Qualifying Variant Set Standard

Swiss Genomics Association

2026

Version 1.0
Standard identifier: SGA-QVSS-1.0
This document defines a normative standard published by the Swiss Genomics Association.

1 Introduction

The **Qualifying Variant Set Standard** (QVSS) defines a minimal, interoperable representation of rule-based criteria used to determine whether records of genetic variation qualify under a declared genomic analysis context.

For the purposes of this standard, a **variant** is a record of genetic variation represented relative to a declared reference, coordinate system, sequence model, or other specified genomic representation. This standard does not restrict variants to a molecular class, reference genome, file format, or variant calling method.

A **qualifying variant set** is a structured and versioned specification. It records criteria used to evaluate variants, but it does not itself call variants, annotate variants, execute filters, classify variants, interpret variants, report variants, or determine downstream conclusions.

This standard defines how qualifying criteria are represented. It does not define which variants should qualify. It is independent of variant callers, annotation systems, interpretation frameworks, pipeline engines, reporting systems, statistical methods, and clinical decision processes.

The purpose of QVSS is to support portable, auditable, and reproducible specification of qualification logic. A QV set can be used by different implementations, provided that the implementation can resolve the declared inputs and evaluate the declared rules deterministically.

2 Scope

This standard governs the representation of QV sets only.

2.1 In scope

QVSS defines:

- the normative definition of a QV set
- required identity fields
- the minimal structure of rules and rule statements
- core logical operators and atomic comparison operators
- datatype and missing value requirements for field-resolving statements
- profile and extension declaration requirements
- integrity and provenance requirements
- minimal QV application record requirements

2.2 Out of scope

QVSS does not define:

- variant calling, normalisation, or variant representation formats
- annotation systems or annotation field names
- genome build conversion or coordinate systems
- disease-specific criteria or clinical gene panels
- clinical classification, pathogenicity interpretation, or reportability
- statistical inference, association testing, or decision thresholds
- pipeline execution, implementation logic, or software interfaces
- how qualified variants should be retained, excluded, ranked, reported, interpreted, or used downstream
- the correctness of any field, annotation, resource, threshold, rule, or external predicate

3 Terminology and conventions

The key words **MUST**, **MUST NOT**, **SHOULD**, and **MAY** are to be interpreted as described in RFC 2119.

A **variant** is an evaluated genomic item supplied by an upstream process. This standard does not restrict its molecular type, representation, coordinate system, or source.

A **qualifying variant** is a variant that satisfies the declared qualification rule of a QV set under the declared application context. If no qualification rule is declared, the QV set defines rule outcomes only and does not define a single set-level qualifying status.

A **QV set** is the normative specification object defined by this standard. It is a structured, versioned set of criteria used to evaluate whether variants qualify.

A **rule** is a named declarative statement in a QV set. A rule evaluates one or more rule statements and produces a rule outcome.

A **rule statement** is an evaluable element within a rule. A rule statement may be an atomic comparison condition, a compound expression, an aggregation statement, or an extension-defined predicate.

An **atomic comparison condition** is a field-resolving statement containing a field, an operator, and, except for unary operators, a value.

A **compound expression** combines one or more child rule statements using a logical operator.

An **aggregation statement** evaluates grouped, counted, or multi-record information. Aggregation statements are outside the core rule language unless defined by a declared conformance profile or extension.

An **extension-defined predicate** is a rule statement whose semantics are defined outside core QVSS by a declared conformance profile or extension.

A **field** is an implementation-resolvable identifier for a quantity, annotation, attribute, or derived value evaluated for a variant or other declared context.

An **operator** defines the comparison or test applied to a field in an atomic comparison condition.

A **value** is the literal, list, or structured value against which a field is evaluated.

A **rule outcome** is the result of evaluating a rule for an evaluated variant. Rule outcomes are implementation-level outputs and are not assigned clinical, biological, or statistical meaning by this standard.

A **QV application** is the act of applying a QV set to one or more evaluated variants.

A **QV application record** is an audit record identifying the QV set and standard version used in a QV application.

