# INDRA Network Service

**K. Karis**

**Oct 05, 2021**

# CONTENTS:

This documentation covers a tutorial and the modules of the INDRA Network Search and is part of a broader set of INDRA derived applications. To read more about INDRA, see the homepage, the documentation and the project on github.

CHAPTER

ONE

# NETWORK SEARCH WEB UI

This document introduces the web interface of the INDRA Network Search Service

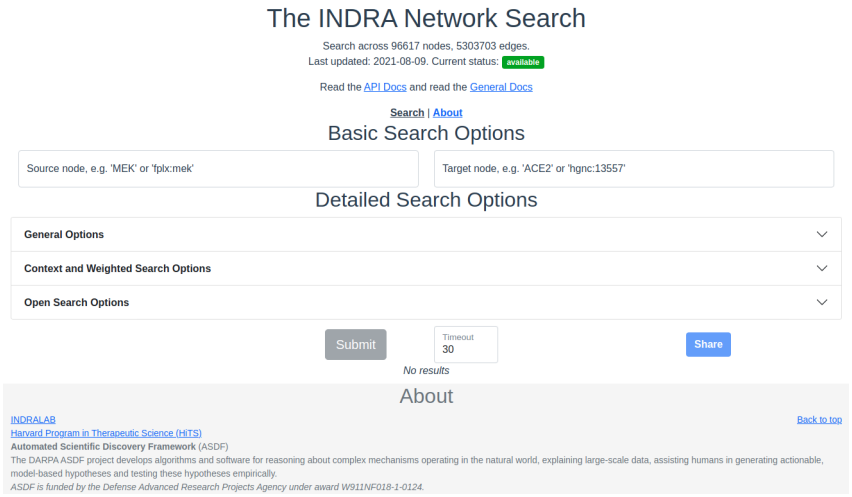


Fig. 1: *The network search interface with no input or results.*

## 1.1 Source and Target

The source and target are the nodes between which to find a path and at least one of source and target is needed to do a search. If only one of source or target is provided, an open ended breadth first search is done instead of a path search. Note that the source and target are not affected by the choice of *allowed namespaces* (see below at *Allowed Node Namespaces*).

### 1.1.1 Autocompleting source/target inputs

Autocompletion of source/target based on prefix and entity identifier are made automatically as an input is typed or pasted into the text boxes. The suggestions are picked from the nodes in the graph and the text box will mark the entered text as correct if it matches an existing node in the graph.

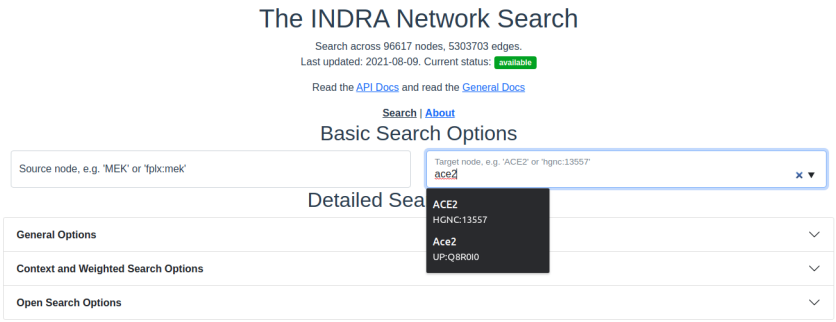


Fig. 2: *Autocompleting an entity. As an entity name is typed into the source or target text boxes, suggestions from the graph nodes are provided.*

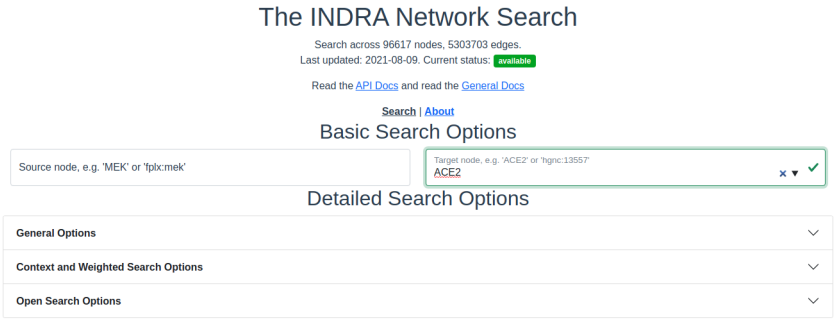


Fig. 3: *Verified entry. Entries are verified as they are being typed or pasted into the text box. When the entered entity is verified to exist in the graph, the text box border will switch to green and a checkmark will appear.*

## 1.2 Detailed Search Options - General Options

The general detailed search options contain filters that apply to most searches, regardless of weighting or openness.

### 1.2.1 Path Length

Only paths of this many edges will be returned. Must be a positive integer.

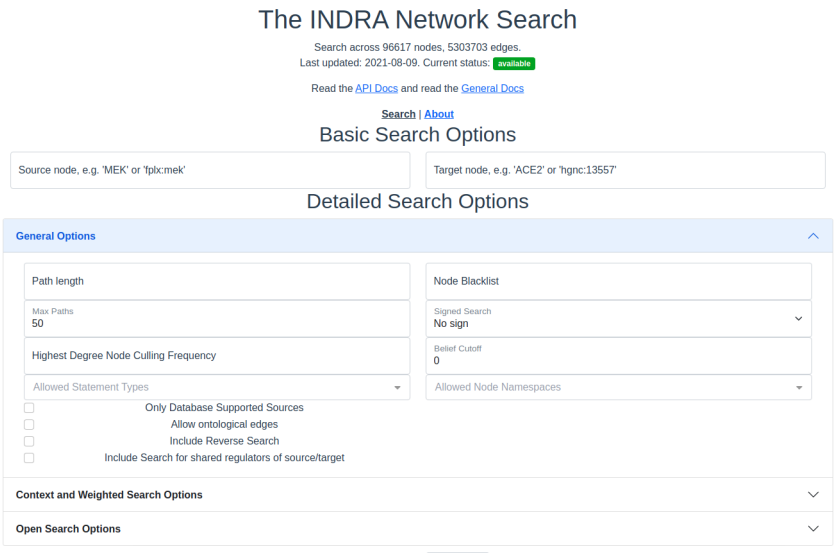


Fig. 4: *The search interface with the general options section expanded.*

### 1.2.2 Node Blacklist

Node names entered here are skipped in the path search. This is a good way to avoid nodes of extremely high degree that overwhelms the results and effectively blocks out results that include nodes of lower degree. *See also Highest Degree Node Culling Frequency below.*

### 1.2.3 Max Paths

The maximum number of results to return per category in the results. The default and the maximum allowed is 50 results. For unweighted searches this number rarely makes a perceivable difference in response time but for weighted searches keep this number low for a faster response time.

