Global Alliance
for Genomics & Health

GA4GH Schemas Documentation

Release 0.0.1

Jeltje van Baren

November 27, 2015

Global Alliance
for Genomics & Health

This is the documentation for the GA4GH API.

CHAPTER 1

Contents

1.1 Introduction

The Data Working Group of the Global Alliance for Genomics and Health developed this web API to facilitate access to and exchange of genomics data across remote sites.

1.1.1 Why use this web API?

This API is specifically designed to allow sharing of genomics data in a standardized manner and without having to exchange complete experiments. With this API, you can

- Exchange genome annotations, DNA sequence reads, reference-based alignments, metadata, and variant calls
- Request alignments and variant calls for one genome or a million.
- Explore data by slicing alignments and variants by genomic location or other feature across one or multiple genomes.
- Interactively process entire cohorts

1.1.2 If you need background

For more details on web APIs, see this wikipedia page. ReST protocols for data transfer are described here. Wikipedia also has good overviews of bioinformatics and DNA sequencing

1.1.3 The GA4GH web API

This API was created to enable researchers to better access and exchange genomic data across remote sites. For example, instead of downloading complete BAM files or whole genome annotations, the API allows retrieval of information on, for instance, single genes or genomic regions.

For a full list of the GA4GH API goals, see *API Goals*

1.1.4 Schemas and formats

The API consists of a series of schemas that define the types of things that API clients and servers exchange: requests for data, server responses, error messages, and objects actually representing pieces of genomics data.

The schemas are written in Avro Interface Description Language (extension .avdl). For more details on Avro and how it is used in the GA4GH APIs, see *Apache Avro*.

Here is an example schema definition for a Variant (with comments removed):

```
record Variant {
  string id;
  string variantSetId;
  array<string> names = [];
  union { null, long } created = null;
  union { null, long } updated = null;
  string referenceName;
  long start;
  long end;
  string referenceBases;
  array<string> alternateBases = [];
  map<array<string>> info = {};
  array<Call> calls = [];
}
```

On the wire, the GA4GH web API takes the form of a client and a server exchanging JSON-serialized objects over HTTP or HTTPS. For more details on JSON, including how the GA4GH web API serializes and deserializes Avro-specified objects in JSON, see *The JSON Format*.

1.2 API Overview

The API is designed for sharing genomic data. It currently has support for sharing sequencing reads, genetic variants, and reference genomes. The following sections give an overview of the API, including general API patterns, individual data types, and links to detailed schema documentation.

1.2.1 API Design

Object Ids

- Many objects need to be referenced by the API (e.g. a search method may return a list of objects) and by applications built outside the API (e.g. a visualization app may refer back to the objects being shown). The API has a standard mechanism for referring to such objects.
- The standard way to programmatically reference an object is via an **id** field. The id is server-assigned, and must provide "id-like" semantics, including:
  - ids are unique within the scope of the server instance
  - ids are durable for the lifetime of the server, and persistent across server restarts – once a user is given an ID for data stored on a server, the id remains valid for as long as the server is still storing that data
- Many objects, including most 'container' objects, also have a **name** field. The name is user-defined, isn't programmatically required to be unique in any given scope (although in practice data owners will often choose unique names), and is intended to be human readable. This can be thought of as a display name.

Cross-repository data federation will need a standard way to refer to a data object, regardless of which repository it's in. There is no such standard currently, and the current API, including the id and name fields, isn't sufficient. A future API may introduce standard cross-repository identifiers using some combination of **content hashes**, **GUIDs**, and central **accession** facilities.

ID and Name

Throughout the API objects have *IDs*. The purpose of IDs is to allow unique identification of all objects within a single server, such that no two objects in a given server have the same ID and no object has more than one ID. The scope of an ID is limited to a given server and an ID may be an arbitrary string.

A name is a user defined identifier. Names need only be uniquely identifying within a specific scope, for example, the names of sequences within a ReferenceSet must be distinct, but there might be two sequences named "chr1" stored in a server, each in a different ReferenceSet. Names may be an arbitrary string.

Object Relationships

- Some objects are contained by other objects. These relationships can be
  - many:1 (e.g. ReadGroupSets in a Dataset); aka single-include
  - many:many (e.g. ReadGroups in a ReadGroupSet); aka multi-include
- Some objects are derived from other objects. These relationships can be
  - many:1 (e.g different aligned ReadGroupSet's derived from an unaligned ReadGroupSet using different alignment algorithms and/or reference sequences)
  - many:many (e.g. different VariantSets derived from a collection of ReadGroupSets using different joint variant calling algorithms)

Dataset

A dataset is a highest level grouping that contains sequence and variant data. It provides the concept of a container which allows high level separation between data.

For the Dataset schema definition see the Metadata schema

Unresolved Issues

- Is the GA4GH object design a conceptual data model that must be followed or only containers for data exchange. If they are containers, where is the conceptual data model defined?
- Are GA4GH objects idempotent? In particular, can one obtain an object with a subset of it's fields?
- Is object life-cycle semantics in the scope of GA4GH API? Which objects are immutable and which are mutable? If objects are mutable, how does one know they have changed? How does one protect against changes while using the objects over a given time-frame?
- What is the definition of the wire protocol? HTTP 1.0? Is HTTP 1.1 chunked encoding allowed? What is the specification for the generate JSON for a given an Avro schema?
- What is the role of Avro? Is it for documentation-only or for use as an IDL?
- Need overall object relationship diagram.

1.2.2 API Goals

- From the GA4GH DWG site:

> The Global Alliance for Genomics and Health (GA4GH) Genomics [API] will allow the interoperable exchange of genomic information across multiple organizations and on multiple platforms. This is a freely available open standard for interoperability, that uses common web protocols to support serving and sharing of data on DNA sequences and genomic variation. The API is implemented as a webservice to create a data source which may be integrated into visualization software, web-based genomics portals or processed as part of genomic analysis pipelines. It overcomes the barriers of incompatible infrastructure between organizations and institutions to enable DNA data providers and consumers to better share genomic data and work together on a global scale, advancing genome research and clinical application.

- The API must allow flexibility in server implementation, including:
  - choice of persistent backend (e.g. files, SQL, NoSQL)
  - choice of implementation language (e.g. Java, Python, Go)
  - choice of authorization model (e.g. all public, all private, fine-grain ACLs)
  - choice of import mechanism (e.g. self-service vs. centrally managed)
  - choice of commercial model (e.g. single payer vs. per-data-owner billing)
  - choice of scale (from a single researcher working with dozens of sequences and homogeneous tools, to government-funded studies with over a million sequences and multiple tool chains)
- The API must allow full-fidelity representation of data that was prepared using today's common methods and stored using today's common file formats.
  - Note that real-world data files sometimes use invalid or ambiguous syntax, making it hard to understand the semantics of the contained data. If a server can't figure out what those semantics are, it can throw an error on import. But whenever the semantics are clear, including when they're specified in valid data files, the API must allow preserving them.
- The API should allow adding more structure to data beyond today's common practices (e.g. formal provenance, versioning).
- The API should allow data owners to organize their data in ways that make sense to them, which implies there often isn't One True Taxonomy. For example, a ReadGroupSet is defined as "a set of ReadGroups that are intended to be analyzed together" – different researchers might choose different sets for different purposes.
  - At the same time, the API should encourage reusable organization, anticipating a future that supports cross-researcher and cross-repository data federation.

Unresolved Issues

- What is the operational scope of the API? Is the capacity goal for a transfer? Should it handle small amounts of data randoman or larger transfers?
- How is sharing defined? Download of data for use in another environment or online, random access to the data?
- What are the performance goals of the API in various configurations?
- What is the scope of interoperability? Code-only interoperability or data interoperability?
- Is the DWG defining an API or a federated network of servers, tools to build a federated network of servers? Need to define the scope.
- Need high-level uses cases for entire API.

1.2.3 Reads

Reads are genetic data generated by a DNA sequencing instrument, including nucleotides and quality scores. Reads may optionally be aligned to a reference sequence. (The data model for reads is similar to SAM/BAM.)

Reads API

See Reads schema for a detailed reference.

Reads Data Model

The Reads data model,although based on the SAM format, allows for more versatile interaction with the data. Instead of sending whole chromosome or whole genome files, the server can send information on specific genomic regions instead.

The model has the following data types:

Record	Description	SAM/BAM rough equivalent
ReadAlignment	One alignment for one read	A single line in a file
ReadGroup	A group of read alignments	A single RG tag
ReadGroupSet	Collecton of ReadGroups that map to the same genome	Single SAM/BAM file
Program	Software version and parameters that were used to align reads to the genome	PN, CL tags in SAM header
ReadStats	Counts of aligned and unaligned reads for a ReadGroup or ReadGroupSet	Samtools flagstats on a file

The relationships are mostly one to many (e.g. each *ReadAlignment* is part of exactly one *ReadGroup*), with the exception that a *ReadGroup* is allowed to be part of more than one *ReadGroupSet*.

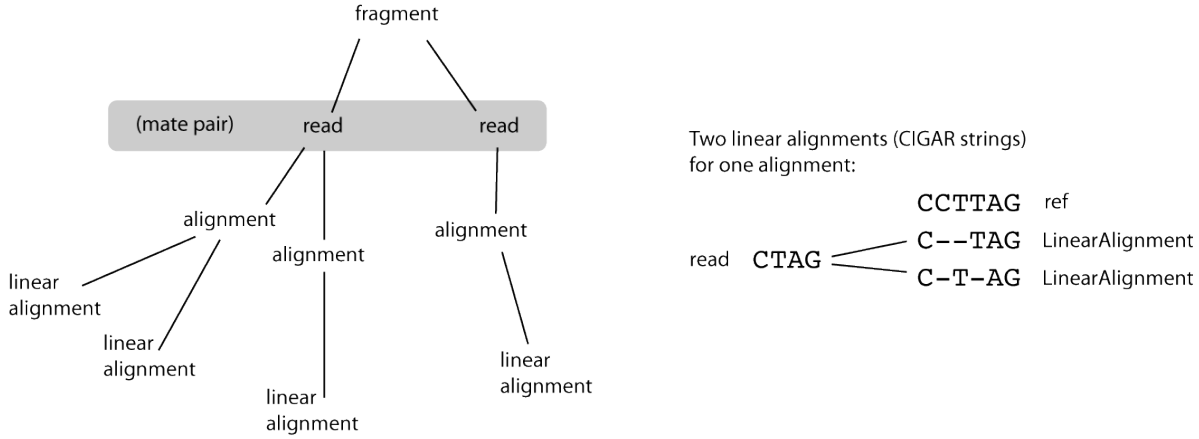
Dataset –< *ReadGroupSet* >–< *ReadGroup* –< *ReadAlignment*

- A *Dataset* is a general-purpose container, defined in metadata.avdl.
- A *ReadGroupSet* is a logical collection of ReadGroups, as determined by the data owner. Typically one *ReadGroupSet* represents all the Reads from one experimental sample, which traditionally would be stored in a single BAM file.

- A *ReadGroup* is all the data that's processed the same way by the sequencer. There are typically 1-10 ReadGroups in a *ReadGroupSet*.
- A *ReadAlignment* object is a flattened representation of several layers of bioinformatics hierarchy, including fragments, reads, and alignments, stored in one object for easy access.

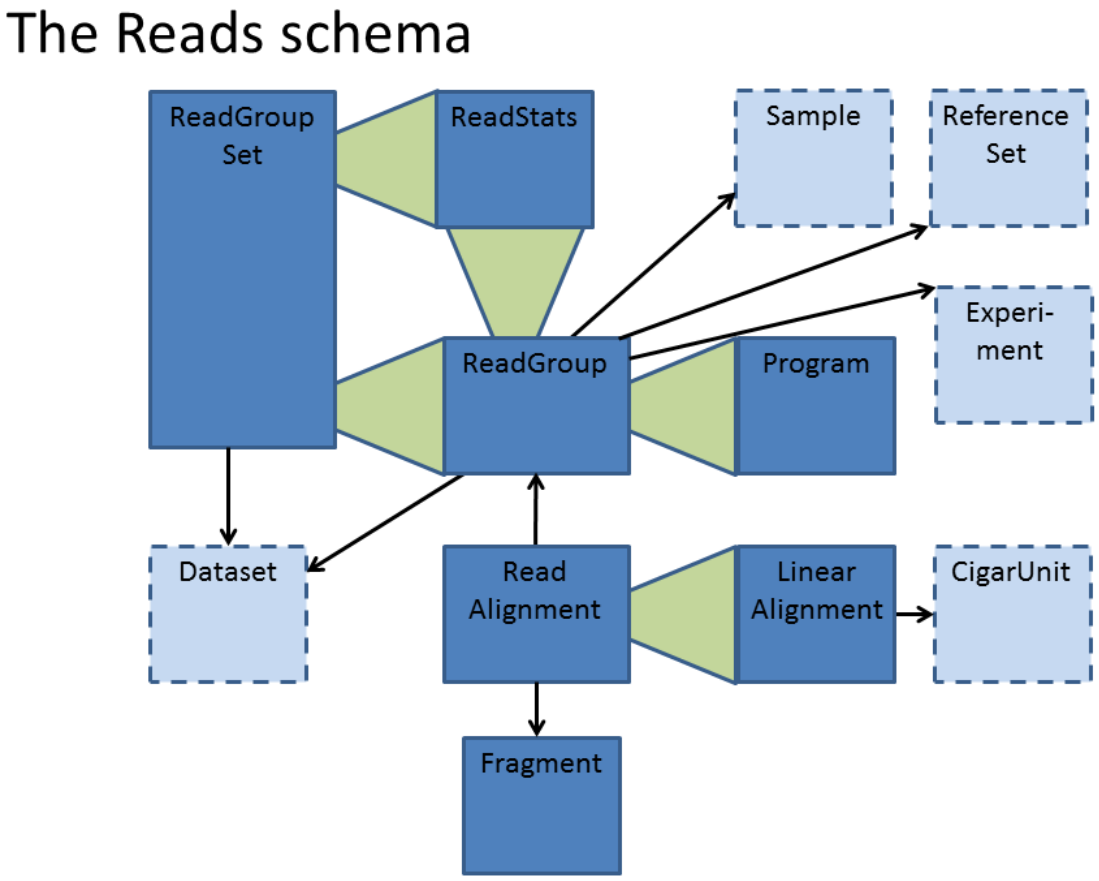
ReadAlignment: detailed discussion

One *ReadAlignment* object represents the following logical hierarchy. See the field definitions in the *ReadAlignment* object for more details.



- A *fragment* is a single stretch of a DNA molecule. There are typically at least millions of fragments in a ReadGroup. A fragment has a name (QNAME in BAM spec), a length (TLEN in BAM spec), and one or more reads.
- A *read* is a contiguous sequence of bases. There are typically only one or two reads in a fragment. If there are two reads, they're known as a mate pair. A read has an array of base values, an array of base qualities, and optional alignment information.
- An *alignment* is the way alignment software maps a read to a reference. There's one primary alignment, and can be one or more secondary alignments. Secondary alignments represent alternate possible mappings.
- A *linear alignment* maps a string of bases to a reference using a single CIGAR string. There's one representative alignment, and can be one or more supplementary alignments. Supplementary alignments represent linear alignments that are subsets of a chimeric alignment.

