Beacon Scouts: Genomic Variants Use Cases & Examples

This document develops a set of genomic variant types and associated query formats to be supported by the Beacon protocol. The initial development focuses on the possibly limited, but unambiguous definition of query formats, driven and documented through real-world use cases.

References **1**

Conventions Followed in the Document **1**

Use of Positional Parameters **2**

Variant Types, Documentation and Example Queries **2**

INS (Insertion) 2
DEL (Deletion) 3
DUP (Duplication) 5
LOH (loss of representation of second allele, with or without copy number change) 8
INV (inversion) 8
TL (Translocation) 9
Proposal: BRK (Breakpoint) 9
ME (Mobile elements insertion /deletion) 10
CNV - (non directional CNVs) - do we allow cnv queries? / complex CNVs 10
Tandem Duplication 11

Name Based Queries **11**

Topics for discussion 11

Technical considerations **12**

Format of GET queries 12

References

- Sequence Ontology structural variation types in dbVar
- Beacon v2 GenomicVariantFields
- B. Nowakowska: Clinical interpretation of copy number variants in the human genome [article]

This document develops a set of structural variant types and associated query formats which will be supported by the Beacon protocol. The focus of the initial development is on the possibly limited, but unambiguous definition of query formats, driven and documented through real-world use cases7-0407-4 for some size discussion

Conventions Followed in the Document

- Example queries are constructed using the parameters from the Beacon v2 GenomicVariantFields
- While Sequence Ontology terms act as a guidance for the definition of Beacon query types - and SO terms may be used as *variantType* values - the use is developed in this document and will become the authoritative definitions in the Beacon specification and associated documentation
- The document uses human example data (where this applies, e.g. for gene locations) and omits the fixed parameters (e.g. `species=NCBI:txid9606`, `assemblyId=GRCh38`, `datasetIds=...` etc.)

- Legend for the sequence graphs:
  - `?????` : bases corresponding to end, start interval of the example query
  - `xxxxx` : bases corresponding to the genomic feature (e.g. TP53)
  - `*****` : bases of matched variants or the matched parts of unmatched
  - `=>` : points to end of the chromosome
  - `!!!--!`: Not matched, with the highlight color indicating conditions breaking the requirement

Example: Find (any) genomic variants involving the specified genomic feature

```
Genomic region: ----------------------xxxxxx-------------------------=>
Start match:    ???????????????????????????
End match:                             ?????????????????????????????

Matched variants
Match1:         --------------------******************----------------=>
Match2:         ----------------------------*****-------------------=>

Not Matched
Non Match1:     -------------------!!!------------------------------=>
Non Match2:     ----------------------------------!!!!!!!-----------=>
```

Example: Find genomic variants completely covering the specified genomic feature

```
Genomic region: ----------------------xxxxxx-------------------------=>
Start match:    ??????????????????????------------------------------=>
End match:      ----------------------??????????????????????????????=>

Matched variants
Match1:         --------------------******************----------------=>
Match2:         ----------------------******------------------------=>

Not Matched
Non Match1:     -------------------!!!------------------------------=>
Non Match2:     ----------------------**********----------------------=>
```

Use of Positional Parameters

The use of positional parameters influences the interpretation of the query. At the moment, the indicated query types (Range Query, Bracket Query …) are *implicit*; however, a specific definition in the schema may be evaluated.

1. single **`start`** parameter
   - used for the occurrence of a variant at this exact position
   - usually for precise replacements (ref > alt) with indicated base values
2. single **`start`** and single **`end`** parameter
   - indicates a **Range Query**
   - used to find any variant (with optional specified **`variantType`** or **`alternateBases`**) inside or with overlap to the specified range

3. two **`start`** and two **`end`** parameters
    - indicates a **Bracket Query**
    - used for finding (structural) variants of a certain, usually variable extend, where the start of the variant is inside the `start[0],start[1]` interval, and the end is in the `end[0],end[1]` interval
    - a precise match for **start** and/or **end** position is indicated with `start[1]=start[0]+1` and/or `end[1]=end[0]+1`; this allows to disambiguate from Range Queries were only single values are provided for **start** and **end**
    - typical examples include
        - finding all duplications of a gene involving it's complete coding region; this is achieved by having the end of the start interval before the gene's start position, and end interval beginning after the gene's last base position
        - finding all "focal" deletions in a gene by limiting the maximum size of a detected deletion start[0] -> end[1]