### 1.2.4 Signed Search

To perform a signed search, click on the drop down menu that says "No sign" and chose a sign. "+" means that the returned paths are upregulations, and "-" means that the returned paths are downregulations. For the purpose of signed search, only statements that imply a clear up- or downregulation are considered. Currently this mean *IncreaseAmount* and *Activation* for upregulation, and *DecreaseAmount* and *Inhibition* for downregulation.

### 1.2.5 Highest Degree Node Culling Frequency

Entering a positive integer here allows the path search to include the highest degree node for the first N returned paths, after which that node is added to the *Node Blacklist*. This is repeated for the second highest degree node for the following N paths, then for the third highest degree node and so forth. *Limitations:* This option is only applied to unweighted open search and source-target searches.

### 1.2.6 Belief Cutoff

Any statement with a belief score below this number will be excluded from the edge support. If all statements are excluded from the edge, all paths containing that edge become invalid and are skipped. It is set to zero by default to include all edges. Read more about belief scores in the belief module of INDRA.

### 1.2.7 Allowed Statement Types

This is a multiselect dropdown which contains multiple statement types to allow in the results. If an edge of a path does not contain any of the selected statement types, the whole path will be skipped from the result. Read more about statement types in the statements module of INDRA.

### 1.2.8 Allowed Node Namespaces

The namespaces included here are the ones that are allowed on any node visited in the path search. The namespace of the source and target are excluded from this restriction. A namespace in INDRA is the prefix or name of the *type* of identifier used to uniquely identify an entity from a specific knowledge source. For example, a chemical can be identified using a *CHEBI* identifier and would then be identified in the *CHEBI* namespace.

### 1.2.9 Checkboxes

The following options are available as checkboxes:

- **Only Database Supported Sources**: Check this box to enforce that all edges must be supported by at least one statement sourced from curated databases like PathwayCommons and Signor
- **Allow Ontological Edges**: Check this box to allow directed edges that go from an entity to its ontological parent, e.g. from the NFKB1 sub-unit to the NFkappaB complex.
- **Include Reverse Search**: Check this box to also search for paths with source and target swapped. With this option, the reverse search *from* target *to* source is done as well as the original search from source to target. If the *timeout* is reached (see below) before the reverse search can start, the reverse search will not return any paths. If the *timeout* is reached during the reverse search, fewer paths than for the original search will be returned.
- **Include Search for Shared Regulators of Source/Target**: Check this box to include a search for common upstream nodes one edge away from both source and target. This option is only available when both source and target specified.

## 1.3 Detailed Search Options - Context and Weighted Search Options

This section of the search options allows control over how to prioritize or *weight* edges in paths differently. During weighted search, the cost along every path encountered is calculated as the sum of the edge weights along the path. The paths are returned in ascending order of cost.

The different ways of weighting the search are available in the dropdown menu "Weighted Search". *Note:* A weighted search is costly and usually takes longer than an unweighted search. It is common that a very heavy weighted search times out, especially for a *signed weighted* search, even with the highest allowed *timeout* of 120 seconds.

The weighted search uses a slightly modified version of the Djikstra weighted search employed in Networkx.

The code implemented for the weighted search is available on github in the functions *shortest_simple_paths()* and *open_dijkstra_search()* for closed and open paths, respectively.

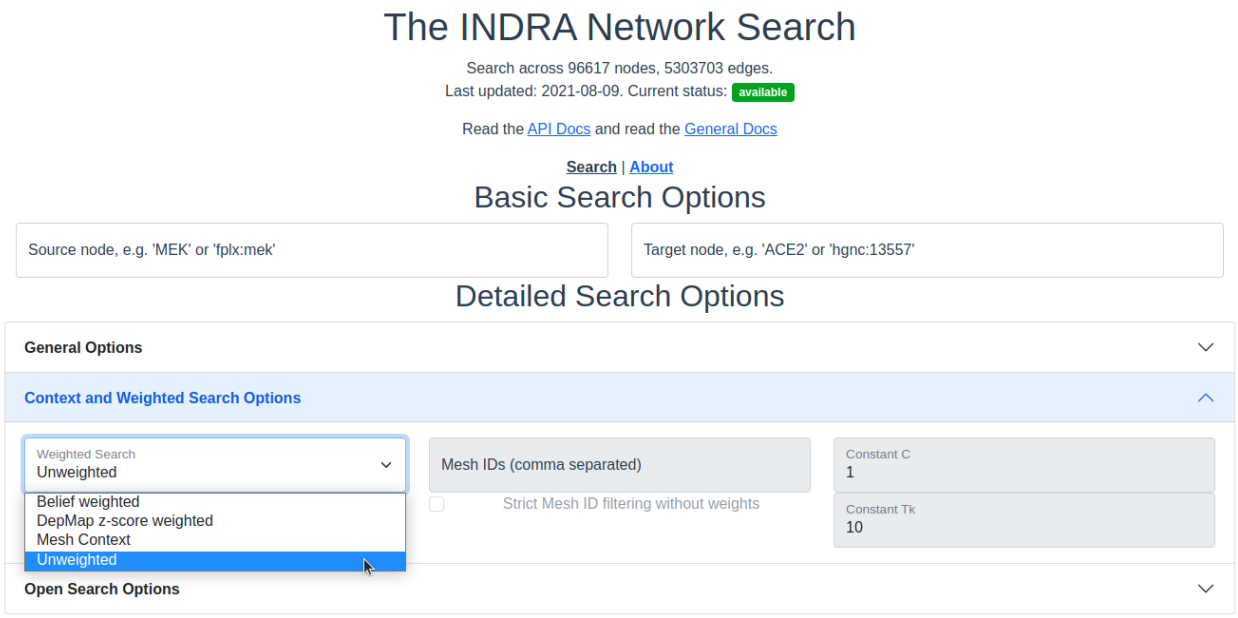


Fig. 5: *The search interface with the Context and Weighted search options section expanded.*

### 1.3.1 Unweighted

This is the default option and imposes no weight on the edges and is equivalent to all edges having a unit weight.

### 1.3.2 Belief Weighted

The belief weight of an edge is calculated as the negative log of the aggregated belief scores of all the statements supporting edge $e$:

$$w_e = -\log\left(1 - \prod_i (1 - b_i)\right)$$

where $b_i$ is the belief score of supporting statement $i$ of edge $e$. Since the belief score is limited to the interval $[0, 1]$, it can interpreted as a probability and the above weight can therefore be seen as the log of the *complement* to the probability that every supporting statement is *incorrect*.

