EKP public API implementation

Introduction

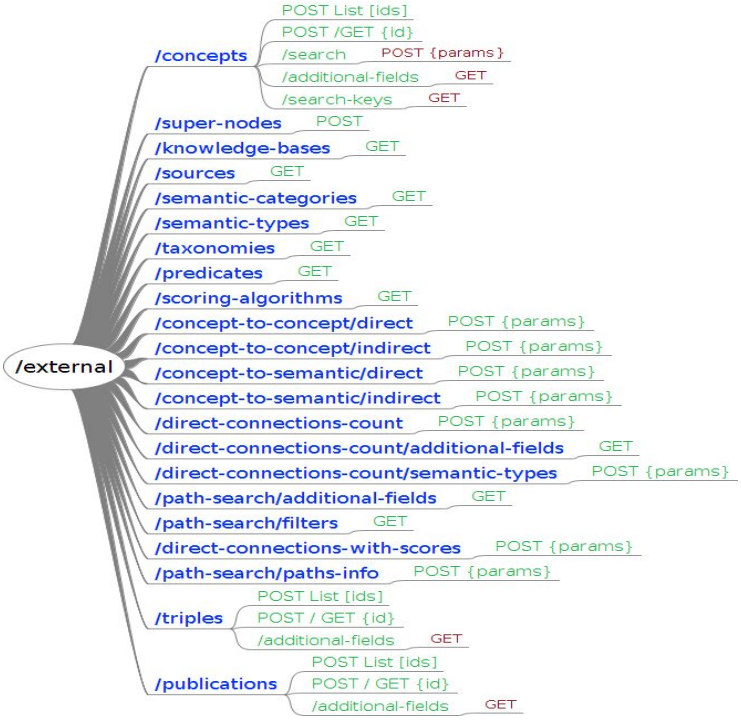
This document describes the API calls available to query the Euretos knowledge network. Most queries use a predefined key assignment which are associated with a predefined value list.

To ensure proper use of the (predefined) key, (predefined) value pairs as available within the API call structure several enumeration calls are implemented which respond with available key or value attributes. It is recommended to initialize a client implementation at the start of the connection with the enumeration attributes to ensure the client is using valid attributes. One of the advantages of the enumeration is that future changes can be automatically incorporated by the client implementation ensuring all future attributes are available for use.

Only few api calls require values which cannot be part of a predefined list. The most obvious one would be the concept search API call where the term value is user based input. For example a search for a thesaurus match on the term “cancer”. The remaining values which are not part of a predefined list are concept, triple and publication identifiers. Often details are desired for a triple and the triple API calls require a triple Id to resolve.

Overview API calls

The diagram below provides an overview of the implemented API calls.



API details

Below the API request / response models as available. A test environment is available through a swagger implementation on the server.

@RequestMapping("/external")

@RequestMapping(value = "/concepts", method = RequestMethod.POST)

Returns concept details for the given id list, "additionalFields" (list) can be included to enrich the response.

@param ids (M) list with string based concept id's.

@param additionalFields (O), see externalconceptsadditional-fields API call.

Info; Most results will return the concept id. This call allows translation to human understandable information. The "additionalFields" allow further response enrichment of the concept details.

Example "additionalfields"; "description", "semanticCategory", "semanticTypes", "taxonomies", "measures", "accessMappings", "hasTriples", "synonyms", "source", "knowledgebase".

Example query; { "ids": ["2790858", "2790859"] } .

Example query; { "additionalFields": [""], "ids": ["2790858", "2790859"] } .

Example query; { "additionalFields": ["synonyms", "source"], "ids": ["2790858", "2790859"] } .

Optional dependency response; externalconceptsadditional-fields API call.

@return default concept; preferred term, id

@RequestMapping(value = "/concepts/{id}", method = {RequestMethod.POST, RequestMethod.GET})

Returns concept details for the given id, "additionalFields" (list) can be included to enrich the response.

@param id (M) string based concept id's.

@param additionalFields (O), see externalconceptsadditional-fields API call.

Info; Most results will return the concept id. This call allows translation to human understandable information. The "additionalFields" allow further response enrichment of the concept details.

Example "additionalFields"; "description", "semanticCategory", "semanticTypes", "taxonomies", "measures", "accessMappings", "hasTriples", "synonyms", "source", "knowledgebase".

Example query; { "id":"2790858" } .

Example query; { "additionalFields": [""], "id": "2790858" } .

Example query; { "additionalFields": ["synonyms", "source"], "id": "2790858"} .

Optional dependency response; externalconceptsadditional-fields API call.

