GENOMENON®

AI-DRIVEN GENOMICS

Mastermind Integration

Technical Documentation

May 10, 2021

hello@genomenon.com | www.genomenon.com | 1-734-794-3075

GENOMENON®

GENOMENON®

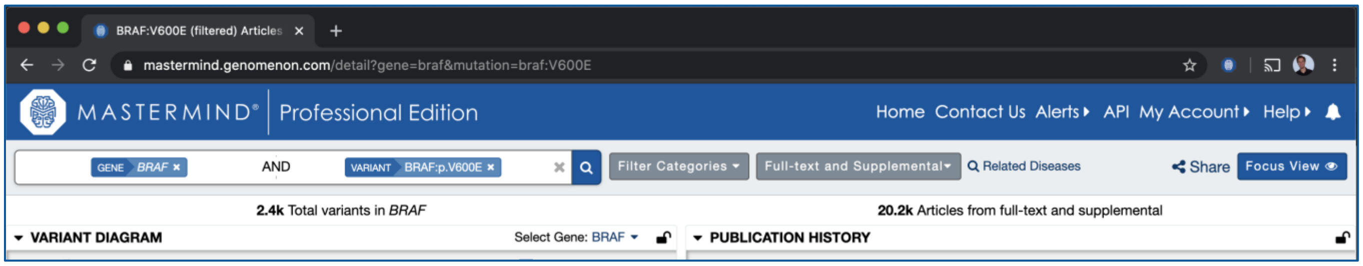
AI-DRIVEN GENOMICS

Linking to Mastermind

Link Structure

The Mastermind Genomic Search Engine provides structured URLs which allow easy deep-linking into search results for a given gene and/or variant. For example, here is a URL which takes the user directly to articles associated with the BRAF:V600E variant:

https://mastermind.genomenon.com/detail?gene=braf&mutation=braf:V600E


BRAF:V600E (filtered) Articles
mastermind.genomenon.com/detail?gene=braf&mutation=braf:V600E
MASTERMIND® Professional Edition
Home Contact Us Alerts API My Account Help
GENE BRAF
AND
VARIANT BRAF:p.V600E
Filter Categories
Full-text and Supplemental
Related Diseases
Share
Focus View
2.4k Total variants in BRAF
20.2k Articles from full-text and supplemental
VARIANT DIAGRAM
Select Gene: BRAF
PUBLICATION HISTORY

This URL format can be used to dynamically generate deep-links into Mastermind searches by simply replacing the disease, gene, and variant in the URL.

https://mastermind.genomenon.com/detail?gene={gene_symbol}&mutation={gene_symbol:variant_key}

The provided parameters may be omitted, with the following constraints:

- For Free Edition users, the gene must be specified. The variant parameter may be omitted.
- For Professional Edition users, at least one parameter other than gene must be specified to get a list of associated genes, or just the gene may be provided to get a list of associated diseases.
- For variant searches, either the gene symbol must also be specified in the variant nomenclature, or the gene symbol may be provided in the separate gene parameter. Some variant nomenclatures such as the fully qualified genomic coordinates or rsIDs may omit the gene entirely.

Referral Code

When integrating Mastermind links into your application, add the `ref` parameter value to the URL. Request a unique code for your organization at hello@genomenon.com:

https://mastermind.genomenon.com/detail?mutation=braf:V600E**&ref=your-org-code**

Parameter Values and Formats

The gene and variant parameters should follow these rules:

Gene

The value for the gene parameter should be the desired HGNC gene symbol in lowercase.

Variant

The value for the variant parameter should include the same gene symbol from the The value for the variant parameter should include the same gene symbol from the gene parameter followed by a colon, followed by the variant key. The variant key may be specified in any of the following nomenclatures:

- Mastermind gene variant nomenclature, which is based on HGVS protein (effect) level nomenclature and optimized for search sensitivity.

  For most variants, this is simply the protein nomenclature. However, to maximize sensitivity for specific types of deletions, insertions, duplications, frame-shifts, and non-coding variants, there is a simple transformation to Mastermind variant nomenclature, outlined in the FAQ.
  Example: https://mastermind.genomenon.com/detail?**mutation=braf:E46int**

- HGVS protein "p." nomenclature.
  Example:
  https://mastermind.genomenon.com/detail?**mutation=cftr:p.Lys684dup**

- HGVS cDNA "c." nomenclature.
  Example:
  https://mastermind.genomenon.com/detail?**mutation=brca1:c.5382insC**

- Reference SNP cluster ID "rsID" identifier from dbSNP.
  Examples:
  https://mastermind.genomenon.com/detail?**mutation=cdkn2a:rs3731249**

  https://mastermind.genomenon.com/detail?**mutation=rs3731249**

- Genomic coordinates using GRCh37. The sequence identifier should be used for genomic coordinates. This is NC_00012.11 for chromosome 12 below:
  Example (the `%3E` below is the URL-encoding of the > character):
  https://mastermind.genomenon.com/detail?**mutation=NC_000012.11:g.57489193T%3EC**

For help with any specific gene or variant, try typing the desired value into the appropriate field in the homepage search interface to see which valid values are suggested as part of the auto-complete drop-down.

Advanced Search

In addition to the standard gene/variant searches described above, Mastermind Professional Edition also offers advanced search functionality for:

- Searching diseases, phenotypes, and/or therapies;
- Searching multiple diseases, genes, and/or variants;
- ACMG/AMP guideline-based searching;
- Filters for clinical significance and genetic mechanisms, such as amplifications, CNVs, and more;
- Exporting of PMID lists with supporting information for any search.

All of these search features are able to be used, combined, and patterned directly within the URL of the links into Mastermind, provided that the users clicking the links have access to Mastermind Professional Edition. If the user has access only to Mastermind Basic Edition, upon clicking, they will be shown a message that the linked search requires Professional Edition to view.

To determine valid URL values or formats for any of the following input parameters, try searching for them using the search field in Mastermind, and then copy/pasting the values from the resulting URL for the search results page.

Disease

The value for the disease should be the canonical MeSH term for the disease, in lower-case and URL-encoded (e.g. spaces are encoded as "%20").

To view results irrespective of citations of any specific disease, diseases may be ignored, or the value of "all%20diseases" may also be explicitly encoded. This will show results for all diseases together.

The valid, canonical disease names can be looked up either directly from the MeSH website, or by typing a disease into the search field in Mastermind, selecting one of the suggestions, clicking search, and then copy/pasting the resulting identifier from the URL of the search results page.

An example URL for searching a disease to show associated genes follows:

https://mastermind.genomenon.com/search?disease=melanoma

Here is an example URL for a standard gene/variant search further filtered to show article results specifically associated with the disease, Melanoma:

https://mastermind.genomenon.com/detail?mutation=braf:V600E&disease=melanoma

Phenotype

The value for the phenotype should be the corresponding Human Phenotype Ontology (HPO) identifier, such as HP:0012649 for the phenotype, "Increased inflammatory response".