### 1.3.3 DepMap z-score weighted

The z-score edge weight is focused around prioritizing edges between human genes that have been targeted in knockout screens performed at the Broad Institute's Dependency Map project. The z-score is obtained from first calculating the pearson correlation between all pairs of genes in the gene knockout screen. Then the log of the p-values of the correlations are calculated using the CDF of the beta distribution. Finally the strength of the z-scores are obtained from the p-values and the signs are recuperated from the original correlation matrix.

The edge weight, assuming both nodes are human genes, is calculated by normalizing the difference between the z-score associated with a self-correlation and the strength of the z-score between the two nodes of the edge. In the case that one or both of the nodes of the edge are non-gene entities, the z-score weight is set to 1:

$$w_e = \begin{cases} 1 & \text{if } z_e = z_0 \\ \frac{z_0 - |z_e|}{z_0} & \text{if } z_e \neq z_0 \end{cases}$$

where $z_0$ is the z-score associated with self correlation and $z_e$ is the z-score of the edge.

### 1.3.4 Mesh Context

The context based search allows a search to prioritize or only include connections that are relevant to the provided context. The context is given as MeSH terms.

- **MeSH IDs:** Enter the MeSH IDs, separated by comma, that should be used in the search.
- **Strict Filtering on MeSH IDs:** Check this box to *only* allow edges with associated with the provided MeSH IDs. If left unchecked, the search is weighted.
- **Constants $C$ and $T_k$:** These two constant allow for changing the importance of the context in a weighted context based search. For any edge $e$, the weight $w_e$ of the edge in the context based search is calculated in the following way:

$$w_e = -C \cdot \log\left(\frac{\text{refcount}}{\text{total} + T_k}\right)$$

Here, *refcount* is the number of references with the associated MeSH ID(s) that are supporting edge $e$ and *total* is the total number of references supporting edge $e$.

## 1.4 Detailed Search Options - Open Search Options

Options under the Open Search Options are only applied during open searches, i.e. when either of source or target is provided.

### 1.4.1 Terminal Namespaces

Namespaces selected here restrict the search to only return paths that *end* (open search from source) or *start* (open search from target) on the given namespaces and then not consider these nodes further. For example: if namespace A is selected, then a downstream path might look like this: X->Y->A, but not like this: X->Y->A->Z, where X, Y, Z are all namespaces other than A.

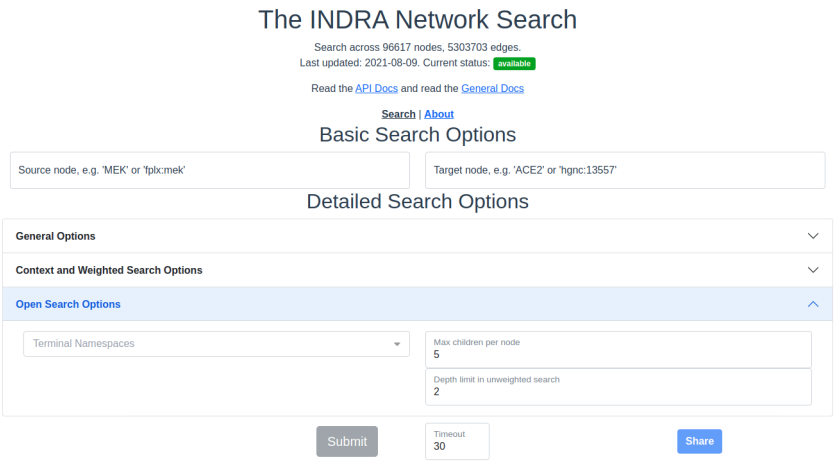


Fig. 6: *The search interface with the Open search options section expanded.*

### 1.4.2 Max Children Per Node (Unweighted Search)

The integer provided here gives a maximum for the number of children to continue to open search from. For example: if N is set here, the first N nodes selected from the starting node are then considered for the next layer in the breadth first search. This option is only available for *unweighted* searches.

### 1.4.3 Depth Limit (Unweighted Search)

This option limits how deep, i.e. how many edges, the returned paths are allowed to be/have. This option is only available for *unweighted* searches.

## 1.5 Timeout

Setting a timeout allows to set a larger (or smaller) timeout than the default 30 seconds timeout. The time since the path search was started is checked before each iteration of data assembly for a returned path during the search. If the time passed is larger than the allowed timeout, the search is stopped. The timeout provided has to be a decimal number smaller than or equal to 120 seconds.

## 1.6 Result Categories

*Note*: If there are no results for a specific category, that section will be hidden.

### 1.6.1 Common Parents

This section shows the result of a search for common ontological parents of source and target. For example with *GP1BA* and *GP1BB* as source and target, respectively, the Platelet membrane glycoprotein complex shows up as a shared ontological parent.

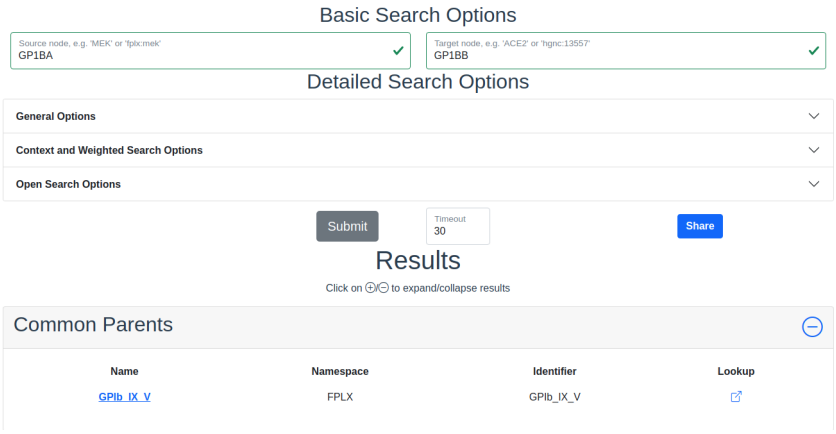


Fig. 7: *The result of a search with GP1BA and GP1BB as source and target, respectively, for Common Parents showing the Platelet membrane glycoprotein complex as their shared protein complex.*

### 1.6.2 Shared Targets

This section shows the direct downstream targets that are shared between *source* and *target*.

### 1.6.3 Shared Regulators

Shared regulators are only searched for if the corresponding checkbox is checked (see *checkboxes* above). The results shown are the direct upstream regulators that are shared between *source* and *target*.

### 1.6.4 Path Results

This section show path results per path length, i.e. all paths with the same number of edges share a specific subsection. The division of paths per subsection is done regardless if the path search is weighted or not.