@return default concept; preferred term, id

@RequestMapping(value = "/super-nodes", method = {RequestMethod.POST})

Returns list of concepts which are part of the super node list, "additionalFields" (list) can be included to enrich the response.

@param additionalFields (O), see externalconceptsadditional-fields API call.

Info; It is possible in several api calls to include or exclude so called super-nodes. Super nodes are considered "large" based on a manual curated list. This api call (super-nodes) can be used to understand which nodes (read concepts) are part of the super node list.The "additionalFields" allow further response enrichment of the concept details.

Example "additionalFields"; "description", "semanticCategory", "semanticTypes", "taxonomies", "measures", "accessMappings", "hasTriples", "synonyms", "source", "knowledgebase".

Example query; { "additionalFields": ["source"] } .

Optional dependency response; externalconceptsadditional-fields API call.

@return default concept; preferred term, id

@RequestMapping(value = "/knowledge-bases", method = RequestMethod.GET)

Returns list of "knowledgebases" values.

@param no parameters required

Info; The concept search API call (conceptssearch) requires a query to be based on a key, value pair. The search-keys response will provide a list of available search keys which can be used within the queryString. Please note that the logical conditions AND, OR, NOT and grouping ( ) are not included in this response.

The values as available for the key "knowledgebases" can be retrieved using the externalknowledge-bases call.

@return "knowledgebases" values

@RequestMapping(value = "/sources", method = RequestMethod.GET)

Returns list of "sources" values.

@param no parameters required

Info; The concept search API call (conceptssearch) requires a query to be based on a key, value pair. The search-keys response will provide a list of available search keys which can be used within the queryString. Please note that the logical conditions AND, OR, NOT and grouping ( ) are not included in this response.

The values as available for the key "sources" can be retrieved using the externalsources call.

@return "sources" values

@RequestMapping(value = "/concepts/search", method = RequestMethod.POST)

Returns list of concepts which match the search criteria, "additionalFields" (list) can be included to enrich the response.

@param queryString (M), Key,value pair based search criteria. Value need to be enclosed using
single quotes '...'. Supported keys are available using externalconcept-search-keys. Associated values can be retrieved through various enum calls, such as externalsources or externalknowledge-bases. Logical conditions AND, OR, NOT can be used within the query with leadtail space. Logical grouping ( ) can be used within query with lead tail space. The "term:" key is required for the API call to succeed.

@param searchType (M) STRING or TOKEN, STRING will consider the term field as one complete
word. TOKEN will consider the term field as multiple words where each word is part of search criteria.

@param additionalFields (O) see externalconceptsadditional-fields API call.

Info; This api call is the starting point for the knowledge network. Here a literal term will be linked to an internal concept identifier. The key "term" is mandatory for the search to succeed. The term will be used to search through the thesaurus and match both preferred and synonyms by default. When specifying "preferred:'true'" within the query, only preferred terms will be matched. Please note that the internal concept identifier is not guaranteed persistent with new data releases. When documenting a resolved concept, please ensure documenting the concept details such as name, synonyms, source,
knowledgebase etc. This will allow resolving the concept in future datasets.

Example "additionalFields"; "description", "semanticCategory", "semanticTypes", "taxonomies", "measures", "accessMappings", "hasTriples", "synonyms", "source", "knowledgebase".

Example query search keys; "term", "semantictype", "semanticcategory", "knowledgebase",
"source", "gi", "preferred".

Example "source" query values; "umls", "uniprot", "entrezgene".

Example "knowledgebase" query values; "mim", "chebi", "umls".

Example "semantictype" query values; "T116", "T028".

Example "semanticcategory" query values; ""Disorders", "Occupations".

Example "preferred" query values; "true", "false".

Example queryString; { "queryString": "term:'egfr'", "searchType": "STRING" } .

Example queryString; { "additionalFields": [""], "queryString": "term:'egfr'", "searchType": "TOKEN" } .

Example queryString; { "additionalFields": ["synonyms", "source"], "queryString": "term:'egfr'", "searchType": "TOKEN" } .

Example queryString; { "additionalFields": ["synonyms", "source"], "queryString": "term:'egfr' AND (source:'umls' OR source:'uniprot')", "searchType": "TOKEN" } .

Dependencies query; externalconcept-search-keys, externalsemantic-categories, externalsemantic-types, externaltaxonomies, externalknowledge-bases, externalsources

Optional dependency response; externalconceptsadditional-fields

@return default concept; preferred term and id

@RequestMapping(value = "/concepts/additional-fields", method = RequestMethod.GET)

Returns list of available concept "additionalfield" values.

@param no parameters required

Info; For most concept related commands "additionalFields" can be specified to enrich the response. Using this call (conceptsadditional-fields) allows you to understand which additionalFields can be used.