The image below shows which Reads records contain other records (represented by green triangles), and which contain IDs that can be used to get information from other records (arrows). The arrow points *from* the record that lists the ID *to* the record that can be identified by that ID. Records are represented by blue rectangles; dotted lines indicate records defined in other schemas.



1.2.4 Variants

Variants are genetic differences between an experimental sample and a reference sequence. (The data model for variants is similar to VCF.)

Variants API

See Variants schema for a detailed reference.

Variants Data Model

The Variants data model, although based on the VCF format, allows for more versaitle interaction with the data. Instead of sending whole VCF files, the server can send information on specific variants or genomic regions instead. And instead of getting the whole genotype matrix, it's possible to just get details for one or more specified individuals.

The API uses four main entities to represent variants. The following diagram illustrates how these entities relate to each other to constitute the genotype matrix.

The lowest-level entity is a Call:

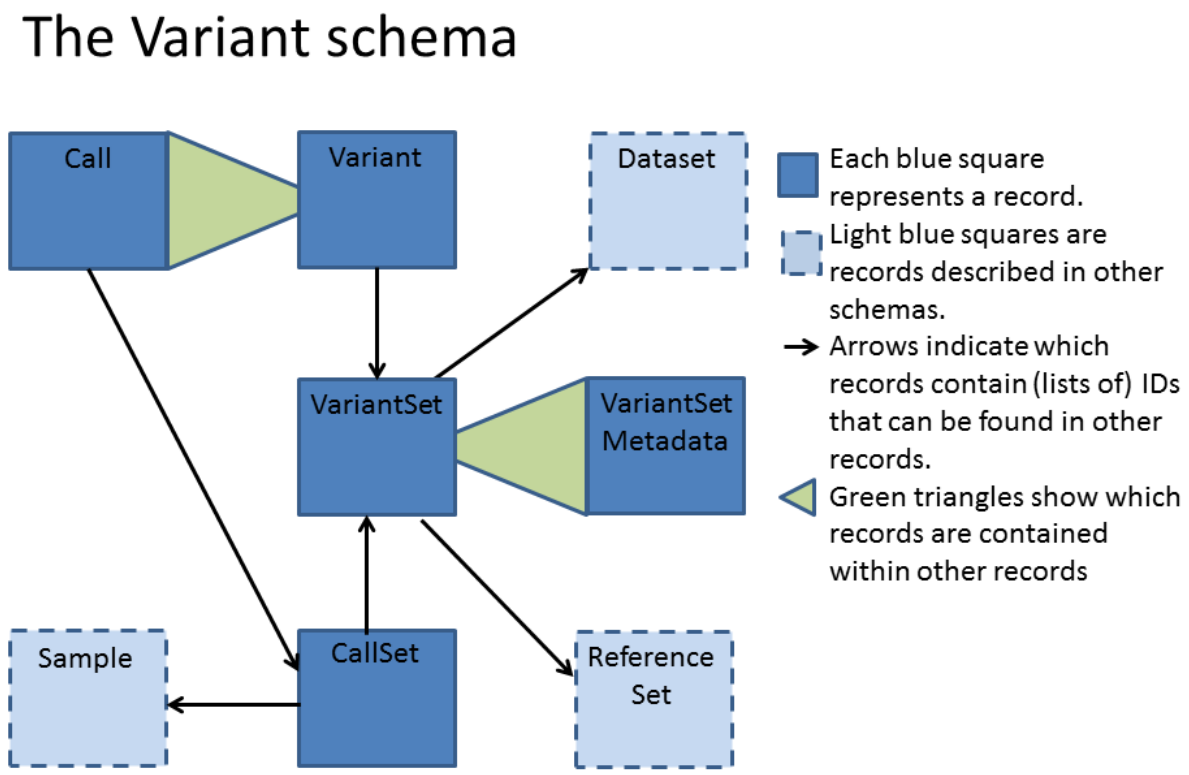
- a `Call` encodes the genotype of an individual with respect to a variant, as determined by some analysis of experimental data.

The other entities can be thought of as collections of Calls that have something in common:

- a *VariantSet* supports working with a collection of Calls intended to be analyzed together.
- a *Variant* supports working with the subset of Calls in a VariantSet that are at the same site and are described using the same set of alleles. The Variant entity contains:
  - a variant description: a potential difference between experimental DNA and a reference sequence, including the site (position of the difference) and alleles (how the bases differ)
  - variant observations: a collection of Calls describing evidence for actual instances of that difference, as seen in analyses of experimental data
- a *CallSet* supports working with the subset of Calls in a VariantSet that were generated by the same analysis of the same sample. The CallSet includes information about which sample was analyzed and how it was analyzed, and is linked to information about what differences were found.

The following diagram shows the relationship of these four entities to each other and to other GA4GH API entities. It shows which entities contain other entities (such as *VariantSetMetadata*), and which contain IDs that can be used to get information from other entities (such as *Variant*'s `variantSetId`). The arrow points *from* the entity that contains the ID *to* the entity that can be identified by that ID.

FIXME: remove the Sample object from the graphic; that object isn't (yet) defined in the API.



1.2.5 References

References are standard genome sequences, used to provide a coordinate system for reads and variants.

References API

See References schema for a detailed reference.

References Data Model

A genome assembly is a digital representation of a genome. It is typically composed of *contigs*, each an uninterrupted string representing a DNA sequence, arranged into *scaffolds*, each of which orders and orients a set of contigs. Scaffolds are typically represented as a string, with runs of wildcard characters (N or n) used to represent interstitial regions of uncertain DNA between contigs.

A reference genome is a genome assembly that other genomes are compared to and described with respect to. For example, sequencing reads are mapped to and described with respect to a reference genome in the API, and genetic variations are described as edits to reference scaffolds/contigs. In the API a reference genome is described by a *ReferenceSet*. In turn a *ReferenceSet* is composed of a set of *Reference* objects, each which represents a scaffold or contig in the assembly. A fairly minimal amount of metadata is associated with the *ReferenceSet* and *Reference* objects.

1.2.6 Metadata

Metadata allows organizing all the primary data types.

Metadata API

Goals and Scope

The metadata API provides information on the primary data objects available via the GA4GH API, and facilities to organize primary data objects.

The current metadata API is immature and will evolve in future.

Datasets

All GA4GH data objects are part of a *dataset*. A dataset is a data-provider-specified collection of related data of multiple types. Logically, it's akin to a folder, where it's up to the provider what goes into the folder. Individual data objects are linked by *datasetId* fields to Dataset objects.

Since the grouping of content in a dataset is determined by the data provider, users should not make semantic assumptions about that data. Subsets of the data in a dataset can be selected for analysis using other metadata or attributes.

Use Cases

For server implementors, datasets are a useful level of granularity for implementing administrative features such as access control (e.g. Data set X is public; data set Y is only available to lab Z's collaborators) and billing (e.g. the costs of hosting Dataset Y should be charged to lab Z).

For data curators, datasets are 'the simplest thing that could possibly work' for grouping data (e.g. Dataset X has all the reads, variants, and expression levels for a particular research project; Dataset Y has all the work product from a particular grant).

For data accessors, datasets are a simple way to scope exploration and analysis (e.g. are there any supporting examples in 1000genomes? what's the distribution of that result in the data from our project?).

Issues (TODO)

- Metadata API is immature and under development.
- *sampleId* is referenced in metadata, reads, and variants records of the schema, however there is no *Sample* object defined in metadata.
- There has been DWG discussion about proposing a new role for the *Dataset* object.
- Lifecycle and version management of metadata objects is not clearly defined. This includes the use of timestamps.
- *Experiment* object is currently copied into the *ReadGroup* object. Given metadata becomes a chain of objects associated with the data, copying records seems less that ideal.

1.3 Schemas

1.3.1 Common

This file defines common types used in other parts of the schema. There are no directly associated methods.

enum `Strand`

Symbols NEG_STRAND|POS_STRAND

Indicates the DNA strand associate for some data item. * *NEG_STRAND*: The negative (-) strand. * *POS_STRAND*: The postive (+) strand.

record `Position`

Fields

- **`referenceName`** (*string*) – The name of the *Reference* on which the *Position* is located.
- **`position`** (*long*) –

  The 0-based offset from the start of the forward strand for that *Reference*. Genomic positions are non-negative integers less than *Reference* length.
- **`strand`** (Strand) – Strand the position is associated with.

A *Position* is an unoriented base in some *Reference*. A *Position* is represented by a *Reference* name, and a base number on that *Reference* (0-based).

record `ExternalIdentifier`

Fields

- **`database`** (*string*) –

  The source of the identifier. (e.g. *Ensembl*)
- **`identifier`** (*string*) –

  The ID defined by the external database. (e.g. *ENST00000000000*)
- **`version`** (*string*) –

  The version of the object or the database (e.g. *78*)

Identifier from a public database

enum `CigarOperation`

Symbols ALIGNMENT_MATCH|INSERT|DELETE|SKIP|CLIP_SOFT|CLIP_HARD|PAD|SEQUENCE_MATCH|SEQU

An enum for the different types of CIGAR alignment operations that exist. Used wherever CIGAR alignments are used. The different enumerated values have the following usage:

- *ALIGNMENT_MATCH*: An alignment match indicates that a sequence can be aligned to the reference without evidence of an INDEL. Unlike the *SEQUENCE_MATCH* and *SEQUENCE_MISMATCH* operators, the *ALIGNMENT_MATCH* operator does not indicate whether the reference and read sequences are an exact match. This operator is equivalent to SAM's *M*.
- *INSERT*: The insert operator indicates that the read contains evidence of bases being inserted into the reference. This operator is equivalent to SAM's *I*.
- *DELETE*: The delete operator indicates that the read contains evidence of bases being deleted from the reference. This operator is equivalent to SAM's *D*.
- *SKIP*: The skip operator indicates that this read skips a long segment of the reference, but the bases have not been deleted. This operator is commonly used when working with RNA-seq data, where reads may skip long segments of the reference between exons. This operator is equivalent to SAM's 'N'.
- *CLIP_SOFT*: The soft clip operator indicates that bases at the start/end of a read have not been considered during alignment. This may occur if the majority of a read maps, except for low quality bases at the start/end of a read. This operator is equivalent to SAM's 'S'. Bases that are soft clipped will still be stored in the read.
- *CLIP_HARD*: The hard clip operator indicates that bases at the start/end of a read have been omitted from this alignment. This may occur if this linear alignment is part of a chimeric alignment, or if the read has been trimmed (e.g., during error correction, or to trim poly-A tails for RNA-seq). This operator is equivalent to SAM's 'H'.
- *PAD*: The pad operator indicates that there is padding in an alignment. This operator is equivalent to SAM's 'P'.
- *SEQUENCE_MATCH*: This operator indicates that this portion of the aligned sequence exactly matches the reference (e.g., all bases are equal to the reference bases). This operator is equivalent to SAM's '='.
- *SEQUENCE_MISMATCH*: This operator indicates that this portion of the aligned sequence is an alignment match to the reference, but a sequence mismatch (e.g., the bases are not equal to the reference). This can indicate a SNP or a read error. This operator is equivalent to SAM's 'X'.

record `CigarUnit`

Fields

- **`operation`** (CigarOperation) – The operation type.
- **`operationLength`** (*long*) – The number of bases that the operation runs for.
- **`referenceSequence`** (*null|string*) –

  ***referenceSequence* is only used at mismatches (*SEQUENCE_MISMATCH*)** and deletions (*DELETE*). Filling this field replaces the MD tag. If the relevant information is not available, leave this field as *null*.

A structure for an instance of a CIGAR operation. *FIXME: This belongs under Reads (only readAlignment refers to this)*

1.3.2 Metadata

This protocol defines metadata used in the other GA4GH protocols.

enum `Strand`

Symbols NEG_STRAND|POS_STRAND

Indicates the DNA strand associate for some data item. * *NEG_STRAND*: The negative (-) strand. * *POS_STRAND*: The postive (+) strand.

record `Position`

Fields

- **`referenceName`** (*string*) – The name of the *Reference* on which the *Position* is located.
- **`position`** (*long*) –

  The 0-based offset from the start of the forward strand for that *Reference*. Genomic positions are non-negative integers less than *Reference* length.
- **`strand`** (Strand) – Strand the position is associated with.

A *Position* is an unoriented base in some *Reference*. A *Position* is represented by a *Reference* name, and a base number on that *Reference* (0-based).

record `ExternalIdentifier`

Fields

- **`database`** (*string*) –

  The source of the identifier. (e.g. *Ensembl*)
- **`identifier`** (*string*) –

  The ID defined by the external database. (e.g. *ENST00000000000*)
- **`version`** (*string*) –

  The version of the object or the database (e.g. *78*)

Identifier from a public database

enum `CigarOperation`

Symbols ALIGNMENT_MATCH|INSERT|DELETE|SKIP|CLIP_SOFT|CLIP_HARD|PAD|SEQUENCE_MATCH|SEQU

An enum for the different types of CIGAR alignment operations that exist. Used wherever CIGAR alignments are used. The different enumerated values have the following usage:

•*ALIGNMENT_MATCH*: An alignment match indicates that a sequence can be aligned to the reference without evidence of an INDEL. Unlike the *SEQUENCE_MATCH* and *SEQUENCE_MISMATCH* operators, the *ALIGNMENT_MATCH* operator does not indicate whether the reference and read sequences are an exact match. This operator is equivalent to SAM's *M*.

•*INSERT*: The insert operator indicates that the read contains evidence of bases being inserted into the reference. This operator is equivalent to SAM's *I*.

•*DELETE*: The delete operator indicates that the read contains evidence of bases being deleted from the reference. This operator is equivalent to SAM's *D*.

•*SKIP*: The skip operator indicates that this read skips a long segment of the reference, but the bases have not been deleted. This operator is commonly used when working with RNA-seq data, where reads may skip long segments of the reference between exons. This operator is equivalent to SAM's 'N'.

•*CLIP_SOFT*: The soft clip operator indicates that bases at the start/end of a read have not been considered during alignment. This may occur if the majority of a read maps, except for low quality bases at the start/end of a read. This operator is equivalent to SAM's 'S'. Bases that are soft clipped will still be stored in the read.

- *CLIP_HARD*: The hard clip operator indicates that bases at the start/end of a read have been omitted from this alignment. This may occur if this linear alignment is part of a chimeric alignment, or if the read has been trimmed (e.g., during error correction, or to trim poly-A tails for RNA-seq). This operator is equivalent to SAM's 'H'.
- *PAD*: The pad operator indicates that there is padding in an alignment. This operator is equivalent to SAM's 'P'.
- *SEQUENCE_MATCH*: This operator indicates that this portion of the aligned sequence exactly matches the reference (e.g., all bases are equal to the reference bases). This operator is equivalent to SAM's '='.
- *SEQUENCE_MISMATCH*: This operator indicates that this portion of the aligned sequence is an alignment match to the reference, but a sequence mismatch (e.g., the bases are not equal to the reference). This can indicate a SNP or a read error. This operator is equivalent to SAM's 'X'.

record `CigarUnit`

Fields

- **`operation`** (CigarOperation) – The operation type.
- **`operationLength`** (*long*) – The number of bases that the operation runs for.
- **`referenceSequence`** (*null|string*) –

  ***referenceSequence* is only used at mismatches (*SEQUENCE_MISMATCH*)** and deletions (*DELETE*). Filling this field replaces the MD tag. If the relevant information is not available, leave this field as *null*.

