefinitionAccessor</b> to find the <b>Channel</b> version(s) overlapping the time interval defined by this <b>QcSegmentVersion's</b> <i>startTime</i> and <i>endTime</i> attributes.</td></tr>
<tr><td><i>category</i></td><td>See the table below.</td></tr>
<tr><td><i>type</i></td><td>See the table below.</td></tr>
<tr><td><i>startTime</i></td><td>Assigns using the following formula: \(QCMASKINFO.time + (QCMASKSEG.startsample / QCMASKINFO.samprate) \)
Note: <i>startTime</i> must correspond to an actual sample time.</td></tr>
<tr><td><i>endTime</i></td><td>Assigns using the following formula: \(QCMASKINFO.time + (QCMASKSEG.endsample / QCMASKINFO.samprate) \)
Note: <i>endTime</i> must correspond to an actual sample time.</td></tr>
<tr><td><i>createdBy</i></td><td>Assigns to the <i>QCMASKSEG</i> record's <i>auth</i> attribute.</td></tr>
<tr><td><i>rejected</i></td><td>Assigns as follows:
1. true - if <i>QCMASKSEG masktype</i> is 300 (Analyst deleted)
2. false - otherwise</td></tr>
</table>

rationale	Assigns to the string: "N/A (bridged)"
stageId	Empty optional.
discoveredOn	Empty optional.

*Table 3: Creating **QcSegmentVersion** objects from legacy database records*

The following table shows how **QcSegmentConverter** assigns the **QcSegmentVersion** *category* and *type* attributes using the *QCMASKSEG masktype* attribute:

***QCMASKSEG masktype* value**	***QCMASKSEG masktype* meaning**	**GMS QcSegmentCategory Literal**	**GMS QcSegmentType Literal**	**Notes**
0	Unprocessed by AutoQC	UNPROCESSED	Empty optional	
10	Missing	WAVEFORM	GAP	
20	Flat	WAVEFORM	FLAT	
30	Noisy	WAVEFORM	NOISY	
40	Bad single point	WAVEFORM	SPIKE	Lossy when saving **QcSegments** created by automatic processing.
50	Multiple data array spike	WAVEFORM	SPIKE	Lossy when saving **QcSegments** created by automatic processing.
60	Single spike data	WAVEFORM	SPIKE	Lossy when saving **QcSegments** created by automatic processing.
70	Spike identified by TOS	WAVEFORM	SPIKE	Lossy when saving **QcSegments** created by automatic processing.
100	Aggregate	WAVEFORM	AGGREGATE	
200	Channel	LONG_TERM	Empty optional	
300	Analyst deleted	Empty optional	Empty optional	Bridged into a **QcSegmentVersion** with *rejected* attribute equal to "true"
400	Analyst	ANALYST_DEFINED	Empty optional	
500	Calibration	STATION_SOH	CALIBRATION	

Table 4: Mapping USNDC *QCMASKSEG masktype* to GMS **QcSegmentCategory** and **QcSegmentType** literals

COI to Legacy Conversion

QcSegmentRepositoryBridged must populate the legacy database *QCMASKINFO* and *QCMASKSEG* records in the same way as the legacy system. To accomplish this, **QcSegmentConverter** must create *QCMASKINFO* and *QCMASKSEG* records according to the following expected attribute populations and invariants:

