This document is stale.

The VMC project was incorporated into GA4GH and renamed to Variation Representation.

The scope of this document was narrowed significantly for the 1.0 release, which was approved by GA4GH in Sep 2019.

The primary repository is https://github.com/ga4gh/vr-spec/, which contains the machine readable schema and links to current documentation.

VMC {{version}}: Variation Modelling Collaboration Data Model and Specifications

Variation Modelling Collaboration Working Group

https://github.com/ga4gh/vmc

This document, written by a partnership among national information resource providers, major public initiatives, and diagnostic testing laboratories, proposes a specification to standardize the exchange of biological sequence variation data.

How to Provide Feedback

Readers are invited to submit comments or feedback on the document itself or through the public discussion forum at http://bit.ly/vmc-discuss. Discussion and announcements will occur on this forum. Your initial comments will be appreciated by {{}}.

The "development" version of the VMC Specification contains the document history and background discussions regarding the complex issues that arose during the development of this specification.

Authors

Chair

Reece Hart <reece@genomemedical.com>, Genome Medical, Invitae, GA4GH

Contributing Authors

Gil Alterovitz, Harvard Medical School/Boston Children's Hospital, FHIR Genomics, GA4GH
Larry Babb, Sunquest/GeneInsight, ClinGen, HL7/FHIR, GA4GH
Karen Eilbeck, University of Utah
Robert Freimuth, Mayo Clinic, ClinGen, HL7/FHIR, GA4GH
Sarah Hunt, European Molecular Biology Laboratory, European Bioinformatics Institute, GA4GH
David Kreda, Consultant to Harvard Medical School DBMI, HL7/FHIR
Jennifer Lee, NCBI/NLM/NIH, ClinVar
Peter Robinson, Jackson Laboratory, HPO
Shawn Rynearson, University of Utah, UCGD

Support

Global Alliance for Genomics and Health
Genome Medical, Inc.
Invitae, Inc.
National Institute of Health NHGRI R01HG008628 to Karen Eilbeck
NIH ClinGen Resource grant U41 HG006834-01A1 supported LB and RF travel
Boston Children's Hospital and Massachusetts Institute of Technology for Gil Alterovitz

September 25, 2017

Dear Colleagues:

While there are several notable successes in sharing genomic variation data, including dbSNP, ClinVar, COSMIC, and DECIPHER, there has been no consensus for the representation of an allele, perhaps the most foundational concept for sharing genomic information, or more complex structures such as haplotypes and genotypes. As a result, the research and medical community's ability to share genetic information broadly has been hampered.

The Variant Modelling Collaboration (VMC) has spent the past 15 months developing **the VMC 0.1 Data Model and Specification to standardize the terminology and exchange of allele, haplotype, and genotype data**. Our initial work covers representing sequences at precise locations as discovered through high-throughput sequencing, which covers the high volume use case. Out of the gate, the specification addresses exact identity matching for represented alleles, haplotypes, and genotypes, which lays the requisite foundation for supporting more complex notions of equivalence. In the specification we also propose and demonstrate an algorithm to compute universally unique ids for modelled alleles, haplotypes, and genotypes, the use of which would remove the need for prior negotiation or assignment of ids by central authorities and is particularly useful in assigning globally unique identifiers to novel variation. In the future, we plan to extend the specification to cover structural variants, variants at approximate locations, copy number variants, mosaicism, and chimerism.

We believe the VMC specification merits your timely consideration. Comments can be added to the Google doc for VMC 0.1 (bit.ly/vmc-0-1) or via the discussion forum at bit.ly/vmc-discuss. Developers can find specification-compliant source code at bit.ly/vmc-github and notebook examples at bit.ly/vmc-notebooks.

We welcome comments at any time via the vmc-discuss group. Feedback received by October 6, 2017 will be considered for inclusion in the next release of the VMC specification and prior to broader dissemination.

We look forward to hearing from you!

Very truly yours,

Reece Hart
on behalf of the Variant Modelling Collaboration

Summary

Maximizing the personal, public, research, and clinical value of genomic information will require that clinicians, researchers, and testing laboratories exchange genetic variation data reliably. This document, written by a partnership among national information resource providers, major public initiatives, and diagnostic testing laboratories, proposes a specification to standardize the exchange of variation data. Four contributions are made:

- **Terminology**. Definitions for biological terms may be abstract or intentionally ambiguous, often accurately reflecting scientific uncertainty or understanding at the time. Abstract and ambiguous terms are not readily translatable into a representation of knowledge. Therefore, the specification begins with precise computational definitions for biological concepts that are essential to representing sequence variation.
- **Information model.** The VMC information model specifies how the computational definitions are to be represented in terms of fields, semantics, objects, and object relationships.
- **Machine readable schema definition.** To be useful for information exchange, the information model should be realized in a schema definition language. VMC is currently implemented using JSON Schema. However, the construction of VMC 1 is intended to support translations to other schema systems, such as XML, OpenAPI, and GraphQL.
- **Globally unique computed identifiers.** This specification also recommends a specific algorithm for constructing distributed and globally-unique identifiers for genetic states. Importantly, this algorithm enables data providers and consumers to computationally generate consistent, globally unique identifiers for variation without a central authority.

The machine readable schema definitions and example code are available at https://github.com/ga4gh/vmc.

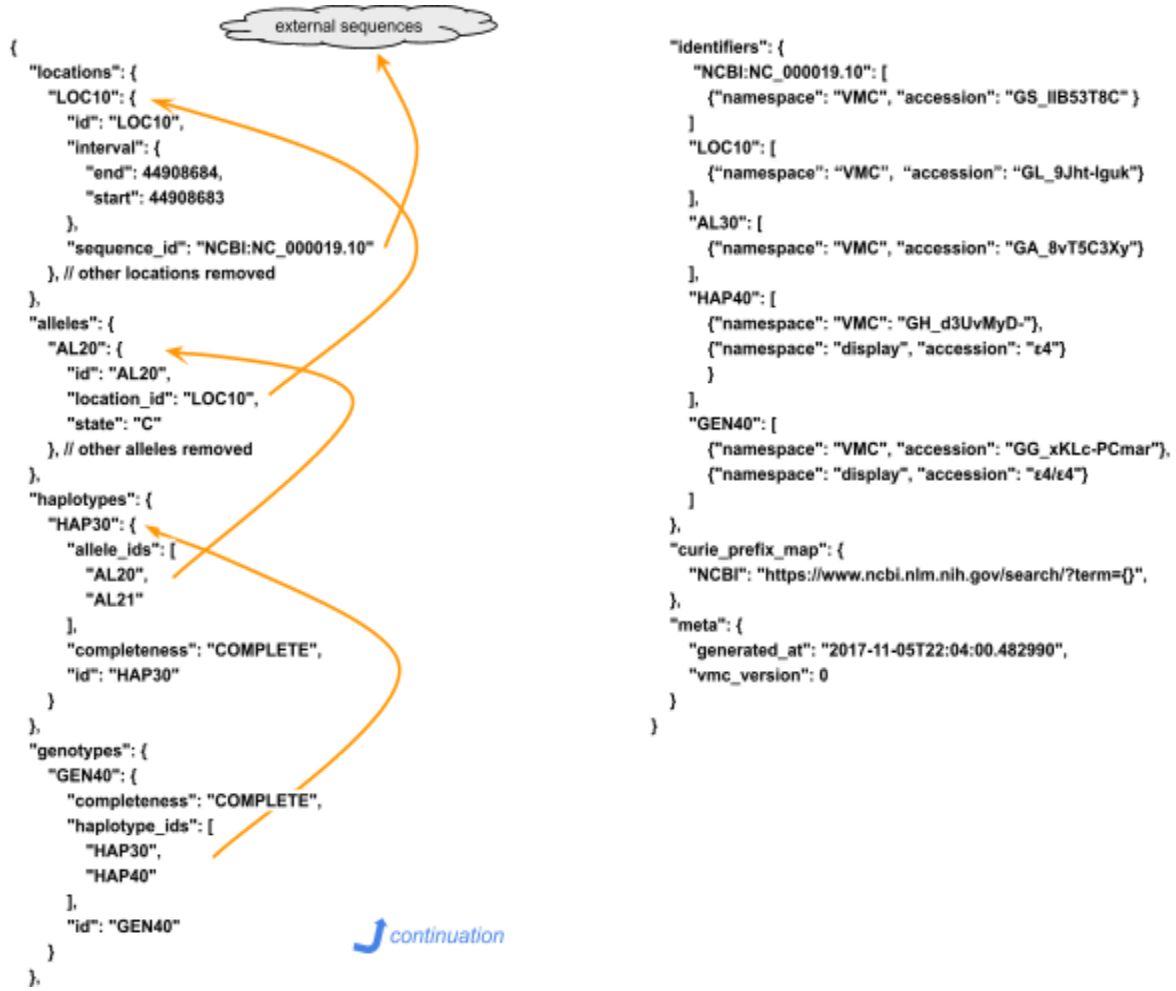
The initial scope of VMC is purposefully limited to provide only the most essential elements for contemporary needs for next generation sequencing. This document includes a roadmap for adoption by key stakeholders, recognized functional gaps in the current specification, and a recommendation for the governance of the specification that promotes technical collaboration and, therefore, long-term benefit to human health through accurate data exchange.

Table of Contents

VMC in One Page

The VMC specification is dense. Before we get into details, let’s whet your appetite with the big picture.

A “VMC Bundle” is a coherent aggregate of sequence locations, alleles, haplotypes, genotypes, and the identifiers associated with each of these. Internal ids may be from any source.. Locations are defined on sequences, which are expected to be defined outside of VMC by authoritative sources. Alleles are defined as assertions of sequence possibly unchanged, or a deletion, insertion, or replacement at a location. Haplotypes are defined as a set of Alleles on the same sequence. Genotypes are defined as a set of Haplotypes. A globally unique Computed Identifier for any object using the VMC Digest, obviating a central authority; a VMC Bundle must provide external identifiers for all objects and must include the VMC Computed Identifier.