A structure for an instance of a CIGAR operation. *FIXME: This belongs under Reads (only readAlignment refers to this)*

record `Experiment`

Fields

- **`id`** (*string*) – The experiment UUID. This is globally unique.
- **`name`** (*null|string*) – The name of the experiment.
- **`description`** (*null|string*) – A description of the experiment.
- **`recordCreateTime`** (*string*) –

  The time at which this record was created. Format: ISO 8601, YYYY-MM-DDTHH:MM:SS.SSS (e.g. 2015-02-10T00:03:42.123Z)
- **`recordUpdateTime`** (*string*) –

  The time at which this record was last updated. Format: ISO 8601, YYYY-MM-DDTHH:MM:SS.SSS (e.g. 2015-02-10T00:03:42.123Z)
- **`runTime`** (*null|string*) –

  The time at which this experiment was performed. Granularity here is variable (e.g. date only). Format: ISO 8601, YYYY-MM-DDTHH:MM:SS (e.g. 2015-02-10T00:03:42)
- **`molecule`** (*null|string*) – The molecule examined in this experiment. (e.g. genomics DNA, total RNA)
- **`strategy`** (*null|string*) –

  The experiment technique or strategy applied to the sample. (e.g. whole genome sequencing, RNA-seq, RIP-seq)

- **selection** (*null|string*) –

  The method used to enrich the target. (e.g. immunoprecipitation, size fractionation, MNase digestion)

- **library** (*null|string*) – The name of the library used as part of this experiment.
- **libraryLayout** (*null|string*) – The configuration of sequenced reads. (e.g. Single or Paired)
- **instrumentModel** (*null|string*) –

  The instrument model used as part of this experiment. This maps to sequencing technology in BAM.

- **instrumentDataFile** (*null|string*) –

  The data file generated by the instrument. TODO: This isn't actually a file is it? Should this be *instrumentData* instead?

- **sequencingCenter** (*null|string*) – The sequencing center used as part of this experiment.
- **platformUnit** (*null|string*) –

  The platform unit used as part of this experiment. This is a flowcell-barcode or slide unique identifier.

- **info** (*map<array<string>>*) – A map of additional experiment information.

An experimental preparation of a sample.

record Dataset

Fields

- **id** (*string*) – The dataset's id, locally unique to the server instance.
- **name** (*null|string*) – The name of the dataset.
- **description** (*null|string*) – Additional, human-readable information on the dataset.

A Dataset is a collection of related data of multiple types. Data providers decide how to group data into datasets. See [Metadata API](../api/metadata.html) for a more detailed discussion.

1.3.3 RPC

error GAException

A general exception type.

1.3.4 ReadMethods

searchDatasets (*request*)

Parameters **request** – SearchDatasetsRequest: This request maps to the body of *POST /datasets/search* as JSON.

Return type SearchDatasetsResponse

Throws GAException

Gets a list of datasets accessible through the API.

TODO: Reads and variants both want to have datasets. Are they the same object?

POST /datasets/search must accept a JSON version of *SearchDatasetsRequest* as the post body and will return a JSON version of *SearchDatasetsResponse*.

searchReads(*request*)

Parameters **request** – SearchReadsRequest: This request maps to the body of *POST /reads/search* as JSON.

Return type SearchReadsResponse

Throws GAException

Gets a list of ‘ReadAlignment‘s for one or more ‘ReadGroup‘s.

searchReads operates over a genomic coordinate space of reference sequence and position defined by the ‘Reference‘s to which the requested ‘ReadGroup‘s are aligned.

If a target positional range is specified, search returns all reads whose alignment to the reference genome *overlap* the range. A query which specifies only read group IDs yields all reads in those read groups, including unmapped reads.

All reads returned (including reads on subsequent pages) are ordered by genomic coordinate (by reference sequence, then position). Reads with equivalent genomic coordinates are returned in an unspecified order. This order must be consistent for a given repository, such that two queries for the same content (regardless of page size) yield reads in the same order across their respective streams of paginated responses.

POST /reads/search must accept a JSON version of *SearchReadsRequest* as the post body and will return a JSON version of *SearchReadsResponse*.

getDataset(*id*)

Parameters **id** – string: The ID of the *Dataset*.

Return type org.ga4gh.models.Dataset

Throws GAException

Gets a *Dataset* by ID. *GET /datasets/{id}* will return a JSON version of *Dataset*.

searchReadGroupSets(*request*)

Parameters **request** – SearchReadGroupSetsRequest: This request maps to the body of *POST /readgroupsets/search* as JSON.

Return type SearchReadGroupSetsResponse

Throws GAException

Gets a list of *ReadGroupSet* matching the search criteria.

POST /readgroupsets/search must accept a JSON version of *SearchReadGroupSetsRequest* as the post body and will return a JSON version of *SearchReadGroupSetsResponse*.

getReadGroupSet(*id*)

Parameters **id** – string: The ID of the *ReadGroupSet*.

Return type org.ga4gh.models.ReadGroupSet

Throws GAException

Gets a *org.ga4gh.models.ReadGroupSet* by ID. *GET /readgroupsets/{id}* will return a JSON version of *ReadGroupSet*.

getReadGroup(*id*)

Parameters **id** – string: The ID of the *ReadGroup*.

Return type org.ga4gh.models.ReadGroup

Throws GAException

Gets a *org.ga4gh.models.ReadGroup* by ID. *GET /readgroups/{id}* will return a JSON version of *ReadGroup*.

enum Strand

Symbols NEG_STRAND|POS_STRAND

Indicates the DNA strand associate for some data item. * *NEG_STRAND*: The negative (-) strand. * *POS_STRAND*: The postive (+) strand.

record Position

Fields

- **referenceName** (*string*) – The name of the *Reference* on which the *Position* is located.
- **position** (*long*) –

  The 0-based offset from the start of the forward strand for that *Reference*. Genomic positions are non-negative integers less than *Reference* length.
- **strand** (Strand) – Strand the position is associated with.

A *Position* is an unoriented base in some *Reference*. A *Position* is represented by a *Reference* name, and a base number on that *Reference* (0-based).

record ExternalIdentifier

Fields

- **database** (*string*) –

  The source of the identifier. (e.g. *Ensembl*)
- **identifier** (*string*) –

  The ID defined by the external database. (e.g. *ENST00000000000*)
- **version** (*string*) –

  The version of the object or the database (e.g. *78*)

Identifier from a public database

enum CigarOperation

Symbols ALIGNMENT_MATCH|INSERT|DELETE|SKIP|CLIP_SOFT|CLIP_HARD|PAD|SEQUENCE_MATCH|SEQU

An enum for the different types of CIGAR alignment operations that exist. Used wherever CIGAR alignments are used. The different enumerated values have the following usage:

•*ALIGNMENT_MATCH*: An alignment match indicates that a sequence can be aligned to the reference without evidence of an INDEL. Unlike the *SEQUENCE_MATCH* and *SEQUENCE_MISMATCH* operators, the *ALIGNMENT_MATCH* operator does not indicate whether the reference and read sequences are an exact match. This operator is equivalent to SAM's *M*.

•*INSERT*: The insert operator indicates that the read contains evidence of bases being inserted into the reference. This operator is equivalent to SAM's *I*.

•*DELETE*: The delete operator indicates that the read contains evidence of bases being deleted from the reference. This operator is equivalent to SAM's *D*.

- *SKIP*: The skip operator indicates that this read skips a long segment of the reference, but the bases have not been deleted. This operator is commonly used when working with RNA-seq data, where reads may skip long segments of the reference between exons. This operator is equivalent to SAM's 'N'.
- *CLIP_SOFT*: The soft clip operator indicates that bases at the start/end of a read have not been considered during alignment. This may occur if the majority of a read maps, except for low quality bases at the start/end of a read. This operator is equivalent to SAM's 'S'. Bases that are soft clipped will still be stored in the read.
- *CLIP_HARD*: The hard clip operator indicates that bases at the start/end of a read have been omitted from this alignment. This may occur if this linear alignment is part of a chimeric alignment, or if the read has been trimmed (e.g., during error correction, or to trim poly-A tails for RNA-seq). This operator is equivalent to SAM's 'H'.
- *PAD*: The pad operator indicates that there is padding in an alignment. This operator is equivalent to SAM's 'P'.
- *SEQUENCE_MATCH*: This operator indicates that this portion of the aligned sequence exactly matches the reference (e.g., all bases are equal to the reference bases). This operator is equivalent to SAM's '='.
- *SEQUENCE_MISMATCH*: This operator indicates that this portion of the aligned sequence is an alignment match to the reference, but a sequence mismatch (e.g., the bases are not equal to the reference). This can indicate a SNP or a read error. This operator is equivalent to SAM's 'X'.

record `CigarUnit`

Fields

- **`operation`** (CigarOperation) – The operation type.
- **`operationLength`** (*long*) – The number of bases that the operation runs for.
- **`referenceSequence`** (*null|string*) –

  ***referenceSequence* is only used at mismatches (*SEQUENCE_MISMATCH*)** and deletions (*DELETE*). Filling this field replaces the MD tag. If the relevant information is not available, leave this field as *null*.

A structure for an instance of a CIGAR operation. *FIXME: This belongs under Reads (only readAlignment refers to this)*

error `GAException`

A general exception type.

record `Experiment`

Fields

- **`id`** (*string*) – The experiment UUID. This is globally unique.
- **`name`** (*null|string*) – The name of the experiment.
- **`description`** (*null|string*) – A description of the experiment.
- **`recordCreateTime`** (*string*) –

  The time at which this record was created. Format: ISO 8601, YYYY-MM-DDTHH:MM:SS.SSS (e.g. 2015-02-10T00:03:42.123Z)
- **`recordUpdateTime`** (*string*) –

  The time at which this record was last updated. Format: ISO 8601, YYYY-MM-DDTHH:MM:SS.SSS (e.g. 2015-02-10T00:03:42.123Z)
- **`runTime`** (*null|string*) –

The time at which this experiment was performed. Granularity here is variable (e.g. date only). Format: ISO 8601, YYYY-MM-DDTHH:MM:SS (e.g. 2015-02-10T00:03:42)

- **molecule** (*null|string*) – The molecule examined in this experiment. (e.g. genomics DNA, total RNA)
- **strategy** (*null|string*) –

  The experiment technique or strategy applied to the sample. (e.g. whole genome sequencing, RNA-seq, RIP-seq)
- **selection** (*null|string*) –

  The method used to enrich the target. (e.g. immunoprecipitation, size fractionation, MNase digestion)
- **library** (*null|string*) – The name of the library used as part of this experiment.
- **libraryLayout** (*null|string*) – The configuration of sequenced reads. (e.g. Single or Paired)
- **instrumentModel** (*null|string*) –

  The instrument model used as part of this experiment. This maps to sequencing technology in BAM.
- **instrumentDataFile** (*null|string*) –

  The data file generated by the instrument. TODO: This isn't actually a file is it? Should this be *instrumentData* instead?
- **sequencingCenter** (*null|string*) – The sequencing center used as part of this experiment.
- **platformUnit** (*null|string*) –

  The platform unit used as part of this experiment. This is a flowcell-barcode or slide unique identifier.
- **info** (*map<array<string>>*) – A map of additional experiment information.

An experimental preparation of a sample.

record Dataset

Fields

- **id** (*string*) – The dataset's id, locally unique to the server instance.
- **name** (*null|string*) – The name of the dataset.
- **description** (*null|string*) – Additional, human-readable information on the dataset.

A Dataset is a collection of related data of multiple types. Data providers decide how to group data into datasets. See [Metadata API](../api/metadata.html) for a more detailed discussion.

record Program

Fields

- **commandLine** (*null|string*) – The command line used to run this program.
- **id** (*null|string*) – The user specified ID of the program.
- **name** (*null|string*) – The name of the program.
- **prevProgramId** (*null|string*) – The ID of the program run before this one.

- **version** (*null|string*) – The version of the program run.

Program can be used to track the provenance of how read data was generated.

record ReadStats

Fields

- **alignedReadCount** (*null|long*) – The number of aligned reads.
- **unalignedReadCount** (*null|long*) – The number of unaligned reads.
- **baseCount** (*null|long*) –

  The total number of bases. This is equivalent to the sum of *alignedSequence.length* for all reads.

ReadStats can be used to provide summary statistics about read data.

record ReadGroup

Fields

- **id** (*string*) – The read group ID.
- **datasetId** (*null|string*) – The ID of the dataset this read group belongs to.
- **name** (*null|string*) – The read group name.
- **description** (*null|string*) – The read group description.
- **sampleId** (*null|string*) –

  The sample this read group's data was generated from. Note: the current API does not have a rigorous definition of sample. Therefore, this field actually contains an arbitrary string, typically corresponding to the SM tag in a BAM file.
- **experiment** (*null|Experiment*) – The experiment used to generate this read group.
- **predictedInsertSize** (*null|int*) – The predicted insert size of this read group.
- **created** (*null|long*) – The time at which this read group was created in milliseconds from the epoch.
- **updated** (*null|long*) –

  The time at which this read group was last updated in milliseconds from the epoch.
- **stats** (*null|ReadStats*) – Statistical data on reads in this read group.
- **programs** (*array<Program>*) – The programs used to generate this read group.
- **referenceSetId** (*null|string*) –

  The ID of the reference set to which the reads in this read group are aligned. Required if there are any read alignments.
- **info** (*map<array<string>>*) – A map of additional read group information.

A ReadGroup is a set of reads derived from one physical sequencing process.

record ReadGroupSet

Fields

- **id** (*string*) – The read group set ID.
- **datasetId** (*null|string*) – The ID of the dataset this read group set belongs to.
- **name** (*null|string*) – The read group set name.

- **stats** (*null|ReadStats*) – Statistical data on reads in this read group set.
- **readGroups** (*array<ReadGroup>*) – The read groups in this set.

A ReadGroupSet is a logical collection of ReadGroups. Typically one ReadGroupSet represents all the reads from one experimental sample.

record LinearAlignment

Fields

- **position** (Position) – The position of this alignment.
- **mappingQuality** (*null|int*) –

  The mapping quality of this alignment, meaning the likelihood that the read maps to this position.

  Specifically, this is -10 log10 Pr(mapping position is wrong), rounded to the nearest integer.
- **cigar** (*array<CigarUnit>*) –

  Represents the local alignment of this sequence (alignment matches, indels, etc) versus the reference.

A linear alignment describes the alignment of a read to a Reference, using a position and CIGAR array.

record Fragment

Fields

- **id** (*string*) – The fragment ID.

A fragment represents a contiguous stretch of a DNA or RNA molecule. Reads can be associated with a fragment to specify they derive from the same molecule. TODO: this Fragment object is essentially unused, and may be removed in a future PR.

record ReadAlignment

Fields

- **id** (*null|string*) –

  The read alignment ID. This ID is unique within the read group this alignment belongs to.