1. The stored *QCMASKSEG* records must mask only the samples masked by the latest **QcSegmentVersion** objects of each stored **QcSegment**.
   a. The **QcSegmentConverter** does not attempt to create *QCMASKSEG* records directly corresponding to **QcSegmentVersion** objects. Mismatches in the COI and legacy datamodels prevent this.
2. Two *QCMASKSEG* records must not mask the same samples.
   a. Two *QCMASKSEG* records for the same (*sta, chan*) must be separated by at least one sample (i.e., one *QCMASKSEG* record's *startsample* cannot equal another *QCMASKSEG* record's *endsample*).
   b. This rule applies whether the *QCMASKSEG* records have the same *masktype* values or have different *masktype* values.
3. Each *QCMASKSEG* record is associated to a *QCMASKINFO* record.
   a. A *QCMASKINFO* record's *time* and *endtime* attributes define a time interval. When combined with a *QCMASKINFO* record's *time* and *samprate* attributes, a *QCMASKSEG* record's *startsample* and *endsample* attributes also define a time interval. A *QCMASKSEG* record's time interval must occur within time interval defined by its associated *QCMASKINFO* record.
   b. When the **QcSegmentConverter** converts a **QcSegmentVersion** object that does not fall entirely within an existing *QCMASKINFO* record's time interval, then it also creates one or more new *QCMASKINFO* objects spanning the **QcSegmentVersion** object's time interval.
   c. If **QcSegmentConverter** needs to create a new *QCMASKINFO* record to associate with a *QCMASKSEG* record, it:
      i. Uses a seed *QCMASKINFO* object to populate the *QCMASKINFO* record as described in the table below.
      ii. Creates "Unprocessed by AutoQC" *QCMASKSEG* records (*masktype* 0) for the portions of the *QCMASKINFO* records' time intervals not masked by the *QCMASKSEG* records it created from the **QcSegmentVersion** object.
4. "Analyst deleted" *QCMASKSEG* records *(masktype* 300) represent previously masked samples that the Analyst unmasked. A GMS Analyst unmasks samples by rejecting a **QcSegment** or by shortening the masked time range of an existing **QcSegment**. In both cases, the **QcSegmentConverter** creates "Analyst deleted" *QCMASKSEG* records.
5. A *QCMASKINFO* record's *nseg* attribute must contain the number of *QCMASKSEG* record's in its time interval (i.e., those *QCMASKSEG* records with the same *qcmaskid*).
6. A *QCMASKINFO* record's *samprate* attribute is valid for the record's entire time interval.

QcSegmentConverter uses the following approach to create the *QCMASKSEG* records corresponding to a **QcSegmentVersion**. This approach uses the **QcSegment** object being stored, other existing **QcSegment** objects, and the existing *QCMASKSEG* records:

1. Determines which samples to unmask:
    a. If the latest **QcSegmentVersion** of the provided **QcSegment** object is rejected, then **QcSegmentConverter** creates Analyst Deleted (*masktype 300*) QCMASKSEG record(s) only for the samples that are not included in the latest **QcSegmentVersion** objects of other stored **QcSegment** objects.
    b. If the latest **QcSegmentVersion** of the provided **QcSegment** object masks overlapping but fewer samples than the **QcSegment** object's previous **QcSegmentVersion** (i.e., its *startTime* is later, its *endTime* is earlier, or both), then the **QcSegmentConverter** creates Analyst Deleted (*masktype 300*) *QCMASKSEG* record(s) only for the samples that are not included in the latest **QcSegmentVersion** objects of other stored **QcSegment** objects.
2. Determines which samples to mask:
    a. If the latest **QcSegmentVersion** of the provided **QcSegment** object is not rejected, then the **QcSegmentRepositoryBridged** creates Analyst Defined (*masktype 400*) *QCMASKSEG* record(s) for all the samples in the **QcSegmentVersion** object.

The tables below describe how **QcSegmentConverter** assigns values to *QCMASKINFO* and *QCMASKSEG* records using a **QcSegment** and its latest **QcSegmentVersion** object.