Introduction

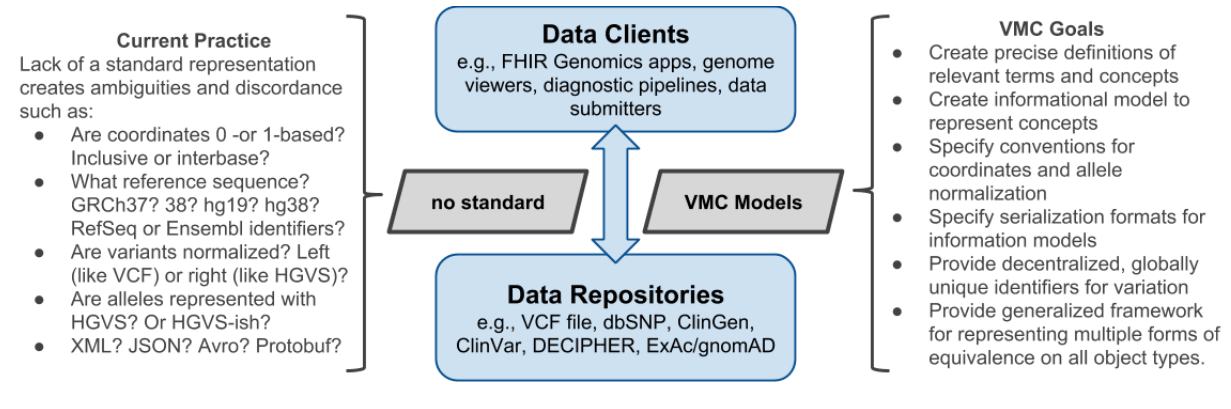
Ensuring that precision genomic medicine is effective for individuals and for health systems will require that clinicians, researchers, and testing laboratories communicate genomic variation[1] and related information reliably. Although multiple academic and industry organizations are developing tools to manipulate and communicate data associated with alleles, thus far these efforts have been independent. There are several notable successes in sharing variation data, such as dbSNP, ClinGen, ClinVar, COSMIC, and DECIPHER. Despite these efforts, there is currently no consensus for the representation of an allele[2], perhaps the most foundational concept for sharing genomic information. As a result, the ability to share genetic information broadly among a loosely connected network of data providers and consumers is uncertain.

The Variation Modelling Collaboration (VMC) data model provides conventions and nomenclature that facilitate reliable, accurate, and efficient communication of biological sequence variation data. The model is primarily focused on representing variation with precise locations on specific DNA, RNA, and protein sequences. Although motivated by applications to human health, the model applies to species with any ploidy. This document describes the model and provides implementation guidance for its use.

The VMC data model defines representations for use at the interfaces *between computer systems*. The model is *not* intended to prescribe how “back-end” software services are designed or how “front-end” systems present data to human users. For example, while the specification uses interbase coordinates for information exchange, user interfaces should likely continue to use conventional one-based inclusive numbering.

The VMC data model does not specify a complete transfer format. Instead, it is intended to be used within the context of a broader complete data model or serialization format. For example, a clinical laboratory delivering variation data to an electronic health record (EHR) would need to structure variation data within the context of a larger construct; the model proposes how the variation data should be structured.

[1] “Variation” refers abstractly to all forms of biological sequence changes. It is distinct from “variant”, which refers to one such specific change.
[2] For reasons discussed in detail later in this document, “allele” is preferred over “variant”. Although we make conceptual distinctions between these terms later in the specification, readers may read these terms interchangeably for now. The rationale for preferring “allele” is in the appendix.



VMC Context Currently, there is no standard computational interface between repositories and consumers of variation data. This creates unnecessary heterogeneity that makes software development inefficient at best and unreliable at worst. The narrow goal for the VMC is to define how systems exchange alleles, haplotypes, and genotypes by defining relevant terms and creating a simple data model using those terms. [figure source]

In order to anticipate implementations in multiple computer languages, VMC distinguishes the logical model (*i.e.*, objects and typed data elements) from the physical model used for communications. For example, the logical model specifies the names and types of fields that might be concurrently realized in JSON, GraphQL, XML, protobuf, or a tab-separated value file. While the VMC envisions many physical representations and is unbiased with respect to them, it *does* prescribe conventions regarding element names, types, and semantics.

The VMC data model is minimalistic: entities include only the data essential to an object. All annotations, such as alternative names for a haplotype or population frequencies of an allele, are associated with VMC entities by way of an identifier.

Object identification is critical to establishing relationships between objects within a single system and between systems. VMC's approach to identifiers follows common practice and adopts the terminology for object identification from the HL7 FHIR specification[3]. In particular, VMC distinguishes the *logical identifier*, which is a private identifier used to establish relationships *within* a system, and *business identifier*, which is used for communication *between* systems. In VMC, business identifiers are represented as URIs, which are unambiguously interconvertible with FHIR identifiers (see Identifier).

The VMC specification defines and requires the use of globally-unique computed identifiers, based on the VMC Digest, that are generated from object data. Computed identifiers enable the collation of variation and annotations from multiple sources *without prior negotiation, a crucial capability for eliminating both a practical bottleneck and a source of error in exchanging data between systems*[4]. Computed identifiers work for shared and novel variation, thus providing a

[3] http://build.fhir.org/resource.html#id
[4] Computed identifiers, when used as internal identifiers, also facilitate deduplication and identity errors within a system.

suitable replacement for identifiers within a system. Computed identifiers may be used as logical (internal) identifiers, business (external) identifiers, or both. In addition, because computed identifiers are based on the underlying data, the technique is readily extensible to future versions of VMC that will support, for example, translocations and inversions, or coordinate systems more complex than that provided in the current version of the VMC specification.

Relationship of VMC specification to existing standards

Because a primary objective of the VMC effort is to unify disparate efforts, it is important to describe how this document relates to previous work in order to avoid “reinventing the wheel”.

- The Variant Call Format (VCF) is the de facto standard for representing alleles, particularly for use during primary analysis in high-throughput sequencing pipelines. VCF permits a wide range of annotations on alleles, such as quality and likelihood scores. VCF is a file-based format and is exclusively for genomic alleles. In contrast, the VMC data model abstractly represents Alleles, Haplotypes, and Genotypes on all sequence types, is independent of medium, and is well-suited to secondary analyses, allele interpretation, aggregation, and system-level interoperability.
- The HGVS nomenclature recommendations describe how sequence variation should be *presented to human beings*. In addition to representing a wide variety of sequence changes from single residue variation through large cytogenetic events, HGVS attempts to also encode in strings notions of biological mechanism (e.g., inversion as a kind of deletion-insertion event), predicted events (e.g., parentheses for computing protein sequence), and complex states (e.g., mosaicism). In practice, HGVS recommendations are difficult to implement fully and consistently, leading to ambiguity in presentation. In contrast, the VMC is a formal specification that improves consistency of *representation among computer systems*. VMC is currently less expressive than HGVS for rarer cases of variation, such as cytogenetic variation or context-based allele representations (e.g., insT written as dupT when the insertion follows a T). Future versions of the specification will seek to address limitations while preserving principles of conceptual clarity and precision.

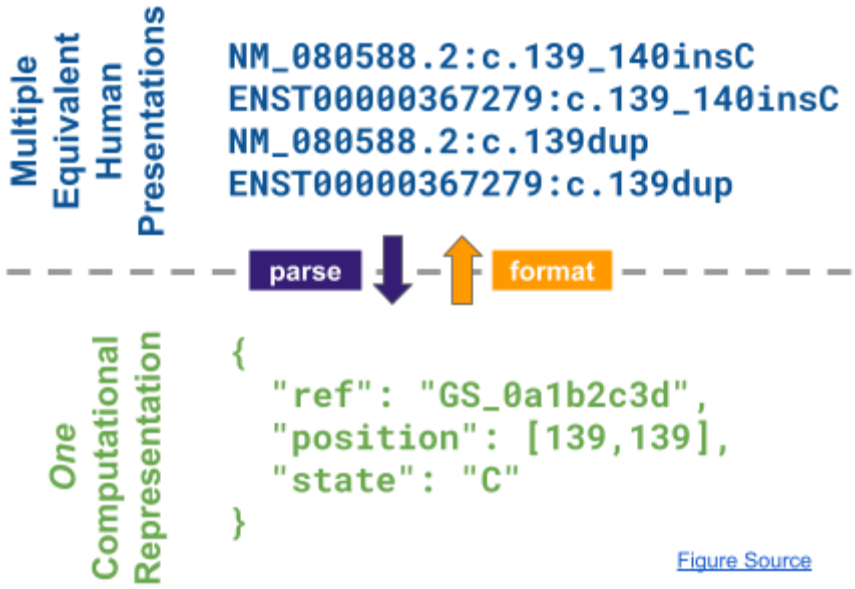


Figure Source

- The Sequence Ontology (SO) is a set of terms and relationships used to describe the features and attributes of biological sequence. The focus of the SO has been the annotation of, or placement of 'meaning', onto genomic sequence regions. The VMC effort seeks to use the same descriptive definitions where possible, and to inform the refinement of SO.
- The Genotype Ontology (GENO) builds on the SO to include richer modeling of genetic variation at different levels of granularity that are captured in genotype representations. Unlike the SO which is used primarily for annotation of genomic features, GENO was developed by the Monarch Initiative to support semantic data models for integrated representation of genotypes and genetic variants described in human and model organism databases. The core of the GENO model decomposes a genotype specifying sequence variation across an entire genome into smaller components of variation (e.g. allelic composition at a particular locus, haplotypes, gene alleles, and specific sequence alterations). GENO also enables description of biological attributes of these genetic entities (e.g. zygosity, phase, copy number, parental origin, genomic position), and their causal relationships with phenotypes and diseases.
- ClinVar is an archive of clinically reported relationships between variation and phenotypes along with interpretations and supporting evidence. Data in ClinVar are submitted primarily by diagnostic labs. ClinVar includes expert reviews and data links to other clinically-relevant resources at NCBI. The VMC specification is expected to facilitate data submissions by providing unified guidelines for data structure and allele normalization.
- ClinGen provides a centralized database of genomic and phenotypic data provided by clinicians, researchers, and patients. It standardizes clinical annotation and interpretation of genomic variants and provides evidence-based expert consensus for curated genes and variants. ClinGen has informed the VMC effort and is committed to harmonizing and

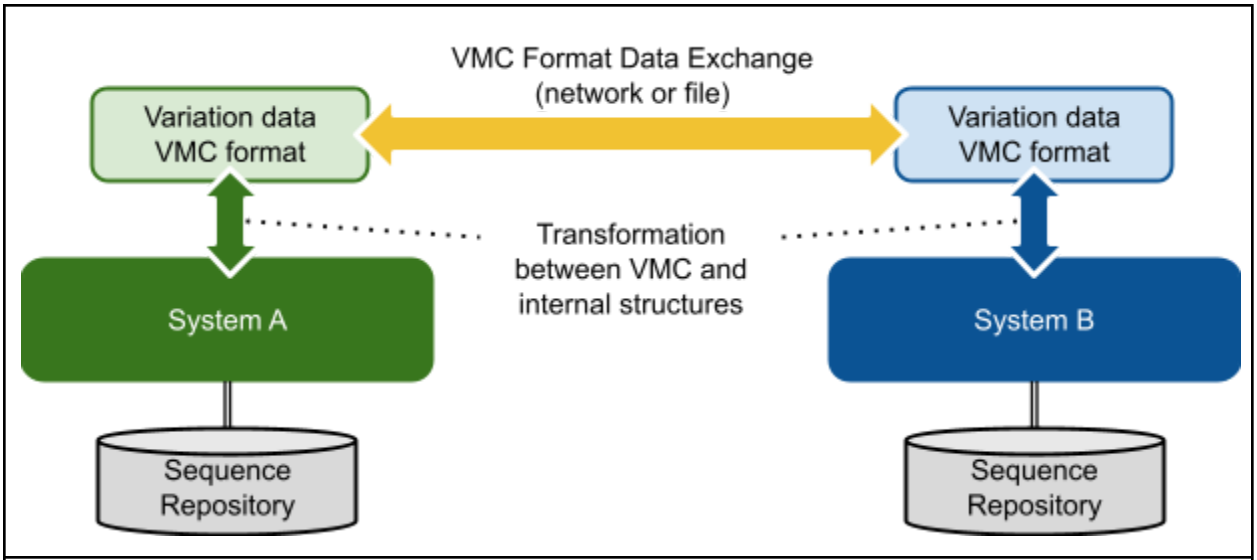
collaborating on the evolution of the VMC specification to achieve improved data sharing.