  For performance reasons, this field may be omitted by a backend. If provided, its intended use is to make caching and UI display easier for genome browsers and other lightweight clients.
- **readGroupId** (*string*) –

  The ID of the read group this read belongs to. (Every read must belong to exactly one read group.)
- **fragmentId** (*string*) –

  The fragment ID that this ReadAlignment belongs to. TODO: this is the only reference to the Fragment object, which may be removed in a future PR.
- **fragmentName** (*string*) – The fragment name. Equivalent to QNAME (query template name) in SAM.
- **properPlacement** (*null|boolean*) –

The orientation and the distance between reads from the fragment are consistent with the sequencing protocol (equivalent to SAM flag 0x2)

- **duplicateFragment** (*null|boolean*) – The fragment is a PCR or optical duplicate (SAM flag 0x400).
- **numberReads** (*null|int*) – The number of reads in the fragment (extension to SAM flag 0x1)
- **fragmentLength** (*null|int*) – The observed length of the fragment, equivalent to TLEN in SAM.
- **readNumber** (*null|int*) –

  The read ordinal in the fragment, 0-based and less than numberReads. This field replaces SAM flag 0x40 and 0x80 and is intended to more cleanly represent multiple reads per fragment.
- **failedVendorQualityChecks** (*null|boolean*) – The read fails platform or vendor quality checks (SAM flag 0x200).
- **alignment** (*null|LinearAlignment*) –

  The alignment for this alignment record. This field will be null if the read is unmapped.
- **secondaryAlignment** (*null|boolean*) –

  Whether this alignment is secondary. Equivalent to SAM flag 0x100. A secondary alignment represents an alternative to the primary alignment for this read. Aligners may return secondary alignments if a read can map ambiguously to multiple coordinates in the genome.

  By convention, each read has one and only one alignment where both secondaryAlignment and supplementaryAlignment are false.
- **supplementaryAlignment** (*null|boolean*) –

  Whether this alignment is supplementary. Equivalent to SAM flag 0x800. Supplementary alignments are used in the representation of a chimeric alignment. In a chimeric alignment, a read is split into multiple linear alignments that map to different reference contigs. The first linear alignment in the read will be designated as the representative alignment; the remaining linear alignments will be designated as supplementary alignments. These alignments may have different mapping quality scores.

  In each linear alignment in a chimeric alignment, the read will be hard clipped. The *alignedSequence* and *alignedQuality* fields in the alignment record will only represent the bases for its respective linear alignment.
- **alignedSequence** (*null|string*) –

  The bases of the read sequence contained in this alignment record (equivalent to SEQ in SAM).

  alignedSequence and *alignedQuality* may be shorter than the full read sequence and quality. This will occur if the alignment is part of a chimeric alignment, or if the read was trimmed. When this occurs, the CIGAR for this read will begin/end with a hard clip operator that will indicate the length of the excised sequence.
- **alignedQuality** (*array<int>*) –

  The quality of the read sequence contained in this alignment record (equivalent to QUAL in SAM).

alignedSequence and *alignedQuality* may be shorter than the full read sequence and quality. This will occur if the alignment is part of a chimeric alignment, or if the read was trimmed. When this occurs, the CIGAR for this read will begin/end with a hard clip operator that will indicate the length of the excised sequence.

- **nextMatePosition** (*null|Position*) –

  The mapping of the primary alignment of the *(readNumber+1)%numberReads* read in the fragment. It replaces mate position and mate strand in SAM.

- **info** (*map<array<string>>*) – A map of additional read alignment information.

Each read alignment describes an alignment with additional information about the fragment and the read. A read alignment object is equivalent to a line in a SAM file.

record SearchReadsRequest

Fields

- **readGroupIds** (*array<string>*) – The ReadGroups to search. At least one id must be specified.
- **referenceId** (*null|string*) –

  The reference to query. Leaving blank returns results from all references, including unmapped reads - this could be very large.

- **start** (*null|long*) –

  The start position (0-based) of this query. If a reference is specified, this defaults to 0. Genomic positions are non-negative integers less than reference length. Requests spanning the join of circular genomes are represented as two requests one on each side of the join (position 0).

- **end** (*null|long*) –

  The end position (0-based, exclusive) of this query. If a reference is specified, this defaults to the reference's length.

- **pageSize** (*null|int*) –

  Specifies the maximum number of results to return in a single page. If unspecified, a system default will be used.

- **pageToken** (*null|string*) –

  The continuation token, which is used to page through large result sets. To get the next page of results, set this parameter to the value of *nextPageToken* from the previous response.

This request maps to the body of *POST /reads/search* as JSON.

If a reference is specified, all queried *ReadGroup's must be aligned to 'ReferenceSet's containing that same 'Reference*. If no reference is specified, all queried *ReadGroup's must be aligned to the same 'ReferenceSet*.

record SearchReadsResponse

Fields

- **alignments** (*array<org.ga4gh.models.ReadAlignment>*) –

  The list of matching alignment records, sorted by position. Unmapped reads, which have no position, are returned last.

- **nextPageToken** (*null|string*) –

The continuation token, which is used to page through large result sets. Provide this value in a subsequent request to return the next page of results. This field will be empty if there aren't any additional results.

This is the response from *POST /reads/search* expressed as JSON.

record `SearchReadGroupSetsRequest`

Fields

- **`datasetId`** (*string*) – The dataset to search.
- **`name`** (*null|string*) – Only return read group sets with this name (case-sensitive, exact match).
- **`pageSize`** (*null|int*) –

  Specifies the maximum number of results to return in a single page. If unspecified, a system default will be used.

- **`pageToken`** (*null|string*) –

  The continuation token, which is used to page through large result sets. To get the next page of results, set this parameter to the value of *nextPageToken* from the previous response.

This request maps to the body of *POST /readgroupsets/search* as JSON.

TODO: Factor this out to a common API patterns section. - If searching by a resource ID, and that resource is not found, the method will return a *404* HTTP status code (*NOT_FOUND*). - If searching by other attributes, e.g. *name*, and no matches are found, the method will return a *200* HTTP status code (*OK*) with an empty result list.

record `SearchReadGroupSetsResponse`

Fields

- **`readGroupSets`** (*array<org.ga4gh.models.ReadGroupSet>*) – The list of matching read group sets.
- **`nextPageToken`** (*null|string*) –

  The continuation token, which is used to page through large result sets. Provide this value in a subsequent request to return the next page of results. This field will be empty if there aren't any additional results.

This is the response from *POST /readgroupsets/search* expressed as JSON.

record `SearchDatasetsRequest`

Fields

- **`pageSize`** (*null|int*) –

  Specifies the maximum number of results to return in a single page. If unspecified, a system default will be used.

- **`pageToken`** (*null|string*) –

  The continuation token, which is used to page through large result sets. To get the next page of results, set this parameter to the value of *nextPageToken* from the previous response.

This request maps to the body of *POST /datasets/search* as JSON.

record `SearchDatasetsResponse`

Fields

- **datasets** (*array<org.ga4gh.models.Dataset>*) – The list of datasets.
- **nextPageToken** (*null|string*) –

  The continuation token, which is used to page through large result sets. Provide this value in a subsequent request to return the next page of results. This field will be empty if there aren't any additional results.

This is the response from *POST /datasets/search* expressed as JSON.

1.3.5 Reads

This file defines the objects used to represent a reads and alignments, most importantly ReadGroupSet, ReadGroup, and ReadAlignment. See {TODO: LINK TO READS OVERVIEW} for more information.

enum Strand

Symbols NEG_STRAND|POS_STRAND

Indicates the DNA strand associate for some data item. * *NEG_STRAND*: The negative (-) strand. * *POS_STRAND*: The postive (+) strand.

record Position

Fields

- **referenceName** (*string*) – The name of the *Reference* on which the *Position* is located.
- **position** (*long*) –

  The 0-based offset from the start of the forward strand for that *Reference*. Genomic positions are non-negative integers less than *Reference* length.
- **strand** (Strand) – Strand the position is associated with.

A *Position* is an unoriented base in some *Reference*. A *Position* is represented by a *Reference* name, and a base number on that *Reference* (0-based).

record ExternalIdentifier

Fields

- **database** (*string*) –

  The source of the identifier. (e.g. *Ensembl*)
- **identifier** (*string*) –

  The ID defined by the external database. (e.g. *ENST00000000000*)
- **version** (*string*) –

  The version of the object or the database (e.g. *78*)

Identifier from a public database

enum CigarOperation

Symbols ALIGNMENT_MATCH|INSERT|DELETE|SKIP|CLIP_SOFT|CLIP_HARD|PAD|SEQUENCE_MATCH|SEQU

An enum for the different types of CIGAR alignment operations that exist. Used wherever CIGAR alignments are used. The different enumerated values have the following usage:

- *ALIGNMENT_MATCH*: An alignment match indicates that a sequence can be aligned to the reference without evidence of an INDEL. Unlike the *SEQUENCE_MATCH* and *SEQUENCE_MISMATCH* operators, the *ALIGNMENT_MATCH* operator does not indicate whether the reference and read sequences are an exact match. This operator is equivalent to SAM's *M*.
- *INSERT*: The insert operator indicates that the read contains evidence of bases being inserted into the reference. This operator is equivalent to SAM's *I*.
- *DELETE*: The delete operator indicates that the read contains evidence of bases being deleted from the reference. This operator is equivalent to SAM's *D*.
- *SKIP*: The skip operator indicates that this read skips a long segment of the reference, but the bases have not been deleted. This operator is commonly used when working with RNA-seq data, where reads may skip long segments of the reference between exons. This operator is equivalent to SAM's 'N'.
- *CLIP_SOFT*: The soft clip operator indicates that bases at the start/end of a read have not been considered during alignment. This may occur if the majority of a read maps, except for low quality bases at the start/end of a read. This operator is equivalent to SAM's 'S'. Bases that are soft clipped will still be stored in the read.
- *CLIP_HARD*: The hard clip operator indicates that bases at the start/end of a read have been omitted from this alignment. This may occur if this linear alignment is part of a chimeric alignment, or if the read has been trimmed (e.g., during error correction, or to trim poly-A tails for RNA-seq). This operator is equivalent to SAM's 'H'.
- *PAD*: The pad operator indicates that there is padding in an alignment. This operator is equivalent to SAM's 'P'.
- *SEQUENCE_MATCH*: This operator indicates that this portion of the aligned sequence exactly matches the reference (e.g., all bases are equal to the reference bases). This operator is equivalent to SAM's '='.
- *SEQUENCE_MISMATCH*: This operator indicates that this portion of the aligned sequence is an alignment match to the reference, but a sequence mismatch (e.g., the bases are not equal to the reference). This can indicate a SNP or a read error. This operator is equivalent to SAM's 'X'.

record `CigarUnit`

Fields

- **`operation`** (CigarOperation) – The operation type.
- **`operationLength`** (*long*) – The number of bases that the operation runs for.
- **`referenceSequence`** (*null|string*) –

  ***referenceSequence* is only used at mismatches (*SEQUENCE_MISMATCH*)** and deletions (*DELETE*). Filling this field replaces the MD tag. If the relevant information is not available, leave this field as *null*.

A structure for an instance of a CIGAR operation. *FIXME: This belongs under Reads (only readAlignment refers to this)*

record `Experiment`

Fields

- **`id`** (*string*) – The experiment UUID. This is globally unique.
- **`name`** (*null|string*) – The name of the experiment.
- **`description`** (*null|string*) – A description of the experiment.
- **`recordCreateTime`** (*string*) –

The time at which this record was created. Format: ISO 8601, YYYY-MM-DDTHH:MM:SS.SSS (e.g. 2015-02-10T00:03:42.123Z)

- **recordUpdateTime** (*string*) –

  The time at which this record was last updated. Format: ISO 8601, YYYY-MM-DDTHH:MM:SS.SSS (e.g. 2015-02-10T00:03:42.123Z)

- **runTime** (*null|string*) –

  The time at which this experiment was performed. Granularity here is variable (e.g. date only). Format: ISO 8601, YYYY-MM-DDTHH:MM:SS (e.g. 2015-02-10T00:03:42)

- **molecule** (*null|string*) – The molecule examined in this experiment. (e.g. genomics DNA, total RNA)

- **strategy** (*null|string*) –

  The experiment technique or strategy applied to the sample. (e.g. whole genome sequencing, RNA-seq, RIP-seq)

- **selection** (*null|string*) –

  The method used to enrich the target. (e.g. immunoprecipitation, size fractionation, MNase digestion)

- **library** (*null|string*) – The name of the library used as part of this experiment.

- **libraryLayout** (*null|string*) – The configuration of sequenced reads. (e.g. Single or Paired)

- **instrumentModel** (*null|string*) –

  The instrument model used as part of this experiment. This maps to sequencing technology in BAM.

- **instrumentDataFile** (*null|string*) –

  The data file generated by the instrument. TODO: This isn't actually a file is it? Should this be *instrumentData* instead?

- **sequencingCenter** (*null|string*) – The sequencing center used as part of this experiment.

- **platformUnit** (*null|string*) –

  The platform unit used as part of this experiment. This is a flowcell-barcode or slide unique identifier.

- **info** (*map<array<string>>*) – A map of additional experiment information.

An experimental preparation of a sample.

record Dataset

Fields

- **id** (*string*) – The dataset's id, locally unique to the server instance.
- **name** (*null|string*) – The name of the dataset.
- **description** (*null|string*) – Additional, human-readable information on the dataset.

A Dataset is a collection of related data of multiple types. Data providers decide how to group data into datasets. See [Metadata API](../api/metadata.html) for a more detailed discussion.

record `Program`

Fields

- **`commandLine`** (*null|string*) – The command line used to run this program.
- **`id`** (*null|string*) – The user specified ID of the program.
- **`name`** (*null|string*) – The name of the program.
- **`prevProgramId`** (*null|string*) – The ID of the program run before this one.
- **`version`** (*null|string*) – The version of the program run.

Program can be used to track the provenance of how read data was generated.

record `ReadStats`

Fields

- **`alignedReadCount`** (*null|long*) – The number of aligned reads.
- **`unalignedReadCount`** (*null|long*) – The number of unaligned reads.
- **`baseCount`** (*null|long*) –

  The total number of bases. This is equivalent to the sum of *alignedSequence.length* for all reads.

ReadStats can be used to provide summary statistics about read data.

record `ReadGroup`

Fields

- **`id`** (*string*) – The read group ID.
- **`datasetId`** (*null|string*) – The ID of the dataset this read group belongs to.
- **`name`** (*null|string*) – The read group name.
- **`description`** (*null|string*) – The read group description.
- **`sampleId`** (*null|string*) –

  The sample this read group's data was generated from. Note: the current API does not have a rigorous definition of sample. Therefore, this field actually contains an arbitrary string, typically corresponding to the SM tag in a BAM file.
- **`experiment`** (*null|Experiment*) – The experiment used to generate this read group.
- **`predictedInsertSize`** (*null|int*) – The predicted insert size of this read group.
- **`created`** (*null|long*) – The time at which this read group was created in milliseconds from the epoch.
- **`updated`** (*null|long*) –

  The time at which this read group was last updated in milliseconds from the epoch.
- **`stats`** (*null|ReadStats*) – Statistical data on reads in this read group.
- **`programs`** (*array<Program>*) – The programs used to generate this read group.
- **`referenceSetId`** (*null|string*) –

  The ID of the reference set to which the reads in this read group are aligned. Required if there are any read alignments.
- **`info`** (*map<array<string>>*) – A map of additional read group information.