The identifiers can be looked up either directly from the HPO website, or by typing a phenotype into the search field in Mastermind, selecting one of the suggestions, clicking search, and then copy/pasting the resulting identifier from the URL of the search results page.

An example URL for searching a phenotype to show associated genes follows:

https://mastermind.genomenon.com/search?hpo=HP:0012649

Here is an example URL for a standard gene/variant search further filtered to show article results specifically associated with the phenotype, Increased inflammatory response:

https://mastermind.genomenon.com/detail?mutation=cd8a:D30N&hpo=HP:0012649

Therapy

The value for the therapy should be the corresponding FDA Unique Ingredient Identifier (UNII), such as UNII:207SMY3FQT for the therapy Vemurafenib.

The identifiers can be looked up either directly from the UNII website, or by typing a therapy into the search field in Mastermind and copy/pasting the resulting identifier from the suggestions.

Linking to Mastermind (continued)

An example URL for searching a therapy to show associated genes follows:

https://mastermind.genomenon.com/search?unii=UNII:207SMY3FQT

Here is an example URL for a standard gene/variant search further filtered to show article results specifically associated with the therapy Vemurafenib.

https://mastermind.genomenon.com/detail?mutation=braf:V600K&unii=UNII:207SMY3FQT

GENOMENON®

AI-DRIVEN GENOMICS

Single Sign-on

Benefits

Single Sign-on (SSO) integration into Mastermind allows users of your platform to move seamlessly back and forth between your platform and Mastermind without needing to sign up for or log into Mastermind separately. Their account in your platform acts as their account in Mastermind.

This works by linking to or embedding Mastermind from within your application using special Mastermind SSO URLs.

Getting Started

1. Contact us to configure SSO access to Mastermind from within your application. We will set up SSO credentials with a shared key that your servers can use to sign Single Sign-on access tokens.
2. Generate links to Mastermind within your application using your SSO credentials as described in the documentation below.

Generating SSO URLs

The SSO URLs to Mastermind follow the same format described in the "Linking to Mastermind" section of this document, with the addition of the following SSO parameters to be added to every link:

Name	Description
`provider`	The SSO partner ID assigned to your application by Genomenon.
`email`	The unique email address associated with the user's account in your application.
`first_name`	The user's first (given) name.
`last_name`	The user's last (family) name.
`organization`	The company or institution for which the user is affiliated.
`session_id`	The unique identifier used to indicate the user's current case or session within your application.
`timestamp`	The timestamp right now, in epoch (Unix) time. The time must be between five minutes behind and one minute ahead of the time on the Mastermind servers.
`token`	The Base-64 encoded SHA-256-digest-based HMAC hash of the URL request parameters

Example SSO URL

The initial SSO URL will be generated on your application's server using your Mastermind SSO secret key. This key should remain safe and secure on your application's servers and never be shared with anyone.

For this example, we'll assume your assigned `provider` ID, secret key, and the user's request parameters are as follows:

Parameter	Value
`disease`	`"melanoma"`
`gene`	`"braf"`
`variant`	`"v600e"`
`secret_key`	`"YourAppIsAwesome!"`
`provider`	`"yourapp"`
`email`	`"johnsmith@example.com"`
`first_name`	`"John"`
`last_name`	`"Smith"`
`organization`	`"The Lab"`
`session_id`	`"case-1234567"`
`timestamp`	`1551376523` (2019-02-28 12:55:23 EST converted to epoch time)
`token`	`"L2JhPqsSOYElxcyLlqofORR/cTePN7lIoSF7zy/+C+4="`

The token above was calculated by finding the HMAC of the concatenation of the request parameters using the secret_key. If your server uses the Ruby programming language, that would be done as follows:

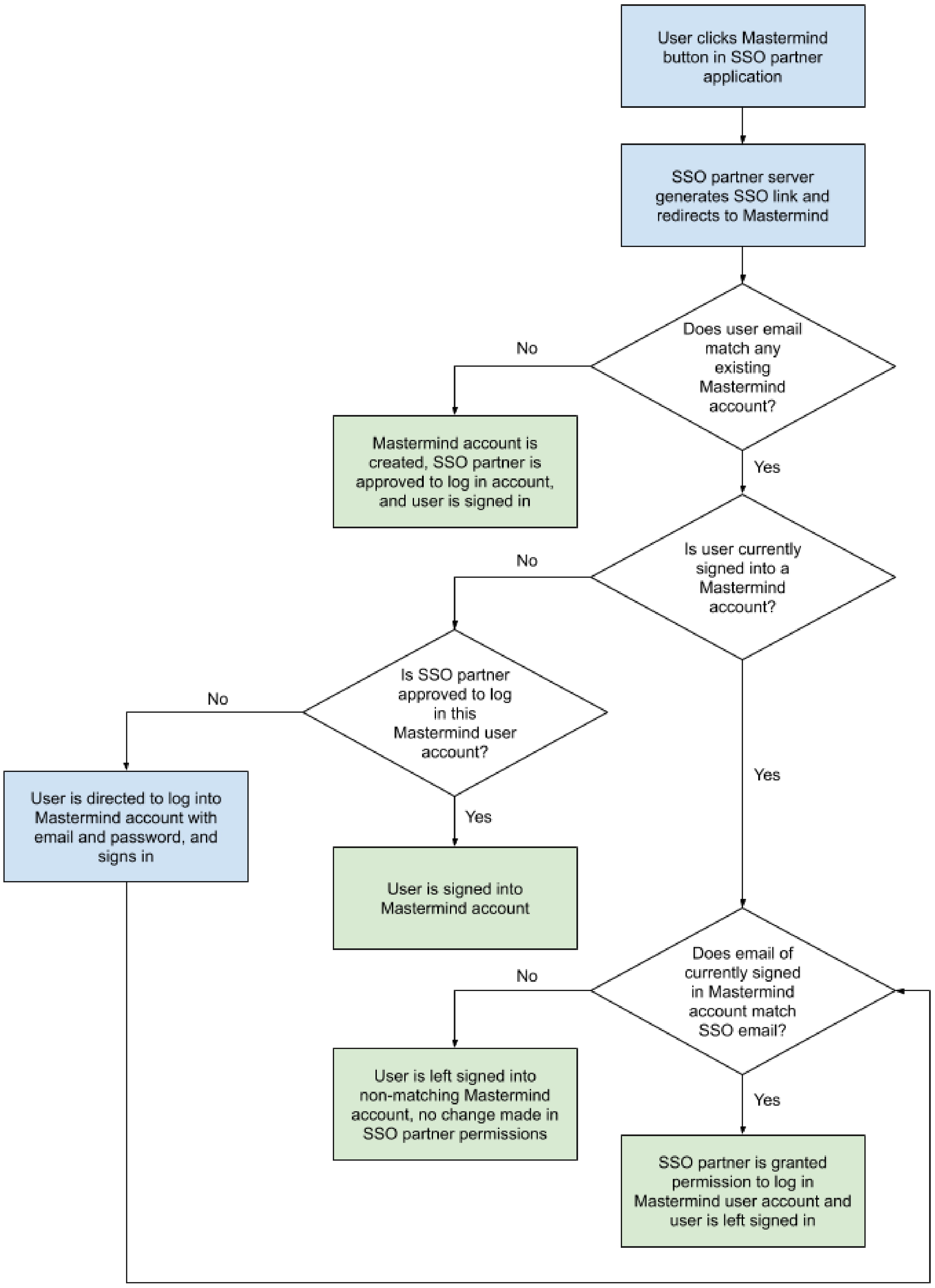
```
POST https://mastermind.genomenon.com/sso/accept\n
body:disease=melanoma&gene=braf&variant=v600eprovider=yourapp&email=johnsmith@example.com&first_name=Jo
hn&last_name=Smith&organization=The%20Lab&session_id=case-
1234567&timestamp=1551376523&token=L2JhPqsSOYElxcyLlqofORR%2FcTePN7lIoSF7zy%2F%2BC%2B4%3D
```

```
secret_key="YourAppIsAwesome!"
provider="yourapp"
email="johnsmith@example.com"
first_name="John"
last_name="Smith"
organization="The Lab"
session_id="case-1234567"
timestamp = Time.now.to_i #=> 1551376523

request_data = disease + gene + variant + provider + email + first_name + last_name + organization + session_id +
timestamp.to_s

token = Base64.encode64(OpenSSL::HMAC.digest(OpenSSL::Digest::Digest.new("sha256"), secret_key, request_data)).strip
```