4 Definition of a qualifying variant set

A **qualifying variant set** is a structured, versioned specification containing one or more rules used to evaluate whether variants qualify under a declared context.

A QV set **MUST** be represented in a machine-readable structured format. JSON and YAML are suitable representations, provided that the required fields and rule semantics defined by this standard are preserved.

An active QV set **MUST** contain stable identity fields and at least one rule. Registry

records for deprecated or withdrawn releases **MAY** omit active rule content if they point to an archived released representation.

A QV set does not itself execute an analysis. It is interpreted by an implementation. The implementation is responsible for resolving fields, applying operators, evaluating logical combinations, applying declared profiles or extensions, and recording the QV application.

A QV authoring tool **MAY** use a line-based, form-based, or key-value input syntax to create a QV set. Such authoring syntax is not itself the normative QV set unless it serialises to the structured representation required by this standard.

5 QV set structure

A QV set **MUST** contain the following information:

Information	Required	Meaning
`qvss_version`	yes	version of this standard
`qv_set_id`	yes	stable identifier for the QV set
`version`	yes	version of the QV set
`title`	yes	human-readable title
`rules`	yes	named rules used for qualification or rule outcomes

A QV set **MAY** contain the following information:

Information	Meaning
`description`	human-readable description
`created`	creation or release date
`authors`	authors or responsible organisation
`licence`	reuse terms
`status`	release status, such as draft, active, deprecated, or withdrawn
`tags`	indexing or discovery terms
`context`	declared application context
`inputs`	declared input fields or resources
`qualification`	rule used to determine final set-level qualification
`descriptions`	non-normative patient, PPIE, or other descriptive text
`notes`	non-normative free text notes
`profiles`	declared conformance profiles
`extensions`	implementation-specific or profile-specific extensions

The required information **MAY** be represented as top-level fields or inside a structured metadata object, provided that the mapping is unambiguous.

A QV set **MAY** use a grouped layout containing `meta`, `filters`, `criteria`, and `notes`, provided that the required identity information and rules can be resolved deterministically.

Additional fields **MAY** be present. Implementations **MUST NOT** treat unrecognised fields as changing the normative meaning of the QV set unless those fields are declared by a recognised conformance profile or extension.

5.1 Identity fields

The `qvss_version` field **MUST** identify the version of this standard used by the QV set.

The `qv_set_id` field **MUST** be stable within the namespace in which the QV set is published, stored, or exchanged.

The `version` field **MUST** identify the version of the QV set.

Published QV set releases **MUST NOT** be modified without changing the version or release identifier.

The `title` field **MUST** provide a human-readable name for the QV set.

5.2 Context

The `context` field **MAY** describe the intended application context of the QV set.

Context information **MAY** include analysis context, variant domain, organism, genome build, intended use, source protocol, or other application information.

This standard does not define a required vocabulary for context values. Context information is descriptive unless a declared conformance profile defines it as normative.

5.3 Inputs

The `inputs` field **MAY** declare fields, resources, or other dependencies required to evaluate the QV set.

If input fields are declared, each declared field **MUST** have a stable identifier within the QV set or declared profile. A field declaration **SHOULD** include source, path, datatype, cardinality, and missing value behaviour where these are required for reproducible evaluation.

If external resources are required for deterministic rule evaluation, the QV set **SHOULD** declare sufficient provenance to identify the resource. Registered QV sets **MUST** include or reference checksums for resources distributed as part of the release.

This standard does not define required genomic fields, annotation names, file formats, coordinate conventions, or resource types.

5.4 Qualification

A QV set **MAY** declare a `qualification` object identifying the rule that determines final set-level qualification.

If no qualification rule is declared, the QV set defines rule outcomes only and does not define a single final qualifying status.

A qualification rule **MUST** refer to a rule defined in the QV set or by a declared profile or extension.

5.5 Rules

A QV set **MUST** contain one or more named rules.

Each rule identifier **MUST** be unique within the QV set.

Each rule **MUST** define at least one rule statement.

A rule **MAY** include a human-readable description. A description does not alter the normative evaluation semantics of the rule.