@return concept "additionalfield" values

@RequestMapping(value = "/concepts/search-keys", method = RequestMethod.GET)

Returns list of available (conceptssearch) "queryString" key names which can be used within the concept search call.

@param no parameters required

Info; The concept search API call (conceptssearch) requires a query to be based on a key, value pair. The conceptssearch-keys response will provide a list of available search keys which can be used within the queryString. Please note that the logical conditions AND, OR, NOT and grouping ( ) are not included in this response.

The values which can be used to make the (conceptssearch) queryString key,value pair complete can be retrieved using the following API calls; externalsemantic-categories, externalsemantic-types, externaltaxonomies, externalknowledge-bases, externalsources.

Example query search keys; "term", "semantictype", "semanticcategory", "knowledgebase",
"source", "gi", "preferred".

Example source query values; "umls", "uniprot", "entrezgene".

Example knowledgebase query values; "mim", "chebi", "umls".

Example semantictypes query values; "T116", "T028".

Example semanticcategory query values; ""Disorders", "Occupations".

Example preferred query values; "true", "false".

Example queryString as used with conceptsearch API call; { "queryString": "term:'egfr' AND source:'umls'", "searchType": "STRING" } .

Associated API calls; externalsemantic-categories, externalsemantic-types, externaltaxonomies, externalknowledge-bases, externalsources

@return concept search "queryString" key names

@RequestMapping(value = "/semantic-categories", method = {RequestMethod.GET, RequestMethod.POST})

Returns list of semantic category values

@param no parameters required

Info; The concept search API call (conceptssearch) requires a query to be based on a key, value pair. The search-keys response will provide a list of available search keys which can be used within the queryString. Please note that the logical conditions AND, OR, NOT and grouping ( ) are not included in this response.

The values as available for the key "semanticcategory" can be retrieved using the externalsemantic-categories call.

Example queryString as used with conceptsearch API call; { "queryString": "term:'egfr' AND semanticcategory:'Disorders'", "searchType": "STRING" } .

@return semantic category values

@RequestMapping(value = "/semantic-types", method = {RequestMethod.GET, RequestMethod.POST})

Returns list of semantic type values

@param no parameters required

Info; The concept search API call (conceptssearch) requires a query to be based on a key, value pair. The search-keys response will provide a list of available search keys which can be used within the queryString. Please note that the logical conditions AND, OR, NOT and grouping ( ) are not included in this response.

The values as available for the key "semantictypes" can be retrieved using the externalsemantic-types call.

Example queryString as used with conceptsearch API call; { "queryString": "term:'egfr' AND semantictypes:'T116'", "searchType": "STRING" } .

@return semantic type values

@RequestMapping(value = "/taxonomies", method = RequestMethod.POST)

Returns list of taxonomy values

@param no parameters required

Info; The concept search API call (conceptssearch) requires a query to be based on a key, value pair. The search-keys response will provide a list of available search keys which can be used within the queryString. Please note that the logical conditions AND, OR, NOT and grouping ( ) are not included in this response.

The values as available for the key "taxonomies" can be retrieved using the externaltaxonomies call.

Example queryString as used with conceptsearch API call; { "queryString": "term:'egfr' AND taxonomies:'Homo Sapiens'", "searchType": "STRING" } .

@return taxonomy values

@RequestMapping(value = "/concept-to-concept/direct", method = RequestMethod.POST)

Returns direct path between "left hand" concept id's and "right hand" concepts id's.

@param leftInputs (M) list with string based concept id's.

@param rightInputs (M) list with string based concept id's.

@param additionalFields (O) list with string based additional fields from externalpath-searchadditional-fields.

@param negativeFilters (Obsolete, please remove) list with filter parameters from externalpath-searchfilters.

@param positiveFilters (Obsolete, please remove) list with filter parameters from externalpath-searchfilters.

@param relationshipWeightAlgorithm (M) string based parameter from externalscoring-algorithms.

@param sort (M) string based value using ASC or DESC.

Notes; Returns direct path between "left hand" concept id's and "right hand" concepts id's. AdditionalFields (list) can be included to enrich the response based on externalpath-searchadditional-fields.

Relationship weight algorithm must be included to rank the paths based on externalscoring-algorithms. Sorting is mandatory and could be ASC or DESC.

Negative and positive filters are obsolete. Please remove from the query.

Example query; { "leftInputs": ["4047995"], "relationshipWeightAlgorithm": "PWS", "rightInputs": ["3062402"], "sort": "ASC" }.

Example query; { "additionalFields":[""], "leftInputs": ["4047995"], "relationshipWeightAlgorithm": "PWS", "rightInputs": ["3062402"], "sort": "ASC" }.

