**This document outlines the Knowledge Beacon ("KB") API Workflow**

# I. Discover available semantics

## A. Concept ("Data") Types

**Input:** none

**Output:** (JSON-encoded) list of concept categories known by the beacon, with code 'id', (optional) uri to the associated concept, and 'frequency' of use in the knowledge beacon, e.g.

```
[
      …
        {
          "category": "chemical_substance",
          "frequency": 58812,
          "id": "BLM:ChemicalSubstance",
          "uri": "http://bioentity.io/vocab/ChemicalSubstance"
        },
      …
]
```

**API endpoint:**

```
GET /categories
```

## B. Predicates

**Input:** none

**Output:** (JSON-encoded) list of predicate relationships used by the beacon with similar information about predicates.

```
[
      …
        {
          "id": "NDEX:involved_in",
          "uri": null,
          "edge_label": "related_to",
          "relation": "related_to",
          "local_id": "NDEX:involved_in",
          "local_uri": null,
          "local_relation": "related_to",
          "description": "",
          "frequency": 270
        },…
]
```

**API endpoint:**

```
GET /predicates
```

## C. Knowledge Map

**Input:** none

**Output:** (JSON-encoded) generic list of knowledge statement type triples ("subject category", predicate, "object category") published by the beacon, e.g.


```
[
      …
      {
          "subject": {
            "category": "gene",
            "prefixes": [
              "hgnc.symbol",
              "ncbigene",
              "genecards",
              "uniprot knowledgebase",
              "HPRD",
              "UniProt Isoform",
              "NCBI Gene",
              "HGNC"
            ]
          },
          "predicate": {
            "edge_label": "related_to",
            "relation": "NDEX:3",
            "negated": null
          },
          "object": {
            "category": "named thing",
            "prefixes": [
                  "BIOCYC"
             ]
          },
          "frequency": 1,
          "description": "gene - 3 - named thing"
        }, …
]
```


**API endpoint:**

```
GET /kmap
```

# II. Discover a list of candidate concepts of interest (by keyword)

**Input:** array (comma separated or proper URL array) of keywords (URL encoded, space delimited) to match against the known names of a concept.

**Output:** (paged) list of matching concepts with simple metadata including CURIE[1], name, semantic group, synonyms and definition, insofar available

**Variants:** Additional keywords and semantic group filters can constrain the list; results may be batched as pages by page number and size

**API endpoint:**

`GET /concepts?keywords=<keyword1%2Ckeyword2%2C…keywordn>`

# III. User selects a concept to get at a table of all its relationships

Once a user *selects* a particular concept of interest from the list of concepts originally matched by keyword and semantic group constraints, the intensional meaning of the user's search has become anchored. However, the chosen concept record originated from only a single Knowledge Beacon source. In order to retrieve all related data from all available Knowledge Beacons, the workflow must identify *all* equivalent concepts across *all* those KB's. To "prime the pump" of the search, the CURIE of the originally matched concept is provided as a list of one element to the **/exactmatches** API call.

## A. System identifies equivalent concepts identifiers across KS's

### 1. Given a list of exact match concepts, return equivalent concepts

Concept equivalency is discerned through exact matches (*sensa* **skos:exactMatches** and **owl:sameAs**) of any concept canonical or equivalent identifiers known to the given KB, with at least one of the input CURIEs. Any such "exact match concept identifiers" ('emci') not seen in the original input list, are returned as additional identifiers deemed equivalent, to allow iterative discovery of equivalent concepts using this API call.

**Input:** Array of *1..n* CURIEs of concepts currently known as exact matches to the current concept of interest

**Output:** List of exact matches (sense **skos:exactMatches** and **owl:sameAs**) associated with the concept

**API endpoint:**

`GET /exactmatches?c=<emci1>&c=<emci2>…&c=<emcin>`

[1] Compact Uniform Resource Identifiers: https://en.wikipedia.org/wiki/CURIE

### 2. Iterative discovery of an "equivalent concept clique"

The lists of input and additional exact matching concept identifiers from II.A.1 above are consolidated into a union set, which is then re-used in iterative calls to the `/exactmatches` API endpoint until the none of the KBs return additional "exact matching concept" identifiers, suggesting that a complete clique of equivalent identifiers is compiled (insofar known by the available KBs)

## B. Set of equivalent concepts are used to retrieve related statements.

**Input:** Array of "Equivalence clique" of *1..m* exact matching concept identifiers ('emci') from II.A.2 above)

**Output:** List of subject-predicate statements in which either the subject or object concepts match at least one member of the input concept list.

**API endpoint:**

`GET /statements?c=<emci1>&c=<emci2>…&c=<emcim>`

# IV. User requests full details about a specific concept

**Input:** A specified globally unique user-selected concept identifier

**Output:** A more complete report of properties of the concept from a given KB

**API endpoint:**

`GET /concepts/{conceptId}`

# V. User selects a specific statement to get at statement details

**Input:** The `statementId` associated with a given statement (from the output of III.B)

**Output:** A list of metadata documenting the statement, including evidence list of citations in which each entry is specified as a (internet resolvable) CURIE, human readable label ("title") and date of origin ("publication date").

**API endpoint:**

`GET /statements/{statementId}`