- HL7 FHIR Genomics, Version 2 Clinical Genomics Implementation Guide, CDA Genetic Test Report: There are several standards developed under the HL7 umbrella that include a genomics component. The FHIR Genomics component was released as part of the overall FHIR specification (latest is Release 3) based on standardized use cases. The HL7 Clinical Genomics (CG) Work Group focuses on developing standards for clinical genomic data and related relevant information within EHRs. The specifications developed by the CG work group primarily utilize the HL7 v2 messaging standard and the newer HL7 FHIR (Fast Healthcare Interoperability Resources) framework.
- The SPDI format created to represent alleles in NCBI's Variation Services has four components: the sequence identifier, which is specified with a sequence accession and version; the 0-based interbase coordinate where the deletion starts; the deleted sequence (or its length) and the inserted sequence. The Variation Services return the minimum deleted sequence required to avoid over precision. For example, a deletion of one G in a run of 4 is specified with deleted and inserted sequences of GGGG and GGG respectively, avoiding the need to left or right shift the minimal representation. This reduces ambiguity, but can lead to long allele descriptions.

Anticipated Uses

As the proposed *lingua franca* for exchanging structured information for sequence variation, the VMC specification is anticipated to have many uses.

Anticipated use	**Examples and Value**
Data Sharing VMC provides conventions for sharing and submitting variation data	• Facilitate submissions to ClinVar • Submit alleles to, and receive alleles from, the ClinGen Allele Registry • Resolve variant equivalence from heterogeneous sources
Knowledge representation VMC enables the representation of precise and unambiguous relationships between variation and associated information	• Provide precise, immutable ids for instances of essential data • Associate annotations with variation on an appropriate sequence type (e.g., protein consequence for protein allele) • Associate pathogenicity with alleles, Haplotypes, and Genotypes (e.g., PhenoPacket Exchange Format) • Create relationships between classes of variation • Associate publications, pathogenicity, and frequency data with all classes of variation
Clinical practice VMC provides a common representation of variation within clinical settings	• Standardize submission from testing labs to institutional EHRs • Represent patient sequence variation (germline DNA, somatic DNA, RNA, or protein) • Retrieve knowledge associated with variation on-demand via FHIR • Enable clinical decision support to distinguish precise alleles from equivalent representations
Research and discovery VMC accommodates advanced research needs for allele representation	• Represent variation on sequence graphs • Enable identifiers for novel variation
Data processing VMC fits within existing software analysis pipelines	• Add VMC ids to VCF INFO fields • Convert HGVS to VMC • Provide REST-based VMC converter and validator



Conceptual diagram for the exchange of information using VMC. Typically systems that exchange variation data required bespoke interfaces for each pair of systems. Such interfaces do not generally facilitate interactions with a third system, etc. With VMC, both systems A and B implement methods to transform variation data between the system format and VMC format. A third system (not shown) that implements VMC would be able to readily exchange information with either or both A and B. To complete an exchange of variation information, these systems must all assign ids to VMC Identifiable Objects. While such ids can be ones assigned by centralized databases, the VMC specification also recommends using Computed Identifiers generated with the VMC Digest to make possible an exchange of information without recourse to central authorities (not depicted).

VMC Scope

The focus of the VMC data model specification is on representing variation that is defined by Alleles — assertions of a specific sequence at precise locations on DNA, RNA, and protein sequences — as well as containers for Haplotypes and Genotypes that build on an Allele primitive. The VMC data model is minimalistic, meaning that they represent only core data that are fundamental to the definition of the data type. The core entities — Sequence, Location, Allele, Haplotype, and Genotype — all have unique internal identifiers.

A consequence of the minimalistic design strategy is that users of the VMC data model are expected to associate data with variation separately, using internal identifiers, rather than store the annotation within the Allele entity itself.

The VMC data model is intended to be used within the context of a broader complete data model, which more fully captures the semantics of a given domain and/or use case, and they may be instantiated using multiple serialization methods. The model uses open standards for which there is wide support in multiple computer languages and systems.

Table: VMC In Summary: Present Scope, Future Scope, Out-of-Scope

Present Scope	
Internal (logical) and external (business) identifiers	Following HL7 FHIR convention, VMC differentiates between logical identifiers used to connect data within a system and business identifiers used to exchange data between systems. These conventions allow systems to implement local naming schemes while preserving the ability to exchange data.
Positions on a sequence	The VMC data model provides specifications for representing sequences and precise, contiguous intervals on sequences at precise locations discovered by high-throughput sequencing.
Primitive Allele type	The VMC data model provides an Allele entity, which represents the *state* of a subsequence with respect to a reference sequence. *This differs from conventional uses of "variant", which connotes a change in sequence.*
Composition of primitive types	The VMC data model provides entities that represent Haplotypes and Genotypes. *The VMC definition of Genotype generalizes conventional notions of diplotypes and single location Genotypes.*
Global computed identifier	The VMC specification includes an algorithm for computing globally-unique identifiers for any genetic state without a central authority.
Machine-readable specification	The VMC specification is implemented in JSON Schema, which enables data validation and consistency in multiple languages. *Implementations of the VMC specification in other schema languages are anticipated.*
Future Scope	
Defining and representing comparisons and equivalence among variation	Except for exact identity, the VMC specification does not define Allele, Haplotype, or Genotype comparisons or equivalence. For example, there is no notion that two Alleles normalize to the same Allele, or that two Alleles result in the same protein amino acid. Future work will add equivalence functions and equivalence sets among Alleles, Haplotypes, and Genotypes.
Quantitative Allele entities	Future work will address important use cases that involve quantitative allelic ratios, such as representation of mosaicism, chimerism, heterogeneous tissues, and heteroplasmy of mitochondrial sequences.
Structural, copy number, and "fuzzy" variation	Future work will address "structural variation" — translocations, large deletion or insertion events — and variation at approximate locations, such as those by cytogenetic techniques.
Out of Scope	
Sequence repository	VMC relies on the existence of a database of sequences, accessible using conventional sequence accessions. If an implementation implements the VMC Digest, the database must support either native VMC sequence accessions or computing such accessions on-the-fly.
Integration with specific systems	With the exception of proof-of-concept implementations, the project does not intend to provide an interface for any specific service. Instead, it will inform the development and consumption of such interfaces by member groups, such as GA4GH, ClinVar, ClinGen, and FHIR Genomics.
Entities for data associated with sequence variation	Sequence variation is commonly associated with annotations from diverse domains, including statistics for an observation in a sample (e.g., VCF quality), population frequencies, disease and trait associations, and a myriad of functional predictions and interpretations. The VMC effort focuses solely on data structures required to represent variation. While these data structures will be used to associate variation with other data — some code examples may

	demonstrate techniques for doing so — defining entities for associated data types is out of scope.
Nested or Recursive Haplotypes	While it is convenient to represent Haplotypes recursively, such a definition is out of scope because it introduces significant challenges in the data model, including prevention of recursive definitions and ambiguity about the correct reference sequence for intermediate levels of a Haplotype definition. (The latter is critical when insertions or deletions change the coordinate system in which an Allele is interpreted.)

Conceptual and Logical Model

When biologists define terms in order to describe phenomena and observations, they rely on a background of human experience and intelligence for interpretation. Definitions may be abstract, perhaps correctly reflecting uncertainty of our understanding at the time. Unfortunately, such terms are not readily translatable into an unambiguous representation of knowledge.

For example, "allele" might refer to "an alternative form of a gene or locus" [Wikipedia], "one of two or more forms of the DNA sequence of a particular gene" [ISOGG], or one of a set of coexisting sequence alleles of a gene [Sequence Ontology]. Even for human interpretation, these definitions are inconsistent: does the definition describe a precise sequence change or a qualitative one? In addition, all three definitions are inconsistent with the practical need for a way to describe sequence changes outside regions associated with genes.

The computational representation of biological concepts requires translating precise biological definitions into data structures that can be used by implementers. This translation should result in a representation of information that is consistent with conventional biological understanding, and, ideally, be able to accommodate future data as well. The resulting *computational representation* of information should also be cognizant of computational performance, the minimization of opportunities for misunderstanding, and ease of manipulating and transforming data.

Accordingly, for each term we define below, we begin by describing the term as used by biologists (**biological definition**). When a term has multiple biological definitions, we explicitly choose one of them for the purposes of this specification. We then provide a computer modeling definition (**computational definition**) that reformulates the biological definition in terms of information content. We then translate each of these computational definitions into precise specifications for representing information (**logical model**). Terms are ordered "bottom-up" so that definitions depend only on previously-defined terms.

Model Overview

The VMC Specification provides five core entities for genetic states:

- Interval represents start and end positions of a range of residues, possibly with length zero, and specified using interbase coordinates.
- Location represents a contiguous region of a Sequence.
- Allele represents a single contiguous Sequence at a Location. The Sequence may differ from the reference Sequence at the Location, including being longer or shorter.
- Haplotype represents Alleles known to be "in cis" or “in phase”; that is, occurring on the same covalent sequence.

- Genotype represents a list of Haplotypes. This entity represents a generalization of two concepts in common use: an alternative notion of Genotype as an assertion of Alleles at the same location (such as rs1345(A;T)), and a diplotype, which connotes a pair of Haplotypes. The intention of the VMC generalization is to subsume both of these narrower definitions into a single structure. The length of the list is the ploidy at that location in that sample, and may reflect typical states of 1 or 2 Alleles, or less common states such as 0, 3, or more due to chimerism, mosaicism, tumor/normal contamination, or copy number variation.