Comparison of Range and Bracket Query

A **Range Query** in principle could also be expressed as **Bracket Query** (see below). However, the recommendation is to use the Range Query type if possible; this convention has been followed throughout the examples.
Whereas in a **Range Query** single start and end values indicate a required match with any bases _between_ those values, in a **Bracket Query** the same range is defined through the overlap between the start and end query brackets.

Range Query

```
Start pars:     <=--------------------?----------=>
End pars:       <=----------------------?--------=>
```

Bracket Query

```
Start pars:     <=??????????????????????????----------=>
End pars:       <=----------????????????????????????=>
```

Variant Types, Documentation and Example Queries

The following examples detail "real world use cases", using the parameters from the Beacon v2 GenomicVariantFields (referencename, start, end, variantType)

INS (Insertion)

Definitions:

- SO:0000667: The sequence of one or more nucleotides added between two adjacent nucleotides in the sequence
- Beacon: A sequence or structural variation even in which a known or unknown piece of DNA is inserted at a genomic position.

Example: Find insertion event in gene *TP53* (17:7669607-7676593) or in close proximity (±~5000bp)

Provided by: Diana Lemos

Notes:

- This is an application of a **Range Query**
- For this query the mapping position of TP53 (17:7669607-7676593) has to be known
- 7669607 - ~5000 =7664000
- 7676593 + ~5000 = 7682000

Query structure:

```
referenceName: "17"
start: 7664000
end: 7682000
variantType: "INS"

?referenceName=17&start=7664000&end=7682000&variantType=INS
```

```
No Match1:      <=!!!!!!!!!!!!!!!!----------------------------------------=>
No Match2:      <=----------------------------------------!!!!!!!!----------=>
No Match3:      <=--------------!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!=>
```

Example: Is an insertion described in a region ranging from chromosome 2 from 47475025 to 47475270?

Provided by: David Salgado
Notes:

- This is an application of a **Range Query** (*MSH2*, ENST00000233146, exon 12).

Query structure:

```
referenceName: "2"
start: 47475025
end: 47475270
variantType: "SO:0000667"

?referenceName=2&start=0&end=10000&variantType=SO:0000667

   Genomic region:<=------|XXXXXXXXXXXXX|-----------------------------------=>
Range Query
   Start pars:     <=------?-------------------------------------------------=>
   End pars:       <=---------------------?-----------------------------------=>
Matched variants
   Match1:         <=--------************-------------------------------------=>
   Match2:         <=********************-------------------------------------=>
   Match3:         <=------*****----------------------------------------------=>
Not Matched
   No Match1:      <=-------------------------------------------!!!!!!!!!!!!=>
   No Match2:      <=-------------------------------------!!!!!!!!!!!!!!!!!!!!=>
   No Match3:      <=!!!----------------------------------------------------=>
```

DEL (Deletion)

Definitions:

- SO: The point at which one or more contiguous nucleotides were excised
- Beacon: Any variation of copy number of a genomic segment w/o size limitation imposed by the protocol[1,2] resulting in a net loss of copies compared to the expected allelic number at this locus.

Example 1: Find (any) deletions involving the *TP53* gene locus

Provided by: Michael Baudis
Note:

- This is a **Range Query**, in which a specified event with full or partial overlap between start and end is matched. This means than *any* overlap will be counted as match
- For this query the mapping position of *TP53* (17:7669607-7676593) has to be known.

Query structure:

```
referenceName: "17"
start: 7669608
end: 7676593
variantType: "SO:0001743"

?referenceName=17&start=7669608&end=7676593&variantType=SO:0001743
```

Example: Find deletions involving the whole *TP53* locus (17:7669607-7676593)

Provided by: Michael Baudis
Notes:

- This is an application of a **Bracket Query**
- Here, matched deletion events start 5` of the gene's CDS and end 3` of it.
- Besides the gene's mapping positions, this requires knowledge about the maximum value of the reference base (or use of a very large one exceeding chromosome size).