***QCMASKINFO* attribute**	**Storage type**	**Description**	**How QcSegmentConverter assigns this attribute's value when updating an existing record**	**How QcSegmentConverter assigns this attribute's value when creating a new record based on a seed *QCMASKINFO* record**
qcmaskid	number(9)	The record's identifier	Leaves unchanged.	Uses the legacy database's *get_next_id()* function to generate a new identifier.
sta	varchar2(6)	The station associated with this *QCMASKINFO* and its associated *QCMASKSEG* records.	Leaves unchanged.	Assigns to the same value as the seed record.
chan	varchar2(8)	The channel associated with this *QCMASKINFO* and its associated *QCMASKSEG* records.	Leaves unchanged.	Assigns to the same value as the seed record.
time	float(53)	The start time of the QC mask processing time interval represented by this record.	Leaves unchanged.	1. Assumes nearby *QCMASKINFO* records have the same time interval duration as the seed record. 2. Uses the seed record's *endtime* and duration to find this record's start *time*.
endtime	float(53)	The end time of the QC mask processing time interval represented by this record.	Leaves unchanged.	1. Assumes nearby *QCMASKINFO* records have the same time interval duration as the seed record. 2. Uses the seed record's time interval duration and this record's start *time* to determine this record's *endtime*.
samprate	float(24)	The sample rate for the sta/chan represented by this record.	Leaves unchanged.	Assigns to the *samprate* attribute from the *WFDISC* record with this record's (*sta, chan)* and which includes this record's entire time interval.
nseg	number(8)	The number of *QCMASKSEG* records associated to this record.	Updates to the number of *QCMASKSEG* records that will be associated to this *QCMASKINFO* record in the legacy database after the storage request completes.	Assigns to the number of *QCMASKSEG* records that will be associated to this *QCMASKINFO* record in the legacy database after the storage request completes.
qcdefid	number(9)	Identifier of the *QCMASKDEF* record describing the QC masking processing parameters used by automatic processing algorithms to create the *QCMASKSEG* records associated to this record.	Leaves unchanged.	Assigns to the same value as the seed record.
auth	varchar2 (15)	The author of this record.	Assigns to the string: "GMSDataBridge"	Assigns to the string: "GMSDataBridge"
lddate	date	The date and time this record was inserted into the database (load date).	Assigns to the date and time when it updates this record.	Assigns to the date and time when it creates this record.

Table 5: Creating legacy QCMASKINFO records from GMS COI objects

***QCMASKSEG* attribute**	**Storage type**	**Description**	**How QcSegmentConverter assigns this attribute's value from a QcSegmentVersion object**

qcmaskid	number(9)	An identifier linking this record to a *QCMASKINFO* record.	1. Finds the *QCMASKINFO* record with (*sta, chan*) matching the **QcSegmentVersion** object's associated **Channel** entity and time interval overlapping this *QCMASKSEG* objec. a. The *QCMASKINFO* record may be an existing record or a new record created by the **QcSegmentConverter**. 2. Assigns to that *QCMASKINFO* record's *qcmaskid* attribute.
startsample	number(8)	The start sample of this QC mask segment.	Assigns using the following formula, rounding to the nearest integer: \(round([startTime - QCMASKINFO.time] * QCMASKINFO.samprate) \) Note: "startTime*"* could be a **QcSegmentVersion** object's *startTime*, but it may be a different time if the **QcSegmentConverter** splits the **QcSegmentVersion** across multiple *QCMASKSEG* records or combines the **QcSegmentVersion** with other stored **QcSegmentVersion** objects to determine the masked (or unmasked) samples.
endsample	number(8)	The end sample of this QC mask segment.	Assigns using the following formula, rounding to the nearest integer: \(round([endTime - QCMASKINFO.time] * QCMASKINFO.samprate) \) Note: "endTime" could be a **QcSegmentVersion** object's *endtime*, but it may be a different time if the **QcSegmentConverter** splits the **QcSegmentVersion** across multiple *QCMASKSEG* records or combines the **QcSegmentVersion** with other stored **QcSegmentVersion** objects to determine the masked (or unmasked) samples.
masktype	number(5)	The type of this QC mask segment (e.g., gap, spike, noise, etc.).	See the conversion table.
auth	varchar2 (15)	The author of this record.	1. When the **QcSegmentConverter** creates the *QCMASKSEG* record directly from a **QcSegmentVersion** object: assigns to the value of the **QcSegmentVersion** object's *createdBy* attribute. 2. When the **QcSegmentConverter** creates the *QCMASKSEG* record by combining multiple **QcSegmentVersion** objects: a. If all the **QcSegmentVersion** object's have have the same *createdBy* attribute: assigns to the value of the **QcSegmentVersion** objects' *createdBy* attribute. b. Otherwise: assigns to the string: "GMSDataBridge" 3. When the **QcSegmentConverter** modifies or creates a *QCMASKSEG* to satisfy the legacy database invariants and expected record population described above: assigns to the string: "GMSDataBridge".
lddate	date	The date and time this record was inserted into the database (load date).	Assigns to the date and time when **QcSegmentConverter** creates or updates the record.

Table 6: Creating legacy QCMASKSEG records from GMS COI objects

Bridged QCMASKSEG Records

BridgedQcMaskSegRecords is a legacy data bridge component responsible for tracking the *QCMASKSEG* records that **QcSegmentRepositoryBridged** has previously read or written.