Example query; { "additionalFields":["publicationIds", "tripleIds", "predicateIds"], "leftInputs": ["4047995"], "relationshipWeightAlgorithm": "PWS", "rightInputs": ["3062402", "12345"], "sort": "ASC" }.

Example additional Fields; "description", "semanticCategory", "semanticTypes", "taxonomies", "measures", "accessMappings", "hasTriples", "synonyms", "source",

"knowledgebase", "concepts", "publicationsCount", "publicationIds".

Example relation weight algorithms; "lws", "jis", "pws", "plws".

Mandatory dependency request: externalscoring-algorithms, "sort":"ASC" or "DESC".

Optional dependency response: externalpath-searchadditional-fields API call.

@return content:[{concepts:[{id, name},{id,name}], relationships:[{concept0Id, concept1Id}], score}]

@RequestMapping(value = "/concept-to-concept/indirect", method = RequestMethod.POST)

Returns indirect path between "left hand" concept id's and "right hand" concepts id's.

@param leftInputs (M) list with string based concept id's

@param rightInputs (M) list with string based concept id's

@param additionalFields (O) list with string based additional fields from externalpath-searchadditional-fields

@param negativeFilters (O) list with filter parameters from path-searchfilters

@param positiveFilters (O) list with filter parameters from path-searchfilters

@param relationshipWeightAlgorithm (M) string based parameter from externalscoring-algorithms

@param sort (M) string based value using ASC or DESC

Info; Returns indirect path between "left hand" concept id's and "right hand" concepts id's. AdditionalFields (list) can be included to enrich the response based on externalpath-searchadditional-fields.

Negative and positive filters (lists) are based on a prefix,value pair. To make a filter for a concept id 12345, the filter construct would look like 'concept:12345'. For semantic type the filter construct would look like 'st:T116'. When providing multiple filter prefix,value pairs keep in mind that the filter is based on intersect.

Relationship weight algorithm must be included to rank the paths based on externalscoring-algorithms. Sorting is mandatory and could be ASC or DESC.

Example query; { "leftInputs": ["4047995"], "relationshipWeightAlgorithm": "PWS", "rightInputs": ["3062402"], "sort": "ASC" }.

Example query; { "additionalFields":[""], "leftInputs": ["4047995"], "relationshipWeightAlgorithm": "PWS", "rightInputs": ["3062402"], "sort": "ASC" }.

Example query; { "additionalFields":["publicationIds", "tripleIds", "predicateIds"], "leftInputs": ["4047995"], "relationshipWeightAlgorithm": "PWS", "rightInputs": ["3062402"], "sort": "ASC" }.

Example query; { "additionalFields": [""], "leftInputs": ["4047995"],

"positiveFilters": ["sc:Disorders"], "relationshipWeightAlgorithm": "pws", "rightInputs": ["sc:Disorders"], "sort": "ASC" }.

Example additional Fields; "description", "semanticCategory", "semanticTypes", "taxonomies", "measures", "accessMappings", "hasTriples", "synonyms", "source", "knowledgebase", "concepts", "publicationsCount", "publicationIds".

Example relation weight algorithms; "lws", "jis", "pws", "plws".

Example search filter prefixes; "concept:", "tax:", "st:", "sc:", "supernodes", "predicate:".

prefix concept: is associated with a node in the network

prefix tax: is associated with a taxonomy value (externaltaxonomies)

prefix st: is associated with a semantic type value (externalsemantic-types)

prefix sc: is associated with a semantic category value (externalsemantic-categories)

prefix predicate: is associated with a predicate value (externalpredicates)

Relationship weight algorithm must be included to rank the paths based on externalscoring-algorithms. Sorting is mandatory and could be ASC or DESC.

Mandatory dependency request: externalscoring-algorithms, "sort":"ASC" or "DESC".

Optional dependency request: externalpath-searchfilter,externalsemantic-categories, externalsemantic-types, externaltaxonomies, externalpredicates.

Optional dependency response: externalpath-searchadditional-fields API call.

@return content:[{concepts:[{id, name},{id, name},{id, name}] , relationships:[{concept0Id, concept1Id},{concept0Id, concept1Id}] , score}]

@RequestMapping(value = "/concept-to-semantic/direct", method = RequestMethod.POST)

Returns direct path between "left hand" concept id's and "right hand" semantic field(s).

@param leftInputs (M) list with string based concept id's

@param rightInputs (M) list with string based semantic field prefix, value pairs

@param negativeFilters (Obsolete, please remove) list with filter parameters from externalpath-searchfilters.