Query structure:

```
referenceName: "17"
Start:[0,7669607]
End:[7676593,83257441]
variantType: "SO:0001743"

?referenceName=17&start=0&start=7669607&end=7676593&end=83257441&variantType=SO:
0001743
```

Example: Find any "focal" deletion involving the *CDKN2A* (chr9:21967751-21975099) locus

Provided by: Michael Baudis
Notes:

- This is an application of a **Bracket Query**
- A "focal" CNV event is (in cancer genomics) smaller than 1-5Mb; therefore, deletion events in this example should begin less than 1Mb 5' and end less than 1Mb 3' of the *CDKN2A* CDS
- This is the standard Progenetix Beacon+ DEL example query (with added NCIT disease filter and other required Beacon parameters).

```
referenceName: "9"
Start:[21000000,21975098]
end:[21967752,23000000]
variantType: "DEL"

?referenceName=9&variantType=DEL&start=21500000,21975098&end=21967752,22500000
```

```
    Genomic region:<=----------------------|XXXXXXXX|------------------------=>
Query Range
    Start pars:    <=----------------???????????????-------------------------=>
    End pars:      <=-----------------------????????????????-----------------=>
Matched variants
    Match1:        <=------------------********-----------------------------=>
    Match2:        <=-----------------------------**************------------------=>
    Match3:        <=--------------------------****------------------------------=>
Not Matched
    No Match1:     <=-------------------------------------------------!!!!!!!!!=>
    No Match2:     <=-----------!!!!!!!!!!!!****-----------------------------=>
    No Match3:     <=!!!!!!!!!!!!!!!!!!!!!!!********!!!!!!!!!!!!!!!!!!!!!!!!!=>
```

Example: Find deletion events in gene *TP53*, *excluding* those that extend beyond the gene's CDS

Provided by: Diana Lemos

Notes:

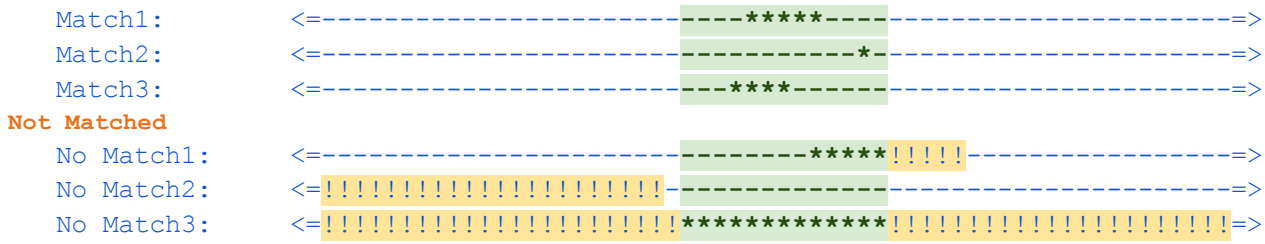
- This is an application of a **Bracket Query**.
- Any DEL that starts and ends inside the region is matched (the parameters would allow for single base deletions, which is correct from the "we do not define cutoff lengths", but also an example why additional `minLength` / `maxLength` parameters would make sense)
- assemblyId=GRCh38, *TP53* (17:7668402-7687538)

Query structure:

```
referenceName: "17"
start: [7668402, 7687537]
End: [7668403, 7687538]
variantType: "DEL"

?assemblyId=GRCh38&referenceName=17&start=7668402,7687537&end=7668403,7687538&va
riantType=DEL
```

```
    Genomic region:<=----------------------|XXXXXXXXXXXXX|---------------------=>
Query Range
    Start pars:    <=-----------------------????????????-----------------------=>
    End pars:      <=------------------------????????????----------------------=>
Matched variants
```



??Example 5: Find any deletion with reciprocal overlap > 80% of a known deletion (11:66265431-66278247) AND involving the *BBS1* gene (chr11:66278119-66301084).