A ReadGroup is a set of reads derived from one physical sequencing process.

record `ReadGroupSet`

Fields

- **`id`** (*string*) – The read group set ID.
- **`datasetId`** (*null|string*) – The ID of the dataset this read group set belongs to.
- **`name`** (*null|string*) – The read group set name.
- **`stats`** (*null|ReadStats*) – Statistical data on reads in this read group set.
- **`readGroups`** (*array<ReadGroup>*) – The read groups in this set.

A ReadGroupSet is a logical collection of ReadGroups. Typically one ReadGroupSet represents all the reads from one experimental sample.

record `LinearAlignment`

Fields

- **`position`** (Position) – The position of this alignment.
- **`mappingQuality`** (*null|int*) –

  The mapping quality of this alignment, meaning the likelihood that the read maps to this position.

  Specifically, this is -10 log10 Pr(mapping position is wrong), rounded to the nearest integer.

- **`cigar`** (*array<CigarUnit>*) –

  Represents the local alignment of this sequence (alignment matches, indels, etc) versus the reference.

A linear alignment describes the alignment of a read to a Reference, using a position and CIGAR array.

record `Fragment`

Fields

- **`id`** (*string*) – The fragment ID.

A fragment represents a contiguous stretch of a DNA or RNA molecule. Reads can be associated with a fragment to specify they derive from the same molecule. TODO: this Fragment object is essentially unused, and may be removed in a future PR.

record `ReadAlignment`

Fields

- **`id`** (*null|string*) –

  The read alignment ID. This ID is unique within the read group this alignment belongs to.

  For performance reasons, this field may be omitted by a backend. If provided, its intended use is to make caching and UI display easier for genome browsers and other lightweight clients.

- **`readGroupId`** (*string*) –

  The ID of the read group this read belongs to. (Every read must belong to exactly one read group.)

- **fragmentId** (*string*) –

  The fragment ID that this ReadAlignment belongs to. TODO: this is the only reference to the Fragment object, which may be removed in a future PR.

- **fragmentName** (*string*) – The fragment name. Equivalent to QNAME (query template name) in SAM.

- **properPlacement** (*null|boolean*) –

  The orientation and the distance between reads from the fragment are consistent with the sequencing protocol (equivalent to SAM flag 0x2)

- **duplicateFragment** (*null|boolean*) – The fragment is a PCR or optical duplicate (SAM flag 0x400).

- **numberReads** (*null|int*) – The number of reads in the fragment (extension to SAM flag 0x1)

- **fragmentLength** (*null|int*) – The observed length of the fragment, equivalent to TLEN in SAM.

- **readNumber** (*null|int*) –

  The read ordinal in the fragment, 0-based and less than numberReads. This field replaces SAM flag 0x40 and 0x80 and is intended to more cleanly represent multiple reads per fragment.

- **failedVendorQualityChecks** (*null|boolean*) – The read fails platform or vendor quality checks (SAM flag 0x200).

- **alignment** (*null|LinearAlignment*) –

  The alignment for this alignment record. This field will be null if the read is unmapped.

- **secondaryAlignment** (*null|boolean*) –

  Whether this alignment is secondary. Equivalent to SAM flag 0x100. A secondary alignment represents an alternative to the primary alignment for this read. Aligners may return secondary alignments if a read can map ambiguously to multiple coordinates in the genome.

  By convention, each read has one and only one alignment where both secondaryAlignment and supplementaryAlignment are false.

- **supplementaryAlignment** (*null|boolean*) –

  Whether this alignment is supplementary. Equivalent to SAM flag 0x800. Supplementary alignments are used in the representation of a chimeric alignment. In a chimeric alignment, a read is split into multiple linear alignments that map to different reference contigs. The first linear alignment in the read will be designated as the representative alignment; the remaining linear alignments will be designated as supplementary alignments. These alignments may have different mapping quality scores.

  In each linear alignment in a chimeric alignment, the read will be hard clipped. The *alignedSequence* and *alignedQuality* fields in the alignment record will only represent the bases for its respective linear alignment.

- **alignedSequence** (*null|string*) –

  The bases of the read sequence contained in this alignment record (equivalent to SEQ in SAM).

alignedSequence and *alignedQuality* may be shorter than the full read sequence and quality. This will occur if the alignment is part of a chimeric alignment, or if the read was trimmed. When this occurs, the CIGAR for this read will begin/end with a hard clip operator that will indicate the length of the excised sequence.

- **alignedQuality** (*array<int>*) –

  The quality of the read sequence contained in this alignment record (equivalent to QUAL in SAM).

  alignedSequence and *alignedQuality* may be shorter than the full read sequence and quality. This will occur if the alignment is part of a chimeric alignment, or if the read was trimmed. When this occurs, the CIGAR for this read will begin/end with a hard clip operator that will indicate the length of the excised sequence.

- **nextMatePosition** (*null|Position*) –

  The mapping of the primary alignment of the *(readNumber+1)%numberReads* read in the fragment. It replaces mate position and mate strand in SAM.

- **info** (*map<array<string>>*) – A map of additional read alignment information.

Each read alignment describes an alignment with additional information about the fragment and the read. A read alignment object is equivalent to a line in a SAM file.

1.3.6 ReferenceMethods

getReferenceSet(*id*)

Parameters **id** – string: The ID of the *ReferenceSet*.

Return type org.ga4gh.models.ReferenceSet

Throws GAException

Gets a *ReferenceSet* by ID. *GET /referencesets/{id}* will return a JSON version of *ReferenceSet*.

getReference(*id*)

Parameters **id** – string: The ID of the *Reference*.

Return type org.ga4gh.models.Reference

Throws GAException

Gets a *Reference* by ID. *GET /references/{id}* will return a JSON version of *Reference*.

searchReferences(*request*)

Parameters **request** – SearchReferencesRequest: This request maps to the body of *POST /references/search*

as JSON. :return type: SearchReferencesResponse :throws: GAException

Gets a list of *Reference* matching the search criteria.

POST /references/search must accept a JSON version of *SearchReferencesRequest* as the post body and will return a JSON version of *SearchReferencesResponse*.

getReferenceBases(*id*, *request*)

Parameters

- **id** – string: The ID of the *Reference*.

- **request** – ListReferenceBasesRequest: Additional request parameters to restrict the query.

Return type ListReferenceBasesResponse

Throws GAException

Lists *Reference* bases by ID and optional range. *GET /references/{id}/bases* will return a JSON version of *ListReferenceBasesResponse*.

searchReferenceSets(*request*)

Parameters **request** – SearchReferenceSetsRequest: This request maps to the body of *POST /referencesets/search*

as JSON. :return type: SearchReferenceSetsResponse :throws: GAException

Gets a list of *ReferenceSet* matching the search criteria.

POST /referencesets/search must accept a JSON version of *SearchReferenceSetsRequest* as the post body and will return a JSON version of *SearchReferenceSetsResponse*.

enum Strand

Symbols NEG_STRAND|POS_STRAND

Indicates the DNA strand associate for some data item. * *NEG_STRAND*: The negative (-) strand. * *POS_STRAND*: The postive (+) strand.

record Position

Fields

- **referenceName** (*string*) – The name of the *Reference* on which the *Position* is located.
- **position** (*long*) –

  The 0-based offset from the start of the forward strand for that *Reference*. Genomic positions are non-negative integers less than *Reference* length.
- **strand** (Strand) – Strand the position is associated with.

A *Position* is an unoriented base in some *Reference*. A *Position* is represented by a *Reference* name, and a base number on that *Reference* (0-based).

record ExternalIdentifier

Fields

- **database** (*string*) –

  The source of the identifier. (e.g. *Ensembl*)
- **identifier** (*string*) –

  The ID defined by the external database. (e.g. *ENST00000000000*)
- **version** (*string*) –

  The version of the object or the database (e.g. *78*)

Identifier from a public database

enum CigarOperation

Symbols ALIGNMENT_MATCH|INSERT|DELETE|SKIP|CLIP_SOFT|CLIP_HARD|PAD|SEQUENCE_MATCH|SEQU

An enum for the different types of CIGAR alignment operations that exist. Used wherever CIGAR alignments are used. The different enumerated values have the following usage:

•*ALIGNMENT_MATCH*: An alignment match indicates that a sequence can be aligned to the reference without evidence of an INDEL. Unlike the *SEQUENCE_MATCH* and *SEQUENCE_MISMATCH* operators, the *ALIGNMENT_MATCH* operator does not indicate whether the reference and read sequences are an exact match. This operator is equivalent to SAM's *M*.

•*INSERT*: The insert operator indicates that the read contains evidence of bases being inserted into the reference. This operator is equivalent to SAM's *I*.

•*DELETE*: The delete operator indicates that the read contains evidence of bases being deleted from the reference. This operator is equivalent to SAM's *D*.

•*SKIP*: The skip operator indicates that this read skips a long segment of the reference, but the bases have not been deleted. This operator is commonly used when working with RNA-seq data, where reads may skip long segments of the reference between exons. This operator is equivalent to SAM's 'N'.

•*CLIP_SOFT*: The soft clip operator indicates that bases at the start/end of a read have not been considered during alignment. This may occur if the majority of a read maps, except for low quality bases at the start/end of a read. This operator is equivalent to SAM's 'S'. Bases that are soft clipped will still be stored in the read.

•*CLIP_HARD*: The hard clip operator indicates that bases at the start/end of a read have been omitted from this alignment. This may occur if this linear alignment is part of a chimeric alignment, or if the read has been trimmed (e.g., during error correction, or to trim poly-A tails for RNA-seq). This operator is equivalent to SAM's 'H'.

•*PAD*: The pad operator indicates that there is padding in an alignment. This operator is equivalent to SAM's 'P'.

•*SEQUENCE_MATCH*: This operator indicates that this portion of the aligned sequence exactly matches the reference (e.g., all bases are equal to the reference bases). This operator is equivalent to SAM's '='.

•*SEQUENCE_MISMATCH*: This operator indicates that this portion of the aligned sequence is an alignment match to the reference, but a sequence mismatch (e.g., the bases are not equal to the reference). This can indicate a SNP or a read error. This operator is equivalent to SAM's 'X'.

record `CigarUnit`

Fields

- **`operation`** (CigarOperation) – The operation type.
- **`operationLength`** (*long*) – The number of bases that the operation runs for.
- **`referenceSequence`** (*null|string*) –

  ***referenceSequence* is only used at mismatches (*SEQUENCE_MISMATCH*)** and deletions (*DELETE*). Filling this field replaces the MD tag. If the relevant information is not available, leave this field as *null*.

A structure for an instance of a CIGAR operation. *FIXME: This belongs under Reads (only readAlignment refers to this)*

error `GAException`

A general exception type.

record `Reference`

Fields

- **`id`** (*string*) – The reference ID. Unique within the repository.
- **`length`** (*long*) – The length of this reference's sequence.
- **`md5checksum`** (*string*) –

The MD5 checksum uniquely representing this *Reference* as a lower-case hexadecimal string, calculated as the MD5 of the upper-case sequence excluding all whitespace characters (this is equivalent to SQ:M5 in SAM).

- **`name`** (*string*) – The name of this reference. (e.g. '22').
- **`sourceURI`** (*null|string*) –

  The URI from which the sequence was obtained. Specifies a FASTA format file/string with one name, sequence pair. In most cases, clients should call the *getReferenceBases()* method to obtain sequence bases for a *Reference* instead of attempting to retrieve this URI.
- **`sourceAccessions`** (*array<string>*) –

  All known corresponding accession IDs in INSDC (GenBank/ENA/DDBJ) which must include a version number, e.g. *GCF_000001405.26*.
- **`isDerived`** (*boolean*) –

  A sequence X is said to be derived from source sequence Y, if X and Y are of the same length and the per-base sequence divergence at A/C/G/T bases is sufficiently small. Two sequences derived from the same official sequence share the same coordinates and annotations, and can be replaced with the official sequence for certain use cases.
- **`sourceDivergence`** (*null|float*) –

  The *sourceDivergence* is the fraction of non-indel bases that do not match the reference this record was derived from.
- **`ncbiTaxonId`** (*null|int*) – ID from http://www.ncbi.nlm.nih.gov/taxonomy (e.g. 9606->human).

A *Reference* is a canonical assembled contig, intended to act as a reference coordinate space for other genomic annotations. A single *Reference* might represent the human chromosome 1, for instance.

'Reference's are designed to be immutable.

record `ReferenceSet`

Fields

- **`id`** (*string*) – The reference set ID. Unique in the repository.
- **`name`** (*null|string*) – The reference set name.
- **`md5checksum`** (*string*) – Order-independent MD5 checksum which identifies this *ReferenceSet*.

  To compute this checksum, make a list of *Reference.md5checksum* for all 'Reference's in this set. Then sort that list, and take the MD5 hash of all the strings concatenated together. Express the hash as a lower-case hexadecimal string.
- **`ncbiTaxonId`** (*null|int*) –

  ID from http://www.ncbi.nlm.nih.gov/taxonomy (e.g. 9606->human) indicating the species which this assembly is intended to model. Note that contained *Reference's may specify a different 'ncbiTaxonId*, as assemblies may contain reference sequences which do not belong to the modeled species, e.g. EBV in a human reference genome.
- **`description`** (*null|string*) – Optional free text description of this reference set.
- **`assemblyId`** (*null|string*) – Public id of this reference set, such as *GRCh37*.
- **`sourceURI`** (*null|string*) – Specifies a FASTA format file/string.

- **sourceAccessions** (*array<string>*) –

  All known corresponding accession IDs in INSDC (GenBank/ENA/DDBJ) ideally with a version number, e.g. *NC_000001.11*.

- **isDerived** (*boolean*) –

  A reference set may be derived from a source if it contains additional sequences, or some of the sequences within it are derived (see the definition of *isDerived* in *Reference*).

A *ReferenceSet* is a set of *Reference's which typically comprise a reference assembly, such as 'GRCh38*. A *ReferenceSet* defines a common coordinate space for comparing reference-aligned experimental data.

record SearchReferenceSetsRequest

Fields

- **md5checksum** (*null|string*) –

  If not null, return the reference sets for which the *md5checksum* matches this string (case-sensitive, exact match). See *ReferenceSet::md5checksum* for details.

- **accession** (*null|string*) –

  If not null, return the reference sets for which the *accession* matches this string (case-sensitive, exact match).

- **assemblyId** (*null|string*) –

  If not null, return the reference sets for which the *assemblyId* matches this string (case-sensitive, exact match).

- **pageSize** (*null|int*) –

  Specifies the maximum number of results to return in a single page. If unspecified, a system default will be used.

- **pageToken** (*null|string*) –

  The continuation token, which is used to page through large result sets. To get the next page of results, set this parameter to the value of *nextPageToken* from the previous response.

This request maps to the body of *POST /referencesets/search* as JSON.

record SearchReferenceSetsResponse

Fields

- **referenceSets** (*array<org.ga4gh.models.ReferenceSet>*) – The list of matching reference sets.

- **nextPageToken** (*null|string*) –

  The continuation token, which is used to page through large result sets. Provide this value in a subsequent request to return the next page of results. This field will be empty if there aren't any additional results.