The **QcSegments** stored in the **BridgedQcSegmentCache** already include these *QCMASKSEG* records, so **QcSegmentRepositoryBridged** does not need to convert these records into **QcSegment** objects. **QcSegmentRepositoryBridged** uses **BridgedQcMaskSegRecords** as an optimization to avoid reading and converting the same *QCMASKSEG* records multiple times.

Implementation Note

1. **QcSegmentRepositoryBridged** can operate correctly without **BridgedQcMaskSegRecords**.
2. **BridgedQcMaskSegRecords** details are left as an implementation decision. Examples of how **BridgedQcMaskSegRecords** may represent bridged *QCMASKSEG* records include: representing collections of *QCMASKSEG* records using a combination of **Channel** name, time interval, and the time when **QcSegmentRepositoryBridged** read or wrote the corresponding records; using *QCMASKSEG qcmaskid, startsample*, and *lddate*; using some other approach.

Bridged QC Segment Cache

BridgedQcSegmentCache is a legacy data bridge component responsible for providing access to previously bridged **QcSegment** and **QcSegmentVersion** objects (i.e. objects previously read or written by **QcSegmentRepositoryBridged**).

QcSegmentRepositoryBridged uses **BridgedQcSegmentCache** to avoid issues related to mismatches between the legacy data model and the COI data model (see the Overview for details). **BridgedQcSegmentCache** expires a cache entry when it has not been read or written for longer than the operational time period.

BridgedQcSegmentCache implements the query operations the **QcSegmentRepositoryBridged** needs to implement the **QcSegmentRepository** operations and the **QcSegmentConverter** needs to convert between the legacy and COI data models, which includes operations to:

1. Find the **QcSegment** entity with a provided *id*.
2. Find the **QcSegmentVersion** object with a provided **QcSegmentVersionId**.

3. Find the **QcSegmentVersion** objects of a provided **QcSegment** (particularly the latest **QcSegmentVersion**).
4. Find **QcSegmentVersion** objects matching a query predicate of **Channel** and time range.

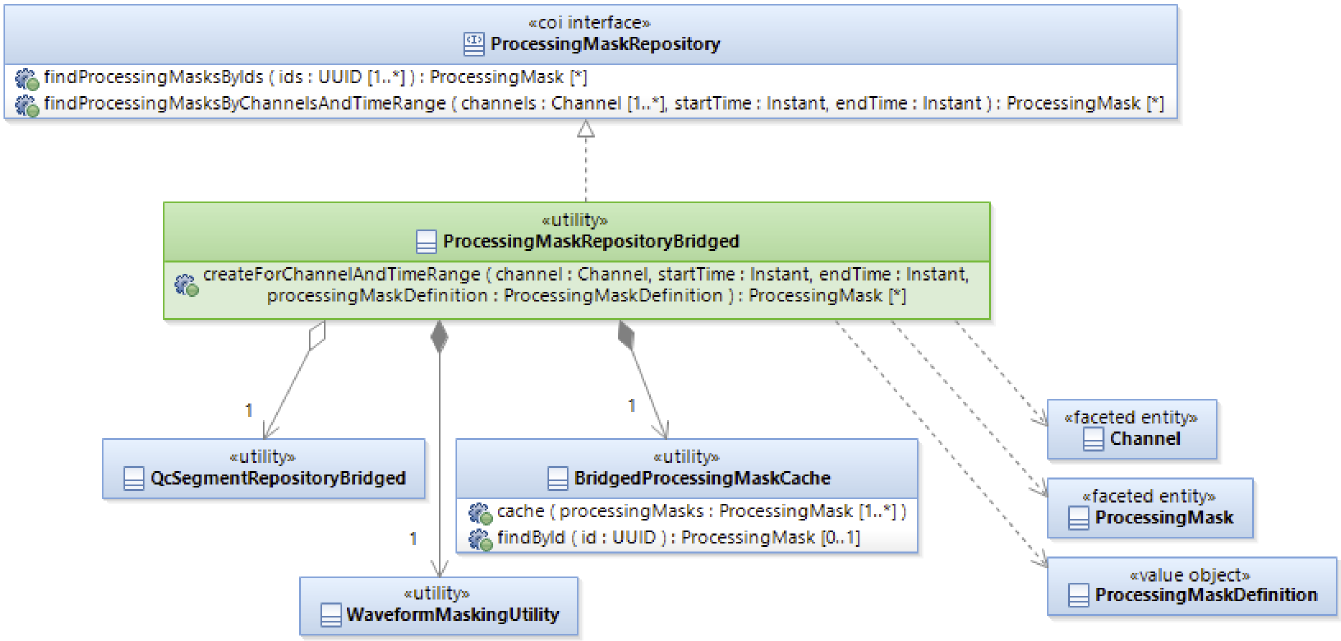
> ⚠ **Implementation Note**
>
> **BridgedQcSegmentCache** details are left as an implementation decision. The implementation may use Apache Ignite scan queries, Apache Ignite index queries (if no longer experimental), design cache key and value objects with data structures supporting custom implementations of these queries, etc.