@param positiveFilters (Obsolete, please remove) list with filter parameters from externalpath-searchfilters.

@param relationshipWeightAlgorithm (M) string based parameter from externalscoring-algorithms.

@param sort (M) string based value using ASC or DESC.

Info; Returns direct path between "left hand" concept id's and "right hand" semantic field(s). AdditionalFields (list) can be included to enrich the response based on externalpath-searchadditional-fields.

The semantic field(s) are based on a prefix,value pair. To make a semantic field for a concept id 12345, the filter construct would look like 'concept:12345'. For semantic type the semantic field construct would look like 'st:T116'. When providing multiple semantic fields keep in mind that the combination is based on intersect.

Relationship weight algorithm must be included to rank the paths based on externalscoring-algorithms. Sorting is mandatory and could be ASC or DESC.

Example query; { "leftInputs": ["4047995"], "relationshipWeightAlgorithm": "PWS", "rightInputs": ["sc:Disorders"], "sort": "ASC" }.

Example query; { "additionalFields":[""], "leftInputs": ["4047995"], "relationshipWeightAlgorithm": "PWS", "rightInputs": ["st:T116"], "sort": "ASC" }.

Example query; { "additionalFields":["publicationIds", "tripleIds", "predicateIds"], "leftInputs": ["4047995"], "relationshipWeightAlgorithm": "PWS", "rightInputs": ["sc:Disorders", "st:T116"], "sort": "ASC" }.

Example additional Fields; "description", "semanticCategory", "semanticTypes", "taxonomies", "measures", "accessMappings", "hasTriples", "synonyms", "source",

"knowledgebase", "concepts", "publicationsCount", "publicationIds".

Example relation weight algorithms; "lws", "jis", "pws", "plws".

Example search filter prefixes; "concept:", "tax:", "st:", "sc:", "supernodes", "predicate:".

prefix concept: is associated with a node in the network

prefix tax: is associated with a taxonomy value (externaltaxonomies)

prefix st: is associated with a semantic type value (externalsemantic-types)

prefix sc: is associated with a semantic category value (externalsemantic-categories)

prefix predicate: is associated with a predicate value (externalpredicates)

Negative and positive filters are obsolete. Please remove from the query.

Mandatory dependency request: externalscoring-algorithms, "sort":"ASC" or "DESC", externalpath-searchfilter,externalsemantic-categories, externalsemantic-types, externaltaxonomies, externalpredicates.

Optional dependency response: externalpath-searchadditional-fields API call.

@return content:[{concepts:[{id, name},{id,name}], relationships:[{concept0Id, concept1Id}], score}]

@RequestMapping(value = "/concept-to-semantic/indirect", method = RequestMethod.POST)

Returns indirect path between "left hand" concept id's and "right hand" semantic field(s).

@param leftInputs (M) list with string based concept id's

@param rightInputs (M) list with string based semantic field prefix, value pairs

@param additionalFields (O) list with string based additional fields from externalpath-searchadditional-fields

@param negativeFilters (O)list with string based prefixes from path-searchfilters combined with value

@param positiveFilters (O)list with string based prefixes from path-searchfilters combined with value

@param relationshipWeightAlgorithm (M) string based parameter from externalscoring-algorithms

@param sort (M) string based value using ASC or DESC

Info; Returns indirect path between "left hand" concept id's and "right hand" semantic field(s). AdditionalFields (list) can be included to enrich the response based on externalpath-searchadditional-fields.

The semantic field(s) are based on a prefix,value pair. To make a semantic field for a concept id 12345, the semantic field construct would look like 'concept:12345'. For semantic type the semantic field construct would look like 'st:T116'. When providing multiple semantic fields keep in mind that the combination is based on intersect.

Negative and positive filters (lists) are based on a prefix,value pair. To make a filter for a concept id 12345, the filter construct would look like 'concept:12345'. For semantic type the filter construct would look like 'st:T116'. When providing multiple filter prefix,value pairs keep in mind that the filter is based on intersect.

Relationship weight algorithm must be included to rank the paths based on externalscoring-algorithms. Sorting is mandatory and could be ASC or DESC.

Example query; { "leftInputs": ["4047995"], "relationshipWeightAlgorithm": "PWS", "rightInputs": ["sc:Disorders"], "sort": "ASC" }.

Example query; { "additionalFields":[""], "leftInputs": ["4047995"],

"relationshipWeightAlgorithm": "PWS", "rightInputs": ["st:T116"], "sort": "ASC" }.

Example query; { "additionalFields":["publicationIds", "tripleIds", "predicateIds"], "leftInputs": ["4047995"], "relationshipWeightAlgorithm": "PWS", "rightInputs": ["sc:Disorders", "st:T116"], "sort": "ASC" }.