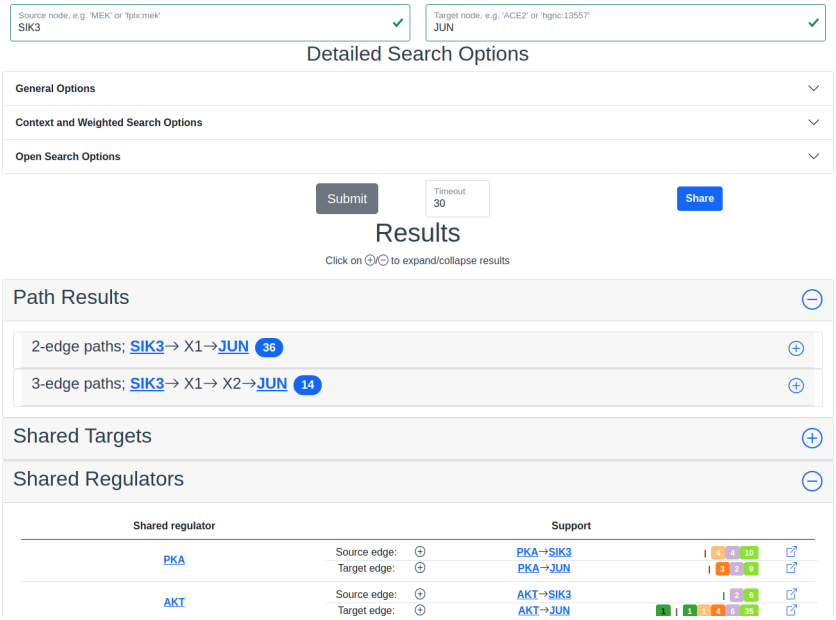


Fig. 8: *Search results with SIK3 as source and JUN as target with the Shared Regulator section expanded.*

## 1.7 Detailed Results

For each result section, excluding Common Parents, there are two levels of detail. Results for Common Parents only have one level of results: name, namespace, identifier and entity lookup. The first level shows path (for Path Results), target (for Shared Targets) or regulator (for Shared Regulators) together with weight (if the search is weighted) the edge, source counts and a link to the INDRA DB for that specific edge.

The second level of results is collapsed by default. To expand it, the circled "+" (⊕) need to be clicked. Once expanded, source counts and a link to more specific information in the INDRA DB per statement type are shown.

As the network search results can be filtered in more detail than what is possible using the INDRA DB, the statements shown in the DB can sometimes be a superset of the statements shown in the second level of the results.

## 1.8 The Graphs Used

The two graphs used for the network search are assembled from a full snapshot of the INDRA DataBase that is updated regularly. Any statement that includes two or three agents are assembled into the support for the edges for the graphs, with one edge containing one or more statements. The two types of graphs used are:

1. Unsigned directed graph
2. Signed node directed graph

The edges in the signed graph only contain statements that have clear up- or downreguations associated with them, which currently are *IncreaseAmount* and *Activation* for upregulation, and *DecreaseAmount* and *Inhibition* for down-regulation.

The code assembling the graphs can be found in net_functions.py in the function *sif_dump_df_to_digraph()* in the *depmap_analysis* repository.

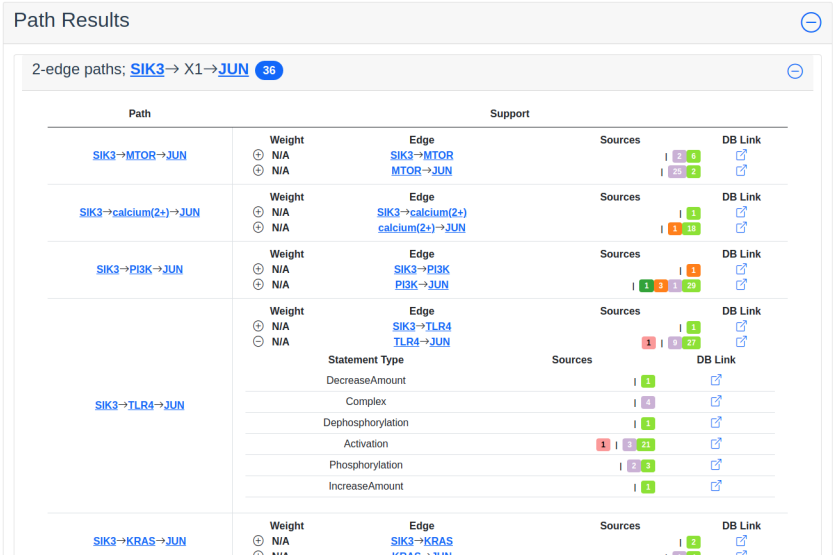


Fig. 9: *Path search results with one of the edges of a path expanded to more detail.*

CHAPTER

TWO

# INDRA NETWORK SEARCH MODULES REFERENCE

## 2.1 Autocomplete (indra_network_search.autocomplete.autocomplete)

An API wrapping SortedStringTrie from pytrie (see https://github.com/gsakkis/pytrie)

**class** indra_network_search.autocomplete.autocomplete.**NodesTrie**(**args*, ***kwargs*)

A Trie structure that has case insensitive search methods

**case_items**(*prefix=None*, *top_n=100*)

Case insensitive wrapper around NodeTrie.items()

**Parameters** **prefix** (Optional[str]) – The prefix to search

**Return type** List[Tuple[str, str, str]]

**Returns** Return a list of (name, namespace, id) tuples

**case_keys**(*prefix=None*, *top_n=100*)

Case insensitive wrapper around NodeTrie.keys()

**Parameters**

- **prefix** (Optional[str]) – The prefix to search
- **top_n** (Optional[int]) – The top ranked entities (by node degree)

**Returns** Return a list of this trie's keys

**Return type** List[str]

**classmethod from_node_names**(*graph*)

Produce a NodesTrie instance from a graph with node names as keys

**Parameters** **graph** (Union[DiGraph, MultiDiGraph]) – Graph from which nodes should be searchable. It is assumed the nodes are all keyed by strings

**Return type** *NodesTrie*

**Returns** An instance of a NodesTrie containing the node names of the graph as keys and the corresponding (name, ns, id, node degree) tuple as values

**classmethod from_node_ns_id**(*graph*)

Produce a NodesTrie instance from a graph using ns:id as key

**Parameters** **graph** (Union[DiGraph, MultiDiGraph]) – Graph from which nodes should be searchable. It is assumed the nodes have the attributes 'ns' and 'id' accessible via g.nodes[node]['ns'] and g.nodes[node]['id']

**Return type** *NodesTrie*

**Returns** An instance of a NodesTrie containing ns:id of each node of the graph as keys and the corresponding (name, ns, id, node degree) tuple as values

## 2.2 Data Models (indra_network_search.data_models)

This module contains all data models used in the repository. They are all built around the the Pydantic BaseModel.