This is the response from *POST /referencesets/search* expressed as JSON.

record SearchReferencesRequest

Fields

- **referenceSetId** (*string*) – The *ReferenceSet* to search.

- **md5checksum** (*null|string*) –

If not null, return the references for which the *md5checksum* matches this string (case-sensitive, exact match). See *ReferenceSet::md5checksum* for details.

- **accession** (*null|string*) –

  If not null, return the references for which the *accession* matches this string (case-sensitive, exact match).

- **pageSize** (*null|int*) –

  Specifies the maximum number of results to return in a single page. If unspecified, a system default will be used.

- **pageToken** (*null|string*) –

  The continuation token, which is used to page through large result sets. To get the next page of results, set this parameter to the value of *nextPageToken* from the previous response.

This request maps to the body of *POST /references/search* as JSON.

record SearchReferencesResponse

Fields

- **references** (*array<org.ga4gh.models.Reference>*) – The list of matching references.
- **nextPageToken** (*null|string*) –

  The continuation token, which is used to page through large result sets. Provide this value in a subsequent request to return the next page of results. This field will be empty if there aren't any additional results.

This is the response from *POST /references/search* expressed as JSON.

record ListReferenceBasesRequest

Fields

- **start** (*long*) –

  The start position (0-based) of this query. Defaults to 0. Genomic positions are non-negative integers less than reference length. Requests spanning the join of circular genomes are represented as two requests one on each side of the join (position 0).

- **end** (*null|long*) –

  The end position (0-based, exclusive) of this query. Defaults to the length of this *Reference*.

- **pageToken** (*null|string*) –

  The continuation token, which is used to page through large result sets. To get the next page of results, set this parameter to the value of *nextPageToken* from the previous response.

The query parameters for a request to *GET /references/{id}/bases*, for example:

GET /references/{id}/bases?start=100&end=200

record ListReferenceBasesResponse

Fields

- **offset** (*long*) –

The offset position (0-based) of the given sequence from the start of this *Reference*. This value will differ for each page in a paginated request.

- **`sequence`** (*string*) –

  A substring of the bases that make up this reference. Bases are represented as IUPAC-IUB codes; this string matches the regexp *[ACGTMRWSYKVHDBN]**.

- **`nextPageToken`** (*null|string*) –

  The continuation token, which is used to page through large result sets. Provide this value in a subsequent request to return the next page of results. This field will be empty if there aren't any additional results.

The response from *GET /references/{id}/bases* expressed as JSON.

1.3.7 References

Defines types used by the GA4GH References API.

enum `Strand`

Symbols NEG_STRAND|POS_STRAND

Indicates the DNA strand associate for some data item. * *NEG_STRAND*: The negative (-) strand. * *POS_STRAND*: The postive (+) strand.

record `Position`

Fields

- **`referenceName`** (*string*) – The name of the *Reference* on which the *Position* is located.
- **`position`** (*long*) –

  The 0-based offset from the start of the forward strand for that *Reference*. Genomic positions are non-negative integers less than *Reference* length.

- **`strand`** (Strand) – Strand the position is associated with.

A *Position* is an unoriented base in some *Reference*. A *Position* is represented by a *Reference* name, and a base number on that *Reference* (0-based).

record `ExternalIdentifier`

Fields

- **`database`** (*string*) –

  The source of the identifier. (e.g. *Ensembl*)

- **`identifier`** (*string*) –

  The ID defined by the external database. (e.g. *ENST00000000000*)

- **`version`** (*string*) –

  The version of the object or the database (e.g. *78*)

Identifier from a public database

enum `CigarOperation`

Symbols ALIGNMENT_MATCH|INSERT|DELETE|SKIP|CLIP_SOFT|CLIP_HARD|PAD|SEQUENCE_MATCH|SEQU

An enum for the different types of CIGAR alignment operations that exist. Used wherever CIGAR alignments are used. The different enumerated values have the following usage:

•*ALIGNMENT_MATCH*: An alignment match indicates that a sequence can be aligned to the reference without evidence of an INDEL. Unlike the *SEQUENCE_MATCH* and *SEQUENCE_MISMATCH* operators, the *ALIGNMENT_MATCH* operator does not indicate whether the reference and read sequences are an exact match. This operator is equivalent to SAM's *M*.

•*INSERT*: The insert operator indicates that the read contains evidence of bases being inserted into the reference. This operator is equivalent to SAM's *I*.

•*DELETE*: The delete operator indicates that the read contains evidence of bases being deleted from the reference. This operator is equivalent to SAM's *D*.

•*SKIP*: The skip operator indicates that this read skips a long segment of the reference, but the bases have not been deleted. This operator is commonly used when working with RNA-seq data, where reads may skip long segments of the reference between exons. This operator is equivalent to SAM's 'N'.

•*CLIP_SOFT*: The soft clip operator indicates that bases at the start/end of a read have not been considered during alignment. This may occur if the majority of a read maps, except for low quality bases at the start/end of a read. This operator is equivalent to SAM's 'S'. Bases that are soft clipped will still be stored in the read.

•*CLIP_HARD*: The hard clip operator indicates that bases at the start/end of a read have been omitted from this alignment. This may occur if this linear alignment is part of a chimeric alignment, or if the read has been trimmed (e.g., during error correction, or to trim poly-A tails for RNA-seq). This operator is equivalent to SAM's 'H'.

•*PAD*: The pad operator indicates that there is padding in an alignment. This operator is equivalent to SAM's 'P'.

•*SEQUENCE_MATCH*: This operator indicates that this portion of the aligned sequence exactly matches the reference (e.g., all bases are equal to the reference bases). This operator is equivalent to SAM's '='.

•*SEQUENCE_MISMATCH*: This operator indicates that this portion of the aligned sequence is an alignment match to the reference, but a sequence mismatch (e.g., the bases are not equal to the reference). This can indicate a SNP or a read error. This operator is equivalent to SAM's 'X'.

record `CigarUnit`

Fields

- **`operation`** (CigarOperation) – The operation type.
- **`operationLength`** (*long*) – The number of bases that the operation runs for.
- **`referenceSequence`** (*null|string*) –

  ***referenceSequence* is only used at mismatches (*SEQUENCE_MISMATCH*)** and deletions (*DELETE*). Filling this field replaces the MD tag. If the relevant information is not available, leave this field as *null*.

A structure for an instance of a CIGAR operation. *FIXME: This belongs under Reads (only readAlignment refers to this)*

record `Reference`

Fields

- **`id`** (*string*) – The reference ID. Unique within the repository.
- **`length`** (*long*) – The length of this reference's sequence.
- **`md5checksum`** (*string*) –

The MD5 checksum uniquely representing this *Reference* as a lower-case hexadecimal string, calculated as the MD5 of the upper-case sequence excluding all whitespace characters (this is equivalent to SQ:M5 in SAM).

- **`name`** (*string*) – The name of this reference. (e.g. '22').
- **`sourceURI`** (*null|string*) –

  The URI from which the sequence was obtained. Specifies a FASTA format file/string with one name, sequence pair. In most cases, clients should call the *getReferenceBases()* method to obtain sequence bases for a *Reference* instead of attempting to retrieve this URI.
- **`sourceAccessions`** (*array<string>*) –

  All known corresponding accession IDs in INSDC (GenBank/ENA/DDBJ) which must include a version number, e.g. *GCF_000001405.26*.
- **`isDerived`** (*boolean*) –

  A sequence X is said to be derived from source sequence Y, if X and Y are of the same length and the per-base sequence divergence at A/C/G/T bases is sufficiently small. Two sequences derived from the same official sequence share the same coordinates and annotations, and can be replaced with the official sequence for certain use cases.
- **`sourceDivergence`** (*null|float*) –

  The *sourceDivergence* is the fraction of non-indel bases that do not match the reference this record was derived from.
- **`ncbiTaxonId`** (*null|int*) – ID from http://www.ncbi.nlm.nih.gov/taxonomy (e.g. 9606->human).

A *Reference* is a canonical assembled contig, intended to act as a reference coordinate space for other genomic annotations. A single *Reference* might represent the human chromosome 1, for instance.

'Reference's are designed to be immutable.

record `ReferenceSet`

Fields

- **`id`** (*string*) – The reference set ID. Unique in the repository.
- **`name`** (*null|string*) – The reference set name.
- **`md5checksum`** (*string*) – Order-independent MD5 checksum which identifies this *ReferenceSet*.

  To compute this checksum, make a list of *Reference.md5checksum* for all 'Reference's in this set. Then sort that list, and take the MD5 hash of all the strings concatenated together. Express the hash as a lower-case hexadecimal string.
- **`ncbiTaxonId`** (*null|int*) –

  ID from http://www.ncbi.nlm.nih.gov/taxonomy (e.g. 9606->human) indicating the species which this assembly is intended to model. Note that contained *Reference's may specify a different 'ncbiTaxonId*, as assemblies may contain reference sequences which do not belong to the modeled species, e.g. EBV in a human reference genome.
- **`description`** (*null|string*) – Optional free text description of this reference set.
- **`assemblyId`** (*null|string*) – Public id of this reference set, such as *GRCh37*.
- **`sourceURI`** (*null|string*) – Specifies a FASTA format file/string.

- **sourceAccessions** (*array<string>*) –

  All known corresponding accession IDs in INSDC (GenBank/ENA/DDBJ) ideally with a version number, e.g. *NC_000001.11*.

- **isDerived** (*boolean*) –

  A reference set may be derived from a source if it contains additional sequences, or some of the sequences within it are derived (see the definition of *isDerived* in *Reference*).

A *ReferenceSet* is a set of *Reference's which typically comprise a reference assembly, such as 'GRCh38*. A *ReferenceSet* defines a common coordinate space for comparing reference-aligned experimental data.

1.3.8 VariantMethods

searchVariants (*request*)

Parameters **request** – SearchVariantsRequest: This request maps to the body of *POST /variants/search* as JSON.

Return type SearchVariantsResponse

Throws GAException

Gets a list of *Variant* matching the search criteria.

POST /variants/search must accept a JSON version of *SearchVariantsRequest* as the post body and will return a JSON version of *SearchVariantsResponse*.

getCallSet (*id*)

Parameters **id** – string: The ID of the *CallSet*.

Return type org.ga4gh.models.CallSet

Throws GAException

Gets a *CallSet* by ID. *GET /callsets/{id}* will return a JSON version of *CallSet*.

searchVariantSets (*request*)

Parameters **request** – SearchVariantSetsRequest: This request maps to the body of *POST /variantsets/search* as JSON.

Return type SearchVariantSetsResponse

Throws GAException

Gets a list of *VariantSet* matching the search criteria.

POST /variantsets/search must accept a JSON version of *SearchVariantSetsRequest* as the post body and will return a JSON version of *SearchVariantSetsResponse*.

getVariantSet (*id*)

Parameters **id** – string: The ID of the *VariantSet*.

Return type org.ga4gh.models.VariantSet

Throws GAException

Gets a *VariantSet* by ID. *GET /variantsets/{id}* will return a JSON version of *VariantSet*.

getVariant (*id*)

Parameters **id** – string: The ID of the *Variant*.

Return type org.ga4gh.models.Variant

Throws GAException

Gets a *Variant* by ID. *GET /variants/{id}* will return a JSON version of *Variant*.

searchCallSets (*request*)

Parameters **request** – SearchCallSetsRequest: This request maps to the body of *POST /callsets/search* as JSON.

Return type SearchCallSetsResponse

Throws GAException

Gets a list of *CallSet* matching the search criteria.

POST /callsets/search must accept a JSON version of *SearchCallSetsRequest* as the post body and will return a JSON version of *SearchCallSetsResponse*.

error GAException

A general exception type.

enum Strand

Symbols NEG_STRAND|POS_STRAND

Indicates the DNA strand associate for some data item. * *NEG_STRAND*: The negative (-) strand. * *POS_STRAND*: The postive (+) strand.

record Position

Fields

- **referenceName** (*string*) – The name of the *Reference* on which the *Position* is located.
- **position** (*long*) –

  The 0-based offset from the start of the forward strand for that *Reference*. Genomic positions are non-negative integers less than *Reference* length.
- **strand** (Strand) – Strand the position is associated with.

A *Position* is an unoriented base in some *Reference*. A *Position* is represented by a *Reference* name, and a base number on that *Reference* (0-based).

record ExternalIdentifier

Fields

- **database** (*string*) –

  The source of the identifier. (e.g. *Ensembl*)
- **identifier** (*string*) –

  The ID defined by the external database. (e.g. *ENST00000000000*)
- **version** (*string*) –

  The version of the object or the database (e.g. *78*)

Identifier from a public database

enum CigarOperation

Symbols ALIGNMENT_MATCH|INSERT|DELETE|SKIP|CLIP_SOFT|CLIP_HARD|PAD|SEQUENCE_MATCH|SEQU

An enum for the different types of CIGAR alignment operations that exist. Used wherever CIGAR alignments are used. The different enumerated values have the following usage:

- *ALIGNMENT_MATCH*: An alignment match indicates that a sequence can be aligned to the reference without evidence of an INDEL. Unlike the *SEQUENCE_MATCH* and *SEQUENCE_MISMATCH* operators, the *ALIGNMENT_MATCH* operator does not indicate whether the reference and read sequences are an exact match. This operator is equivalent to SAM's *M*.
- *INSERT*: The insert operator indicates that the read contains evidence of bases being inserted into the reference. This operator is equivalent to SAM's *I*.
- *DELETE*: The delete operator indicates that the read contains evidence of bases being deleted from the reference. This operator is equivalent to SAM's *D*.
- *SKIP*: The skip operator indicates that this read skips a long segment of the reference, but the bases have not been deleted. This operator is commonly used when working with RNA-seq data, where reads may skip long segments of the reference between exons. This operator is equivalent to SAM's 'N'.
- *CLIP_SOFT*: The soft clip operator indicates that bases at the start/end of a read have not been considered during alignment. This may occur if the majority of a read maps, except for low quality bases at the start/end of a read. This operator is equivalent to SAM's 'S'. Bases that are soft clipped will still be stored in the read.
- *CLIP_HARD*: The hard clip operator indicates that bases at the start/end of a read have been omitted from this alignment. This may occur if this linear alignment is part of a chimeric alignment, or if the read has been trimmed (e.g., during error correction, or to trim poly-A tails for RNA-seq). This operator is equivalent to SAM's 'H'.
- *PAD*: The pad operator indicates that there is padding in an alignment. This operator is equivalent to SAM's 'P'.
- *SEQUENCE_MATCH*: This operator indicates that this portion of the aligned sequence exactly matches the reference (e.g., all bases are equal to the reference bases). This operator is equivalent to SAM's '='.
- *SEQUENCE_MISMATCH*: This operator indicates that this portion of the aligned sequence is an alignment match to the reference, but a sequence mismatch (e.g., the bases are not equal to the reference). This can indicate a SNP or a read error. This operator is equivalent to SAM's 'X'.

record `CigarUnit`

Fields

- **`operation`** (CigarOperation) – The operation type.
- **`operationLength`** (*long*) – The number of bases that the operation runs for.
- **`referenceSequence`** (*null|string*) –

  ***referenceSequence* is only used at mismatches (*SEQUENCE_MISMATCH*)** and deletions (*DELETE*). Filling this field replaces the MD tag. If the relevant information is not available, leave this field as *null*.