Example additional Fields; "description", "semanticCategory", "semanticTypes", "taxonomies", "measures", "accessMappings", "hasTriples", "synonyms", "source", "knowledgebase", "concepts", "publicationsCount", "publicationIds".

Example relation weight algorithms; "lws", "jis", "pws", "plws".

Example search filter prefixes; "concept:", "tax:", "st:", "sc:", "supernodes", "predicate:".

prefix concept: is associated with a node in the network

prefix tax: is associated with a taxonomy value (externaltaxonomies)

prefix st: is associated with a semantic type value (externalsemantic-types)

prefix sc: is associated with a semantic category value (externalsemantic-categories)

prefix predicate: is associated with a predicate value (externalpredicates)

Mandatory dependency request: externalscoring-algorithms, "sort":"ASC" or "DESC", externalpath-searchfilter,externalsemantic-categories, externalsemantic-types, externaltaxonomies, externalpredicates.

Optional dependency response: externalpath-searchadditional-fields API call.

@return content:[{concepts:[{id, name},{id, name},{id, name}] , relationships:[{concept0Id, concept1Id},{concept0Id, concept1Id}] , score}]

@RequestMapping(value = "/scoring-algorithms", method = RequestMethod.GET)

Returns list of "relationshipWeightAlgorithm" values

@param no parameters required

Info; For most path API calls such as concept to concept, concept to semantics and path search the query requires a relation weight algorithm.

Example relation weight algorithms; "lws", "jis", "pws", "plws".

@return "relationshipWeightAlgorithm" values

@RequestMapping(value = "/direct-connections-count", method = RequestMethod.POST)

Returns list of category, predicate, taxonomy count to directly connected concepts.

@param ids (M) list with string based concept id's.

@param additionalFields (M) see direct-connections-countadditional-fields.

Additional fields; "totalCount", "categories", "taxonomies", "predicates".

Info; Returns list of category, predicate andor taxonomy count to direct neighbors, additionalFields (list) can be included to enrich the response.

Example query; { "additionalFields": ["categories"], "ids": ["4047995"] }.

Example query; { "additionalFields": ["categories", "predicates"], "ids": ["4047995"] }.

@return category list (name, count), predicate list(name, count), taxonomy list(name, count)

@RequestMapping(value = "/direct-connections-count/additional-fields", method = RequestMethod.GET)

Returns list of available "additionalField" names which can be used within the direct connection count API.

@param no parameters required

Info; provides available "additionalField" names for direct connection count API calls.

Example "additionalFields"; "totalCount", "categories", "taxonomies", "predicates".

@return field names

@RequestMapping(value = "/direct-connections-count/semantic-types", method = RequestMethod.POST)

Returns list of semantic type count to direct neighbors within given semantic category list.

@param ids (M) list with string based concept id's

@param semanticCategories (M) list with string based semantic categories, see externalsemantic-categories

@param additionalFields (M) see direct-connections-countadditional-fields

Example "additionalFields"; "totalCount", "categories", "taxonomies", "predicates".

Info; Returns list of semantic type count to direct neighbors within given semantic category list, "additionalFields" (list) can be included to enrich the response.

@return semantic type list (name,id, count)

@RequestMapping(value = "/path-search/additional-fields", method = RequestMethod.GET)

Returns list of path-search "additionalField" names.

@param no parameters required

Example "additionalFields"; "description", "semanticCategory", "semanticTypes", "taxonomies", "measures", "accessMappings", "hasTriples", "synonyms", "source", "knowledgebase", "concepts", "publicationsCount", "publicationIds".

@return path-search "additionalField"

@RequestMapping(value = "/path-search/filters", method = RequestMethod.GET)

Returns list of path "negativeFilters", "positiveFilters" or "rightInputs" semantic field prefixes

@param no parameters required

Info; For "negativeFilters" or "positiveFilters" each prefix is part of a prefix, value pair. To make a filter for a concept id 12345, the filter construct would look like 'concept:12345'. For semantic type the filter construct would look like 'st:T116'.

The "rightInputs" semantic field(s) are based on a prefix,value pair. To make a semantic field for a concept id 12345, the semantic field construct would look like 'concept:12345'. For semantic type the semantic field construct would look like 'st:T116'. When providing multiple semantic fields keep in mind that the combination is based on intersect.

Filter prefixes; "concept:", "tax:", "st:", "sc:", "supernodes", "predicate:".

prefix concept: is associated with a node in the network

prefix tax: is associated with a taxonomy value (externaltaxonomies)

prefix st: is associated with a semantic type value (externalsemantic-types)

prefix sc: is associated with a semantic category value (externalsemantic-categories)

prefix predicate: is associated with a predicate value (externalpredicates)

One exception is the key supernodes which is a fixed key without further parameters.