Processing Mask Components

Processing Mask Repository Bridged

ProcessingMaskRepositoryBridged is a legacy data bridge component responsible for providing access to bridged **ProcessingMask** objects.

ProcessingMaskRepositoryBridged implements the **ProcessingMaskRepository** interface by querying records from a legacy USNDC database and converting those legacy records into the equivalent **ProcessingMask** COI objects. The legacy USNDC database does not contain records with information equivalent to the COI **ProcessingMask** class. However, **ProcessingMaskRepositoryBridged** can create bridged **ProcessingMasks** using bridged **QcSegmentVersions** and a **ProcessingMaskDefinition**.



«coi interface»
ProcessingMaskRepository
findProcessingMasksByIds (ids : UUID [1..*]) : ProcessingMask [*]
findProcessingMasksByChannelsAndTimeRange (channels : Channel [1..*], startTime : Instant, endTime : Instant) : ProcessingMask [*]
«utility»
ProcessingMaskRepositoryBridged
createForChannelAndTimeRange (channel : Channel, startTime : Instant, endTime : Instant, processingMaskDefinition : ProcessingMaskDefinition) : ProcessingMask [*]
1
1
1
«utility»
QcSegmentRepositoryBridged
«utility»
BridgedProcessingMaskCache
cache (processingMasks : ProcessingMask [1..*])
findById (id : UUID) : ProcessingMask [0..1]
«utility»
WaveformMaskingUtility
«faceted entity»
Channel
«faceted entity»
ProcessingMask
«value object»
ProcessingMaskDefinition

*Figure 4: **ProcessingMaskRepositoryBridged** static structure*

ProcessingMaskRepositoryBridged implements the **ProcessingMaskRepository** interface operations using the following components:

1. **QcSegmentRepositoryBridged** - **ProcessingMaskRepositoryBridged** uses **QcSegmentRepositoryBridged** to load **QcSegments** so it can combine them with **ProcessingMaskDefinitions** to create **ProcessingMasks**.

   > ⚠ **Implementation Note**
   >
   > Rather than using **QcSegmentRepositoryBridged** to load **QcSegments**, **ProcessingMaskRepositoryBridged** could instead call **WaveformAccessor** to load the **QcSegments**. Some consequences of this approach are:
   >
   > a. It creates a circular dependency since **WaveformAccessor** includes both **ProcessingMaskRepositoryBridged** and **QcSegmentRepositoryBridged**.
   > b. There may be a performance benefit if **WaveformAccessor** includes entity caches for **QcSegment** and **QcSegmentVersion** objects rather than always delegating to its aggregated **QcSegmentRepository** implementation.

2. **WaveformMaskingUtility** - **ProcessingMaskRepositoryBridged** uses **WaveformMaskingUtility** to create **ProcessingMasks** from **ProcessingMaskDefinitions** and bridged **QcSegments**.
3. **BridgeProcessingMaskCache** - **ProcessingMaskRepositoryBridged** caches bridged **ProcessingMasks** in **BridgedProcessingMaskCache** and then uses the cache to implement its *findProcessingMasksByIds(...)* operation.

Note that **ProcessingMaskRepositoryBridged** does not include the standard Software Bridge components (i.e. database connector, converter, and id utility) because the legacy USNDC database does not include records with information equivalent to **ProcessingMask**.