This file contains data models for queries, results and arguments to algorithm functions.

**pydantic model** indra_network_search.data_models.__init__.**ApiOptions**

Options that determine API behaviour


```
{
   "title": "ApiOptions",
   "description": "Options that determine API behaviour",
   "type": "object",
   "properties": {
      "sign": {
         "title": "Sign",
         "type": "integer"
      },
      "fplx_expand": {
         "title": "Fplx Expand",
         "default": false,
         "type": "boolean"
      },
      "user_timeout": {
         "title": "User Timeout",
         "default": false,
         "anyOf": [
            {
               "type": "number"
            },
            {
               "type": "boolean"
            }
         ]
      },
      "two_way": {
         "title": "Two Way",
         "default": false,
         "type": "boolean"
      },
      "shared_regulators": {
         "title": "Shared Regulators",
         "default": false,
         "type": "boolean"
      },
      "format": {
         "title": "Format",
         "default": "json",
         "type": "string"
      }
```


(continues on next page)

(continued from previous page)

```
    }
}
```

**Fields**

- *sign (Optional[int])*
- *fplx_expand (Optional[bool])*
- *user_timeout (Optional[Union[float, bool]])*
- *two_way (Optional[bool])*
- *shared_regulators (Optional[bool])*
- *format (Optional[str])*

**field format: Optional[str] = 'json'**

**field fplx_expand: Optional[bool] = False**

**field shared_regulators: Optional[bool] = False**

**field sign: Optional[int] = None**

**field two_way: Optional[bool] = False**

**field user_timeout: Optional[Union[float, bool]] = False**

**pydantic model** indra_network_search.data_models.__init__.**BreadthFirstSearchOptions**

Arguments for indra.explanation.pathfinding.bfs_search


```
{
   "title": "BreadthFirstSearchOptions",
   "description": "Arguments for indra.explanation.pathfinding.bfs_search",
   "type": "object",
   "properties": {
      "source_node": {
         "title": "Source Node",
         "anyOf": [
            {
               "type": "string"
            },
            {
               "type": "array",
               "items": [
                  {
                     "type": "string"
                  },
                  {
                     "type": "integer"
                  }
               ]
            }
         ]
      },
      "reverse": {
         "title": "Reverse",
```


(continues on next page)

(continued from previous page)


```
            "default": false,
            "type": "boolean"
        },
        "depth_limit": {
            "title": "Depth Limit",
            "default": 2,
            "type": "integer"
        },
        "path_limit": {
            "title": "Path Limit",
            "type": "integer"
        },
        "max_per_node": {
            "title": "Max Per Node",
            "default": 5,
            "type": "integer"
        },
        "node_filter": {
            "title": "Node Filter",
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "node_blacklist": {
            "title": "Node Blacklist",
            "type": "array",
            "items": {
                "type": "string"
            },
            "uniqueItems": true
        },
        "terminal_ns": {
            "title": "Terminal Ns",
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "sign": {
            "title": "Sign",
            "type": "integer"
        },
        "max_memory": {
            "title": "Max Memory",
            "default": 536870912,
            "type": "integer"
        },
        "hashes": {
            "title": "Hashes",
            "type": "array",
            "items": {
```


(continues on next page)

(continued from previous page)


```
            "type": "integer"
         }
      },
      "strict_mesh_id_filtering": {
         "title": "Strict Mesh Id Filtering",
         "default": false,
         "type": "boolean"
      }
   },
   "required": [
      "source_node"
   ]
}
```


**Fields**

- *source_node (Union[str, Tuple[str, int]])*
- *reverse (Optional[bool])*
- *depth_limit (Optional[int])*
- *path_limit (Optional[int])*
- *max_per_node (Optional[int])*
- *node_filter (Optional[List[str]])*
- *node_blacklist (Optional[Set[str]])*
- *terminal_ns (Optional[List[str]])*
- *sign (Optional[int])*
- *max_memory (Optional[int])*
- *hashes (Optional[List[int]])*
- *allow_edge (Optional[Callable[[networkx.classes.digraph.DiGraph, Union[str, Tuple[str, int]], Union[str, Tuple[str, int]]], bool]])*
- *edge_filter (Optional[Callable[[networkx.classes.digraph.DiGraph, Union[str, Tuple[str, int]], Union[str, Tuple[str, int]]], bool]])*
- *strict_mesh_id_filtering (Optional[bool])*

**field allow_edge: Optional[Callable[[networkx.classes.digraph.DiGraph, Union[str, Tuple[str, int]], Union[str, Tuple[str, int]]], bool]] = None**

**field depth_limit: Optional[int] = 2**

**field edge_filter: Optional[Callable[[networkx.classes.digraph.DiGraph, Union[str, Tuple[str, int]], Union[str, Tuple[str, int]]], bool]] = None**

**field hashes: Optional[List[int]] = None**

**field max_memory: Optional[int] = 536870912**

**field max_per_node: Optional[int] = 5**

**field node_blacklist: Optional[Set[str]] = None**

**field node_filter: Optional[List[str]] = None**

**field path_limit: Optional[int] = None**

**field reverse: Optional[bool] = False**

**field sign: Optional[int] = None**

**field source_node: Union[str, Tuple[str, int]] [Required]**

**field strict_mesh_id_filtering: Optional[bool] = False**

**field terminal_ns: Optional[List[str]] = None**

**pydantic model** indra_network_search.data_models.__init__.**DijkstraOptions**

Arguments for open_dijkstra_search


```
{
   "title": "DijkstraOptions",
   "description": "Arguments for open_dijkstra_search",
   "type": "object",
   "properties": {
      "start": {
         "title": "Start",
         "anyOf": [
            {
               "type": "string"
            },
            {
               "type": "array",
               "items": [
                  {
                     "type": "string"
                  },
                  {
                     "type": "integer"
                  }
               ]
            }
         ]
      },
      "reverse": {
         "title": "Reverse",
         "default": false,
         "type": "boolean"
      },
      "path_limit": {
         "title": "Path Limit",
         "type": "integer"
      },
      "hashes": {
         "title": "Hashes",
         "type": "array",
         "items": {
            "type": "integer"
         }
      },
```


(continues on next page)

(continued from previous page)


```
      "ignore_nodes": {
         "title": "Ignore Nodes",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "ignore_edges": {
         "title": "Ignore Edges",
         "type": "array",
         "items": {
            "type": "array",
            "items": [
               {
                  "type": "string"
               },
               {
                  "type": "string"
               }
            ]
         }
      },
      "terminal_ns": {
         "title": "Terminal Ns",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "weight": {
         "title": "Weight",
         "type": "string"
      },
      "const_c": {
         "title": "Const C",
         "default": 1,
         "type": "integer"
      },
      "const_tk": {
         "title": "Const Tk",
         "default": 10,
         "type": "integer"
      }
   },
   "required": [
      "start"
   ]
}
```