The resulting SSO URL would then be a POST request from your server as follows:

Single Sign-on (continued)

The user experience workflow from the SSO partner application into Mastermind will follow this path, depending on the Mastermind user account matching the provided user email address in the request:



User clicks Mastermind button in SSO partner application
SSO partner server generates SSO link and redirects to Mastermind
Does user email match any existing Mastermind account?
No
Mastermind account is created, SSO partner is approved to log in account, and user is signed in
Yes
Is user currently signed into a Mastermind account?
No
Is SSO partner approved to log in this Mastermind user account?
No
User is directed to log into Mastermind account with email and password, and signs in
Yes
User is signed into Mastermind account
Yes
Does email of currently signed in Mastermind account match SSO email?
No
User is left signed into non-matching Mastermind account, no change made in SSO partner permissions
Yes
SSO partner is granted permission to log in Mastermind user account and user is left signed in

GENOMENON®

AI-DRIVEN GENOMICS

Mastermind API Endpoints

The Mastermind API is a set of programmatic endpoints available to interact with the Mastermind data at a more detailed and granular level.

API Endpoints

`/suggestions` API endpoint BASE API

Returns the canonical name(s) to query in Mastermind for the given disease, gene, or variant.

`/counts` API endpoint BASE API

Returns the number of articles that match the given query.

`/articles` API endpoint ADVANCED API

Returns the list of articles, ranked by relevance, for the given query.

`/genes` API endpoint ADVANCED API

Returns the top genes, ranked by number of matched articles, for the given query.

`/variants` API endpoint ADVANCED API

Returns the top variants, ranked by number of matched articles, for the given query.

`/diseases` API endpoint ADVANCED API

Returns the top 5 diseases, ranked by number of matched articles, for the given query.

`/phenotypes` API endpoint ADVANCED API

Returns the top phenotypes, ranked by number of matched articles, for the given query.

`/therapies` API endpoint ADVANCED API

Returns the top therapies, ranked by number of matched articles, for the given query.

`/article_info` API endpoint ADVANCED API

Returns the matched diseases, genes, variants, and article meta-data for the given article PMID.

`/file_annotations/counts` API endpoint ADVANCED API

Returns an annotated VCF from an input VCF file containing multiple variants as genomic coordinates with each variant in the output file annotated with the number of articles and Mastermind URL for each variant.

Authentication

All API requests to both Basic and Advanced endpoints must be authenticated using your API token. Simply pass the API token as the `X-API-TOKEN` header in each request. An example request to the `/counts` endpoint would look like this, using Curl from the command line:

```
curl -H "Content-type: application/json" -H "X-API-TOKEN: [[your API token here]]" -X GET
"https://mastermind.genomenon.com/api/v2/counts?&disease=melanoma&gene=braf&variant=braf:v600e"
```

If you do not know your API token, you can fetch it by visiting the API homepage and clicking the link labeled "Click here to fetch your API token."

If you do not have an API token assigned, you may reach out to sales@genomenon.com.

Request Prefix

All of the following endpoints should be prefixed with the current version of the API, which is v2:

```
https://mastermind.genomenon.com/api/v2
```

So, for example, the `/suggestions` endpoint would be queried via:

```
https://mastermind.genomenon.com/api/v2/suggestions
```

Request Type

As all endpoints in the Mastermind API V2 are to retrieve data, all API requests should be `GET` requests (not `POST` or other type).

URL-encoding Query Values

All parameter values to the API should be encoded. For example, querying the `/suggestions` endpoint for "non-small cell lung cancer" would be encoded as follows:

```
https://mastermind.genomenon.com/api/v2/suggestions?disease=non-small%20cell%20lung%20cancer
```

Likewise, querying the `/suggestions` endpoint for "braf:c.1799t>a" would be encoded as:

```
https://mastermind.genomenon.com/api/v2/suggestions?variant=braf%3Ac.1799t%3Ea
```

Multiple Parameter Values

Some parameter inputs allow multiple values to be input simultaneously, as Boolean queries. For API `GET` requests, these are specified by repeating the parameter name in the URL with each of the different values, such as:

```
https://mastermind.genomenon.com/api/v2/counts?gene=braf&category=in%20vivo&category=de%20novo
```

In addition to being able to specify multiple values for each parameter, the Boolean operation can also be specified. Boolean parameter values are allowed for the following input parameters:

- `gene` – optionally specify `gene_op=and` (default) or `gene_op=or`
- `variant` – optionally specify `variant_op=and` (default) or `variant_op=or`
- `disease` – optionally specify `disease_op=and` (default) or `disease_op=or`
- `hpo_id` – optionally specify `hpo_op=and` (default) or `hpo_op=or`
- `unii_id` – optionally specify `unii_op=and` (default) or `unii_op=or`
- `category` – only works as "or"

And example searching for articles citing either gene *BRAF or KRAS* might look like:

```
https://mastermind.genomenon.com/api/v2/counts?gene=braf&gene=kras&gene_op=or
```

Root endpoint

The "root" URL of the API can also be queried as a `GET` request with no input parameters (other than the API token). This endpoint will return the endpoints active for your API subscription, the renewal date for your access, and the number of queries performed through your API subscription relative to the limit for each endpoint.

```
https://mastermind.genomenon.com/api/v2
```

Example Response

```
{
  "subscription_renewal_date": "2023-03-31",
  "limits_reset": "2022-03-31",
  "endpoints": [
    {
      "path": "v2/counts",
      "yearly_limit": 100000,
      "queries_remaining": 98234
    }
  ]
}
```

Code Examples

For live code examples using the Mastermind API, see the following repository on Github: https://github.com/Genomenon/mastermind-api-cookbook

The `/suggestions` API endpoint returns the canonical name(s) to query in Mastermind for the given gene, variant, disease, phenotype, or therapy. See API demo.