Operation Implementations

1. *findProcessingMasksByIds(ids : UUID[*]) : ProcessingMask[*]* - The legacy database does not include records with information equivalent to **ProcessingMask**, so the **ProcessingMaskRepositoryBridged** implements this operation using **BridgedProcessingMaskCache**. To implement this operation's behavior, **ProcessingMaskRepositoryBridged**:
    - a. Uses **BridgedProcessingMaskCache** to find the collection of **ProcessingMask** objects with the identifiers provided in the query predicate.
    - b. Returns the **ProcessingMask** collection.

2. *findProcessingMasksByChannelsAndTimeRange(channels : Channel[*], startTime : Instant, endTime : Instant) : ProcessingMask[*]*

    > **Implementation Note**
    >
    > The legacy database does not include records with information equivalent to **ProcessingMask**, making it difficult to implement this operation. It is not required for the QC Mask Bridging Capability and likely does not need to be implemented. If necessary, some implementation options include:
    >
    > - a. (Preferred) **ProcessingMaskRepositoryBridged** implements this operation by resolving **ProcessingMaskDefinitions** from configuration (the **ProcessingMaskDefinitions** vary by **Channel**; this implementation uses the same configuration as **StationDefinitionRepositoryBridged** and **SignalEnhancementConfiguration**) and then calling the *createForChannelAndTimeRange(...)* operation to create the **ProcessingMasks**. If this option is selected, **ProcessingMaskRepositoryBridged** implements the following behavior:
    >   - i. Resolves **ProcessingMaskDefinitions** for each provided **Channel**. There may be a variety of **ProcessingMaskDefinitions** describing the **ProcessingMask** objects that **ProcessingMaskRepositoryBridged** needs to create for a variety of **ProcessingOperations**.
    >   - ii. Calls *createForChannelAndTimeRange(...)* to create **ProcessingMasks** for each of the provided **Channels** and each resolved **ProcessingMaskDefinition**.
    >   - iii. Returns the **ProcessingMasks** collection.
    > - b. **ProcessingMaskRepositoryBridged** implements this operation using **BridgedProcessingMaskCache**. However, this implementation likely won't meet expectations for users who intend to search for **ProcessingMasks** within a datastore, rather than querying for previously loaded **ProcessingMasks**.

3. *createForChannelAndTimeRange(channel : Channel, startTime : Instant, endTime : Instant, processingMaskDefinition : ProcessingMaskDefinition) : ProcessingMask[*]* - this operation creates bridged **ProcessingMask** objects masking the provided raw **Channel** within the provided time range (*startTime* and *endTime* both inclusive). The legacy database does not include records with information equivalent to **ProcessingMask**, so **ProcessingMaskRepositoryBridged** creates **ProcessingMask** objects using bridged **QcSegments** and the provided **ProcessingMaskDefinitio** . This operation returns a collection of bridged **ProcessingMasks**. This operation returns an error response if it is provided a derived **Channel** in its parameters. To implement this behavior, **ProcessingMaskRepositoryBridged**:

    - a. Calls the **QcSegmentRepositoryBridged** *findQcSegmentsByChannelsAndTimeRange(...)* operation to load all the **QcSegments**, populated with their current **QcSegmentVersions**, for the provided raw **Channel** and time range.
    - b. Extracts the current **QcSegmentVersion** from each of the **QcSegment** objects.
    - c. Calls the **WaveformMaskingUtility's** *createProcessingMasksFromQcSegmentVersions(QcSegmentVersions[1..*], ProcessingMaskDefinition) : ProcessingMask[*]* operation to obtain the appropriate **ProcessingMasks** for the bridged **QcSegmentVersions** and the provided **ProcessingMaskDefinition**.
    - d. Reassigns each **ProcessingMask** object's *effectiveAt* attribute to the latest *effectiveAt* time selected from the **ProcessingMask** object's *maskedQcSegmentVersions* collection of **QcSegmentVersion** objects.
    - e. Caches the **ProcessingMasks** in the **BridgedProcessingMaskCache**.
    - f. Returns the bridged **ProcessingMask** collection.

