Usage of /Search and /Count Services

Last updated on 2019-04-05

Basic usages

Complex queries are required to interact with the /search and /count endpoint. It has 4 mandatory fields: *dataset_id*, *logic*, *components*, and *results*.

You always need to specify *dataset_id* in your query.

Part I: Logic

Logic is where you specify the relationship between various components. The only operators you will need to specify are either *AND* or *OR***.** When writing the logic, the operation becomes the key. For example, ***conditionA and conditionB and condition C*** would be written as:

```
"logic": {
      "and": [
            {
                  "id": "conditionA"
            },
            {
                  "id": "conditionB"
            }.
            {
                  "id": "conditionC"
            }
      ]
}
```

There is one exception to this rule. When you only have 1 component in your query, your logic part of the query will only have an id, which would look like this:

```
"logic": {
      "id": "condition1"
      }
}
```

The logic can get very complicated, with multiple nested operations involved, the following example is the equivalent to *(condition1 AND condition2) AND (condition3 OR condition4)*

```
"logic": {
        "and": [
            {
                "and": [
                    {
                        "id": "condition1"
                    },
                    {
                        "id": "condition2"
                    }
                ]
            },
            {
                "or": [
                    {
                        "id": "condition3"
                    },
                    {
                        "id": "condition4"
                    }
                ]
            }
        ]
    }
```

When you want to apply the NOT operator, simply specify *negate: true* along with the id, for example:

```
"logic": {
      "id": "condition1",
      "negate": true
      }
}
```

We have a few more examples of the complex queries at the end of this documentation, please refer to them for some of the most common types of complex queries.

Part II: Components

The components part of a complex query is a bit more straightforward. It is a list of components, each corresponding to a filter of a specified table. Be careful that the id has to match with the one you specified in the logic part of your query. It can be any string, but they have to match.

In a component, you specify the tables you want to search on to be the key, in this case, it was “patients”. You will also need to specify filters, where you specify the field, operator and value.

```
"components": [
      {
            "id": "conditionA",
            "patients": {
                  "filters": [
                        {
                              "field": "provinceOfResidence",
                              "operator": "!=",
                              "value": "Ontario"
                        }
                  ]
            }
      }
]
```

We support a number of common operators in the filters, including >, < , >=, <=, =, ==, *!=, contains* and *in.* Note that = and == are equivalent in our query. Operator ‘in’ only works if you specify a list of values under *values*, instead of *value*.

Part III: Results

In the “results” part of your query, you will need to specify the table you want the server to return. For a query made to the /search endpoint, you can simply specify the table name.

```
"results": [
      {
            "table": "patients"
      }
]
```

However, this will *not* work for queries made to the /count endpoint. For queries made to the /count endpoint, you will have to specify a list of fields, so that the server can return aggregated stats on the fields you provided. As example would be:

```
"results": [
      {
            "table": "patients",
            "fields": ["gender", "ethnicity", "provinceOfResidence"]
      }
]
```

Advanced usages

Part I: Complex queries that involve variants search

Queries that involve the variants table are largely the same, however, since variants require different parameters, the way you specify them in the components or results will be a bit different.

To specify variants as one of your components, you will need to supply *start*, *end*, *reference_name*. If you have a list of *variant_set_ids* that you want to search from, specify them in your component.

For example, the following component, with the id being A, is querying on variants within the range between the *start* and *end* that are on *chromesome 22, f*rom the *designated variantSet.*

```
{
	"id": "A",
	"variants": {
		"start": "50158561",
		"end": "50158565",
		"reference_name": "22",
		"variant_set_ids": [
			"yourVariantSetId"
		]
	}
}
```

However, if you simply want to search across **all** variantsets that are associated with one particular dataset, then do not specify the *variant_set_ids*.

```
{
	"id": "A",
	"variants": {
		"start": "50158561",
		"end": "50158565",
		"reference_name": "22"
	}
}
```

When you query variants in the results section of your query, you will also need *start*, *end*, and *reference_name*. But you should *not* supply *variant_set_ids*, this is because the components you wrote in your query should refer to one or more patients, whose variants data will then be returned to you.