Rules **MAY** be grouped by purpose, including but not limited to `filters` and `criteria`. Such grouping **MUST NOT** alter rule semantics unless grouping semantics are declared by this standard, a conformance profile, or an extension.

5.6 Descriptions and notes

The `descriptions` field **MAY** contain patient, consent, PPIE, or other descriptive text. These descriptions are non-normative unless a declared profile defines otherwise.

Patient-specific descriptions **SHOULD NOT** be included in public QV set releases unless explicitly intended and appropriately authorised.

The `notes` field **MAY** contain non-normative free text.

Descriptions and notes **MUST NOT** change the normative meaning of a QV set.

6 Profiles and extensions

QVSS defines a limited core representation.

A QV set **MAY** declare one or more conformance profiles. A conformance profile defines additional constraints, required fields, vocabularies, rule types, operator semantics, resource semantics, or domain-specific behaviour.

A QV set **MAY** declare extensions. Extensions are implementation-specific or profile-specific additions outside core QVSS.

A QV set that uses additional rule statement types, operators, aggregation semantics, resource semantics, or external predicates **MUST** declare the relevant profile or extension if those constructs are normative for evaluation.

A declared profile or extension **SHOULD** include an identifier and version. A profile or extension used to alter normative evaluation semantics **MUST** be identifiable by name

and version. Recognition **MAY** be local, organisational, registry-based, or community-endorsed.

An extension field **MUST NOT** change the normative meaning of required QVSS fields unless the change is explicitly defined by a declared conformance profile.

Implementations **MUST** ignore unrecognised extension fields unless the declared conformance profile requires otherwise.

7 Rule representation

A rule is atomic, compound, aggregation-based, or extension-defined.

An **atomic rule** contains a single atomic comparison condition.

A **compound rule** contains one or more rule statements or nested rule expressions combined using a declared logical operator.

An **aggregation-based rule** evaluates grouped, counted, or multi-record information. Aggregation-based rules are not part of the core QVSS rule language unless defined by a declared profile or extension.

An **extension-defined rule** uses a rule type, predicate, or evaluation model defined by a declared profile or extension.

7.1 Atomic comparison conditions

A core atomic comparison condition using a binary comparison operator **MUST** contain:

- a `field`
- an `operator`
- a `value`

A core atomic comparison condition using a unary operator **MUST** contain:

- a `field`
- an `operator`

Unary operators **MUST NOT** require a value.

A core atomic comparison condition **MAY** contain `description`, `datatype`, and `missing` fields.

Not all QV statements are atomic comparison conditions. Metadata, notes, logical grouping, aggregation rules, implementation hints, and extension-defined predicates **MAY** be represented without a core comparison operator, provided that their role is declared unambiguously.

7.2 Datatypes

QVSS core datatypes are `string`, `number`, `integer`, `boolean`, `date`, `enum`, `list`, and `object`.

A declared field or value **MAY** include a datatype. Implementations **MUST** evaluate comparisons deterministically. Implementations **MUST NOT** silently coerce incompatible datatypes unless coercion behaviour is defined by this standard, a declared profile, or an extension.

Profiles and extensions **MAY** define additional datatypes or stricter datatype rules.

7.3 Logical operators

A compound rule in core QVSS **MUST** declare one of the following logical operators:

Operator	Meaning
`all_of`	all child rule statements must be satisfied
`any_of`	at least one child rule statement must be satisfied
`none_of`	no child rule statement may be satisfied
`not`	the single child rule statement is negated

The logical operator names `all_of`, `any_of`, `none_of`, and `not` are normative in QVSS version 1.0.

7.4 Comparison operators

QVSS version 1.0 defines a core set of comparison operators for atomic comparison conditions.

A core atomic comparison condition **MUST** declare an `operator`. The operator **MUST** be one of the core operators defined in this section, unless the QV set declares a recognised extension or conformance profile that defines additional operators or condition types.

A QV set **MAY** use additional operators only when those operators are declared by a recognised extension or conformance profile.

A QV set that uses additional operators is not core QVSS compliant unless the relevant extension or profile is also declared.