Query Parameters

Name	Description	Example
`gene`	The gene name to be looked up.	`"v-Raf murine sarcoma viral oncogene homolog B"`
`variant`	The variant to be looked up, which may be specified in HGVS cDNA (c.) nomenclature, protein (p.) nomenclature, or rsID. When querying by cDNA or protein nomenclatures, be sure to include the gene symbol.	`"braf:c.1799t>a"`
`disease`	The disease name you would like to look up.	`"acute myeloid leukemia"`
`hpo`	The phenotype name you would like to look up.	`"increased inflammatory response"`
`unii`	The therapy or drug name you would like to look up.	`"arachnodactyly"`

Response Format

```
{
  "type":      [["disease", "gene", or "variant"]],
  "matched":   [[synonym or nomenclature matched]],
  "name":      [[normalized name using standard nomenclature]],
  "canonical": [[canonical name to use for Mastermind queries]],
  "url":       [[link to Mastermind query using the canonical name]]
```

Example Response

```
[
  {
    "type": "disease",
    "matched": "acute myeloid leukemia",
    "name": "Leukemia, Myeloid, Acute",
    "canonical": "leukemia, myeloid, acute",
    "url": "https://mastermind.genomenon.com/search?disease=leukemia%2C+myeloid%2C+acute"
  }
]
```

The /counts API endpoint returns the number of articles that match the given query. See API demo.

Query Parameters

All inputs must use properly formatted and exact values for inputs. Use the "Suggestions" endpoint to obtain these "canonical" accepted values for any given input.

Name	Description	Example
gene	The gene associated with the articles of interest, specified by their canonical HGNC gene symbol.	"braf"
variant	The variant associated with the articles of interest, specified by their protein effect, or directly using HGVS cDNA, genomic coordinates, or dbSNP rsID. Use the /suggestions endpoint to get the generalized variant nomenclature used by Mastermind for your variant, or query the direct variant nomenclatures ensuring they are properly formatted (including no spaces) and correct. NOTE: See this blog post and this FAQ explaining the generalized nomenclature used by Mastermind.	"braf:v600e"
disease	The canonical disease name associated with the articles of interest, specified in lowercased MeSH terminology.	"melanoma"
hpo_id	The HPO phenotype identifier associated with the articles of interest.	"HP:0001166"
unii_id	The UNII therapy identifier associated with the articles of interest.	"UNII:207SMY3FQT"
category	The Mastermind categories and keywords associated with the articles of interest, in lowercase. These should be chosen from the category/keyword list found in Mastermind, including ACMG/AMP criteria and other clinical significance indicators. *NOTE: Passing multiple categories will **increase** the scope of the search, as superset of articles will be returned that match **any** of the specified categories.*	"in vivo", "de novo", "functional-sub"
keyword	A plain-text keyword to filter articles, can pass in any word or phrase.	"basket"
scope	The scope of the article search, from most specific PubMed-only searches, to more sensitive search including the full-text, to most sensitive including supplemental materials.	"abstract", "fulltext", "supplemental" (default)
since	The starting date from which to return matches, as an Epoch Unix timestamp. When specified, only results matched since the provided date are returned. This can be used for example as part of a weekly cronjob to query for all new results since the last time the endpoint was queried for data. The since parameter allows querying as far back as the previous month.	1556918275 (for May 3, 2019 9:17:55 PM GMT)

Response Format

```
{
  "article_count": [[total number of articles]]
  "url": [[link to Mastermind results]]
}
```

Example Response

```
{
  "article_count": 4613,
  "url":
"https://mastermind.genomenon.com/detail?disease=melanoma&gene=braf&mutation=braf:v600e"
```

The `/articles` API endpoint returns the top articles with PMIDs, 5 per page and ranked by relevance, for the given query. See API demo.

Query Parameters

See "Counts" endpoint. Addiitonal parameters for this endpoint are below:

Name	Description	Example
`page`	The desired page of results to return in the "articles" results array. Valid values range from 1 up to a maximum of 2,000.	2

Response Format

```
{
  "article_count": [[total number of articles]],
  "pages": [[number of results pages available]],
  "page": [[current results page]],
  "articles": [
    {
      "pmid": [[pmid]],
      "title": [[title]],
      "publication_date": [[normalized publication date]],
      "journal": [[journal ISO code]],
      "authors": [[list of author names]],
      "url": [[link to view article in Mastermind]]
    }
  ],
  "url": [link to Mastermind results]
}
```

NOTE: Specifying a variant directly using HGVS cDNA, genomic coordinates, or dbSNP rsID will prioritize articles with nucleotide-specific matches first in the results, just as in the user interface, and will also add a `matched_dna: true` property value to the corresponding article results in the response.

Example Response (truncated to first article for clarity)

```
{
  "article_count": 8392,
  "pages": 1679,
  "page": 1,
  "articles": [
    {
      "pmid": "23543365",
      "title": "Gossypin as a novel selective dual inhibitor of V-RAF murine sarcoma viral oncogene
homolog B1 and cyclin-dependent kinase 4 for melanoma.",
      "journal": "Mol. Cancer Ther.",
      "publication_date": "2013-03-29",
      "authors": [
        "Bhaskaran S",
        …
      ],
      "url": "https://mastermind.genomenon.com/detail?disease=melanoma&gene=braf&pmid=23543365"
    }
  ],
  "url": "https://mastermind.genomenon.com/detail?disease=melanoma&gene=braf"
}
```

"Genes" Endpoint

The `/genes` API endpoint returns the top genes, 5 per page and ranked by number of matched articles, for the given query. See API demo.

Query Parameters

See "Counts" and "Articles" endpoints for all valid query parameters.