Provided by: Véronique Geoffroy
Context: Currently, structural variations (SV) are compared in the genetics community mainly thanks to reciprocal overlaps (gnomAD, DGV…). Genomic content is also crucial to achieve such a comparison.
Notes: This is a combination of 3 query:

- a **Range Query:** Find any deletion involving a genomic region (11:66265431-66278247)
- a **Range Query:** Find any deletion involving a the *BBS1* gene (chr11:66278119-66301084)
- reciprocal overlap > 80% between the match and the known deletion (11:66265431-66278247)
  How to solve this 3d query? additional `Length` parameter would make sense for this, but it is really workable?

Query structure:

```
referenceName: "11"
start: 66265431
end: 66278247
variantType: "DEL"

referenceName=11&start=66265431&end=66278247&variantType=DEL

referenceName: "11"
start: 66278119
end: 66301084
variantType: "DEL"

referenceName=11&start=66278119&end=66301084&variantType=DEL
```


BBS1 gene:
Genomic region:
Query Range
Start pars:
End pars:
Matched variants
Match1:
Match2:
Match3:
Not Matched
No Match1:
(reciprocal overlap > 80% not respected)
No Match2:
(reciprocal overlap > 80% respected but BBS1 gene not overlapped)

DUP (Duplication)

Definitions:

- SO:0001742 - A sequence alteration whereby the copy number of a given region is greater than the reference sequence (copy number gain).
- CNV query resolves to DUP or DEL or CNV (and their equivalents -> see DUP, DEL), tandem dups?
- CNV loss / CNV gain)

Examples below based on a specific study (https://doi.org/10.1016/j.tjog.2018.06.018)

Example: Find duplications involving the whole locus (chr2:54,700,000-63,900,000)

Provided by: David Salgado & Michael Baudis

Notes:

- This is an application of a **Bracket Query**
- Here, matched duplication events start 5` of the region and end 3` of it.
- Besides the positions, this requires knowledge about the maximum value of the reference base (or use of a very large one exceeding chromosome size; this example here uses a lazy "just bigger than chr2" value).

Query structure:

```
referenceName: "2"
start:[0,54700000]
end:[63900000,242193529]
variantType: "SO:0001742"

?referenceName=2&start=0,54700000&end=63900000,242193529&variantType=DUP
```

```
   Genomic region:<=----------------------|XXXXXXXXXXXXX|-----------------------=>
Query Range
   Start pars:    <=???????????????????????-------------------------------------=>
   End pars:      <=------------------------------------????????????????????????=>
Matched variants
   Match1:        <=---------------------***************-----------------------=>
   Match2:        <=-----------------------**************************-----------=>
   Match3:        <=-----------------------*************------------------------=>
   Match4:        <=--******************************************************----=>
Not Matched
   No Match1:     <=----------!!!!!!!!!!!!!!!!!!!!!-----------------------------=>
   No Match2:     <=-----------------------------!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!=>
   No Match1:     <=-------!!!!!!!!!!!------------------------------------------=>
```

??Example 2:

Query structure:`?assemblyId=GRCh37&referenceName=2&start=58200000&start=58300000&end=61500000&end=61550000&variantType=DUP&CN=3+

Genomic region: ----------|-----*Locus*----|----------
Mapping location: ----------|XXXXXXXXXX|----------|XXXXXX|------|XXXXX|------|XXXXXX|

Amp (DUP more than 2) CN type of approach

```
   Genomic region:<=----------|XXXXXXXXXXXXX|----------=>
Query Range
   Start pars:    <=??????????????????????????????????-----------=>
   End pars:      <=----------??????????????????????????????????=>
Matched variants
   Match1:        <=----------**********----------=>
   Match2:        <=----------**********----------=>
   Match3:        <=----------****----------=>
Not Matched
   No Match1:     <=----------=>
   No Match2:     <=----------=>
```

LOH (loss of representation of second allele, with or without copy number change)

INV (inversion)

Definitions:

- SO:1000036 - A continuous nucleotide sequence is inverted in the same position.
- Beacon:

- Use cases? Specific inversions? (make this separate query types)-ins, inv, conv,transposition, ME

Example 1:
Query structure:

Example 2:
Query structure:

TRA[1] (Translocation)

Definitions:

- SO: A region of nucleotide sequence that has translocated to a new position. The observed adjacency of two previously separated regions.
- Beacon: The fusion of two previously non-contiguous genomic regions from the same or - usually - different chromosomes.