An example would look like this:

```
"results": [
      {
            "table": "variants",
            "start": "1",
            "end": "50158565",
            "reference_name": "22"
      }
]
```

Part II: The use of Negate

While you cannot specify a NOT operator the same way you write a AND/OR operator, you can achieve the same thing by writing “negate: true” next to the id, in the logic part of the query.

For example, to apply NOT to conditionB, write a “negate: true” next to that particular condition.

```
"logic": {
      "and": [
            {
                  "id": "conditionA"
            },
            {
                  "id": "conditionB",
                  "negate": true
            }
      ]
}
```

Appendix

Example Query 1

Description: Return a list of patients, whose *diseaseResponseOrStatus* is “Complete Response”, AND have a *courseNumber* that is not 100.

Note: The example query below only works for the /search endpoint, as it did not specify a list of fields to aggregate on in the results section. Refer to Part III: Results under Basic Usages to remind yourself how to do it, or refer to Example Query 2.

Query:

```
{
      "dataset_id": "yourDatasetId",
      "logic": {
            "and": [
                  {
                        "id": "A"
                  },
                  {
                        "id": "B"
                  }
            ]
      },
      "components": [
            {
                  "id": "A",
                  "outcomes": {
                        "filters": [
                              {
                                    "field": "diseaseResponseOrStatus",
                                    "operator": "==",
                                    "value": "Complete Response"
                              }
                        ]
                  }
            },
```

```
            {
                  "id": "B",
                  "treatments": {
                        "filters": [
                              {
                                    "field": "courseNumber",
                                    "operator": "!=",
                                    "value": "100"
                              }
                        ]
                  }
            }
      ],
      "results": [
            {
                  "table": "patients"
            }
      ]
}
```

Example Query 2

Description: Return the aggregated stats on patients' gender and ethnicity who have mutations present between "50158561" and "50158565" on chromosome 22, from the list of variantsetIds.

```
{
      "dataset_id": "yourDatasetId",
      "logic": {
            "id": "A"
      },
      "components": [
            {
                  "id": "A",
                  "variants": {
                        "start": "50158561",
                        "end": "50158565",
                        "reference_name": "22",
                        "variant_set_ids": [
                              "yourVariantSetId_1",
                              "yourVariantSetId_2",
                              "yourVariantSetId_3",
                              "yourVariantSetId_4",
                              "yourVariantSetId_5",
                              "yourVariantSetId_6",
                              "yourVariantSetId_7",
                              "yourVariantSetId_8"
                        ]
                  }
            }
      ],
      "results": [
            {
                  "table": "patients",
                  "fields": [
                        "gender",
                        "ethnicity"
                  ]
            }
      ]
}
```

Example Query 3

Description: Return the aggregated stats on patients' gender and ethnicity, who have mutations present between "50158561" and "50158565" on chromosome 22.

Note: Since a list of variantSetIds was not specified, the server will attempt to locate all variantSets associated with the dataset. If you have a lot of variantSets associated with this particular dataset, the query might take some time.

```
{
      "dataset_id": "yourDatasetId",
      "logic": {
            "id": "A"
      },
      "components": [
            {
                  "id": "A",
                  "variants": {
                        "start": "50158561",
                        "end": "50158565",
                        "reference_name": "22"
                  }
            }
      ],
      "results": [
            {
                  "table": "patients",
                  "fields": [
                        "gender",
                        "ethnicity"
                  ]
            }
      ]
}
```

Example Query 4

Description: Retrieve all the variants between 50100000 and 50158565 on chromosome 22 associated with an individual [HG00105].

```
{
      "dataset_id": "yourDatasetId",
      "logic": {
            "id": "A"
      },
      "components": [
            {
                  "id": "A",
                  "patients": {
                        "filters": [
                              {
                                    "field": "patientId",
                                    "operator": "==",
                                    "value": "HG00105"
                              }
                        ]
                  }
            }
      ],
      "results": [
            {
                  "table": "variants",
                  "start": "50100000",
                  "end": "50158565",
                  "reference_name": "22"
            }
      ]
}
```