Response Format

```
{
  "article_count": [[total number of articles]],
  "pages": [[total number of pages]],
  "page": [[current page of results]],
  "gene_count": [[total number of genes]],
  "genes": [
    {
      "symbol": [[gene symbol]],
      "article_count": [[number of articles for gene and given query]],
      "url": [[link to Mastermind for gene and given query]]
    }
  ],
```

Example Response

```
{
  "article_count": 12278,
  "pages": 2255,
  "page": 1,
  "gene_count": 11273,
  "genes": [
    {
      "symbol": "IFNA1",
      "article_count": 2723,
      "url": "https://mastermind.genomenon.com/detail?disease=melanoma&gene=IFNA1&hpo=HP:0012649"
    },
    {
      "symbol": "IL2",
      "article_count": 2709,
      "url": "https://mastermind.genomenon.com/detail?disease=melanoma&gene=IL2&hpo=HP:0012649"
    },
    {
      "symbol": "CD8A",
      "article_count": 2671,
      "url": "https://mastermind.genomenon.com/detail?disease=melanoma&gene=CD8A&hpo=HP:0012649"
    },
    {
      "symbol": "CTLA4",
      "article_count": 2507,
      "url": "https://mastermind.genomenon.com/detail?disease=melanoma&gene=CTLA4&hpo=HP:0012649"
    },
    {
      "symbol": "BRAF",
      "article_count": 2444,
      "url": "https://mastermind.genomenon.com/detail?disease=melanoma&gene=BRAF&hpo=HP:0012649"
    }
  ],
  "url": "https://mastermind.genomenon.com/search?disease=melanoma&gene=&hpo=HP:0012649"
}
```

The `/variants` API endpoint returns the top variants, 5 per page and ranked by number of matched articles, for the given query. See API demo.

Query Parameters

See “Counts” and “Articles” endpoints for all valid query parameters.

Response Format

```
{
  "article_count": [[total number of articles]],
  "pages": [[total number of pages]],
  "page": [[current page of results]],
  "variant_count": [[total number of variants]],
  "variants": [
    {
      "key": [[Mastermind variant key name]]},
      "gene": [[HGVS gene symbol]],
      "article_count": [[number of articles for variant and given query]],
      "url": [[link to Mastermind for variant and given query]]
    }
  ],
```

Example Response

```
{
  "article_count": 12278,
  "pages": 1658,
  "page": 1,
  "variant_count": 8286,
  "variants": [
    {
      "gene": "braf",
      "key": "braf:V600E",
      "article_count": 1297,
      "url": "https://mastermind.genomenon.com/detail?disease=melanoma&gene=braf&hpo=HP:0012649&mutation=braf:V600E"
    },
    {
      "gene": "braf",
      "key": "braf:V600K",
      "article_count": 469,
      "url": "https://mastermind.genomenon.com/detail?disease=melanoma&gene=braf&hpo=HP:0012649&mutation=braf:V600K"
    },
    {
      "gene": "braf",
      "key": "braf:V600D",
      "article_count": 124,
      "url": "https://mastermind.genomenon.com/detail?disease=melanoma&gene=braf&hpo=HP:0012649&mutation=braf:V600D"
    },
    {
      "gene": "braf",
      "key": "braf:V600R",
      "article_count": 118,
      "url": "https://mastermind.genomenon.com/detail?disease=melanoma&gene=braf&hpo=HP:0012649&mutation=braf:V600R"
    },
    [clipped]
],
  "url": "https://mastermind.genomenon.com/search?disease=melanoma&gene=&hpo=HP:0012649"
}
```

The `/diseases` API endpoint returns the top diseases, 5 per page and ranked by number of matched articles for the given query. See API demo.

Query Parameters

See “Counts” and “Articles” endpoints for all valid query parameters.

Response Format

```
{
  "article_count": [[total number of articles]],
  "pages": [[total number of pages]],
  "page": [[current page of results]],
  "disease_count": [[total number of diseases]],
  "diseases": [
    {
      "key": [[MeSH disease name]],
      "article_count": [[number of articles for disease and given query]],
      "url": [[link to Mastermind for disease and given query]]
    }
  ],
  "url": [[link to Mastermind query for the given gene and variant]]
}
```

Example Response

```
{
  "article_count": 26,
  "pages": 25,
  "page": 1,
  "disease_count": 123,
  "diseases": [
    {
      "key": "congenital abnormalities",
      "article_count": 7,
      "url": "https://mastermind.genomenon.com/detail?disease=congenital+abnormalities&gene=braf&hpo=HP:0001166"
    },
    {
      "key": "neoplasms",
      "article_count": 5,
      "url": "https://mastermind.genomenon.com/detail?disease=neoplasms&gene=braf&hpo=HP:0001166"
    },
    {
      "key": "ectodermal dysplasia",
      "article_count": 3,
      "url": "https://mastermind.genomenon.com/detail?disease=ectodermal+dysplasia&gene=braf&hpo=HP:0001166"
    },
    {
      "key": "gene expression regulation",
      "article_count": 3,
      "url":
"https://mastermind.genomenon.com/detail?disease=gene+expression+regulation&gene=braf&hpo=HP:0001166"
    },
    [clipped]
  ],
  "url": "https://mastermind.genomenon.com/search?disease=&gene=braf&hpo=HP:0001166"
}
```

The `/phenotypes` API endpoint returns the top phenotypes, 5 per page and ranked by number of matched articles for the given query. See API Demo.

Query Parameters

See “Counts” and “Articles” endpoints for all valid query parameters.

Response Format

```
{
  "article_count": [[total number of articles]],
  "pages": [[total number of pages]],
  "page": [[current page of results]],
  "disease_count": [[total number of diseases]],
  "diseases": [
    {
      "key": [[MeSH disease name]],
      "article_count": [[number of articles for disease and given query]],
      "url": [[link to Mastermind for disease and given query]]
    }
  ],
  "url": [[link to Mastermind query for the given gene and variant]]
}
```

Example Response

```
{
  "article_count": 6067,
  "pages": 513,
  "page": 1,
  "phenotype_count": 2562,
  "phenotypes": [
    {
      "key": "HP:0000001",
      "term": "All",
      "article_count": 5914,
      "url": "https://mastermind.genomenon.com/detail?disease=melanoma&gene=braf&hpo=HP:0000001&mutation=braf:V600E"
    },
    {
      "key": "HP:0000118",
      "term": "Phenotypic abnormality",
      "article_count": 5914,
      "url": "https://mastermind.genomenon.com/detail?disease=melanoma&gene=braf&hpo=HP:0000118&mutation=braf:V600E"
    },
    {
      "key": "HP:0002664",
      "term": "Neoplasm",
      "article_count": 5914,
      "url": "https://mastermind.genomenon.com/detail?disease=melanoma&gene=braf&hpo=HP:0002664&mutation=braf:V600E"
    },
    {
      "key": "HP:0011792",
      "term": "Neoplasm by histology",
      "article_count": 5908,
      "url": "https://mastermind.genomenon.com/detail?disease=melanoma&gene=braf&hpo=HP:0011792&mutation=braf:V600E"
    },
    [clipped]
  ],
  "url": "https://mastermind.genomenon.com/search?disease=melanoma&gene=braf&mutation=braf:V600E"
}
```

“Therapies” Endpoint

ADVANCED API

GENOMENON®

The `/therapies` API endpoint returns the top therapies, 5 per page and ranked by number of matched articles for the given query. [See API Demo](#).