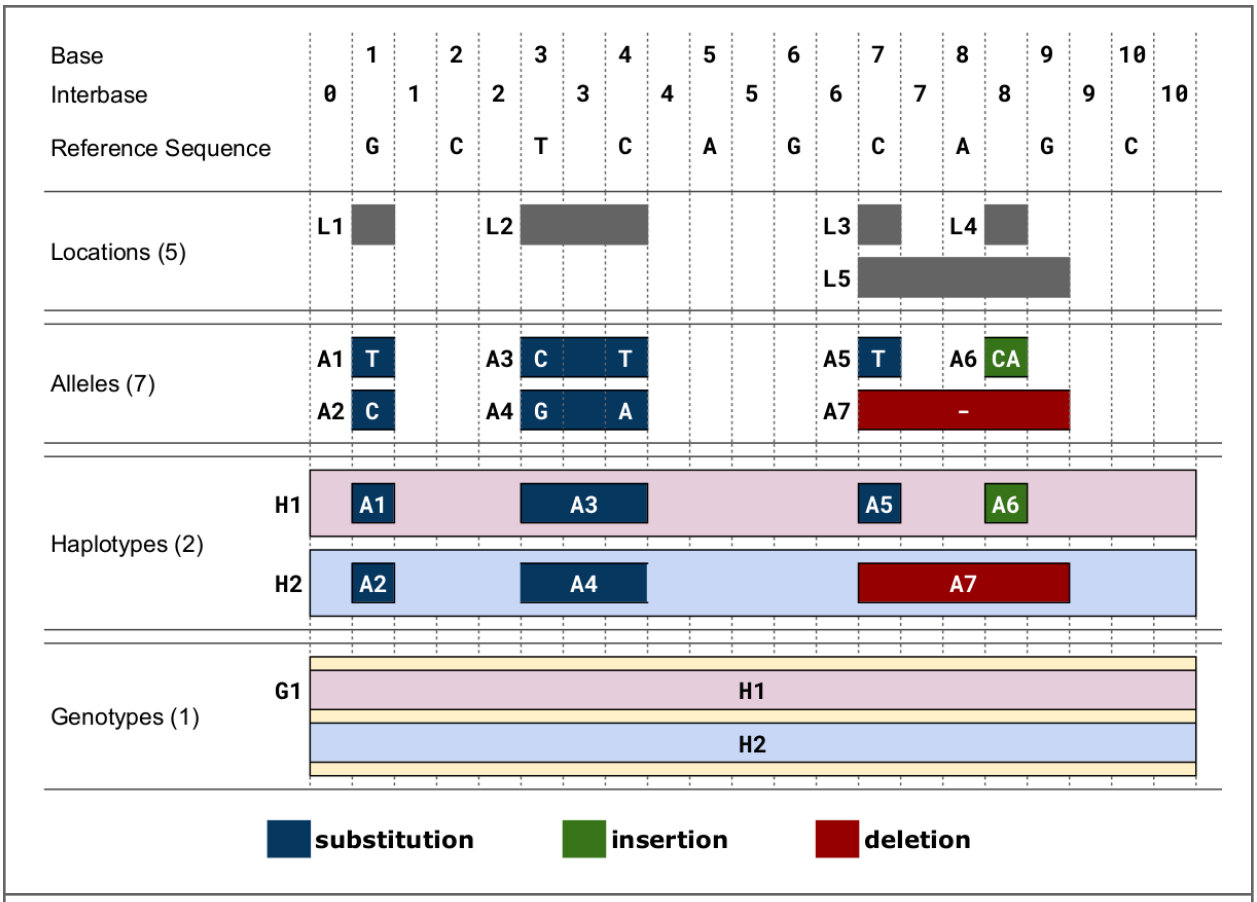


Figure. Representation of a reference Sequence, base and interbase coordinate systems, Locations, Alleles, Haplotypes, and Genotypes. Each type has an internal identifier (Id), shown above with L, A, H, and G prefixes. A Location is a span of reference Sequence with width >= 0. An Allele specifies a Sequence at a Location; the Sequence may be identical to the subsequence of the reference at that Location. The subsequence need not be the same length or the same Sequence as that at the Location. Allele Sequences that are larger or smaller than the Location span correspond to net insertions and deletions respectively. Alleles are colored by type for clarity: blue=substitution, red=deletion, green=insertion. A1 and A2 occur at the same location L1, and A3 and A4 at the same location L2. Allele A7 overlaps A5 and A6, but the three occur at distinct locations. A Haplotype is a collection of Alleles, referenced by identifiers, asserted to occur on a covalently contiguous Sequence (i.e., in phase); two distinct Haplotypes are represented by pink and blue containers. A Genotype, represented by a yellow container, is a combination of Haplotypes, referenced by id. The number of Haplotypes in a Genotype equals the number of chromosomes represented (e.g., 2 for autosomes in diploid cells). [figure source]

The VMC data model makes extensive use of internal Ids in order to refer to Sequences, Locations, Alleles, Haplotypes, and Genotypes (the Identifiable Objects). For example, Haplotypes refer to constituent Alleles by id, and Genotypes refer to constituent Haplotypes by Haplotype Ids. The identifiability of objects is critical to creating relationships between objects, aggregating objects in lists and sets, and associating data with objects. Following the conventions of HL7 FHIR in which string Ids are distinguished from structured Identifiers, the VMC specification enables objects to be associated with an arbitrary number of identifiers in order to facilitate interoperability with external data sets.

The VMC data model distinguishes internal Ids and external Identifiers. Internal Ids are identifiers that are proprietary to the data source and used to associate VMC objects; the specification places no restrictions on their form or content. External Identifiers are used to connect data between systems and the VMC specification provides for multiple Identifiers to be associated with each object; minimally, the VMC specification *requires* that a globally unique Identifier based on the VMC Digest be defined for every object.

General Implementation Notes

This section provides specifications for data elements in the VMC conceptual model, including the types of data members for each structure and constraints on values of those members.

> The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

- Entity names SHOULD use TitleCase, regardless of language or protocol.
- Field names SHOULD use snake_case, regardless of language or protocol.
- String values MUST be encoded with ASCII.
- All entities are singletons. There is never more than one instance of an entity with equal elements.
- The capitalized word NULL in this specification refers to the special value that indicates a lack of value of the defined type.

Most implementations of the VMC specification will require access to a database of sequences in order to normalize (also known as shift or shuffle) Alleles and, optionally, to implement the VMC Digest algorithm for the generation of consistent and globally unique identifiers for all object types. The choice of database is left to the implementation. (The included demonstration code uses SeqRepo, which is publicly available.)

The VMC data model makes extensive use of object identifiers to define relationships among objects and to enable external tools to associate data with objects.

VMC defines ten terms covering primitive concepts and data types:

- Id
- Identifier
- Residue
- Interval
- Sequence
- Location
- Allele
- Haplotype
- Genotype
- VMCBundle

VMC-defined terms, when used subsequently, are capitalized to distinguish them from colloquial use.

Primitive Concepts and Types

Id

Biological definition: None.

Computational definition: A string that uniquely identifies a specific instance of an object within a document.

Implementation Guidance

- Ids are opaque byte-strings: there are no formatting, content, or character set constraints.
- A FHIR Id, which is limited to 64 characters from a restricted character set, may be used as a VMC Id.
- Ids must correspond 1:1 to object instances: An id refers to exactly one object, and an object has only one id. Therefore, equivalence of objects implies equivalence of ids, and vice versa.
- Implementations MAY change internal identifiers at any time. Therefore, receiving systems SHOULD NOT persist Ids from remote sources. Instead, Identifiers (below) should be used for communication between systems.
- Ids are not locatable references. An Id may not be used to retrieve objects from remote databases. Instead, Identifiers should be used for retrieval.
- The VMC specification requires a canonical ordering (sorting) of Ids. Sorting a list of Ids MUST be performed using the C locale or, equivalently, by first encoding Ids as ASCII.

Identifier

Biological definition: An identifier for an object, such as a Sequence or Allele, that is assigned by an organization or algorithm. The identifier may be used to name data in order to reference it, or to locate data in order to retrieve it from its source.

Computational definition: An VMC Identifier is represented using a Compact URI (CURIE)[5], a W3C standard, with a *prefix* and *reference* that correspond to a namespace and local identifier within that namespace.

Information Model

Field	Type	Label	Description
prefix	string	required	Namespace for the identifier
reference	string	required	Unique key within *prefix* namespace

Implementation Guidance

- CURIEs may be represented as structured data objects or as strings. The two forms are deterministically convertible as defined in the CURIE specification.
- Within VMC models, CURIEs MUST be represented as objects.
- Implementations MAY display to users, and accept from users, CURIEs represented as strings.
- The CURIE specification requires the use of a map from each *prefix* value to a URI template, shown below. URIs are generated from the CURIE map by substituting `{reference}` in the URI, if it exists, with the CURIE *reference*. This mapping provides URIs to descriptive information for a prefix rather than to structured data.

prefix	mapped URI template
Ensembl	https://www.ensembl.org/Multi/Search/Results?q={reference}
GRCh37	https://www.ncbi.nlm.nih.gov/grc/human
GRCh38	https://www.ncbi.nlm.nih.gov/grc/human
LRG	http://ftp.ebi.ac.uk/pub/databases/lrgex/{reference}.xml
NCBI	https://www.ncbi.nlm.nih.gov/gquery/?term={reference}
VMC	https://github.com/ga4gh/vmc

- Implementations MAY provide additional mappings in the VMC Bundle.
- Implementations SHALL use *prefix* and *reference*, and the Identifiers derived from them, verbatim. These entities MAY NOT be modified in any way; for example, they may not be case folded or modified by the addition of prefixes or suffixes.

[5] https://www.w3.org/TR/curie/

- CURIEs and FHIR Business Identifiers are convertible: For the purposes of interoperability with FHIR, the Identifier *namespace* and *accession* SHALL map, using the CURIE prefix map, to the *system* and *value* components of a FHIR Identifier[6].
- It is essential for the durability of information that an Identifier refer to exactly one object for all time. (For example a Sequence reference should refer to only one Sequence.) Implementations may assume this uniqueness, but adherence is the responsibility of source databases. For this reason, databases (or versions of databases) that do not provide this guarantee SHALL NOT be used with the VMC data model.

Residue

Biological definition: A residue refers to a specific monomer within the polymeric chain of a protein or nucleic acid (Source: Wikipedia Residue page).

Computational definition: Specific residues (i.e., molecular species) as well as categories or groupings of these ("ambiguity codes") are represented using one-letter IUPAC abbreviations.

Notes

- The term “base” is commonly used to refer to a nucleic acid residue.

Interval

Biological definition: None.

Computational definition: Two integers that define the start and end positions of a range of residues, possibly with length zero, and specified using interbase coordinates.

Information Model

Field	Type	Label	Description
start	uint64	required	start position
end	uint64	required	end position

Implementation Guidance

- Implementations MUST require that 0 ≤ start ≤ end. (In the case of double-stranded DNA, this constraint holds even when a feature is on the complementary strand.)

Notes

- VMC uses Interbase coordinate because they provide conceptual consistency that is not possible with residue-based systems. (See rationale in Sequence ranges use an interbase coordinate system.) Implementations will need to convert between interbase and 1-based inclusive residue coordinates familiar to most human users.

[6] http://build.fhir.org/datatypes.html#identifier

- Interbase coordinates start at 0 (zero).
- The length of an interval is *end - start.*
- An interval in which start == end is a zero width point between two residues.
- An interval of length == 1 may be colloquially referred to as a position.
- Two intervals are equal if the their start and end coordinates are equal.
- Two intervals *intersect* if the start or end coordinate of one is strictly between the start and end coordinates of the other. That is, if:
    - `b.start < a.start < b.end OR`
    - `b.start < a.end < b.end OR`
    - `a.start < b.start < a.end OR`
    - `a.start < b.end < a.end`
- Two intervals a and b *coincide* if they *intersect* or if they are *equal*. (The equality condition is required to handle the case of two identical zero-width Intervals.)
- Alleles within a Haplotype may not coincide; see Haplotype for elaboration.