Example 1: Find *BCR-ABL1* translocations

Provided by: Michael Baudis
Notes:

- This is an application of a **Bracket Query**, where a `mateName` parameter identifies that the 2 bracketed regions indicate where the breakpoints of first (`referenceName`) and second (`mateName`) fusion partner should be located. Positions are approximated.

Query structure:

```
referenceName: "9"
start:[130838500,130840500]
mateName: "22"
end:[23180509,23185000]
variantType: "SO:0000199"
```

Proposal: BRK (Breakpoint)

Definitions:

- Any event with a structural genome variant (translocation, start/end of a CNV …)

This would basically be a range query where certain events (SNPs, INDELs …) are excluded from the response.

[1] Some callers used the "TRA" shorthand, e.g. novobreak : https://github.com/czc/nb_distribution or SURVIVOR https://github.com/fritzsedlazeck/SURVIVOR

ME (Mobile elements insertion /deletion)

Definition: (from SO: ME Insertion: A kind of insertion where the inserted sequence is a mobile element.
ME Deletion: A deletion of a mobile element when comparing a reference sequence (has mobile element) to a individual sequence (does not have mobile element)..)

Example 1:
Query structure:

Example 2:
Query structure:

CNV - (non directional CNVs) - do we allow cnv queries? / complex CNVs

Definition: (from SO: A variation that increases or decreases the copy number of a given region.

CNV query resolves to DUP *or* DEL *or* CNV (and their equivalents -> see DUP, DEL), tandem dups?
CNV loss / CNV gain)

Example 1:
Query structure:

Example 2:
Query structure:

Tandem Duplication

Definitions:

- SO:1000173 - A duplication consisting of 2 identical adjacent regions.
- Beacon: Recommended to follow SO codes where they apply

Example: A duplication between 132927700 and 132932000 (ClinVar RCV000850068)

> Provided by: Diana Lemos
> Notes:
>
> - a range query w/o specified sequence but including the variant type
> - id of variant type here SO
>
> Query structure:

```
referenceName: "9"
start:132927700
end:132932000
variantType: "SO:1000173"
```

Name Based Queries

For many practical applications, users of Beacon services will be interested in exploring "named" genomic features - e.g. gene regions, chromosomal bands or other genomic hallmarks.
So far, the Beacon protocol only supports positional queries (single "start", "start"->"end" ranges, precise "bracketed" combinations of start and end regions). However, there is a clear need to provide support for at least some of the possible "named element" based query scenarios. While this is not a topic specific only for structural variants, it has a somewhat more pressing nature here due to the frequent use of "fuzzy"/approximate queries for positionally imprecise variants.
The Beacon protocol assumes that all variants have been mapped against a reference genome. Therefore, any "name based query" can be executed if the positions of the item are known, either

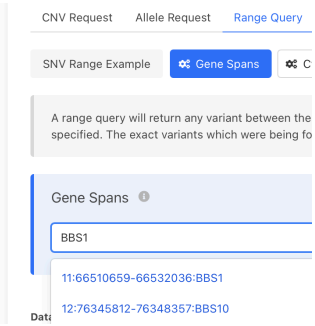
- to the front end (i.e. not a Beacon API issue, but possibly part of recommendation / reference implementation / service)
- the Beacon implementations (i.e. Beacon resources provide internal mapping & announce this ability through the /info endpoint

Topics for discussion

- separate parameter(s) in Beacon query (probably, as are filters) or service requests from front-end with translation to positional queries
- recommended name spaces (HUGO gene symbols, cytobands … ?)
- categorical and/or quantitative parameters for e.g. fractional coverage
- additional qualifiers (e.g. gene symbol + "exon")
- …