Query Parameters

See “Counts” and “Articles” endpoints for all valid query parameters.

Response Format

```
{
  "article_count": [[total number of articles]],
  "pages": [[total number of pages]],
  "page": [[current page of results]],
  "disease_count": [[total number of diseases]],
  "diseases": [
    {
      "key": [[MeSH disease name]],
      "article_count": [[number of articles for disease and given query]],
      "url": [[link to Mastermind for disease and given query]]
    }
  ],
```

Example Response

```
{
  "article_count": 6067,
  "pages": 1461,
  "page": 1,
  "therapy_count": 7303,
  "therapies": [
    {
      "key": "UNII:207SMY3FQT",
      "term": "VEMURAFENIB",
      "article_count": 3455,
      "url":
"https://mastermind.genomenon.com/detail?disease=melanoma&gene=braf&mutation=braf:V600E&unii=UNII:207SMY3FQT"
    },
    {
      "key": "UNII:QGP4HA4G1B",
      "term": "DABRAFENIB",
      "article_count": 2319,
      "url":
"https://mastermind.genomenon.com/detail?disease=melanoma&gene=braf&mutation=braf:V600E&unii=UNII:QGP4HA4G1B"
    },
    {
      "key": "UNII:33E86K87QN",
      "term": "TRAMETINIB",
      "article_count": 1874,
      "url":
"https://mastermind.genomenon.com/detail?disease=melanoma&gene=braf&mutation=braf:V600E&unii=UNII:33E86K87QN"
    },
    {
      "key": "UNII:6T8C155666",
      "term": "IPILIMUMAB",
      "article_count": 1634,
      "url":
"https://mastermind.genomenon.com/detail?disease=melanoma&gene=braf&mutation=braf:V600E&unii=UNII:6T8C155666"
    },
    [clipped]
  ],
```

The `/article_info` API endpoint returns the matched diseases, genes, variants, and article meta-data for a given article PMID. See API demo.

Query Parameters

Name	Description	Example
`pmid`	The PubMed ID for the desired article.	23543365

Response Format

```
{
  "pmid": [[PMID of article]],
  "title": [[PubMed title]],
  "abstract": [[list of abstract parts, each with "index", "text", and "label"]]
  "journal": [[journal ISO abbreviation]],
  "publication_date": [[normalized publication date]],
  "doi": [[DOI from PubMed]],
  "authors": [[list of author names]],
  "mesh_headings": [[list of MeSH heading names from PubMed]],
  "substances": [[list of substance names from PubMed]],
  "gene_symbols": [[list of gene symbols from PubMed]],
  "included_in_mastermind": [[true/false, whether the article is shown in Mastermind]],
  "pubmed_indexed": [[true/false, whether Mastermind has indexed the PubMed data for article]],
  "fulltext_indexed": [[true/false, whether Mastermind has indexed the full-text for article]],
  "supplemental_data_indexed": [[true/false, whether Mastermind has indexed the supplemental data for article]],
  "english_language": [[true/false, whether article is English-language]],
  "categories": [
    { "keywords": [
        { "term": [[keyword label]],
          "type": [[keyword type (either term or pattern)]],
          "key": [[keyword identifier]]
        },…
      ],
      "key": [[category identifier]],
      "name": [[category description]]
    },…
  ],
  "diseases": [
    { "key": [[MeSH disease name]],
      "url": [[link to Mastermind query for disease]]
    }
  ]
  "hpo_terms": [
    { "key": [[HPO identifier]],
      "term": [[phenotype description]]
    },
  ],
  "unii_terms": [
    { "key": [[UNII identifier]],
      "term": [[therapy description]]
    },
  ],
  "genes": [
    {
      "symbol": [[HGVS gene symbol]],
      "mentions": [[number of gene citations in article]],
      "url": [[link to Mastermind for gene]],
      "variants": [
        {
          "key": [[Mastermind variant key name]],
          "matched": [[list of nomenclatures that were matched in the text]],
          "mentions": [[number of variant citations in article]],
          "cdna_effects": [[cdna nomenclatures corresponding to rsID and IVS nomenclatures in "mastched"]],
          "url": [[link to Mastermind for variant]]
        }
      ]
    }
  ],
  "url": [[link to Mastermind query for the article]]
}
```

“Article Info” Endpoint (continued)

Example Response (truncated results, not real data, for illustrative purposes)

```
{
  "pmid": "28717669",
  "title": "Pitfalls in genetic testing: a case of a SNP in primer-annealing region leading to allele dropout in <i>BRCA1</i>.",
  "abstract": [
    {
      "index": 0,
      "text": "Hereditary breast and ovarian cancer is characterized by mutations in <i>BRCA1</i> or <i>BRCA2</i> genes and PCR-based screening
techniques…",
      "label": "BACKGROUND"
    },…
  ],
  "journal": "Mol Genet Genomic Med",
  "publication_date": "2017-05-11",
  "doi": "10.1002/mgg3.295",
  "authors": ["Silva FC",…],
  "mesh_headings": ["BRCA1 Protein",…],
  "substances": ["BRCA1 Protein",…],
  "keywords": ["Allele dropout",…],
```

PubMed meta-data

```
  "included_in_mastermind": true,
  "pubmed_indexed": true,
  "fulltext_indexed": true,
  "supplemental_data_indexed": false,
  "english_language": true,
```

Mastermind meta-data

```
  "categories": [
    {
      "keywords": [
        {
          "term": "wild-type",
          "type": "term",
          "key": "wild-type"
        },…
      ],
      "key": "in vivo-grp",
      "name": "ACMG Interpretation::Functional::in vivo"
    },…
  ],
```

Mastermind Categories/Keywords

```
  "diseases": [
    {
      "key": "breast neoplasms",
      "url": "https://mastermind.genomenon.com/search?disease=breast+neoplasms"
    },…
  ],
```

Mastermind Diseases

```
  "hpo_terms": [
    {
      "key": "HP:0001396",
      "term": "Cholestasis"
    },…
  ],
```

Mastermind Phenotypes

```
  "unii_terms": [
    {
      "key": "UNII:207SMY3FQT",
      "term": "VEMURAFENIB"
    },…
  ],
```

Mastermind Therapies

```
  "genes": [
    {
      "symbol": "BRCA1",
      "mentions": 66,
      "variants": [
        {
          "key": "D1692sd",
          "matched": [
            "c.5074+2T>C",
            "rs80358089"
          ],
          "mentions": 16,
          "cdna_effects": [
            {
              "hgvsc": "NM_007294.3:c.5074+2T>C",
              "rsid": "rs80358089"
            },…
          ],
          "url": "https://mastermind.genomenon.com/detail?gene=brca1&mutation=brca1:D1692sd"
        },…
      ],
      "url": "https://mastermind.genomenon.com/detail?gene=brca1"
    },…
  ],
```

Mastermind Genes/Variants

```
  "url": "https://mastermind.genomenon.com/detail?gene=brca1&pmid=28717669"
}
```

The `/file_annotations/counts` API endpoint provides bulk counts for variants by returning an annotated VCF from an input VCF file containing multiple variants as genomic coordinates with each variant in the output file annotated with the number of articles and Mastermind URL for each variant. **NOTE: This is not HIPAA compliant. Please ensure VCF files are de-identified before uploading to this endpoint.**

This process is made up of four sequential endpoints to ease integration from different workflows and allow robust handling of potentially large files. The four steps are:

1. Create a new bulk annotation process
2. Upload the VCF data
3. Check the status of the annotation process
4. Download the annotated VCF file when ready

The easiest way to use the “File Annotation Counts” endpoint is to use the “Annotate with Evidence” Python script provided in the Mastermind API Cookbook repository.