Operator	Meaning
`==`	equal to
`!=`	not equal to
`<`	less than
`<=`	less than or equal to
`>`	greater than
`>=`	greater than or equal to
`in`	field value is in the declared value set
`not_in`	field value is not in the declared value set
`exists`	field is present
`not_exists`	field is absent
`contains`	field contains the declared value
`matches`	field matches the declared pattern
`overlaps`	field overlaps the declared value, resource, or interval

The operators `exists` and `not_exists` are unary operators.

This standard does not define field-specific semantics for any operator. For example, it does not define how genomic interval overlap, string matching, set membership, or pattern matching is implemented.

Implementations **MUST** evaluate declared operators deterministically and **SHOULD** document implementation-specific semantics where ambiguity is possible.

When `overlaps` is used for genomic intervals, the QV set or declared profile **SHOULD** specify reference assembly, coordinate convention, interval closure, minimum overlap requirement, and resource provenance.

The core operator set is intentionally limited. It supports portable representation of common rule-based qualification logic. It is not a complete language for all bioinformatic computation.

Complex assessments, including inheritance models, compound heterozygosity, phasing, transcript consequence selection, external pathogenicity scoring, statistical inference, probabilistic classification, RNA outlier calling, methylation classification, or structural variant interpretation, **SHOULD** be represented either as declared input fields evaluated using core operators, or by an explicitly declared extension or conformance profile.

Arbitrary executable predicates are outside the scope of core QVSS.

7.5 Aggregation statements

Aggregation statements evaluate grouped, counted, or multi-record properties.

Aggregation statements **MAY** be used only when their semantics are defined by a declared conformance profile or extension.

Aggregation statements **MUST** identify the fields, grouping variables, counted records, threshold, and missing value behaviour required for deterministic evaluation, unless the declared profile or extension defines an equivalent representation.

A QV set using aggregation statements **MUST NOT** claim core rule-language compliance for those aggregation statements. It **MAY** claim QVSS profile compliance if the aggregation semantics are defined by a declared profile or extension.

7.6 External predicates

A QV set **MAY** reference externally computed predicates, such as model outputs, annotation flags, inheritance classifications, or prior pipeline assessments.

An external predicate used as a declared input field **MAY** be evaluated using core atomic comparison operators.

An external predicate whose computation is part of the QV rule semantics **MUST** be defined by a declared conformance profile or extension.

QVSS does not define the correctness of any external predicate.

7.7 Missing values

A field-resolving rule statement **MAY** declare missing value behaviour using the `missing` field.

The allowed values of `missing` are:

Value	Meaning
`fail`	a missing field does not satisfy the rule statement
`pass`	a missing field satisfies the rule statement
`unknown`	the rule statement outcome is unknown
`error`	application is invalid if the field is missing

If `missing` is not declared for a field-resolving rule statement, the missing value behaviour **MUST** be treated as `unknown`.

Implementations **MUST** handle missing values deterministically.

Profiles and extensions **MAY** define additional missing value behaviour for rule statement types outside core QVSS.

7.8 Unknown outcome propagation

Unknown outcomes **MUST** propagate through core logical operators according to the following three-valued logic.

Expression	Result
`all_of(true, true)`	`true`
`all_of(true, unknown)`	`unknown`
`all_of(true, false)`	`false`
`all_of(false, unknown)`	`false`
`any_of(false, false)`	`false`
`any_of(false, unknown)`	`unknown`
`any_of(true, unknown)`	`true`
`none_of(false, false)`	`true`
`none_of(false, unknown)`	`unknown`
`none_of(true, unknown)`	`false`
`not(true)`	`false`
`not(false)`	`true`
`not(unknown)`	`unknown`

For more than two child statements, implementations **MUST** apply the same logic across all children.

7.9 Rule outcomes

A rule outcome **MUST** be derived from declared rule statements, operators, values, logical structure, missing value behaviour, and declared profiles or extensions.

QVSS does not require a specific serialisation of rule outcomes.

If rule outcomes are recorded, the representation **SHOULD** distinguish satisfied, not satisfied, and unknown outcomes.