@return path filter prefixes

@RequestMapping(value = "/direct-connections-with-scores", method = RequestMethod.POST)

Returns list of direct neighbors with scores based on filterGroups.

@param ids (M) list of concept ids.

@param additionalFields (O) see direct-connections-countadditional-fields.

@param filterGroups (M) filter groups to define the search criteria.

@param groupExternalCombinationType (O), currently fixed to INTERSECT.

@param groupInternalCombinationType (O), currently fixed to UNION.

@param relationshipWeightAlgorithm (M) string based parameter from externalscoring-algorithms

Info; Returns concept ids (list) to direct neighbors using filterGroups.

"filterGroups" represents a list of "filters". The amount of "filters" is not restricted. Each filter contains one or multiple prefix,value pairs.

A single filter is based on a prefix,value pair. To make a filter for a concept id 12345, the filter construct would look like "concept:12345". For semantic type the filter construct would look like "st:T116". When providing multiple filter prefix,value pairs within a "filters" construct, keep in mind that the inner logic is based on union.

Example of a single "filters" with single filter;
{"filterGroups":[{"filters":["sc:Disorders"]}]}

Example of a single "filters" with multiple filters. Within the "filters" construct this condition will match "sc:Disorders" OR "st:T116";
{"filterGroups":[{"filters":["sc:Disorders", "st:T116"]}]}

Example of a multiple "filters", each with single filter. Within the "filterGroup" construct each "filters" construct logic is based on intersect. This means that this condition will match "sc:Disorders" AND "st:T116";
{"filterGroups":[{"filters":["sc:Disorders"]},{"filters":["st:T116"]}]}

Example of a multiple "filters", one with multiple filter. Within the "filterGroup" construct each "filters" construct logic is based on intersect. This means that this condition will match "sc:Disorders" AND ("st:T116" OR "st:028");
{"filterGroups":[{"filters":["sc:Disorders"]},{"filters":["st:T116", "st:028"]}]}

Relationship weight algorithm must be included to rank the paths based on externalscoring-algorithms. Sorting is mandatory and could be ASC or DESC.

Example query; { "additionalFields": [""], "filterGroups": [{ "filters": ["sc:Disorders"] }], "ids": ["4047995"], "relationshipWeightAlgorithm": "pws" }

Example query;{ "additionalFields": ["categories"], "filterGroups": [{ "filters": ["sc:Disorders"] }], "groupExternalCombinationType": "INTERSECTION", "groupInternalCombinationType": "INTERSECTION", "ids": ["4047995"], "relationshipWeightAlgorithm": "pws" }

Example "additionalFields"; "totalCount", "categories", "taxonomies", "predicates".

Example "relationshipWeightAlgorithm"; "lws", "jis", "pws", "plws".

Mandatory dependency request: externalscoring-algorithms, externalpath-searchfilter,externalsemantic-categories, externalsemantic-types, externaltaxonomies, externalpredicates.

Optional dependencies response: direct-connections-countadditional-fields,

@return {content: [{ neighbour: { id,name}, score:, sourceNodes: [{ id,name,score}] }}

@RequestMapping(method = RequestMethod.POST, value = "/path-search/paths-info")

Finds the publications containing all the concepts from a received path

@param paths list of paths. Path is an array of ids of concepts in that path

Info; After building a "path" step by step and storing them as a List [node1, node2, node3, node4] representing a path between node1,2,3 and 4 this API call can be used to collect the associated publication ids. When multiple paths are available the query needs to be repeated for each list.

@return for each path returns publications count and publication ids

---

@RequestMapping(value = "/predicates", method = {RequestMethod.GET, RequestMethod.POST})

Returns list of predicate values

@param no parameters required

Info; the predicate value can be used for the filters as available within the concept to concept, concept to semantics and direct connections with scores api calls. The api call path-searchfilters provides the options as available for those calls and predicate is one of them. Please note that a prefix is required to make the filter work. In this case the prefix will be "predicate:".

@return predicate values; predicate id, name, definition

@RequestMapping("/external/publications")

@RequestMapping(value = "", method = RequestMethod.POST)

Returns publication details for the given id list, additionalFields (list) can be included to enrich the response.

@param ids (M) list with string based publication id's.

@param additionalFields (O), see externalpublicationsadditional-fields API call.

Info; when working with triples andor paths often the publication id can be included in the response. Based on score you can decide which publications are required to make the result complete.