1. Create a new bulk annotation process

This is a `POST` request to `/file_annotations/counts`. The response will create a unique identifier for your bulk annotation job and respond with a URL for submitting the VCF data as well as the URL endpoint to check the status of the annotation process.

Query Parameters

Name	Description	Example
`assembly`	The build version used to produce the VCF file, either `grch37` or `grch38`.	`grch37`
`filename`	The name of the input VCF file to upload.	`my_file.vcf.gz`

Example Request

```
curl -XPOST -L \
-H "X-API-TOKEN: [[your API token here]]" \
"https://mastermind.genomenon.com/api/v2/file_annotations/counts?filename=my_file.vcf.gz&assembly=grch37"
```

Response Format

```
{
  "job_id": [[unique annotation job identifier]],
  "state": [[new annotation job status]],
  "assembly": [[assembly used for file]],
  "upload_url": [[URL to upload VCF file]],
  "input_filename": [[file name of input file]],
  "created_at": [[timestamp of when job was created]],
  "job_url": [[URL to check status of annotation job]]
}
```

Example Response

```
{
  "job_id": "ff91654c-5ad4-4f1c-9ed3-60a8492ce689",
  "state": "created",
  "assembly": "GRCh37",
  "upload_url": "https://genomenon-vcf-in.s3.us-west-2.amazonaws.com/…",
  "input_filename": "my_file.vcf.gz",
  "created_at": "2019-08-29T08:37:11.081Z",
  "job_url": "https://mastermind.genomenon.com/api/v2/file_annotations/counts/ff91654c-5ad4…"
}
```

2. Upload the VCF Data

This is a `PUT` request to the URL provided in the `upload_url` attribute of the response of step 1. For the upload URL to work, be sure that the response from step 1 is properly JSON-decoded. This will respond with a 200 status code once the upload is complete.

Example Request

```
curl -X PUT --data-binary @/path/to/my_file.vcf -L \
-H "Content-Type: application/octet-stream" \
"https://genomenon-vcf-in.s3.us-west-2.amazonaws.com/…"
```

3. Check the status of the annotation process

This is a `GET` request to `/file_annotations/counts/[[job ID]]` endpoint provided in the `job_url` attribute in the response of step 1. The response will include the status of the job, and once complete will provide the download URL.

Response Format

Example Response

```
{
  "job_id": "ff91654c-5ad4-4f1c-9ed3-60a8492ce689",
  "state": "succeeded",
  "assembly": "GRCh37",
  "input_filename": "my_file.vcf.gz",
  "download_url": "https://mastermind.genomenon.com/api/v2/file_annotations/counts/ff91654c-5ad4…",
  "input_size": 325103,
  "input_etag": "\"087450a21b0bffc946b716f9f18f73cf\"",
  "output_filename": "my_file.vcf.gz",
  "output_size": 431216,
  "output_etag": "\"321b3e7233cdbed6d0821083a1b14e47\"",
  "created_at": "2019-08-29T08:37:11.081Z",
  "started_at": "2019-08-29T08:37:22.649Z",
  "completed_at": "2019-08-29T08:47:32.716Z",
  "elapsed": 284,
  "records": 6980,
  "annotated": 4623,
  "job_url": "https://mastermind.genomenon.com/api/v2/file_annotations/counts/ff91654c-5ad4…"
}
```

4. Download the annotated VCF file when ready

This is a `GET` request to `/file_annotations/counts/[[job ID]]/download` endpoint provided in the `download_url` attribute in the response of step 3. The response will be a gzip-compressed annotated VCF file.

The statuses possible from the `state` attribute returned by the endpoint in step 3 are:

- `created`
- `started`
- `succeeded`
- `failed`

The output and completion attributes will show up once the state has been updated to succeeded. This is also when the download_url will be returned, at which point the file may be downloaded.

Example Annotation

```
#CHROM  POS     ID              REF     ALT     QUAL    FILTER  INFO
1       2160390 rs28384811      C       G       .       PASS
MMCNT=3;MMURI=https://mastermind.genomenon.com/detail?disease=all%20diseases&gene=SKI&mutation=SKI:A
```

200: Success

When querying endpoints, successful responses will return an HTTP response code of `200` (success).

400: Bad Request

Queries lacking an API token will return an HTTP response code of `400` (bad response). A JSON body containing an error key will also be returned explaining what cased the issue, for example:

```
{
  "error": "API token missing"
}
```

Some issues, such as querying an endpoint without URL-encoding parameter values may cause a `400` response without a JSON body.

401: Unauthorized

Queries to API endpoints not granted to your API subscription, using an inactive or expired API token, or hitting your overall annual query limits, will return an HTTP response code of `401` (unauthorized). A JSON body containing an `error` key will also be returned explaining what caused the issue.

404: Not Found

If a given gene, variant, or other input parameter is not found, the API will return an HTTP response code of `404` (not found). A JSON body containing an `errors` key will also be returned, including which input(s) were not recognized, for example:

```
{
  "errors": [
    {
      "message": "specified variant 'blah' not found",
      "parameter": "variant"
    }
  ]
}
```

If this happens, considering querying the `/suggestions` endpoint with your disease, gene, or variant to get the appropriate value for querying the other Mastermind API endpoints.

NOTE: The `/suggestions` endpoint will return an empty result body with a 200 success response status for inputs that are not found, instead of a 404 error response.

422: Unprocessable Entity

Invalid queries, such as querying an endpoint with required fields omitted, will return an HTTP response code of `422` (unprocessable entity). A JSON body containing an `errors` key will also be returned explaining what caused the issue, for example:

```
{
  "errors": [
    {
      "message": "at least one parameter (disease, gene, or variant) must be specified"
    }
  ]
}
```

429: Too Many Requests

Queries which exceed the per-second or per-day throughput rate limits will temporarily return an HTTP response code of `429` (too many requests). If this happens, wait or throttle requests before trying again.

408: Timed Out

Occasionally queries may time out and temporarily return an HTTP response code of `408` (timed out). If this happens, wait or throttle requests before trying again. If it happens for a given request multiple times in a row, consider skipping the request, and letting us know.