A qualifying variant is identified only under the declared QV set, implementation context, input data, and application record used for the QV application.

8 Integrity and provenance requirements

A QV set **MUST** include sufficient identity information to allow unambiguous reference to the QV set and standard version.

At minimum, a QV set **MUST** specify:

- the QVSS version applied
- the QV set identifier
- the QV set version
- the QV set title
- one or more uniquely identified rules

A QV set that is published, exchanged, or registered **SHOULD** include or reference a checksum.

A registered QV set **MUST** include or reference a checksum sufficient to verify the exact released file or canonical representation.

A registry **MUST** state whether a checksum applies to the exact released file bytes or to a canonical representation. QVSS 1.0 registries **SHOULD** use exact released file byte checksums unless a canonicalisation profile is declared.

Rule identifiers **MUST** be stable within a QV set version.

Changing a rule identifier, rule statement, operator, value, logical structure, missing value behaviour, declared profile, or normative extension creates a different QV set version and **MUST** be documented.

9 QV publication and registry records

A QV set **MAY** be published, exchanged, or registered in a public or private QV database.

A registered QV set **MUST** include:

- the QV set identifier
- the QV set version
- the QVSS version
- the released file or canonical representation
- a checksum for the released file or canonical representation

A registered QV set **SHOULD** include:

- release date
- status
- authors or responsible organisation
- licence
- declared profiles or extensions
- provenance for external resources
- a human-readable description

Publication or registration of a QV set does not assert that the criteria are clinically valid, statistically valid, biologically correct, or suitable for a particular use.

10 QV application record

A QV application record identifies the QV set used in a QV application.

A QV application record **MUST** include:

- the QV set identifier
- the QV set version
- the QVSS version

A QV application record **SHOULD** include:

- the QV set checksum, where available
- the date or time of application
- the implementation name and version, where available
- the declared profiles or extensions used
- a reference to the input data or upstream dataset, where appropriate

This standard does not define an output format for qualified variants. It defines only the minimal audit information required to identify the QV set applied.

11 Conformance requirements

A QV set is **core QVSS compliant** if it satisfies all required structural, identity, rule, logical operator, core atomic comparison operator, datatype, and missing value requirements defined by this standard, and does not require any undeclared extension or profile for evaluation.

A QV set is **QVSS profile compliant** if it satisfies the core QVSS requirements and declares a recognised conformance profile that defines any additional required fields, vocabularies, rule types, operators, resource semantics, aggregation semantics, or domain-specific behaviour used by the QV set.

A QV set is **QVSS extension compliant** if it satisfies the core QVSS requirements and declares the extension required to interpret any additional operators, predicates, aggregation rules, resource semantics, or implementation-specific rule statement types.

A QV application record is **QVSS compliant** if it records the QV set identifier, QV set version, and QVSS version used in the application.

An implementation is **QVSS aware** if it can read a QV set and evaluate its declared rules according to the semantics of this standard, or can report which required QVSS features, profiles, or extensions it does not support.

An implementation **MUST NOT** claim full QVSS compliance for a QV application if it ignores required fields, unsupported operators, declared missing value behaviour, required logical structure, declared profiles, or normative extensions.

12 Examples

12.1 Minimal core QV set

The following informative example illustrates a minimal QV set using generic field names. The example does not define a clinical, biological, or statistical interpretation of the variant.

```
qvss_version: "1.0"
qv_set_id: "example_minimal_qv_set"
version: "1.0.0"
title: "Minimal QV set"

inputs:
  fields:
    quality:
      source: "implementation"
      datatype: "number"
      cardinality: "single"

rules:
  minimum_quality:
    field: "quality"
    operator: ">="
    value: 20
    datatype: "number"
    missing: "unknown"

qualification:
  rule: "minimum_quality"
```

In this example, `quality` is an implementation-resolvable field. QVSS does not define how that field is produced or what biological quantity it represents.