A structure for an instance of a CIGAR operation. *FIXME: This belongs under Reads (only readAlignment refers to this)*

record `VariantSetMetadata`

Fields

- **`key`** (*string*) – The top-level key.
- **`value`** (*string*) – The value field for simple metadata.
- **`id`** (*string*) –

User-provided ID field, not enforced by this API. Two or more pieces of structured metadata with identical id and key fields are considered equivalent. *FIXME: If it's not enforced, then why can't it be null?*

- **type** (*string*) – The type of data.
- **number** (*string*) –

  The number of values that can be included in a field described by this metadata.
- **description** (*string*) – A textual description of this metadata.
- **info** (*map<array<string>>*) – Remaining structured metadata key-value pairs.

Optional metadata associated with a variant set.

record VariantSet

Fields

- **id** (*string*) – The variant set ID.
- **name** (*null|string*) – The variant set name.
- **datasetId** (*string*) – The ID of the dataset this variant set belongs to.
- **referenceSetId** (*string*) – The ID of the reference set that describes the sequences used by the variants in this set.
- **metadata** (*array<VariantSetMetadata>*) –

  Optional metadata associated with this variant set. This array can be used to store information about the variant set, such as information found in VCF header fields, that isn't already available in first class fields such as "name".

A VariantSet is a collection of variants and variant calls intended to be analyzed together.

record CallSet

Fields

- **id** (*string*) – The call set ID.
- **name** (*null|string*) – The call set name.
- **sampleId** (*null|string*) –

  The sample this call set's data was generated from. Note: the current API does not have a rigorous definition of sample. Therefore, this field actually contains an arbitrary string, typically corresponding to the sampleId field in the read groups used to generate this call set.
- **variantSetIds** (*array<string>*) – The IDs of the variant sets this call set has calls in.
- **created** (*null|long*) – The date this call set was created in milliseconds from the epoch.
- **updated** (*null|long*) –

  The time at which this call set was last updated in milliseconds from the epoch.
- **info** (*map<array<string>>*) – A map of additional call set information.

A CallSet is a collection of calls that were generated by the same analysis of the same sample.

record Call

Fields

- **callSetName** (*null|string*) –

The name of the call set this variant call belongs to. If this field is not present, the ordering of the call sets from a *SearchCallSetsRequest* over this *VariantSet* is guaranteed to match the ordering of the calls on this *Variant*. The number of results will also be the same.

- **callSetId** (*null|string*) – The ID of the call set this variant call belongs to.

  If this field is not present, the ordering of the call sets from a *SearchCallSetsRequest* over this *VariantSet* is guaranteed to match the ordering of the calls on this *Variant*. The number of results will also be the same.

- **genotype** (*array<int>*) – The genotype of this variant call.

  A 0 value represents the reference allele of the associated *Variant*. Any other value is a 1-based index into the alternate alleles of the associated *Variant*.

  If a variant had a referenceBases field of "T", an alternateBases value of ["A", "C"], and the genotype was [2, 1], that would mean the call represented the heterozygous value "CA" for this variant. If the genotype was instead [0, 1] the represented value would be "TA". Ordering of the genotype values is important if the phaseset field is present.

- **phaseset** (*null|string*) –

  If this field is not null, this variant call's genotype ordering implies the phase of the bases and is consistent with any other variant calls on the same contig which have the same phaseset string.

- **genotypeLikelihood** (*array<double>*) –

  The genotype likelihoods for this variant call. Each array entry represents how likely a specific genotype is for this call as log10(P(data | genotype)), analogous to the GL tag in the VCF spec. The value ordering is defined by the GL tag in the VCF spec.

- **info** (*map<array<string>>*) – A map of additional variant call information.

A *Call* represents the determination of genotype with respect to a particular *Variant*.

It may include associated information such as quality and phasing. For example, a call might assign a probability of 0.32 to the occurrence of a SNP named rs1234 in a call set with the name NA12345.

record Variant

Fields

- **id** (*string*) – The variant ID.
- **variantSetId** (*string*) –

  The ID of the *VariantSet* this variant belongs to. This transitively defines the *ReferenceSet* against which the *Variant* is to be interpreted.

- **names** (*array<string>*) – Names for the variant, for example a RefSNP ID.
- **created** (*null|long*) – The date this variant was created in milliseconds from the epoch.
- **updated** (*null|long*) –

  The time at which this variant was last updated in milliseconds from the epoch.

- **referenceName** (*string*) –

  The reference on which this variant occurs. (e.g. *chr20* or *X*)

- **start** (*long*) –

The start position at which this variant occurs (0-based). This corresponds to the first base of the string of reference bases. Genomic positions are non-negative integers less than reference length. Variants spanning the join of circular genomes are represented as two variants one on each side of the join (position 0).

- **`end`** (*long*) –

  The end position (exclusive), resulting in [start, end) closed-open interval. This is typically calculated by *start + referenceBases.length*.

- **`referenceBases`** (*string*) – The reference bases for this variant. They start at the given start position.

- **`alternateBases`** (*array<string>*) –

  The bases that appear instead of the reference bases. Multiple alternate alleles are possible.

- **`info`** (*map<array<string>>*) – A map of additional variant information.

- **`calls`** (*array<Call>*) –

  The variant calls for this particular variant. Each one represents the determination of genotype with respect to this variant. *Call's in this array are implicitly associated with this 'Variant*.

A *Variant* represents a change in DNA sequence relative to some reference. For example, a variant could represent a SNP or an insertion. Variants belong to a *VariantSet*. This is equivalent to a row in VCF.

record `SearchVariantSetsRequest`

Fields

- **`datasetId`** (*string*) – The *Dataset* to search.

- **`pageSize`** (*null|int*) –

  Specifies the maximum number of results to return in a single page. If unspecified, a system default will be used.

- **`pageToken`** (*null|string*) –

  The continuation token, which is used to page through large result sets. To get the next page of results, set this parameter to the value of *nextPageToken* from the previous response.

This request maps to the body of *POST /variantsets/search* as JSON.

record `SearchVariantSetsResponse`

Fields

- **`variantSets`** (*array<org.ga4gh.models.VariantSet>*) – The list of matching variant sets.

- **`nextPageToken`** (*null|string*) –

  The continuation token, which is used to page through large result sets. Provide this value in a subsequent request to return the next page of results. This field will be empty if there aren't any additional results.

This is the response from *POST /variantsets/search* expressed as JSON.

record `SearchVariantsRequest`

Fields

- **`variantSetId`** (*string*) – The *VariantSet* to search.

- **callSetIds** (*null|array<string>*) –

  Only return variant calls which belong to call sets with these IDs. If an empty array, returns variants without any call objects. If null, returns all variant calls.

- **referenceName** (*string*) – Required. Only return variants on this reference.

- **start** (*long*) –

  Required. The beginning of the window (0-based, inclusive) for which overlapping variants should be returned. Genomic positions are non-negative integers less than reference length. Requests spanning the join of circular genomes are represented as two requests one on each side of the join (position 0).

- **end** (*long*) –

  Required. The end of the window (0-based, exclusive) for which overlapping variants should be returned.

- **pageSize** (*null|int*) –

  Specifies the maximum number of results to return in a single page. If unspecified, a system default will be used.

- **pageToken** (*null|string*) –

  The continuation token, which is used to page through large result sets. To get the next page of results, set this parameter to the value of *nextPageToken* from the previous response.

This request maps to the body of *POST /variants/search* as JSON.

record SearchVariantsResponse

Fields

- **variants** (*array<org.ga4gh.models.Variant>*) –

  The list of matching variants. If the *callSetId* field on the returned calls is not present, the ordering of the call sets from a *SearchCallSetsRequest* over the parent *VariantSet* is guaranteed to match the ordering of the calls on each *Variant*. The number of results will also be the same.

- **nextPageToken** (*null|string*) –

  The continuation token, which is used to page through large result sets. Provide this value in a subsequent request to return the next page of results. This field will be empty if there aren't any additional results.

This is the response from *POST /variants/search* expressed as JSON.

record SearchCallSetsRequest

Fields

- **variantSetId** (*string*) – The VariantSet to search.

- **name** (*null|string*) – Only return call sets with this name (case-sensitive, exact match).

- **pageSize** (*null|int*) –

  Specifies the maximum number of results to return in a single page. If unspecified, a system default will be used.

- **pageToken** (*null|string*) –

The continuation token, which is used to page through large result sets. To get the next page of results, set this parameter to the value of *nextPageToken* from the previous response.

This request maps to the body of *POST /callsets/search* as JSON.

record `SearchCallSetsResponse`

Fields

- **`callSets`** (*array<org.ga4gh.models.CallSet>*) – The list of matching call sets.
- **`nextPageToken`** (*null|string*) –

  The continuation token, which is used to page through large result sets. Provide this value in a subsequent request to return the next page of results. This field will be empty if there aren't any additional results.

This is the response from *POST /callsets/search* expressed as JSON.

1.3.9 Variants

This file defines the objects used to represent variant calls, most importantly VariantSet, Variant, and Call. See {TODO: LINK TO VARIANTS OVERVIEW} for more information.

enum `Strand`

Symbols NEG_STRAND|POS_STRAND

Indicates the DNA strand associate for some data item. * *NEG_STRAND*: The negative (-) strand. * *POS_STRAND*: The postive (+) strand.

record `Position`

Fields

- **`referenceName`** (*string*) – The name of the *Reference* on which the *Position* is located.
- **`position`** (*long*) –

  The 0-based offset from the start of the forward strand for that *Reference*. Genomic positions are non-negative integers less than *Reference* length.
- **`strand`** (Strand) – Strand the position is associated with.

A *Position* is an unoriented base in some *Reference*. A *Position* is represented by a *Reference* name, and a base number on that *Reference* (0-based).

record `ExternalIdentifier`

Fields

- **`database`** (*string*) –

  The source of the identifier. (e.g. *Ensembl*)
- **`identifier`** (*string*) –

  The ID defined by the external database. (e.g. *ENST00000000000*)
- **`version`** (*string*) –

  The version of the object or the database (e.g. *78*)

Identifier from a public database

enum `CigarOperation`

Symbols ALIGNMENT_MATCH|INSERT|DELETE|SKIP|CLIP_SOFT|CLIP_HARD|PAD|SEQUENCE_MATCH|SEQU

An enum for the different types of CIGAR alignment operations that exist. Used wherever CIGAR alignments are used. The different enumerated values have the following usage:

- *ALIGNMENT_MATCH*: An alignment match indicates that a sequence can be aligned to the reference without evidence of an INDEL. Unlike the *SEQUENCE_MATCH* and *SEQUENCE_MISMATCH* operators, the *ALIGNMENT_MATCH* operator does not indicate whether the reference and read sequences are an exact match. This operator is equivalent to SAM's *M*.
- *INSERT*: The insert operator indicates that the read contains evidence of bases being inserted into the reference. This operator is equivalent to SAM's *I*.
- *DELETE*: The delete operator indicates that the read contains evidence of bases being deleted from the reference. This operator is equivalent to SAM's *D*.
- *SKIP*: The skip operator indicates that this read skips a long segment of the reference, but the bases have not been deleted. This operator is commonly used when working with RNA-seq data, where reads may skip long segments of the reference between exons. This operator is equivalent to SAM's 'N'.
- *CLIP_SOFT*: The soft clip operator indicates that bases at the start/end of a read have not been considered during alignment. This may occur if the majority of a read maps, except for low quality bases at the start/end of a read. This operator is equivalent to SAM's 'S'. Bases that are soft clipped will still be stored in the read.
- *CLIP_HARD*: The hard clip operator indicates that bases at the start/end of a read have been omitted from this alignment. This may occur if this linear alignment is part of a chimeric alignment, or if the read has been trimmed (e.g., during error correction, or to trim poly-A tails for RNA-seq). This operator is equivalent to SAM's 'H'.
- *PAD*: The pad operator indicates that there is padding in an alignment. This operator is equivalent to SAM's 'P'.
- *SEQUENCE_MATCH*: This operator indicates that this portion of the aligned sequence exactly matches the reference (e.g., all bases are equal to the reference bases). This operator is equivalent to SAM's '='.
- *SEQUENCE_MISMATCH*: This operator indicates that this portion of the aligned sequence is an alignment match to the reference, but a sequence mismatch (e.g., the bases are not equal to the reference). This can indicate a SNP or a read error. This operator is equivalent to SAM's 'X'.

record `CigarUnit`

Fields

- **`operation`** (CigarOperation) – The operation type.
- **`operationLength`** (*long*) – The number of bases that the operation runs for.
- **`referenceSequence`** (*null|string*) –

  ***referenceSequence* is only used at mismatches (*SEQUENCE_MISMATCH*)** and deletions (*DELETE*). Filling this field replaces the MD tag. If the relevant information is not available, leave this field as *null*.

A structure for an instance of a CIGAR operation. *FIXME: This belongs under Reads (only readAlignment refers to this)*

record `VariantSetMetadata`

Fields

- **`key`** (*string*) – The top-level key.
- **`value`** (*string*) – The value field for simple metadata.

- **id** (*string*) –

  User-provided ID field, not enforced by this API. Two or more pieces of structured metadata with identical id and key fields are considered equivalent. *FIXME: If it's not enforced, then why can't it be null?*

- **type** (*string*) – The type of data.
- **number** (*string*) –

  The number of values that can be included in a field described by this metadata.

- **description** (*string*) – A textual description of this metadata.
- **info** (*map<array<string>>*) – Remaining structured metadata key-value pairs.

Optional metadata associated with a variant set.

record VariantSet

Fields

- **id** (*string*) – The variant set ID.
- **name** (*null|string*) – The variant set name.
- **datasetId** (*string*) – The ID of the dataset this variant set belongs to.
- **referenceSetId** (*string*) – The ID of the reference set that describes the sequences used by the variants in this set.
- **metadata** (*array<VariantSetMetadata>*) –

  Optional metadata associated with this variant set. This array can be used to store information about the variant set, such as information found in VCF header fields, that isn't already available in first class fields such as "name".

A VariantSet is a collection of variants and variant calls intended to be analyzed together.

record CallSet

Fields

- **id** (*string*) – The call set ID.
- **name** (*null|string*) – The call set name.
- **sampleId** (*null|string*) –

  The sample this call set's data was generated from. Note: the current API does not have a rigorous definition of sample. Therefore, this field actually contains an arbitrary string, typically corresponding to the sampleId field in the read groups used to generate this call set.

- **variantSetIds** (*array<string>*) – The IDs of the variant sets this call set has calls in.
- **created** (*null|long*) – The date this call set was created in milliseconds from the epoch.
- **updated** (*null|long*) –

  The time at which this call set was last updated in milliseconds from the epoch.

- **info** (*map<array<string>>*) – A map of additional call set information.

A CallSet is a collection of calls that were generated by the same analysis of the same sample.

record Call

Fields

- **callSetName** (*null|string*) –

  The name of the call set this variant call belongs to. If this field is not present, the ordering of the call sets from a *SearchCallSetsRequest* over this *VariantSet* is guaranteed to match the ordering of the calls on this *Variant*. The number of results will also be the same.

- **callSetId** (*null|string*) – The ID of the call set this variant call belongs to.