500: Internal Server Error

Occasionally queries may cause an internal server error and return an HTTP response code of `500` (internal server error). If this happens, wait or throttle requests before trying again. If it happens for a given request multiple times in a row, consider skipping the request, and letting us know.

503: Service Unavailable

Occasionally queries may cause a service unavailable error and return an HTTP response code of `503` (service unavailable error). Similarly to 408 errors, if this happens, wait or throttle requests before trying again.

GENOMENON®

AI-DRIVEN GENOMICS

Cited Variants Reference

Cited Variants Reference

The Mastermind Cited Variants Reference file is a VCF file providing a snapshot of variant evidence available in Mastermind. Below is a truncated excerpt:

HGVS	MMCNT1	MMCNT2	MMCNT3	MMURL3
g.3209075C>T	4	4	4	https://mastermind.genomenon.com/… mutation=SLC4A11:Q812sa
g.865595A>G	1	1	1	https://mastermind.genomenon.com/… mutation=SAMD11:K45E
g.874491_874492delinsTA	0	0	1	https://mastermind.genomenon.com/… mutation=SAMD11:R168X

`MMCNT1` column (most specific)

cDNA-level exact matches. This is the number of articles that mention the variant at the nucleotide level in either the title/abstract or the full-text.

`MMCNT2` column

cDNA-level possible matches. This is the number of articles with nucleotide-level matches (from 1) plus articles with protein-level matches in which the publication did not specify the cDNA-level change, meaning they could be referring to this nucleotide-level variant but there is insufficient data in these articles to determine conclusively.

`MMCNT3` column (most sensitive)

This is the number of articles citing any variant resulting in the same biological effect as this variant. This includes the articles from MMCNT1 and MMCNT2 plus articles with alternative cDNA-level variants that result in the same protein effect.

`MMURL3` column

This is a deep-link into Mastermind for the selected variant, which shows all articles from MMCNT3, in order to investigate and explore the evidence in the literature.

If you are integrating the link URLs from the CVR into a customer-facing application, please contact us for instructions on properly embedding your own reference code into your Mastermind URLs to properly attribute users.

Current Version

To fetch the current version of the CVR in JSON format, automated periodic GET requests may be performed using the following URL. This will allow automatic detection of CVR updates to re-import the latest data available via the CVR:

```
curl https://mastermind.genomenon.com/cvr/version
```

The response will be JSON formatted. An example response is as follows:

```
{"version":"2020.04.02"}
```

Limitations

While the file does contain over 7.1 million variants seen in the medical literature, it doesn't include everything in Mastermind's ever-expanding database.

Some citations in the medical literature describe variants ambiguously (e.g. "BRCA1:p.L1795fs" to indicate some frameshift at that location). We index those variants, but it's not possible to definitively specify the cited variant with the nucleotide change causing the frameshift, which is what is required in the VCF format using genomic coordinates with the "ref" and "alt" alleles for each row of data.

The CVR therefore includes all variants cited with nucleotide-level precision, as well as protein-level nomenclatures for amino acid substitutions which have a small, finite number of nucleotide-specific possibilities (each of which will be included in the CVR providing the highest possible sensitivity for amino acid substitution lookups based on genomic coordinates in the CVR).

Additionally, since the CVR static files are released quarterly, the data will not be as up-to-date as the real-time API.

To summarize, the CVR:

- May be up to 3 months out of date.
- Only includes references for substitution variants and other types of variants that are specifically cited in the medical literature using valid nucleotide-level nomenclature.
- For variants that are non-coding, frameshifts, duplications, deletions, insertions, indels, or inversions, we recommend querying the Mastermind API if they are not in the reference file, for maximum sensitivity.

For more information about using the Cited Variants Reference and for licensing information, please see the Mastermind Cited Variants Reference download page.

GENOMENON®

AI-DRIVEN GENOMICS

Integration Effort Overview
API vs. CVR

Integration Effort: API vs. CVR

CVR Overview

- Flat file
- VCF or CSV format

Steps to integrate:

1. Write code to download and store CVR file on some server or in file storage such as S3.
2. Write code to open file on server in-memory, and parse either VCF or CSV format.
3. Write code to store count data in own application's database.
4. Write code to periodically check for update and replace file on server or in file storage.
5. Write code to re-process file on server in-memory and update data in own application's database.
6. Write code to filter or prioritize data based on count information. (*Optional)*
7. Mock up and design the display of count data and link-outs from count information in own application's user interface. (*Optional)*
8. Write code to show count data and link-outs from count information in own application's user interface. (*Optional)*

API Overview

- RESTful API
- JSON format

Steps to integrate:

1. Write code to query RESTful API and parse result.
2. Write code store count data in own application's database.
3. Write code to periodically re-query API and update data in own application's database.
4. Write code to filter or prioritize data based on count information. (*Optional)*
5. Mock up and design the display of count data and link-outs from count information in own application's user interface. (*Optional)*
6. Write code to show count data and link-outs from count information in own application's user interface. (*Optional)*

Commonality

The CVR and API both require writing code to check fetch count data, parse the results, store in database, and design and build integration of count data into the user interface.

Integration Effort: API vs. CVR, cont.

The CVR requires more effort to actually fetch, parse, store, and update the count information.

If the integrator happens to have already built into their pipeline a *reusable* method to download, parse, import, and update annotation data from flat VCF files, then steps 1-4 can be shortened as they'll require less (but not zero) effort to add us as an additional source for information.

However, if the integrator happens to have already built into their pipeline a reusable method to query RESTful APIs, which is not uncommon, then steps 1-3 of the API integration can also be shortened in the same way.

When is the CVR easier to integrate?
When the integrator:

- has built a reusable method for downloading, parsing, importing, and updating annotation data from flat VCF files, *but*
- has not built the ability into their pipeline to query and integrate information from other APIs.

When is the API easier to integrate?
When *either* of the above two criteria are not true, the API will require fewer and simpler steps to integrate the data.

In our industry, this does happen (where both criteria for the CVR being easier are true), but it's not a given. In fact, this is why the CVR was conceived.

However, we also can't know ahead of time for any given integrator which will be easier, since we don't know what other sources they've already built integrations for, or how reusable and modular their existing integrations were made.

The other thing that makes the CVR easier is if they have a large number of variants to annotate at once, because it's easier to bulk-annotate many variants at once with a large file than it is to loop through each variant and query each one at a time. However, the File Annotations endpoint in our API handles this use-case, so it's often the case that the integrator may be thinking the CVR is easier due to the bulk-nature of the data, simply because they don't know the File Annotations endpoint in the API exists, which can bulk annotate and has the API advantage of being up-to-date and complete, compared to the CVR.

NOTE: If integrating the CVR or API into software that will be available outside of your organization, contact us for a unique invitation code prior to development at support@genomenon.com.