# JSON Variant Call Format Specification

Version 0.1

November 16, 2020

## Contents

# 1 Rationale

## 1.1 Variant calls in genome graphs

The Variant Call Format (VCF) is a tab-delimited data format describing genetic variation occurring with respect to a linear reference genome. Here we define an extension to VCF for genome graphs, which are graph structures representing genetic variants occurring on potentially multiple reference genomes.

The format uses JavaScript Object Notation (JSON), a commonly used data format, and we thus call it JSON VCF or jVCF.

### 1.1.1 Graph restrictions

jVCF describes a specific type of genome graph which we call non-crossing, directed, acyclic graphs (NCDAGs). Consider a graph where sequence is stored in the nodes. Then three properties must hold:

1. Edges are *directed*, leading to nodes with higher position on the genomic segment/chromosome.
2. There are no cycles in the graph, so the graph is *acyclic*.
3. Variant sites are *non-crossing*: two variant sites share a node if and only if one site is entirely contained within the other. Contained means all paths inside the contained variant site are also paths in the containing site.

## 1.2 Description of JSON

The JSON format is defined here: https://www.json.org/json-en.html.

Briefly, JSON consists of *objects* and *arrays*:

- An object is an unordered collection of key/value pairs enclosed in curly brackets (‘{’ and ‘}’).
- An array is an ordered collection of values enclosed in square brackets (‘[’ and ‘]’).

In an object, a key is a string. Strings are enclosed in double quotes (‘"’). In objects and arrays, values can be:

- a string
- a number
- one of the literals ‘true’, ‘false’ or ‘null’
- an array
- an object

## 2 Example

Here is a toy genome graph:

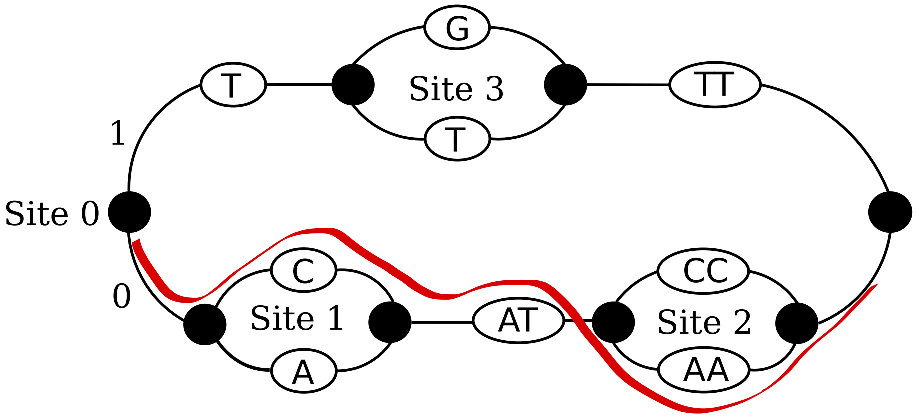

Figure 1: Genome graph with haploid genotyped sample as red path

In Figure 1, black nodes mark site entry and exit points. Each variant site is labeled by a unique positive integer ID, and each outgoing branch from the start of a variant site is labeled by a unique positive integer, called its **haplogroup**.

Assume the ploidy is 1, and we are genotyping a sample whose genotyped path in the graph is the red line. The jVCF of this genotyped sample would look like this:

```
{
  "Sites":[
    {"ALS":["AATAA","CATAA"],"FT":[[]],"GT":[[1]],"HAPG":[[0]],"POS":1,"SEG":"myRef"},
    {"ALS":["A","C"],"FT":[[]],"GT":[[1]],"HAPG":[[1]],"POS":1,"SEG":"myRef"},
    {"ALS":["AA"],"FT":[[]],"GT":[[0]],"HAPG":[[0]],"POS":4,"SEG":"myRef"},
    {"ALS":["T"],"FT":[[]],"GT":[[null]],"HAPG":[[]],"POS":2,"SEG":"myRef"}
  ],
  "Site_Fields": {
    "ALS": {
      "Desc": "Alleles at this site"
    },
    "FT": {
      "Desc": "Filters failed in a sample"
    },
    "GT": {
      "Desc": "Genotype"
    },
    "HAPG": {
      "Desc": "Sample haplogroups of genotyped alleles"
    },
    "POS": {
      "Desc": "Position on reference or pseudo-reference"
    },
    "SEG": {
      "Desc": "Segment ID"
    }
  },
  "Samples": [
    {
      "Desc": "mySample description",
      "Name": "mySample"
    }
  ],
  "Filters": {
    "MINQ": {
      "Desc": "Call is below minimum quality"
    }
  },
  "Model": "myGenotypingModel",
  "Child_Map": {
    "0": {
      "0": [1,2],
      "1": [3]
    }
  },
  "Lvl1_Sites": [0]
}
```

# 3 Specification

There are 7 required keys in jVCF, which can appear in any order. We list each of the required keys below, giving for each the type and description of its corresponding value.