  If this field is not present, the ordering of the call sets from a *SearchCallSetsRequest* over this *VariantSet* is guaranteed to match the ordering of the calls on this *Variant*. The number of results will also be the same.

- **genotype** (*array<int>*) – The genotype of this variant call.

  A 0 value represents the reference allele of the associated *Variant*. Any other value is a 1-based index into the alternate alleles of the associated *Variant*.

  If a variant had a referenceBases field of "T", an alternateBases value of ["A", "C"], and the genotype was [2, 1], that would mean the call represented the heterozygous value "CA" for this variant. If the genotype was instead [0, 1] the represented value would be "TA". Ordering of the genotype values is important if the phaseset field is present.

- **phaseset** (*null|string*) –

  If this field is not null, this variant call's genotype ordering implies the phase of the bases and is consistent with any other variant calls on the same contig which have the same phaseset string.

- **genotypeLikelihood** (*array<double>*) –

  The genotype likelihoods for this variant call. Each array entry represents how likely a specific genotype is for this call as log10(P(data | genotype)), analogous to the GL tag in the VCF spec. The value ordering is defined by the GL tag in the VCF spec.

- **info** (*map<array<string>>*) – A map of additional variant call information.

A *Call* represents the determination of genotype with respect to a particular *Variant*.

It may include associated information such as quality and phasing. For example, a call might assign a probability of 0.32 to the occurrence of a SNP named rs1234 in a call set with the name NA12345.

record Variant

Fields

- **id** (*string*) – The variant ID.

- **variantSetId** (*string*) –

  The ID of the *VariantSet* this variant belongs to. This transitively defines the *ReferenceSet* against which the *Variant* is to be interpreted.

- **names** (*array<string>*) – Names for the variant, for example a RefSNP ID.

- **created** (*null|long*) – The date this variant was created in milliseconds from the epoch.

- **updated** (*null|long*) –

  The time at which this variant was last updated in milliseconds from the epoch.

- **referenceName** (*string*) –

  The reference on which this variant occurs. (e.g. *chr20* or *X*)

- **start** (*long*) –

  The start position at which this variant occurs (0-based). This corresponds to the first base of the string of reference bases. Genomic positions are non-negative integers less than reference length. Variants spanning the join of circular genomes are represented as two variants one on each side of the join (position 0).

- **end** (*long*) –

  The end position (exclusive), resulting in [start, end) closed-open interval. This is typically calculated by *start* + *referenceBases.length*.

- **referenceBases** (*string*) – The reference bases for this variant. They start at the given start position.

- **alternateBases** (*array<string>*) –

  The bases that appear instead of the reference bases. Multiple alternate alleles are possible.

- **info** (*map<array<string>>*) – A map of additional variant information.

- **calls** (*array<Call>*) –

  The variant calls for this particular variant. Each one represents the determination of genotype with respect to this variant. *Call's in this array are implicitly associated with this 'Variant*.

A *Variant* represents a change in DNA sequence relative to some reference. For example, a variant could represent a SNP or an insertion. Variants belong to a *VariantSet*. This is equivalent to a row in VCF.

1.4 Appendix

1.4.1 Apache Avro

Apache Avro is a data serialization ecosystem, comparable to Google's Protocol Buffers.

What does the GA4GH web API take from Avro?

The GA4GH web API uses the Avro IDL (aka AVDL) language and JSON serialization labraries.

The GA4GH web API presents a simple HTTP(S) and JSON interface to clients. It does **not** use Avro's binary serialization format, or Avro's built-in client/server networking and RPC features.

How does the GA4GH web API use Avro schemas?

GA4GH web API objects, including both the data objects actually exchanged and the control messages requesting and returning those objects, are defined in the Avro IDL language, AVDL.

The full documentation for the AVDL language is abvailable here. Bear in mind that the Avro IDL comes with an entire ecosystem; the GA4GH web APIs do not use most of it.

How does the GA4GH Web API use AVDL?

The GA4GH web API schemas are broken up into multiple AVDL files, which reference each other. Each file defines a number of types (mostly Avro Records, with a smattering of Avro Enums), grouped into a "protocol" (which is somewhat of a misnomer) of types defining a facet of the API. Mostly, the files come in pairs: a normal AVDL file defining the types representing actual data, and a "methods" AVDL file defining the control messages to be sent back and forth to query and exchange the representational types, and the URLs associated with various operations.

Each type has a leading comment documenting its purpose, and each field in the type has a description. These are included in the automatically generated API documentation.

Here is an example of an AVDL definition from, in this case defining a genomic *Position* type which is used across the API:

```
/**
A `Position` is an unoriented base in some `Reference`. A `Position` is
represented by a `Reference` name, and a base number on that `Reference`
(0-based).
*/
record Position {
  /**
  The name of the `Reference` on which the `Position` is located.
  */
  string referenceName;

  /**
  The 0-based offset from the start of the forward strand for that `Reference`.
  Genomic positions are non-negative integers less than `Reference` length.
  */
  long position;

  /**
  Strand the position is associated with.
  */
  Strand strand;
}
```

This is a "record", which contains three fields. All of the fields are required to be filled in, and all of the fields can only hold objects of a particular single type. (In cases where this is not desired, see the AVDL documentation on unions). The last field holds a *Strand* object, which is defined elsewhere in the file.

A note on unions and optional fields

Any field which is optional should be defined as a `union<null, ActualType>`, and given a default value of `null`. Note that `null` should always be first in the union, since it is the type of the default value.

The Avro JSON libraries serialize union types strangely, so the GA4GH API schemas have been specifically designed never to include union types that would trigger this behavior. The upshot of this is that the **only** legal union type is `union<null, ActualType>`. Unions with multiple non-`null` types are not allowed.

Todo

- How much of the AVDL tutorial do we want in here?
- Document/show an example for methods (request and response pairing pattern)
- Talk about how we manually specify that some things land in URLs

---

1.4.2 Glossary

A compendium of bioinformatics terminology commonly used across the API.

Short Reads and BAM

High throughput genome and transcriptome sequencing produces millions of short (50-200 nucleotide) sequences. These sequences are usually referred to as reads. Reads can be produced from:

1. Complete genomes. These reads can be used to piece together the full genome of an individual.
2. Exomes. These reads are derived from just the gene regions in the genome (in humans this is a reduction of >97%)
3. Transcriptomes. Here, the RNA that gets transcribed from the genomic DNA is sequenced, representing only the genes that are active in the tissue that was sampled. Transcriptomes differ from tissue to tissue and can be used to determine differences between tumors and their surrounding tissues.

Reads are usually mapped to a reference genome, for example GRCh37 in humans.

These alignments can be displayed like so:

```
ID      CHROM  POS    CIGAR   SEQUENCE
read1   chr1   234    10M     GACAGTCCCA
read2   chr14  1456   10M     AAAGATTGAC
read3   chrX   2837   7M2I1M  TGGGACTCTA
```

In this format, the CIGAR string shows how well the read matches the genome: read1 is identical to the genome sequence over all its 10 bases: 10M. The first seven bases of read3 match the genome (7M), but then it has a 2 base insertion (2I), followed by another 1 base match (1M).

The SAM/BAM Format is a way of representing read data. It includes the fields shown above as well as information on read orientation, sequence quality, and optional fields. The format also allows for reads that do not align to the genome, by leaving the reference sequence ID, position, and cigar fields empty.

SAM is a human readable format, BAM is a condensed binary format. The formats can be readily converted to each other.

Genetic variants and VCF

Genetic variants are differences in the genome sequence from one individual to the next. Such variation can manifest at different scales, from small changes affecting just one or a few DNA base pairs, to copy number variations of whole exons or genes, to large structural variations affecting megabases or more. The GA4GH Variants schema focuses on small variants for now, because there's less consensus on how to represent the larger kinds of variation.

Small genetic variants can be represented as edits to a Reference sequence: typically a tuple of (1) Reference sequence name, (2) starting position of the affected portion on the Reference sequence, (3) DNA sequence of the affected portion of the Reference, and (4) alternate DNA sequence found in place of the Reference sequence. Both the reference and alternate sequences are provided in order to represent sequence insertions and deletions (indels). A few examples:

```
CHROM  POS     REF  ALT
20     14370   G    A
20     17330   TA   T
20     18302   TG   ACC
```

The first variant is a single-nucleotide substitution of G to A. The second variant is a deletion of an A at position 17,331. The third variant is a multi-nucleotide change to a lengthier sequence starting at position 18,302.

Given a list of such variants, we can specify the genotype of one or more individuals with respect to each variant. The genotype of a diploid individual (for an autosomal variant) may take one of three distinct values: homozygous reference (0,0), heterozygous (0,1), or homozygous alternate (1,1). We can then present a matrix of genotypes, where the rows are variants as shown above, the columns are the individuals, and each entry is one of those three genotype *calls* (or marked missing):

```
CHROM  POS     REF  ALT   Alice   Bob
20     14370   G    A     (0,0)   (0,1)
20     17330   TA   T     (0,0)   (1,1)
20     18302   TG   ACC   (0,1)     -
```

(If the phase of an individual's genotypes across several variant positions is known, then the heterozygous genotypes (0,1) and (1,0) may be considered distinct, where the order specifies which homologous chromosome possesses the alternate sequence.)

It's possible to observe multiple different alternate sequences, or *alleles*, affecting the same portion of the reference. This can occur even within one individual, if their two homologous chromosomes contain different alternate sequences, and becomes somewhat common when representing variants observed across a population. To handle these cases, we allow a variant to specify multiple alternate alleles. For example:

```
CHROM  POS     REF  ALT
20     19254   G    A,C,T
20     21672   AT   AC,TGA
```

And in this case the genotypes can take values such as (0,3) or (1,2). This multi-allelic sites model was refined and popularized in the 1000 Genomes Project's Variant Call Format (VCF), upon which the Variants schema is based.

There remain some outstanding challenges with this model of small variants. For example, the same edit to the reference sequence can be represented in multiple ways. There are also different ways to represent clusters of alleles that affect overlapping but non-equal portions of the reference. The GA4GH doesn't yet prescribe resolutions to these ambiguities, and different conventions are used in practice.

(TODO possible additional/advanced topics: homozygous ref vs. no-call; phasing and phase sets; genotype likelihoods; INFO, FORMAT, QUAL, FILTER)

1.4.3 The JSON Format

JSON, or JavaScript Object Notation, is officially defined here. It is the standard data interchange format for web APIs.

The GA4GH Web API uses a JSON wire protocol, exchanging JSON representations of the objects defined in its AVDL schemas. More information on the AVDL schemas is available in *Apache Avro*; basically, the AVDL type definitions say what attributes any given JSON object ought to have, and what ought to be stored in each of them.

GA4GH JSON Serialization

The GA4GH web APIs use Avro IDL to define their schemas, and use the associated Avro JSON serialization libraries. Since the schemas use a restricted subset of AVDL types (see *A note on unions* below), the serialized JSON format is fairly standard. This means that standard non-Avro JSON serialization and deserialization libraries (like, for example, the Python `json` module) can be used to serialize and deserialize GA4GH JSON messages in an idiomatic way.

Serialization example

For example, here is the schema definition for Variants (with comments removed):

```
record Variant {
  string id;
  string variantSetId;
  array<string> names = [];
  union { null, long } created = null;
  union { null, long } updated = null;
  string referenceName;
  long start;
  long end;
  string referenceBases;
  array<string> alternateBases = [];
  map<array<string>> info = {};
  array<Call> calls = [];
}
```

Here is a serialized variant in JSON. It's a bit of an edge case in some respects:

```
{
    "id": "gv79384-3200-11",
    "variantSetId": "vs-44-1",
    "names": [
        "rs110",
        "Victoria"
    ],
    "created": 1446842841,
    "updated": null,
    "start": 1000,
    "end": 1001,
    "referenceBases": "A",
    "alternateBases": [
        "C",
        "CTATCTT"
    ],
    "info": {
        "variantfacts": ["is_interesting", "is_long"],
        "numberOfPapers": ["11"]
    },
}
```

Things to notice:

- A serialized record contains no explicit information about its type.
- Arrays are serialized as JSON arrays.
- Maps are serialized as JSON objects.
- Records are also serialized as JSON objects.
- Enums (not shown here) are serialized as JSON strings.
- Nulls are serialized as JSON nulls.
- Fields with default values may be omitted (see the lack of an `updated` or `calls`) as a way of serializing their default values.
- Unions of `null` and a non-`null` type are serialized as either `null` or the serialized non-null value. No other kinds of unions are present or permitted.

A note on unions

As noted above, a field with union type serialized in GA4GH JSON looks no different from a field of any other type: you just put the field name and its recursively serialized value. In order for the Avro JSON libraries to support this, it is necessary that AVDL `union` types union together only `null` and a single non-`null` type. If there were two or more non-`null` types, the Avro libraries would need to include additional type information to say which to use when deserializing. Since we prohibit those unions, however, API clients and alternative server implementations never need to worry about this additional type information or its syntax. They can just handle "normal" JSON.

Todo

- add example of Python decoder output
- create a python class, if necessary

Wire protocol example

This is from the ga4gh server example.

To get information from the readgroupsets on a server, create a JSON format request:

```
{
  "datasetIds":[],
  "name":null
}
```

Note: What is this actually asking?

To send this to the server, we need to create a HTTP request which tells the server what type of data to expect (JSON format, in this case) In our test case, we have a server running at http://localhost:8000

Since we want to query the readgroupsets, we'll have to make that part of the URL

Note:

- How do we know it's v0.5.1?
- where is the readgroupsets/search part documented or defined?

To create a command line request, we can use cURL:

```
curl --data '{"datasetIds":[], "name":null}' --header 'Content-Type: application/json' http://localho
```

The server returns:

```
{
"nextPageToken": null,
"readGroupSets": [{
"readGroups": [{
"info": {},
"updated": 1432287597662,
"predictedInsertSize": null,
"description": null,
"created": 1432287597662,
"programs": [],
"sampleId": null,
"experiment": null,
"referenceSetId": null,
```

```
"id":
"low-coverage:HG00533.mapped.ILLUMINA.bwa.CHS.low_coverage.20120522",
"datasetId": null,
"name":
"low-coverage:HG00533.mapped.ILLUMINA.bwa.CHS.low_coverage.20120522"
},
{   "info": {},
"updated": 1432287793946,
"predictedInsertSize": null,
"description": null,
"created": 1432287793946,
"programs": [],
"sampleId": null,
"experiment": null,
"referenceSetId": null,
"id":
"low-coverage:HG00096.mapped.ILLUMINA.bwa.GBR.low_coverage.20120522",
"datasetId": null,
"name":
"low-coverage:HG00096.mapped.ILLUMINA.bwa.GBR.low_coverage.20120522"
},
{    "info": {},
"updated": 1432287793946,
"predictedInsertSize": null,
"description": null,
"created": 1432287793946,
"programs": [],
"sampleId": null,
"experiment": null,
"referenceSetId": null,
"id":
"low-coverage:HG00534.mapped.ILLUMINA.bwa.CHS.low_coverage.20120522",
"datasetId": null,
"name":
"low-coverage:HG00534.mapped.ILLUMINA.bwa.CHS.low_coverage.20120522"
}],
"id":
"low-coverage",
"datasetId": null,
"name": null
}
]
}
```