Examples

- <start, end>=<*0,0*> refers to the point with width zero before the first residue.
- <start, end>=<*i,i+1*> refers to the *i+1th* (1-based) residue.
- <start, end>=<*N,N*> refers to the position after the last residue for Sequence of length *N*.
- See Interbase Interval tests in the vmc repo for a diagram and examples.

Sources

- Interbase Coordinates (Chado documentation)
- Interbase coordinate system primer
- SO: sequence_feature (SO:0000110) — Any extent of continuous biological sequence.
- SO: region (SO:0000001) — *A sequence_feature with an extent greater than zero. A nucleotide region is composed of bases and a polypeptide region is composed of amino acids.*

Identifiable Objects

Sequence

Biological definition: A contiguous, linear polymer of nucleic acid or amino acid residues.

Computational definition: A character string of Residues that represents a biological sequence using the conventional sequence order (5'-to-3' for nucleic acid sequences, and amino-to-carboxyl for amino acid sequences). IUPAC ambiguity codes are permitted in Sequences.

Information Model

A Sequence is a string, constrained by characters representing IUPAC nucleic acid or amino acid codes.

Implementation Guidance

- Sequences MAY be empty (zero-length) strings. Empty sequences are used as the replacement Sequence for deletion Alleles.
- Sequences MUST consist of only uppercase IUPAC abbreviations, including ambiguity codes.

Notes

- A Sequence provides a stable coordinate system by which an Allele may be located and interpreted.
- A Sequence may have several roles. A “reference sequence” is any Sequence used to define an Allele (below). A Sequence that replaces another Sequence is called a “replacement sequence”.
- In some contexts outside the VMC specification, “reference sequence” may refer to a member of set of sequences that comprise a genome assembly. In the VMC specification, any sequence may be a “reference sequence”, including those in a genome assembly.
- For the purposes of representing sequence variation, it is not necessary that Sequences be “typed” (i.e., DNA, RNA, or AA).

Location

Biological definition: As used by biologists, the precision of “location” (or “locus”) varies widely; examples include chromosomal bands, named genomic features (e.g., genes, exons, or markers), or specific positions on a reference sequence.

Computational definition: An identifiable position or region on a Sequence, defined by Sequence Id and an Interval.

Information Model

Field	Type	Label	Description
id	Id	optional	Location Id; must be unique within document
sequence_id	Id	required	An id mapping to the Identifier of the external database Sequence
interval	Interval	required	Position of feature on reference sequence specified by sequence_id.

Implementation Guidance

- For a Sequence of length *n*:
  - 0 ≤ *interval.start* ≤ *interval.end* ≤ *n*
  - interbase coordinate 0 refers to the point before the start of the Sequence
  - interbase coordinate *n* refers to the point after the end of the Sequence.

- VMC requires that coordinates refer to valid Sequence. VMC does not support referring to intronic positions within a transcript sequence, extrapolations beyond the ends of sequences, or other implied sequence.
- IMPORTANT: HGVS permits variants that refer to non-existent sequence. Examples include coordinates extrapolated beyond the bounds of a transcript and intronic sequence. Such variants are not representable using VMC and must be projected to a genomic reference in order to be represented.

Allele

Biological definition: One of a number of alternative forms of the same gene or same genetic locus. In the context of biological sequences, “allele” refers to a set of specific changes within a Sequence, including sets with zero (no change), one change (a simple allele), or a multiple changes (a Haplotype). In the context of VMC, allele refers to a Sequence or Sequence change with respect to a reference sequence.

Computational definition: An Allele is a specific, single, and contiguous Sequence at a Location. Each alternative Sequence may be empty, shorter, longer, or the same length as the interval (e.g., due to one or more indels).

Information Model

Field	Type	Label	Description
id	Id	optional	Allele identifier; must be unique within document
location_id	Id	required	Where Allele is located.
state	Sequence|State	required	Sequence at location_id.

Implementation Guidance

- Implementations MUST require that interval.end ≤ sequence_length when the Sequence length is known.
- The implementation MAY infer the Sequence type of Sequence reference by location_id and the type of Sequence in state, and ensure compatibility between them. This behavior is not included in the specification.
- Alleles are *equal* only if the component fields are equal: at the same *location* and with the same *state*.
- Alleles may have multiple *related* representations on the same Sequence type due to shifting (aka shuffling, normalization). A future version of VMC will provide a general framework for flexibly declaring various notions of pairwise Allele relationships.
- Implementations MUST report variation *right* normalized. See the rationale for right normalization.

Notes

- When the alternate Sequence is the same length as the interval, the lengths of the reference Sequence and imputed Sequence are the same. (Here, imputed sequence means the sequence derived by applying the Allele to the reference sequence.) When the replacement Sequence is shorter than the length of the interval, the imputed Sequence is shorter than the reference Sequence, and conversely for replacements that are larger than the interval.
- When the replacement is “” (the empty string), the Allele refers to a deletion at this location.
- The Allele entity is based on Sequence and is intended to be used for intragenic and extragenic variation. Alleles are not explicitly associated with genes or other features.
- Alleles may have multiple representations on the same Sequence type due to shifting (aka shuffling, normalization). Fully resolving such equivalences is beyond the scope of the first phase of this specification.
- Biologically, referring to Alleles is typically meaningful only in the context of empirical alternatives. For modelling purposes, Alleles may exist as a result of biological observation or computational simulation, i.e., virtual Alleles.
- “Single, contiguous” refers the representation of the Allele, not the biological mechanism by which it was created. For instance, two non-adjacent single residue Alleles could be represented by a single contiguous multi-residue Allele.
- The terms "allele" and "variant" are often used interchangeably, although this use may mask subtle distinctions made by some users.
    - In the genetics community, "allele" may also refer to a haplotype.
    - "Allele" connotes a state whereas "variant" connotes a change between states. This distinction makes it awkward to use variant to refer to the concept of an unchanged position in a Sequence and was one of the factors that influenced the preference of “Allele” over “Variant” as the primary subject of annotations.
    - See Use “Allele” rather than “Variant” in the appendix for more discussion.
- When a trait has a known genetic basis, it is typically represented computationally as an association with an Allele.
- The VMC definition of Allele applies to all Sequence types (DNA, RNA, AA).

Sources

- ISOGG: Allele — *An **allele** is one of two or more forms of the DNA sequence of a particular gene.*
- SO: allele (SO:0001023) — *An allele is one of a set of coexisting sequence variants of a gene.*
- SO: sequence_alteration (SO:0001059) — *A sequence_alteration is a sequence_feature whose extent is the deviation from another sequence.*
- Wikipedia: Allele — *One of a number of alternative forms of the same gene or same genetic locus.*
- SO: sequence_variant — *A sequence_variant is a non exact copy of a sequence_feature or genome exhibiting one or more sequence_alteration*.

- GENO: Allele (GENO:0000512) - A sequence feature representing one of a set of coexisting sequences at a particular genomic locus. An allele can represent a 'reference' or 'variant' sequence at a locus.

Haplotype

Biological definition: A specific combination of Alleles that occur together on single sequence in one or more individuals.

Computational definition: A specific combination of non-overlapping Alleles that co-occur on the same reference sequence.

Information Model

Field	Type	Label	Description
id	Id	optional	Haplotype identifier; must be unique within document
location_id	Id	required	Where Haplotype is located.
completeness	enum	required	Declaration of completeness of the Haplotype definition. Values are: • UNKNOWN: Other in-phase Alleles may exist. • PARTIAL: Other in-phase Alleles exist but are unspecified. • COMPLETE: The Haplotype declares a complete set of Alleles.
allele_ids	Id[]	required	List of Alleles that comprise this Haplotype

Implementation Guidance

- The Haplotype location (as specified by the location_id) may refer to a subsequence of the reference sequence, such as a subsequence of an entire chromosome.
- All Alleles in a Haplotype MUST be defined on the same reference sequence as specified by location_id.
- Alleles within a Haplotype MUST not overlap ("overlap" is defined in Interval).
- All Location Intervals are to be interpreted in the context of the underlying reference sequence, irrespective of insertions or deletions by other “upstream” Alleles within the Haplotype.
- When reporting an Haplotype, completeness MUST be set according to these criteria:
    - "COMPLETE" only if the entire reference sequence was assayed and all in-phase Alleles are reported in this Haplotype.
    - "PARTIAL" only if the entire reference sequence was assayed, other in-phase Alleles exist, and are NOT reported in this Haplotype. This is an assertion of unreported variation.
    - "UNKNOWN" otherwise. This value is the default and should be used if neither "COMPLETE" nor "PARTIAL" applies. These cases include, but are not limited to,

assays that do not fully cover the reference sequence and an unwillingness by the reporter to declare the existence or absence of other in-phase Alleles.

- A Haplotype with an empty list of Alleles and completeness set to "COMPLETE" is an assertion of an unchanged reference sequence.
- When projecting a Haplotype from one sequence to a larger sequence, a "complete" Haplotype becomes an "unknown" Haplotype on the target sequence. Furthermore, this change is not reversible.

Notes

- Alleles within a Haplotype are, by definition, “cis” or “in-phase”. (“In phase” and “cis” refer to features that exist on instances of covalently bonded sequences.)
- Haplotypes are often given names, such as *ApoE3* or *A*33:01* for convenience.
    - Examples: A*33:01:01 (IMGT/HLA)
- When used to *report* Haplotypes, the completeness property enables data providers (e.g, diagnostic labs) to indicate that other Alleles exist, may exist, or do not exist. Data providers may not assay the full reference sequence or may withhold other in-phase Alleles in order to protect patient privacy.
- When used to *define* Haplotypes, the completeness property enables implementations to permit (PARTIAL) or preclude (COMPLETE) the existence of other variation when matching a Haplotype to a set of observed Alleles.
- Data consumers may wish to use the completeness property in order to provide accurate context for Allele interpretation or to select data used in association studies.

Sources

- ISOGG: Haplotype — *A **haplotype** is a combination of alleles (DNA sequences) at different places (loci) on the chromosome that are transmitted together. A haplotype may be one locus, several loci, or an entire chromosome depending on the number of recombination events that have occurred between a given set of loci.*
- SO: haplotype (SO:0001024) — *A haplotype is one of a set of coexisting sequence variants of a haplotype block.*
- GENO: Haplotype (GENO:0000871) - A set of two or more sequence alterations on the same chromosomal strand that tend to be transmitted together.

Genotype

Biological definition: The genetic state of an organism, whether complete (defined over the whole genome) or incomplete (defined over a subset of the genome).

Computational definition: A list of Haplotypes.