Example additional fields; "url", "abstract", "sourceId", "sourceName", "authors", "keywords", "geneSymList", "chemSubList", "meshHeadList", "publicationDateAsEpochMillisecondsUTC", "publicationDateHumanReadableUTC", "lastUpdatedDateAsEpochMillisecondsUTC", "LastUpdatedDateHumanReadableUTC", "measures", "accessMappings"

Example; { "ids": ["678126"] } .

Example; { "additionalFields": [""], "ids": ["678126","12345"] } .

Example; { "additionalFields": ["url","keywords"], "ids": ["678126","12345"] } .

Optional dependency response; externalpublicationsadditional-fields API call.

@return default publication details; publication id, document id, title

---

@RequestMapping(value = "/{id}", method = {RequestMethod.POST, RequestMethod.GET})

Returns publication details for the given id, additionalFields (list) can be included to enrich the response.

@param id (M) string based publication id.

@param additionalFields (O), see externalpublicationsadditional-fields API call.

Example additional fields; "url", "abstract", "sourceId", "sourceName", "authors", "keywords", "geneSymList", "chemSubList", "meshHeadList", "publicationDateAsEpochMillisecondsUTC", "publicationDateHumanReadableUTC", "lastUpdatedDateAsEpochMillisecondsUTC", "LastUpdatedDateHumanReadableUTC", "measures", "accessMappings"

Info; when working with triples andor paths often the publication id can be included in the response. Based on score you can decide which publications are required to make the result complete.

Example; { "id": "678126" } .

Example; { "additionalFields": [""], "id": "678126" } .

Example; { "additionalFields": ["url","keywords"], "id": "678126" } .

Optional dependency response; externalpublicationsadditional-fields API call.

@return default publication details; publication id, document id, title

---

@RequestMapping(value = "/additional-fields", method = RequestMethod.GET)

Returns list of available additional field names which can be used within the publication calls.

@param no parameters required

Info; For most publication related commands additionalFields can be specified to enrich the response. Using this call (externalpublicationsadditional-fields) allows you to understand which additionalFields can be used.

Example additional fields; "url", "abstract", "sourceId", "sourceName", "authors", "keywords", "geneSymList", "chemSubList", "meshHeadList", "publicationDateAsEpochMillisecondsUTC", "publicationDateHumanReadableUTC", "lastUpdatedDateAsEpochMillisecondsUTC", "LastUpdatedDateHumanReadableUTC", "measures",
"accessMappings"

@return publication additional field names

@RequestMapping("/external/triples")

@RequestMapping(value = "", method = RequestMethod.POST)

Returns triple details for the given id list, additionalFields (list) can be included to enrich the response.

@param ids (M) list with string based triple id's

@param additionalFields (O) , see externaltriplesadditional-fields API call.

Info; For most triple related commands additionalFields can be specified to enrich the response. Using this call (externaltriplesadditional-fields) allows you to understand which additionalFields can be used.

Example additional fields; "subjectName", "objectName", "predicateName", "subjectSemanticCategory", "objectSemanticCategory", "accessMappings", "measures"

Example; { "ids": ["678126"] } .

Example; { "additionalFields": [""], "ids": ["678126", "1234"] } .

Example; { "additionalFields": ["subjectName","objectName"], "ids": ["678126","1234"] } .

Optional dependency response; externaltriplesadditional-fields API call.

@return default triple details; subject id, predicate id, object id

---

@RequestMapping(value = "/{id}", method = {RequestMethod.POST, RequestMethod.GET})

Returns triple details for the given id, additionalFields (list) can be included to enrich the response

@param id (M) string based triple id

@param additionalFields (O) , see externaltriplesadditional-fields

Info; For most triple related commands additionalFields can be specified to enrich the response. Using this call (externaltriplesadditional-fields) allows you to understand which additionalFields can be used.

Example additional fields; "subjectName", "objectName", "predicateName", "subjectSemanticCategory", "objectSemanticCategory", "accessMappings", "measures"

Example; { "id": "678126" } .

Example; { "additionalFields": [""], "id": "678126" } .

Example; { "additionalFields": ["subjectName","objectName"], "id": "678126" } .

Optional dependency response; externaltriplesadditional-fields API call.

@return default triple details; subject id, predicate id, object id

---

@RequestMapping(value = "/additional-fields", method = RequestMethod.GET)

Returns list of available additional field names which can be used within the triple calls.

Info; For most triple related commands additionalFields can be specified to enrich the response. Using this call (externaltriplesadditional-fields) allows you to understand which additionalFields can be used.

Example fields; "subjectName", "objectName", "predicateName", "subjectSemanticCategory", "objectSemanticCategory", "accessMappings", "measures".

@return triple additional field names