Playground

- "BeaconGenes+" on top of the Progenetix cancer CNV dataset
- https://progenetix.test/beacon-genes/search?
- can be used for flexible testing of optional search parameters
- currently supports
    - gene symbol (selector)
    - Minimum Variant Length
    - Maximal Variant Length



Example 1: Find any events overlapping the BBS1 gene (or its promoter/enhancers)

Provided by: Véronique Geoffroy

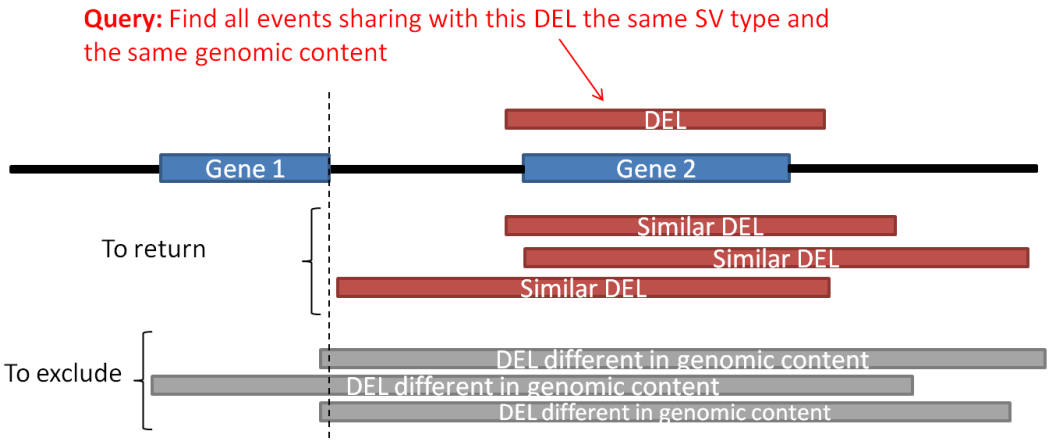
Notes:

- What would be the resource for those positions?
    - should be some reference service which e.g. then is use by the network implementations
    - right now Progenetix provides a "whole CDS coordinates" service "genespans":
        - https://progenetix.org/services/genespans/?geneSymbol=BBS1
        - this is e.g. used in its Beacon front-ends as a helper utility
        - the site also uses this for an internal resolution of "gene symbol based" queries

Example 2: Find all events sharing with a given SV (e.g. DEL 11:66287088-66288847) the same SV type and the same genomic content

Provided by: Véronique Geoffroy

- Notes: Search for ~identical SV (same SV type + same genomic content) with fuzzy coordinates ⇔ same SV type + same genes (or more precisely same exons of the genes), same promoters, same enhancers overlapped.
- In case of frequent (benign) SV matching, this could allow an SV to be filtered out of an analysis



Technical considerations

Format of GET queries

Since the direct interpretation of list parameters in queries is not supported by some server environments (e.g. PHP, GO…), list parameters such as `start` and `end` should be provided as comma-concatenated strings.
An example of the problem is illustrated here:
https://stackoverflow.com/questions/24059773/correct-way-to-pass-multiple-values-for-same-parameter-name-in-get-request
There is no "official" specification on how to provide multiple data to a same url parameter.
Some http request handlers will behave differently. It is therefore recommended to use a formalism that will be handle by any http request consumer the same way.
This problem happen only while
With JSON based query the problem is solved

Pending cases

Insertion

Alex:

There are cases where the specific DNA insertion is less important than the resulting protein insertion event. Here are a few examples from the CIViC knowledgebase: https://civicdb.org/search/variants/267cb7a0-225a-4a4b-9a37-4d8b09a1fa77

Michael:
Well, these are insertions and can be found through either range or specific ref/alt queries. But I wouldn't have a concept for direct "in-frame insertion of unknown sequence" on the variant representation / Beacon positional query side.

However, this could be done by separate "evidence" attributes which are then used to filter variant instances based on the matching annotations in a linked resource (the new Beacon data model has a "variantAnnotations" collection). So this is coming … This is something for our upcoming cell line testbed Beacon by @rahelpaloots@gmail.com ...