Information Model

Field	Type	Label	Description

id	Id	optional	Genotype identifier; must be unique within document
completeness	enum	required	Declaration of completeness of the Genotype definition. Values are: • UNKNOWN: Other Haplotypes may exist. • PARTIAL: Other Haplotypes exist but are unspecified. • COMPLETE: The Genotype declares a complete set of Haplotypes.
haplotype_ids	Id[]	required	List of Haplotypes; length must agree with ploidy of genomic region

Implementation Notes

- Haplotypes in a Genotype may occur at different locations or on different reference sequences. For example, an individual may have haplotypes on two population-specific references.
- Haplotypes in a Genotype MAY contain differing numbers of Alleles or Alleles at different Locations.

Notes

- The term "genotype" has two, related definitions in common use. The narrower definition is a set of alleles observed at a *single* location and with a ploidy of two, such as a pair of single residue variants on an autosome. The broader, generalized definition is a set of alleles at *multiple* locations and/or with ploidy other than two.The VMC Genotype entity is based on this broader definition.
- The term "diplotype" is often used to refer to two haplotypes. The VMC Genotype entity subsumes the conventional definition of diplotype. Therefore, the VMC data model does not include an explicit entity for diplotypes. See Genotypes represent collections of in-phase alleles with arbitrary ploidy in the Appendix for a discussion.
- The VMC data model makes no assumptions about ploidy of an organism or individual. The number of Haplotypes in a Genotype is the observed ploidy of the individual.
- In diploid organisms, there are typically two instances of each autosomal chromosome, and therefore two instances of sequence at a particular location. Thus, Genotypes will often list two Haplotypes. In the case of haploid chromosomes or haploinsufficiency, the Genotype consists of a single Haplotype.
- A consequence of the computational definition is that Haplotypes at overlapping or adjacent intervals may not be included in the same Genotype. However, two or more Alleles may always be rewritten as an equivalent Allele with a common sequence and interval context.
- The rationale for permitting Genotypes with Haplotypes defined on different reference sequences is to enable the accurate representation of segments of DNA with the most appropriate population-specific reference sequence.

Sources

- SO: Genotype (SO:0001027) — *A genotype is a variant genome, complete or incomplete.*

Using VMC within Applications

VMCBundle

The VMCBundle provides a coherent and complete collection of Locations, Alleles, Haplotypes, Genotypes, and external Identifiers. The VMCBundle includes metadata describing the collection.

Information Model

Field	Type	Label	Description
meta	Map	required	Reserved; Key-value map with a description of the VMCBundle.
locations	Map	optional	A map of Id: Location. Must include all Locations referenced by objects in Alleles.
alleles	Map	optional	A map of Id: Allele. Must include all Alleles referenced by objects in Haplotypes.
haplotypes	Map	optional	A map of Id: Haplotype. Must include all Haplotypes referenced by objects in Genotypes.
genotypes	Map	optional	A map of Id: Genotype.
identifiers	Map	optional	A map of Id: Identifier[].

Implementation Guidance

- The meta key is reserved. Future versions of VMC will impose requirements on the content of the meta map.
- The locations, alleles, haplotypes, genotypes, and identifiers fields contain a map of (internal) Ids to objects. In the case identifiers, the map is from Ids to a *list* of Identifiers.
- The identifiers map is one-to-many: An Id may be associated with zero or more Identifiers. There is no requirement for objects to have associated Identifiers.
- The inverse mapping of Identifiers to Ids may be ambiguous. That is, a single Identifier may refer to one or more objects. Implementations MUST NOT prohibit ambiguous mapping of identifiers to ids.
- A naive implementation of the VMCBundle will require traversing and storing a graph of objects that may become large. Implementations may wish to generate multiple bundles grouped by sets of Genotypes or Haplotypes.

Computed Identifiers with the VMC Digest

In the VMC specification, Identifiers are external, shared names for objects that enable data sharing. Identifiers are often provided by naming authorities, such as Ensembl, NCBI, or ClinGen. However, the pace at which sequence variation is discovered makes it difficult to centrally assign identifiers for each entity. Furthermore, entities are often private and are not appropriate for submission to a naming authority.

The VMC specification provides an algorithmic solution to deterministically generate a globally unique identifier from a VMC object itself. Two valid implementations of the VMC digest will generate the same identifier when the objects are identical, and will generate different identifiers when they are not. Adopting the VMC Digest algorithm for id assignment could gradually relieve centralized registries of the need to assign ids to variation data, instead allowing them to focus on adding annotations and relationships using universal computed identifiers.

The VMC Digest algorithm applies two well-established standard algorithms, the SHA-512 hash function[7], which generates a binary SHA-512 digest, and Base64 encoding, which represents binary data using printable characters to create unique ids for each unique VMC object. The computed id has less than a 1 in 10^{30} probability of one or more collisions in 10^{24} unique objects (see Appendix: VMC Digest Collision Analysis for details on digest byte length, etc.).

The three steps to create a VMC Digest id from a VMC object — detailed below — include:

1. Serializing a VMC object into a string form.
2. Generating a URL-safe, base64-encoded SHA512 truncated digest from the serialized object.
3. Constructing an identifier from the digest.

👉 The examples below are taken from the Quick Start and VMC Digest notebooks in the VMC code repository.

[7] While SHA-512 is typically used for cryptography, its application here is not for providing information security, which is out of scope for VMC.

Object Serialization

Serialization converts an object into a string using the templates shown below. All types in the VMC Specification have serializations. In addition, the Identifiable Objects — Sequence, Location, Allele, Haplotype, and Genotype — the serializations may be used to generate a VMC Digest.

Object Type	**Template, Example Serialization, and Resulting VMC Digest id (if an identifiable object)**
Identifier	<Identifier|*namespace*|*accession*> <Identifier|VMC|GS_IIB53T8CNeJJdUqzn9V_JnRtQadwWCbl>
Interval	<Interval|*start*|*end*> <Interval|44908683|44908684>
Sequence	*sequence* NNNN … CTCT … NNNN [58617616 nucleotides] → VMC:GS_IIB53T8CNeJJdUqzn9V_JnRtQadwWCbl
Location	<Location|*sequence_id*|*interval*> <Location|VMC:GS_IIB53T8CNeJJdUqzn9V_JnRtQadwWCbl|<Interval|44908683|44908684>> → VMC:GL_L1IS6jOwSUsOpKihGRcqxHul1IwbV-1s
Allele	<Allele|*location_identifier*|*state*> <Allele|VMC:GL_L1IS6jOwSUsOpKihGRcqxHul1IwbV-1s|C> → VMC:GA_zsJuMckKGajqHCl16sxKQJtBMjGrFHHZ
Haplotype	<Haplotype|*location*|*completeness*|[*allele_id*;...]> <Haplotype||COMPLETE|[VMC:GA__8rLiy7YkQDNy-t536RpVFGxIDiWLr6J;VMC:GA_zsJuMckKGajqHCl16sxKQJtBMjGrFHHZ]> →
Genotype	<Genotype|*completeness*|[*haplotype_id*;...]> <Genotype|COMPLETE|[VMC:GH_xk_4sKZKfwD7ol3H89mDShrBT3dfu5Aq;VMC:GH_xk_4sKZKfwD7ol3H89mDShrBT3dfu5Aq]> → VMC:GG_Pv97fICMeVRmowtCwioFpoFmrsOkZ7es
Template, example serialization, and resulting computed_id for each object type. Italicized terms in templates are replaced by serializations of the appropriate object. The examples above are taken from the VMC Quick Start notebook and the VMC specification validation test set.	

Implementation Notes:

- Implementations of the VMC Digest MUST use Identifiers with the VMC namespace. Conversely, the VMC namespace MUST NOT be used for any other purpose.

- The VMC Digest requires a mechanism to obtain VMC-based Sequence identifiers (e.g., VMC:GS_IIB53T8CNeJJdUqzn9V_JnRtQadwWCbl) for accessions used to define variation. For efficiency, it is recommended that implementations precompute VMC Sequence identifiers for accessions from NCBI and Ensembl. VMC Identifiers are supported natively in SeqRepo.
- Serializing objects requires identifiers for dependent objects. That is, serializing Genotypes requires identifiers for Haplotypes; Haplotypes requires Alleles; Alleles requires Locations; and Locations requires Sequence
- Lists of *allele_identifiers* and *haplotype_identifiers* MUST be arranged in ASCII sorted order (or, equivalently, using POSIX or C locale).

IMPORTANT: URL-safe base64 encoding does *not* sort consistently in all locales. Ids must be sorted using the C or POSIX locale, or, equivalently, by first ASCII-encoding Ids.

Truncated Digest

The truncated digest step generates a string digest from a serialized object. Specifically, the VMC Truncated Digest is generated from the following steps:

1. ASCII encoding the string digest into a binary "blob".
2. Applying SHA-512 to the byte blob.
3. Truncating the resulting (binary) digest to the first *n* bytes. For the VMC Digest, n=24 bytes (192 bits). This truncation length was chosen conservatively, such that the probability of a chance collision for 10^{30} objects is less than 1 in 10^{24}. See Appendix: Digest Collision Analysis for details.
4. Formatting the truncated binary digest using URL-safe Base 64 encoding (RFC3548 Section 4). A Base 64 expands a binary string by a factor of 4/3. For example, a 24-byte binary digest will be expanded to a 32-byte Base 64 ASCII-encoded byte string.

Python

```
>import base64, hashlib
>>> def truncated_digest(blob, n=24):
...     d = hashlib.sha512(blob).digest()
...     return base64.urlsafe_b64encode(d[:n]).decode("ASCII")
>>> truncated_digest("ACGT".encode("ASCII"))
'aKF498dAxcJAqme6QYQ7EZ07-fiw8Kw2'
```

Java

```
import static java.nio.charset.StandardCharsets.*;
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;

class TruncatedDigestExample {
```

```
    public static void main(String args[]){
       System.out.println(truncated_digest("ACGT".getBytes(US_ASCII), 24));
    };

    private static String truncated_digest(byte[] blob, int n) {
       MessageDigest md = null;
       try {
           md = MessageDigest.getInstance("SHA-512");
       } catch (NoSuchAlgorithmException e) {
           e.printStackTrace();
       }

       byte[] digest = md.digest(blob);
       byte[] digest24 = Arrays.copyOfRange(digest, 0, n);
       return Base64.getUrlEncoder().encodeToString(digest24);
    }
}
```

Example: Minimal demonstrations of the Truncated Digest in Python and Java. Both examples generate the same digest string for the Sequence "ACGT", `aKF498dAxcJAqme6QYQ7EZ07-fiw8Kw2.`

Computed Identifier based on VMC Digest

An Identifier is a structure consisting of a namespace and accession. For a computed identifier, the namespace is "VMC". The accession is constructed by concatenating, in order, a type-specific prefix, an underscore ("_"), and the truncated digest determined above. As described for Identifiers, this structure may also be represented as a string using the CURIE syntax. Such Identifiers and Ids are compatible with FHIR.