**Fields**

- *start (Union[str, Tuple[str, int]])*

- *reverse (Optional[bool])*
- *path_limit (Optional[int])*
- *hashes (Optional[List[int]])*
- *ignore_nodes (Optional[List[str]])*
- *ignore_edges (Optional[List[Tuple[str, str]]])*
- *terminal_ns (Optional[List[str]])*
- *weight (Optional[str])*
- *ref_counts_function (Optional[Callable])*
- *const_c (Optional[int])*
- *const_tk (Optional[int])*

**field const_c: Optional[int] = 1**

**field const_tk: Optional[int] = 10**

**field hashes: Optional[List[int]] = None**

**field ignore_edges: Optional[List[Tuple[str, str]]] = None**

**field ignore_nodes: Optional[List[str]] = None**

**field path_limit: Optional[int] = None**

**field ref_counts_function: Optional[Callable] = None**

**field reverse: Optional[bool] = False**

**field start: Union[str, Tuple[str, int]] [Required]**

**field terminal_ns: Optional[List[str]] = None**

**field weight: Optional[str] = None**

**pydantic model** indra_network_search.data_models.__init__.**EdgeData**

Data for one single edge


```
{
   "title": "EdgeData",
   "description": "Data for one single edge",
   "type": "object",
   "properties": {
      "edge": {
         "title": "Edge",
         "type": "array",
         "items": {
            "$ref": "#/definitions/Node"
         }
      },
      "statements": {
         "title": "Statements",
         "type": "object",
         "additionalProperties": {
            "$ref": "#/definitions/StmtTypeSupport"
         }
```


(continues on next page)

(continued from previous page)


```
    },
    "belief": {
       "title": "Belief",
       "minimum": 0,
       "maximum": 1,
       "type": "number"
    },
    "weight": {
       "title": "Weight",
       "minimum": 0,
       "type": "number"
    },
    "context_weight": {
       "title": "Context Weight",
       "default": "N/A",
       "anyOf": [
          {
             "type": "string"
          },
          {
             "type": "number",
             "exclusiveMinimum": 0
          },
          {
             "enum": [
                "N/A"
             ],
             "type": "string"
          }
       ]
    },
    "z_score": {
       "title": "Z Score",
       "type": "number"
    },
    "corr_weight": {
       "title": "Corr Weight",
       "exclusiveMinimum": 0.0,
       "type": "number"
    },
    "sign": {
       "title": "Sign",
       "minimum": 0,
       "maximum": 1,
       "type": "integer"
    },
    "db_url_edge": {
       "title": "Db Url Edge",
       "type": "string"
    },
    "source_counts": {
       "title": "Source Counts",
```


(continues on next page)

(continued from previous page)

```
        "default": {},
        "type": "object",
        "additionalProperties": {
           "type": "integer"
        }
     }
  },
  "required": [
     "edge",
     "statements",
     "belief",
     "weight",
     "db_url_edge"
  ],
  "definitions": {
     "Node": {
        "title": "Node",
        "description": "Data for a node",
        "type": "object",
        "properties": {
           "name": {
              "title": "Name",
              "minLength": 1,
              "type": "string"
           },
           "namespace": {
              "title": "Namespace",
              "minLength": 1,
              "type": "string"
           },
           "identifier": {
              "title": "Identifier",
              "minLength": 1,
              "type": "string"
           },
           "lookup": {
              "title": "Lookup",
              "minLength": 1,
              "type": "string"
           },
           "sign": {
              "title": "Sign",
              "minimum": 0,
              "maximum": 1,
              "type": "integer"
           }
        },
        "required": [
           "namespace",
           "identifier"
        ]
     },
```

(continues on next page)

(continued from previous page)


```
    "StmtData": {
      "title": "StmtData",
      "description": "Data for one statement supporting an edge",
      "type": "object",
      "properties": {
        "stmt_type": {
          "title": "Stmt Type",
          "type": "string"
        },
        "evidence_count": {
          "title": "Evidence Count",
          "minimum": 1,
          "type": "integer"
        },
        "stmt_hash": {
          "title": "Stmt Hash",
          "anyOf": [
            {
              "type": "integer"
            },
            {
              "type": "string",
              "minLength": 1,
              "maxLength": 2083,
              "format": "uri"
            }
          ]
        },
        "source_counts": {
          "title": "Source Counts",
          "type": "object",
          "additionalProperties": {
            "type": "integer"
          }
        },
        "belief": {
          "title": "Belief",
          "minimum": 0.0,
          "maximum": 1.0,
          "type": "number"
        },
        "curated": {
          "title": "Curated",
          "type": "boolean"
        },
        "english": {
          "title": "English",
          "type": "string"
        },
        "weight": {
          "title": "Weight",
          "type": "number"
```


(continues on next page)

(continued from previous page)


```
            },
            "residue": {
               "title": "Residue",
               "default": "",
               "type": "string"
            },
            "position": {
               "title": "Position",
               "default": "",
               "type": "string"
            },
            "initial_sign": {
               "title": "Initial Sign",
               "minimum": 0,
               "maximum": 1,
               "type": "integer"
            },
            "db_url_hash": {
               "title": "Db Url Hash",
               "type": "string"
            }
         },
         "required": [
            "stmt_type",
            "evidence_count",
            "stmt_hash",
            "source_counts",
            "belief",
            "curated",
            "english",
            "db_url_hash"
         ]
      },
      "StmtTypeSupport": {
         "title": "StmtTypeSupport",
         "description": "Data per statement type",
         "type": "object",
         "properties": {
            "stmt_type": {
               "title": "Stmt Type",
               "type": "string"
            },
            "source_counts": {
               "title": "Source Counts",
               "default": {},
               "type": "object",
               "additionalProperties": {
                  "type": "integer"
               }
            },
            "statements": {
               "title": "Statements",
```


(continues on next page)

(continued from previous page)

```
                "type": "array",
                "items": {
                   "$ref": "#/definitions/StmtData"
                }
             }
          },
          "required": [
             "stmt_type",
             "statements"
          ]
       }
    }
}
```

