This document outlines the TKBio workflow as a series of KSAPI calls

I. Discover a list of candidate concepts

Input: A URL encoded character string to match against the canonical or aliases of a concept

Output: (paged) list of matching concepts from all KS's supporting the API call

Variants: Text and semantic type filters can constrain the list; results are batched as pages

API endpoint:

```
GET /concepts?textFilter=<search string>
```

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

A. System identifies equivalent concepts identifiers across KS's

1. Given a concept, return all associated cross references

Input: A specified globally unique user-selected concept identifier

Output: List of cross-references associated with the concept

API endpoint:

```
GET /xref/{conceptId}
```

2. Given a list of qualified cross-references (e.g. output from II.A.1 above), return equivalent concepts

Concept equivalency is discerned through matches with at least one of the input cross-references, either to the primary identifier of the concept or its associated cross references. Concepts are also returned with their associated cross-references, to allow iterative discovery of equivalent concepts using this API call.

Input: List of 1..n cross-references associated with a user selected concept

Output: List of all other concepts with matching cross-references

API endpoint:

GET /xref?xi=<*xref*$_1$>&xi=<*xref*$_2$>…&xi=<*xref*$_n$>

3. Iterative discover of the equivalent concept clique

The lists of cross-references from II.A.3 above are consolidated into a union set and identifiers that were already initially run are subtracted from the resulting set, then the difference set of identifiers are used in iterative calls to II.A.2 until the resulting "equivalent concept" identifier clique ceases to expand, suggestive of a complete clique (to the extent known by the available KS's)

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

Input: "Equivalence clique" of 1..m concept identifiers (from II.A.3 above)

Output: List of matching subject-predicate statements

API endpoint:

GET /statements?xi=<*xref*$_1$>&xi=<*xref*$_2$>…&xi=<*xref*$_m$>

III. User requests details about a specific concept

Input: A specified globally unique user-selected concept identifier

Output: A more complete report of properties of the concept

API endpoint:

```
GET /concepts/{conceptId}
```

IV. User selects a specific statement to get at evidence

Input: The evidenceId associated with a given statement (from the output of II.B)

Output: A list of citation references supporting the statement

API endpoint:

```
GET /evidence/{evidenceId}
```