Type	Prefix	Description
Sequence	GS	Global Sequence identifier
Location	GL	Global Location identifier
Allele	GA	Global Allele identifier
Haplotype	GH	Global Haplotype identifier
Genotype	GG	Global Genotype identifier

Identifier prefixes for VMC entities. When generating a computed identifiers for a type, the identifier is constructed by concatenating the corresponding prefix, an underscore, and the VMC digest applied to the serialization of the object.

Example Construction of Computed Identifier

This section will demonstrate an end-to-end computation of a Computed Identifier for a Location defined as the interbase range 44908683,44908684 on NC_000019.10.

The Location serialization format is

`<Location|sequence_identifier|interval>`

Therefore, the serializations of sequence_identifier and interval are also required.

The VMC Sequence identifier for NC_000019.10 may be obtained by fetching the Sequence, running VMC Digest on it, and constructing a VMC Sequence identifier. **An optimized solution is to maintain a lookup table (e.g., SeqRepo contains VMC ids)**. The VMC id for NC_000019.10 is VMC:GS_IIB53T8CNeJJdUqzn9V_JnRtQadwWCbl. Using the serialization format for given earlier, the serialized Sequence identifier is

`VMC:GS_IIB53T8CNeJJdUqzn9V_JnRtQadwWCbl`

Similarly, the serialization for the above Interval is

`<Interval|44908683|44908684>`

Inserting the Identifier and Interval serialization into the Location format results in:

```
<Location|VMC:GS_IIB53T8CNeJJdUqzn9V_JnRtQadwWCbl|<Interval|4490
8683|44908684>>
```

(N.B. This text is wrapped to fit on the page. The serialization consists solely of contiguous printable characters without whitespace.)

The Location serialization is then ASCII encoded and the SHA-512 digest is computed and truncated to the first 24 bytes. The URL-safe Base64 encoded form of the truncated binary digest is the string:

`L1IS6jOwSUsOpKihGRcqxHul1IwbV-1s`

The prefix for Location types is "GL". The Identifier structure for this Location is (using JSON representation):

```
{
  "namespace": "VMC",
  "accession": "GL_L1IS6jOwSUsOpKihGRcqxHul1IwbV-1s"
}
```

And the CURIE form for this Identifier is:

`VMC:GL_L1IS6jOwSUsOpKihGRcqxHul1IwbV-1s`

Figure: Pseudocode summarizing the construction of the computed id for the VMC representation of the Location object corresponding to nucleotide position 44908684 on NC_000019.10. The sha512 digest and base64us (URL-safe base 64) implementations are assumed to be provided by external libraries.

Implementation and Examples

The VMC specification (this document) refers principally to the conceptual and logical model, not a particular implementation of it. It is expected that implementations in other protocols and schema languages, such as Protobuf, GraphQL, or OpenAPI, might eventually become available.

An implementation of VMC using JSON Schema is available from https://github.com/ga4gh/vmc. JSON Schema is a platform- and language-neutral method for representing and validating objects in JavaScript Object Notation (JSON). Despite its name, JSON has been widely adopting by all common languages for serializing network data.

In addition, numerous examples and Python notebooks are available at https://github.com/ga4gh/vmc-python.

Future Work

The Variation Modelling Collaboration was formed by representatives of various stakeholder organizations in order to overcome barriers to data sharing between those groups. As a first step, the VMC specification delivers an extensible framework for representing variation that provide functionality for many key use cases and a corresponding large number of research and clinical users.

As part of publishing this specification, we now describe our next steps, which include expanding the specification for important additional use cases even while we begin the equally crucial work of encouraging stakeholders to embrace the VMC model and work toward standardizing the exchange of variation data.

Governance and Stewardship

The Variation Modelling Collaboration was formed by representatives of several stakeholder organizations in order to overcome barriers to data sharing between those groups. The goal was to create an *initial* specification and begin the process of adoption. The specification will evolve to meet the needs of those groups that contribute to its development. Therefore, the VMC will seek the participation from groups not already represented.

New types of variation

The VMC specification covers primitive data types to represent the replacement of sequences in a reference. Increasing its applicability will require supporting more complex types of variation, including:

- **copy number variants** and **structural variants**, including inversions and translocations
- DNA segment variation, such as **"fuzzy" intervals** that provide boundaries on imprecise coordinates, including **context intervals** in which boundaries are defined by sequence context, and **feature intervals** in which sequence features such as introns and exons are identifiable.
- **mosaicism** and **chimerism**

Allele comparisons, equivalence, and generalized relationships

The fundamental use case for describing variants is so that knowledge collected about a variant at one place and time can be used at another place and time. The VMC Allele specification provides an unambiguous method for describing an Allele with respect to in the context of a public, stable sequence (i.e. RefSeq, Ensembl). However, the VMC Allele specification does not currently deal with the mapping of these described alleles from one instance of a public, stable sequence to another.

Public, stable sequences are re-released in new versions as knowledge about sequences increases and new transcripts are discovered. In order to ensure that clinically relevant information collected on one version of a sequence is not lost when these public, stable sequences are updated requires understanding the equivalence of alleles mapped to different instances of these sequences.

Currently, this equivalence building is carried out in parallel at institutions around the world, and represents a major (perhaps the major) impediment to adoption of updated public, stable, “reference” sequences. Indeed, we lack even the ability to express, in a structured, sharable way, that two Alleles are equivalent across genome builds.

Gaining VMC Adoption by Stakeholder Organizations

The success of the VMC effort will ultimately depend on adoption by stakeholder organizations, many of which have existing development priorities and processes. As supporters of and contributors to the VMC specification, the GA4GH, ClinGen, ClinVar, and HL7 Clinical Genomics organizations are now considering the process by which the specification can be incorporated into existing systems.

GA4GH

The GA4GH Genomic Knowledge Standards (GKS) Workstream will serve as the organizational home for the VMC. The GKS Workstream will utilize the VMC specification as a core component of its deliverables, collecting feedback on and revising the specification as needed.

ClinGen

ClinGen is a National Institutes of Health (NIH)-funded resource building authoritative central resources that define the clinical relevance of genes and variants for use in precision medicine and research. Our major areas of focus include:

- Sharing genomic and phenotypic data provided by clinicians, researchers, and patients through centralized databases for clinical and research use
- Standardizing the clinical annotation and interpretation of genomic variants and implementing evidence-based expert consensus for curating genes and variants

- Disseminating the collective knowledge and resources for unrestricted use in the community

We believe that the standardization efforts of the Variant Modelling Collaboration and the expansion of that model to include the development of standards to support the capture of evidence related to genomic interpretation is critical to supporting genomic health efforts and should be a high priority of GA4GH. Furthermore, ClinGen is eager to continue our active collaboration with GA4GH and welcomes international additional partnerships to standardize in this critical area of genomic medicine.

ClinGen will integrate the VMC specification into our core products, including our Allele Model, Interpretation Model, Allele Registry, and Variant Curation Interface (VCI). We will encourage and contribute to the development of a set of common use cases that can be used to illustrate how each system has incorporated the VMC model.

ClinVar

The specifications have been shared with NCBI variation group and agreed to be consistent with how variations are represented at NCBI. Adoption of data exchange using the VMC will need to be prioritized among other NCBI initiatives. Once adopted the expectation would be variation submissions to NCBI could be done in the VMC format and will be internally converted to formats needed for internal processing at NCBI.

HL7 Clinical Genomics (FHIR)

The specifications will be shared with the HL7 Clinical Genomics members and applications to FHIR will be examined in the FHIR Genomics subgroup meetings. This will engage the various stakeholders to gather feedback. This feedback will be collected and presented to VMC. As the specification matures, the adoption of relevant definitions, structures, etc. for FHIR Genomics will be explored. For items that involve FHIR overall infrastructure, as opposed to the domain, those will be engaged for the overall product.

Appendices

VMC Normalization

The goal of the VMC normalization scheme is to rewrite an Allele with respect to its reference sequence such that the variant satisfies two criteria:

- It is minimum length. That is, the Allele cannot be written as a shorter Allele that generates the same resulting sequence; and,
- The Allele is maximally right aligned. That is, the Allele cannot be written as another Allele with numerically greater coordinates that would generate the same resulting sequence. (See Alleles must be right normalized for the rationale for preferring right normalization.)

These criteria are similar to those of the vt[8] algorithm.

The VMC normalization method is intended for all sequence types. Pseudocode for normalization and an example applied to a deletion-insertion allele are shown below. This algorithm is implemented in vmc-python/vmc/normalize.py.

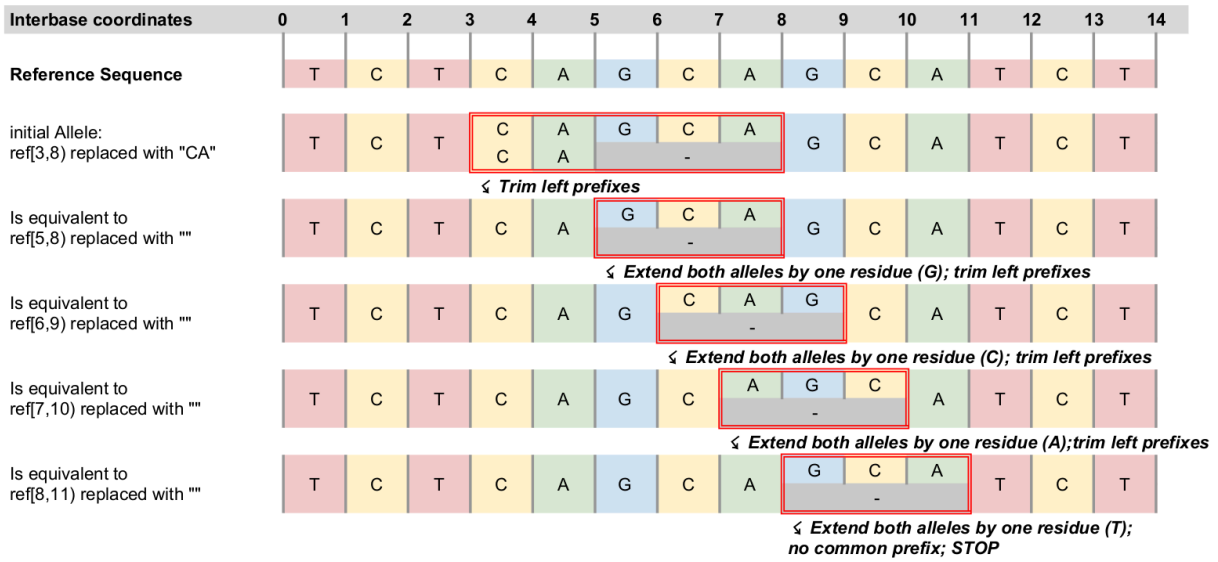
[8] http://genome.sph.umich.edu/wiki/Vt#Normalization

```
vmc_normalize_right(ref_seq, start, end, allele_seq) → pos, allele_seq
      let ref_allele_seq be the subsequence of ref_seq[start:end]
      let alleles be a list of ref_allele_seq and allele_seq
      let pfx_len be the length of common prefix of sequences in alleles
      remove the first pfx_len chars of sequences in alleles
      increment start by pfx_len

      while true do:
            if end equals length of ref_seq:
                  exit while
            let new_alleles = alleles
            append the one residue ref_seq[end:end+1] to all sequences in
                        new_alleles
            let pfx_len be the length of common prefix of sequences in new_alleles
            if pfx_len is 0:
                  exit while
            remove the first pfx_len chars of sequences in new_alleles
            increment start by pfx_len
            let alleles = new_alleles

      let repl_allele = second sequence of alleles
      return start, end, repl_allele
```

Pseudocode for VMC normalization. Alleles are VMC normalized if they are minimal length and right aligned. *ref_seq* is the reference sequence; *start* and *end* are the coordinates of the interbase interval (possibly zero width) at which the allele occurs; *allele_seq* is the sequence that occurs at the specified interval, possibly replacing the corresponding reference sequence at the interval. The notation *ref_seq[start:end]* refers to the span of sequence between (interbase) *start* and *end* positions in *ref_seq*. As an optimization, alleles may be extended by more than one residue in each iteration.