**Fields**

- *edge (List[indra_network_search.data_models.__init__.Node])*
- *statements (Dict[str, indra_network_search.data_models.__init__.StmtTypeSupport])*
- *belief (indra_network_search.data_models.__init__.ConstrainedFloatValue)*
- *weight (indra_network_search.data_models.__init__.ConstrainedFloatValue)*
- *context_weight (Union[str, indra_network_search.data_models.__init__.ConstrainedFloatValue, typing_extensions.Literal[N/A]])*
- *z_score (Optional[float])*
- *corr_weight (Optional[indra_network_search.data_models.__init__.ConstrainedFloatValue])*
- *sign (Optional[indra_network_search.data_models.__init__.ConstrainedIntValue])*
- *db_url_edge (str)*
- *source_counts (Dict[str, int])*

**field belief: indra_network_search.data_models.__init__.ConstrainedFloatValue [Required]**

**Constraints**

- **minimum** = 0
- **maximum** = 1

**field context_weight: Union[str, indra_network_search.data_models.__init__.ConstrainedFloatValue, typing_extensions.Literal[N/A]] = 'N/A'**

**field corr_weight: Optional[indra_network_search.data_models.__init__.ConstrainedFloatValue] = None**

**Constraints**

- **exclusiveMinimum** = 0.0

field db_url_edge: str [Required]

field edge: List[*indra_network_search.data_models.__init__.Node*] [Required]

field sign: Optional[indra_network_search.data_models.__init__.ConstrainedIntValue] [Required]

**Constraints**

- **minimum** = 0
- **maximum** = 1

field source_counts: Dict[str, int] = {}

field statements: Dict[str, *indra_network_search.data_models.__init__.StmtTypeSupport*] [Required]

field weight: indra_network_search.data_models.__init__.ConstrainedFloatValue [Required]

**Constraints**

- **minimum** = 0

field z_score: Optional[float] = None

is_empty()
Return True if len(statements) == 0

**Return type** bool

set_source_counts()
Updates the source count from the contained data in self.statements

pydantic model indra_network_search.data_models.__init__.EdgeDataByHash
Data for one single edge, with data keyed by hash


```
{
   "title": "EdgeDataByHash",
   "description": "Data for one single edge, with data keyed by hash",
   "type": "object",
   "properties": {
      "edge": {
         "title": "Edge",
         "type": "array",
         "items": {
            "$ref": "#/definitions/Node"
         }
      },
      "stmts": {
         "title": "Stmts",
         "type": "object",
         "additionalProperties": {
            "$ref": "#/definitions/StmtData"
         }
      },
      "belief": {
         "title": "Belief",
         "type": "number"
      },
```


(continues on next page)

(continued from previous page)


```
      "weight": {
         "title": "Weight",
         "type": "number"
      },
      "db_url_edge": {
         "title": "Db Url Edge",
         "type": "string"
      },
      "url_by_type": {
         "title": "Url By Type",
         "type": "object",
         "additionalProperties": {
            "type": "string"
         }
      }
   },
   "required": [
      "edge",
      "stmts",
      "belief",
      "weight",
      "db_url_edge",
      "url_by_type"
   ],
   "definitions": {
      "Node": {
         "title": "Node",
         "description": "Data for a node",
         "type": "object",
         "properties": {
            "name": {
               "title": "Name",
               "minLength": 1,
               "type": "string"
            },
            "namespace": {
               "title": "Namespace",
               "minLength": 1,
               "type": "string"
            },
            "identifier": {
               "title": "Identifier",
               "minLength": 1,
               "type": "string"
            },
            "lookup": {
               "title": "Lookup",
               "minLength": 1,
               "type": "string"
            },
            "sign": {
               "title": "Sign",
```


(continues on next page)

(continued from previous page)

```
                "minimum": 0,
                "maximum": 1,
                "type": "integer"
            }
        },
        "required": [
            "namespace",
            "identifier"
        ]
    },
    "StmtData": {
        "title": "StmtData",
        "description": "Data for one statement supporting an edge",
        "type": "object",
        "properties": {
            "stmt_type": {
                "title": "Stmt Type",
                "type": "string"
            },
            "evidence_count": {
                "title": "Evidence Count",
                "minimum": 1,
                "type": "integer"
            },
            "stmt_hash": {
                "title": "Stmt Hash",
                "anyOf": [
                    {
                        "type": "integer"
                    },
                    {
                        "type": "string",
                        "minLength": 1,
                        "maxLength": 2083,
                        "format": "uri"
                    }
                ]
            },
            "source_counts": {
                "title": "Source Counts",
                "type": "object",
                "additionalProperties": {
                    "type": "integer"
                }
            },
            "belief": {
                "title": "Belief",
                "minimum": 0.0,
                "maximum": 1.0,
                "type": "number"
            },
            "curated": {
```

(continues on next page)

(continued from previous page)


```
                "title": "Curated",
                "type": "boolean"
            },
            "english": {
                "title": "English",
                "type": "string"
            },
            "weight": {
                "title": "Weight",
                "type": "number"
            },
            "residue": {
                "title": "Residue",
                "default": "",
                "type": "string"
            },
            "position": {
                "title": "Position",
                "default": "",
                "type": "string"
            },
            "initial_sign": {
                "title": "Initial Sign",
                "minimum": 0,
                "maximum": 1,
                "type": "integer"
            },
            "db_url_hash": {
                "title": "Db Url Hash",
                "type": "string"
            }
        },
        "required": [
            "stmt_type",
            "evidence_count",
            "stmt_hash",
            "source_counts",
            "belief",
            "curated",
            "english",
            "db_url_hash"
        ]
    }
  }
}
```


**Fields**

- *edge (List[indra_network_search.data_models.__init__.Node])*
- *stmts (Dict[int, indra_network_search.data_models.__init__.StmtData])*
- *belief (float)*

- *weight (float)*
- *db_url_edge (str)*
- *url_by_type (Dict[str, str])*

**field belief: float [Required]**

**field db_url_edge: str [Required]**

**field edge: List[*indra_network_search.data_models.__init__.Node*] [Required]**

**field stmts: Dict[int, *indra_network_search.data_models.__init__.StmtData*] [Required]**

**field url_by_type: Dict[str, str] [Required]**

**field weight: float [Required]**

**pydantic model** indra_network_search.data_models.__init__.**FilterOptions**

Options for filtering out nodes or edges


```
{
   "title": "FilterOptions",
   "description": "Options for filtering out nodes or edges",
   "type": "object",
   "properties": {
      "stmt_filter": {
         "title": "Stmt Filter",
         "default": [],
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "allowed_ns": {
         "title": "Allowed Ns",
         "default": [],
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "node_blacklist": {
         "title": "Node Blacklist",
         "default": [],
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "path_length": {
         "title": "Path Length",
         "type": "integer"
      },
      "belief_cutoff": {
         "title": "Belief Cutoff",
```


(continues on next page)