Extra keys beyond these can be used.

## 3.1 Sites

**Type** Array

**Description** Each element is a Site object

### 3.1.1 Site objects

A Site object is like a data line in the VCF format. It describes variant calls at one site. The following keys are required:

| Key | Value Type |
|---|---|
| ALS | Array of strings |
| SEG | String |
| POS | Number |
| GT | Array of array of numbers, 'null' or empty |
| HAPG | Array of array of numbers or empty |
| FT | Array of arrays of strings or empty |

Here is a more detailed explanation of the keys:

- `ALS`: does not need to store all possible alleles at the variant site. It must store at least one allele which is the 'reference' for this site. The reference allele must be the allele obtained by following haplogroup 0 from the start of the site to the end of the site. All called alleles should also be present in `ALS`.

- `SEG`: the name of the genomic segment the site lies on, e.g. chromosome or the name of a reference genome.

- `POS`: 1-based position, offset from the last non-nested site (see Lvl1_Sites). The position of sites contained in other sites is expressed relative to the reference path through the containing site, which is obtained by following haplogroup 0.

- `GT` and `HAPG`: are arrays of arrays. The two array levels are:

  1. Samples. This should have the same size as the array in Samples.
  2. Ploidy. This should have one entry per chromosome copy, or chromosome population (eg tumour subclones or mixed infections).

For example, to access the first sample's genotype calls at a site, you access the 0th element of the GT array. If you have a diploid genotyped sample, the first and second chromosome calls are the 0th and 1st elements of that array.

- FT: also an array of arrays. The first level is the samples, and the second the filters set for that sample.

You can, of course, use more keys. For each used key, Site_Fields needs to provide a description of its meaning. The required keys here must be present in Site_Fields.

## 3.2 Site_Fields

**Type** Object

**Description** Each entry describes the fields that can appear in Site objects.

For each key which appears in Site objects, a corresponding key must be present here. The following keys are required to be present:

| Key | Meaning |
|---|---|
| ALS | Genotyped alleles at this site |
| SEG | Segment (eg chromosome) on which the site lies |
| POS | 1-based position relative to haplogroup reference |
| GT | Genotype calls |
| HAPG | Haplogroup calls: haplogroup the called allele(s) lie on |
| FT | Filters set |

Each key corresponds to an object with at least a "Desc" key which gives a description of the field.

## 3.3 Samples

**Type** Array

**Description** Each entry is an object describing one genotyped sample.

Each entry must contain the following keys: "Name" and "Desc", giving the sample name and its description. The order of the samples matters: it must match that in Site objects.

## 3.4 Filters

**Type** Object

**Description** Describes the filters that can appear in Site objects.

Each filter that appears under the “FT” key in Site objects must be described here. Each key gives the filter name and each value is an object with at least a “Desc” key giving the filter’s description.

### 3.5 Model

**Type** String

**Description** The name, or a description, of the genotyping model that was used.

### 3.6 Child_Map

**Type** Object

**Description** Records parent/site relationships between variant sites and on which haplogroups child sites lie.

Each key corresponds to a site index in the Sites array. The value associated to each key is a Child listing object.

Starting from each entry of the Lvl1_Sites, the Child_Map can be recursively traversed to recover the full parent/child site structure of the graph.

#### 3.6.1 Child listing

A Child listing is an object whose keys are haplogroups and whose values are arrays of site indices. It records, for a given parent site, what child sites it has, and which haplogroups the child sites fall under.

### 3.7 Lvl1_Sites

**Type** Array

**Description** Lists the indices of sites which are not nested in others.

Each element of this array corresponds to a site which has no parents; the value is the index at which to access this site in the Sites array.

Each site referred to in this array may or may not have children, depending on whether or not it appears in Child_Map.