Example of VMC Normalization applied to a substitution at interbase coordinates [3,8) with the sequence “CA”. At each iteration, the normalization algorithm compares the reference allele with the replacement allele. In step 1, the reference allele sequence “CAGCA” and replacement allele sequence “CA” are compared; the common prefix “CA” is removed to result in the equivalent change of replacing [5,8) with an empty sequence. In the subsequent steps, the both alleles are extended by the reference residue to the right of the current window; if this results in a common prefix that can be trimmed, it is removed and the process continues; when there is no common prefix, the current coordinates and corresponding alleles are returned. [Figure Source]

Design Decisions

VMC contributors confronted numerous trade-offs in developing this specification. As these trade-offs may not be apparent to outside readers, this section highlights the most significant ones and the rationale for our design decisions, including:

- Use “Allele” rather than “Variant”
- Genotypes represent collections of in-phase Alleles with arbitrary ploidy
- Alleles must be right normalized
- Sequence ranges use an interbase coordinate system
- Refer to objects by id rather than inline
- Modeling Language and Serialization Options

Use “Allele” rather than “Variant”

The most primitive sequence assertion in VMC is the Allele entity. Colloquially, the words "allele" and "variant" have similar meanings and they are often used interchangeably. However, the VMC contributors believe that it is essential to distinguish the *state* of the sequence from the *change* between states of a sequence. It is imperative that precise terms are used when modeling data. Therefore, within VMC, Allele refers to a state and "variant" refers to the change from one Allele to another.

A few examples may help. The word "variant", which implies change, makes it awkward to refer to the (unchanged) reference allele. Some systems will use an HGVS-like syntax (e.g., NC_000019.10:g.44906586G>G or NC_000019.10:g.44906586=) when referring to an unchanged residue. In some cases, such "variants" are even associated with allele frequencies. Similarly, a predicted consequence is better associated with an allele than with a variant.

Genotypes represent Haplotypes with arbitrary ploidy

The VMC specification defines Haplotypes as a list of Alleles, and Genotypes as a list of Haplotypes. In essence, Haplotypes and Genotypes represent two distinct dimensions of containment: Haplotypes represent the "in phase" relationship of Alleles while Genotypes represents sets of Haplotypes of arbitrary ploidy.

There are two important consequences of these definitions:

- There is no single-location Genotype. Users of SNP data will be familiar with representations like rs7412 C/C, which indicates the diploid state at a position. In VMC, this is merely a special case of a Genotype with two Haplotypes, each of which is defined with only one Allele (the same Allele in this case).
- VMC does not define a diplotype type. A diplotype is a special case of a VMC Genotype with exactly two Haplotypes. In practice, software data types that assume a ploidy of 2 make it very difficult to represent haploid states, copy number loss, and copy number

gain, all of which occur when representing human data. In addition, assuming ploidy=2 makes software incompatible with organisms with other ploidy. VMC makes no assumptions about "normal" ploidy.

In other words, VMC does not represent single-position Genotypes or diplotypes because both concepts are subsumed by the Allele, Haplotype, and Genotypes entities.

Alleles must be right normalized

In order to standardize the presentation of sequence variation, the VMC specification requires that Alleles be right normalized (aka shifted or shuffled) using the vt algorithm. Furthermore, normalization rules should be identical for all sequence types; although this need not be a strict requirement, there is no reason to normalize in different directions based on sequence type.

The choice of algorithm was relatively straightforward: vt is well-documented, published, easily understood, easily implemented, and covers a wide-range of cases.

The choice to right normalize is more contentious. The HGVS nomenclature recommendations, originally published in 1998, require that alleles be right normalized on all sequence types. The Variant Call Format, released as a PDF specification in 2009, made the conflicting choice to write variants left (5') normalized and anchored with the previous nucleotide.

Several factors contributed to requiring right alignment in VMC:

- Right-normalized alleles on transcript and protein sequences are likely closer to their biological consequences than their left-normalized counterparts. By convention, sequences are written in the order of biological processing: 5'-to-3' for genomic sequences and for transcript sequences (regardless of strand), and N-to-C for protein sequences. Broadly speaking and with exceptions, the consequences of variation occur on the downstream end of processing. For example, a right-shifted insertion allele that induces a downstream stop codon is always closer to the consequence (termination of transcription) than its left-shifted counterpart.
- Related to, but distinct from, the above factor is that it is often easier for humans to understand certain sequence changes at their downstream locations. For example, most biologists likely imagine a trinucleotide repeat as extending from the downstream end rather than being inserted upstream. While the information content is clearly the same regardless of the mechanism or representation, there is nonetheless a strong bias toward envisioning variation on the rightmost position.

While consistency with Alleles written according HGVS recommendations was not a factor, it is an advantage of this design choice.

Sequence ranges use an interbase coordinate system

Interbase coordinate conventions are used in this terminology because they provide conceptual consistency that is not possible with residue-based systems.

> IMPORTANT: The choice of *what* to count — residues versus inter-residue positions — has significant semantic implications for coordinates. Because interbase coordinates and the corresponding 0-based residue-counted coordinates are numerically identical in some circumstances, uninitiated readers often conflate the choice of numerical base with the choice of residue or inter-residue counting. Whereas the choice of numerical base is inconsequential, the semantic advantages of interbase are significant.

When humans refer to a range of residues within a sequence, the most common convention is to use an interval of ordinal residue positions in the sequence. While natural for humans, this convention has several shortcomings when dealing with sequence variation.
For example, interval coordinates are interpreted as exclusive coordinates for insertions, but as inclusive coordinates for substitutions and deletions; in effect, the interpretation of coordinates depends on the variant type, which is an unfortunate coupling of distinct concepts.

VMC uses interbase coordinates. Interbase coordinates refer to the zero-width points before and after residues. An interval of interbase coordinates permits referring to any span, including an empty span, before, within, or after a sequence.

Refer to objects by id rather than inline

The VMC specification makes extensive use of references by logical identifiers (ids). While this methodology is common when serializing complex data structures, alternatives in which some elements are "inlined" are possible and were considered. For example, Haplotypes could have been defined as inlined Allele records rather than as sets of ids. The following factors were considered:

- **Association.** In order to associate information with Sequence, Location, Allele, Haplotype, and Genotype entities, each would need to be identifiable (i.e., have an id). Thus, using ids to declare relationships among objects was consistent with that broader use.
- **Computation**. Although most alleles are small, large insertions would create large allele structures. Repeating those for Allele definitions, then repeating such Alleles within dependent Haplotype and Genotype definitions, could be unnecessarily inefficient.
- **Extensibility**. Using ids makes it easier to extend current types. That is, the ids permit a simple, loosely-coupled abstraction while permitting the types to be differentiated as the specification evolves. For example, Haplotypes are currently comprised of Alleles.

Because this container relationship is by id, a new type of Allele, such as a hypothetical Inversion type, will be easily incorporated into Haplotypes by id. This extension would require more careful coordination if the data were inlined.

- **Consistency.** Id-based relationships among Genotypes, Haplotypes, Alleles, Locations, and Sequences requires that subordinate types are always explicitly defined.
- **Rigorousness.** Reference by id makes it more difficult for the model to enforce correctness. For example, several levels of indirection are required to ensure that Alleles in a Haplotype are defined on the same sequences. This burden was deemed insufficient to offset the above merits.

Modeling Language and Serialization Options

The VMC collaborators investigated numerous options for modeling data, generating code, and writing the wire protocol. Required and desired selection criteria included:

- language-neutral -- or at least C/C++, java, python
- high-quality tooling/libraries
- high-quality code generation
- documentation generation
- supported constructs and data types:
    - typedefs/aliases
    - enums
    - lists, maps, and maps of lists/maps
    - nested objects
- protocol versioning (but not necessarily automatic adaptation)

Initial versions of the VMC logical model were implemented in UML, protobuf, and swagger/OpenAPI, and JSON Schema. JSON Schema was deemed to provide the best combination of the above selection criteria and was chosen as the serialization format for the first release of the VMC Specification. Nonetheless, it is anticipated that some adopters of the VMC logical model may implement the specification in other protocols.

VMC Digest Collision Analysis

The current VMC Digest specifies 24-byte digests. This length is chosen extremely conservatively. It is likely that 12-byte digests would be more than adequate and future revisions of the specification may reduce the digest length. Importantly, shorter digests are readily compatible with longer ones because they will share a common prefix.

A summary of the VMC Digest Collision Analysis notebook is shown below; please see that document for details.

Applying a hash function that generates digests of length *b* bits to a corpus of *m* messages (objects), the probability *P* of one or more collisions is:

$$P(b, m) \;=\; m^2 \,/\, 2^{b+1}$$

Solving for *b* as a function of *m* and *P* yields:

$$b(m, P) \;=\; log_2(m^2/P) \;-\; 1$$

Message lengths, in bytes, are shown below for an expected number of messages and expected probabilities of collisions. Because Base64 encoding efficiency is greatest when the digest length is divisible by 3, message lengths are rounded up to nearest integer modulo 3. Thus, a message length of 24 bytes means that the probability of a chance collision in a corpus of 10^{30} objects (bottom row) is less than 1 in 10^{27} (second column).

	P, probability of collision for different digest [computed id] byte lengths							
m, messages	**≤10^{-30}**	**≤10^{-27}**	**≤10^{-24}**	**≤10^{-21}**	**≤10^{-18}**	**≤10^{-15}**	**≤10^{-12}**	**≤10^{-9}**
10^{6}	15	15	15	12	12	9	9	9
10^{9}	18	15	15	15	12	12	9	9
10^{12}	18	18	15	15	15	12	12	9
10^{15}	12	18	18	15	15	15	12	12
10^{18}	21	21	18	18	15	15	15	12
10^{21}	24	21	21	18	18	15	15	15
10^{24}	24	24	21	21	18	18	15	15
10^{30}	27	24	24	24	21	21	18	18