Bridged Processing Mask Cache

BridgedProcessingMaskCache is a utility component responsible for providing access to previously bridged **ProcessingMask** objects.

BridgedProcessingMaskCache operations have the following semantics:

1. *cache(ProcessingMask[1..*])* - adds each of the provided **ProcessingMask** objects to the cache. If the cache already contains a **ProcessingMask** with the same *id* as one of the provided **ProcessingMask** objects, then replaces the **ProcessingMask** already in the cache with the provided **ProcessingMask**.

2. *findById(UUID) : ProcessingMask[0..1]* - finds within the cache and then returns the **ProcessingMask** object with the provided *id*. Returns an empty optional if the cache does not contain a **ProcessingMask** with the provided *id*.

Processing Mask Database Connector

ProcessingMaskRepositoryBridged does not include a **ProcessingMaskDatabaseConnector**.

Processing Mask Converter

ProcessingMaskRepositoryBridged does not include a **ProcessingMaskConverter**.

Processing Mask Id Utility

ProcessingMaskRepositoryBridged does not include a **ProcessingMaskIdUtility**.

Notes

1. The QC Mask Bridging Capability does not require GMS to load "late arriving" QcSegments.

Change History

1. PI22
    a. 12/2022 - Renamed **ProcessingMaskCache** to **BridgedProcessingMaskCache** to avoid a name conflict with the cache used by **WaveformAccessor**.
    b. 12/2022 - Reworked to include **QcSegment** saving.
2. PI20
    a. 07/2022 - Initial Guidance

References

1. See Software Bridge for a description of the OSD data bridge implementation pattern.

Open Issues

1. None.

Data Fabric Bridge Conversion Parameters

Table of Contents

List of Figures

List of Tables

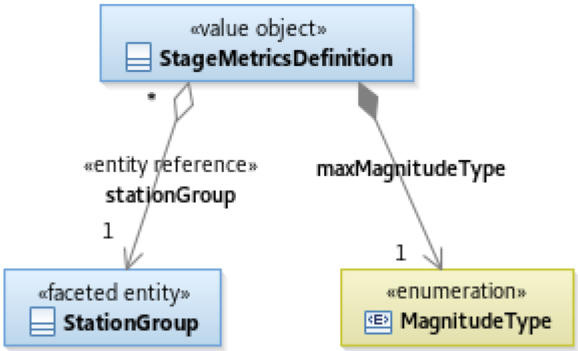
Overview

Some of the Data Fabric conversions from the USNDC database's physical data model to the COI data model require conversion parameters. The **DataFabric** loads these parameters independently of the GMS system. This page describes the conversion parameters classes.

COI Conversion Class Descriptions

Stage Metrics Definition

Figure 1: **StageMetricsDefinition** structure



StageMetricsDefinition includes parameters the Data Fabric needs to construct COI **StageMetrics** objects using the existing USNDC database contents. The Data Fabric must support the possibility of a different **StageMetricsDefinition** object for each **Stage**.

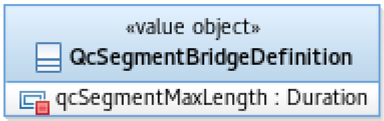
StageMetricsDefinition has the following attributes:

Table 1: **StageMetricsDefinition**

Attribute	DataType	Units	Range	Populated	Description
maxMagnitudeType	**MagnitudeType**	N/A	N/A	Always	Contains the **MagnitudeType** the components computing **StageMetrics** use to determine maximum magnitude values.
stationGroup	**StationGroup**	N/A	N/A	Always	Contains the default **Station** collection the components computing **StageMetrics** use to determine **Waveform** availability. Populated as an entity reference.

QC Segment Bridge Definition

Figure 2: **QcSegmentBridgeDefinition** structure



QcSegmentBridgeDefinition includes parameters the Data Fabric needs to construct COI **QcSegment** objects using the existing USNDC database contents.