12.2 Grouped QV set layout

The following informative example illustrates a grouped layout compatible with authoring systems that separate metadata, filters, criteria, and notes.

```
qvss_version: "1.0"

meta:
  qv_set_id: "qv_gwas_common_v1_20250827"
  version: "1.0.0"
  title: "GWAS common QC"
  description: "Common GWAS criteria expressed in tool-agnostic form."
  created: "2025-08-27"
```

```
  authors:
    - "Alice"
    - "Bob"
  tags:
    - "GWAS"
    - "QC"
    - "PCA"

inputs:
  fields:
    MAF:
      source: "PLINK"
      datatype: "number"
      cardinality: "single"
    HWE_P:
      source: "PLINK"
      datatype: "number"
      cardinality: "single"

filters:
  maf_minimum:
    description: "Minimum MAF"
    field: "MAF"
    operator: ">="
    value: 0.01
    datatype: "number"
    missing: "unknown"

  hwe:
    field: "HWE_P"
    operator: ">="
    value: 1e-6
    datatype: "number"
    missing: "unknown"

rules:
  final_qualification:
    logic: "all_of"
    conditions:
      - ref: "maf_minimum"
      - ref: "hwe"

qualification:
  rule: "final_qualification"

notes:
  - "Implementation details are non-normative unless declared by a profile."
```

This grouped layout is valid only if the implementation maps grouped fields to the

required QV set identity and rule information unambiguously.

12.3 Profile-defined aggregation

The following informative example illustrates an aggregation statement. The aggregation semantics are not core QVSS semantics. They require a declared profile or extension.

```
qvss_version: "1.0"
qv_set_id: "example_compound_het_profile"
version: "1.0.0"
title: "Example aggregation rule"

profiles:
  - id: "https://qvss.org/profiles/aggregation/1.0"
    name: "qvss-aggregation"
    version: "1.0.0"

rules:
  possible_compound_heterozygous:
    type: "aggregation"
    description: "At least two qualifying records grouped by sample and gene."
    group_by:
      - "sample"
      - "SYMBOL"
    count:
      operator: ">="
      value: 2
    missing: "unknown"
```

12.4 QV application record

The following informative example illustrates a minimal QV application record.

```
qv_application:
  qv_set_id: "example_minimal_qv_set"
  qv_set_version: "1.0.0"
  qvss_version: "1.0"
  qv_set_checksum_sha256: "..."
  applied_at: "2026-01-01"
  implementation:
    name: "example implementation"
    version: "1.0.0"
```

The application record does not assert that any variant is causal, pathogenic, reportable, statistically significant, or clinically actionable. It records only which QV set was applied.

13 Compatibility with authoring syntax

This section is informative. QVSS authoring tools may use simpler input syntax if they serialise to a valid structured QV set.

Authoring syntax	Canonical meaning
`logic: and`	`logic: all_of`
`logic: or`	`logic: any_of`
`group: any_of:start`	start of nested `any_of` expression
`group: any_of:end`	end of nested `any_of` expression
`logic: keep_if`	action hint or profile-defined action
`logic: exclude_if`	action hint or profile-defined action
`filters` and `criteria`	grouped rule namespaces

Action hints such as `keep_if` and `exclude_if` are not core logical operators unless a declared profile defines them as normative.

14 Non-goals and interpretation

QVSS does not assert that a qualifying variant is causal, pathogenic, clinically reportable, statistically significant, or clinically actionable.

QVSS does not define whether a qualifying variant should be retained, excluded, reported, prioritised, interpreted, or used in downstream inference. Those actions are external to this standard.

QVSS does not define the correctness of any field, annotation, resource, threshold, rule, profile, extension, or external predicate. It defines only how a QV set represents qualification criteria.

Rules in a QV set are not required to be independent. Correlation, overlap, or dependency between rules is expected in practical applications. QVSS does not address the detection, modelling, or correction of correlated rules.

QVSS does not make a QV database, QV builder, QV registry, software implementation, or submitted QV file clinically valid by default.

15 Versioning

This document defines **QVSS version 1.0**.

Future revisions **MUST** preserve the required identity fields and core rule semantics defined herein, or explicitly define a version transition mechanism.

Published QV set releases **MUST NOT** be modified without changing the version or release identifier.