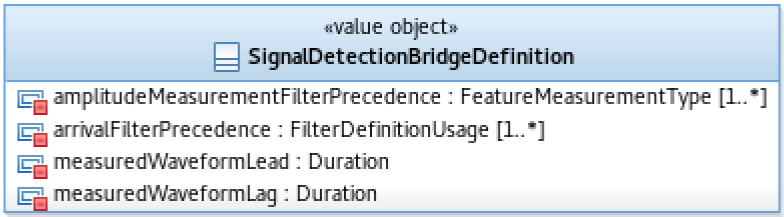
QcSegmentBridgeDefinition has the following attributes:

Table 2: **QcSegmentBridgeDefinition** attribute description

Attribute	DataType	Units	Range	Populated	Description
qcSegmentMaxLength	Duration (ISO-8601 date and time)	Varies / handled by ISO-8601.	N/A	Always	The Data Fabric will ensure the maximum duration of **QcSegmentVersion** objects in the returned **QcSegment** objects does not exceed this value. If a duration exceeds this value, the Data Fabric will split this duration equally into **QcSegment** objects that have a max duration less than or equal to *qcSegmentMaxLength*.

Signal Detection Bridge Definition

Figure 3: **SignalDetectionBridgeDefinition** structure


«value object»
SignalDetectionBridgeDefinition
amplitudeMeasurementFilterPrecedence : FeatureMeasurementType [1..*]
arrivalFilterPrecedence : FilterDefinitionUsage [1..*]
measuredWaveformLead : Duration
measuredWaveformLag : Duration

SignalDetectionBridgeDefinition includes parameters the Data Fabric needs to construct COI **SignalDetection** objects using the existing USNDC database contents.

SignalDetectionBridgeDefinition has the following attributes:

Table 3: **SignalDetectionBridgeDefinition** attribute descriptions

Attribute	DataType	Units	Range	Populated	Description
amplitudeMeasurementFilterPrecedence	**FeatureMeasurementType** ordered collection (non-empty)	N/A	N/A	Always	An ordered collection of amplitude **FeatureMeasurementType** literals providing the order of precedence for which of a **SignalDetectionHypothesis** object's potentially many amplitude **FeatureMeasurement** objects provides the **FilterDefinition** associated with the **FilterDefinitionUsage** literal AMPLITUDE. Ordered from higher precedence to lower precedence.

arrivalFilterPrecedence	**FilterDefinitionUsage** ordered collection (non-empty)	N/A	N/A	Always	An ordered collection of **FilterDefinitionUsage** literals defining the order of precedence for which of the potentially many USNDC format filter definition objects associated with a USNDC format ARRIVAL record provides the COI **FilterDefinition** used to construct **FeatureMeasurement** attributes *channel* and *measuredChannelSegment* and to associate with **FeatureMeasurement** *analysisWaveform* objects. Ordered from higher precedence to lower precedence.
measuredWaveformLag	Duration (ISO-8601 time duration)	Varies / handled by ISO-8601. Will be a unit of elapsed time (e.g. seconds)	>= 0 seconds	Always	Offset after a **SignalDetectionHypothesis** object's measured *ARRIVAL_TIME* used with *measuredWaveformLead* to define: 1. The maximum durations of the **FeatureMeasurement** *measuredChannelSegment* and the **Waveform ChannelSegment** objects in the **FeatureMeasurement** *analysisWaveform* objects (i.e. the duration between their *startTime* and *endTime).* 2. The maximum duration between *effectiveAt* and *effectiveUntil* for derived **Channel** objects created specifically for those **ChannelSegment** objects (e.g. the duration of an event beam steered using a particular **EventHypothesis**).
measuredWaveformLead	Duration (ISO-8601 time duration)	Varies / handled by ISO-8601. Will be a unit of elapsed time (e.g. seconds)	>= 0 seconds	Always	Offset before a **SignalDetectionHypothesis** object's measured *ARRIVAL_TIME* used with *measuredWaveformLag* to define: 1. The maximum durations of the **FeatureMeasurement** *measuredChannelSegment* and the **Waveform ChannelSegment** objects in the **FeatureMeasurement** *analysisWaveform* objects (i.e. the duration between their *startTime* and *endTime).* 2. The maximum duration between *effectiveAt* and *effectiveUntil* for derived **Channel** objects created specifically for those **ChannelSegment** objects (e.g. the duration of an event beam steered using a particular **EventHypothesis**).

Notes

1. None

Change History

1. PI30
    a. 12/2024 - Removed the **FrequencyAmplitudePhaseDefinition** desccription.
2. PI29 - Initial release.