(continued from previous page)


```
            "default": 0.0,
            "type": "number"
        },
        "curated_db_only": {
            "title": "Curated Db Only",
            "default": false,
            "type": "boolean"
        },
        "max_paths": {
            "title": "Max Paths",
            "default": 50,
            "type": "integer"
        },
        "cull_best_node": {
            "title": "Cull Best Node",
            "type": "integer"
        },
        "weighted": {
            "title": "Weighted",
            "enum": [
                "weight",
                "context_weight",
                "corr_weight"
            ],
            "type": "string"
        },
        "context_weighted": {
            "title": "Context Weighted",
            "default": false,
            "type": "boolean"
        },
        "overall_weighted": {
            "title": "Overall Weighted",
            "default": false,
            "type": "boolean"
        }
    }
}
```


**Fields**

- *stmt_filter (List[indra_network_search.data_models.__init__.ConstrainedStrValue])*
- *allowed_ns (List[indra_network_search.data_models.__init__.ConstrainedStrValue])*
- *node_blacklist (List[str])*
- *path_length (Optional[int])*
- *belief_cutoff (float)*
- *curated_db_only (bool)*
- *max_paths (int)*

- *cull_best_node (Optional[int])*
- *weighted (Optional[typing_extensions.Literal[weight, context_weight, corr_weight]])*
- *context_weighted (bool)*
- *overall_weighted (bool)*

**field allowed_ns: List[indra_network_search.data_models.__init__.ConstrainedStrValue] = []**

**field belief_cutoff: float = 0.0**

**field context_weighted: bool = False**

**field cull_best_node: Optional[int] = None**

**field curated_db_only: bool = False**

**field max_paths: int = 50**

**field node_blacklist: List[str] = []**

**field overall_weighted: bool = False**

**field path_length: Optional[int] = None**

**field stmt_filter: List[indra_network_search.data_models.__init__.ConstrainedStrValue] = []**

**field weighted: Optional[typing_extensions.Literal[weight, context_weight, corr_weight]] = None**

**no_filters**()
Return True if all filter options are set to defaults

**Return type** bool

**no_node_filters**()
Return True if the node filter options allow all nodes

**no_stmt_filters**()
Return True if the stmt filter options allow all statements

**pydantic model** indra_network_search.data_models.__init__.**MultiInteractorsOptions**
Multi interactors options


```
{
   "title": "MultiInteractorsOptions",
   "description": "Multi interactors options",
   "type": "object",
   "properties": {
      "nodes": {
         "title": "Nodes",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "downstream": {
         "title": "Downstream",
```


(continues on next page)

(continued from previous page)


```
            "type": "boolean"
        },
        "allowed_ns": {
            "title": "Allowed Ns",
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "stmt_types": {
            "title": "Stmt Types",
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "source_filter": {
            "title": "Source Filter",
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "max_results": {
            "title": "Max Results",
            "default": 50,
            "type": "integer"
        },
        "hash_blacklist": {
            "title": "Hash Blacklist",
            "type": "array",
            "items": {
                "type": "integer"
            },
            "uniqueItems": true
        },
        "node_blacklist": {
            "title": "Node Blacklist",
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "belief_cutoff": {
            "title": "Belief Cutoff",
            "default": 0.0,
            "type": "number"
        },
        "curated_db_only": {
            "title": "Curated Db Only",
            "default": false,
            "type": "boolean"
```


(continues on next page)

(continued from previous page)

```
        }
    },
    "required": [
        "nodes",
        "downstream"
    ]
}
```

**Fields**

- *nodes (List[str])*
- *downstream (bool)*
- *allowed_ns (Optional[List[str]])*
- *stmt_types (Optional[List[str]])*
- *source_filter (Optional[List[str]])*
- *max_results (int)*
- *hash_blacklist (Optional[Set[int]])*
- *node_blacklist (Optional[List[str]])*
- *belief_cutoff (float)*
- *curated_db_only (bool)*

**field allowed_ns: Optional[List[str]] = None**

**field belief_cutoff: float = 0.0**

**field curated_db_only: bool = False**

**field downstream: bool [Required]**

**field hash_blacklist: Optional[Set[int]] = None**

**field max_results: int = 50**

**field node_blacklist: Optional[List[str]] = None**

**field nodes: List[str] [Required]**

**field source_filter: Optional[List[str]] = None**

**field stmt_types: Optional[List[str]] = None**

**pydantic model** indra_network_search.data_models.__init__.**MultiInteractorsRestQuery**

Multi interactors rest query

```
{
    "title": "MultiInteractorsRestQuery",
    "description": "Multi interactors rest query",
    "type": "object",
    "properties": {
        "nodes": {
            "title": "Nodes",
            "type": "array",
            "items": {
```

(continues on next page)

(continued from previous page)


```
            "type": "string"
         }
      },
      "downstream": {
         "title": "Downstream",
         "type": "boolean"
      },
      "allowed_ns": {
         "title": "Allowed Ns",
         "type": "array",
         "items": {
            "type": "string",
            "minLength": 1
         }
      },
      "stmt_types": {
         "title": "Stmt Types",
         "type": "array",
         "items": {
            "type": "string",
            "minLength": 1
         }
      },
      "source_filter": {
         "title": "Source Filter",
         "type": "array",
         "items": {
            "type": "string",
            "minLength": 1
         }
      },
      "max_results": {
         "title": "Max Results",
         "default": 50,
         "type": "integer"
      },
      "node_blacklist": {
         "title": "Node Blacklist",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "belief_cutoff": {
         "title": "Belief Cutoff",
         "default": 0.0,
         "type": "number"
      },
      "curated_db_only": {
         "title": "Curated Db Only",
         "default": false,
         "type": "boolean"
```


(continues on next page)

(continued from previous page)

```
        },
        "timeout": {
            "title": "Timeout",
            "default": 30,
            "minimum": 5.0,
            "maximum": 120.0,
            "type": "number"
        }
    },
    "required": [
        "nodes",
        "downstream"
    ]
}
```

**Fields**

- *nodes (List[str])*
- *downstream (bool)*
- *allowed_ns (Optional[List[indra_network_search.data_models.__init__.ConstrainedStrValue]])*
- *stmt_types (Optional[List[indra_network_search.data_models.__init__.ConstrainedStrValue]])*
- *source_filter (Optional[List[indra_network_search.data_models.__init__.ConstrainedStrValue]])*
- *max_results (int)*
- *node_blacklist (Optional[List[str]])*
- *belief_cutoff (float)*
- *curated_db_only (bool)*
- *timeout (indra_network_search.data_models.__init__.ConstrainedFloatValue)*

**field allowed_ns:**
**Optional[List[indra_network_search.data_models.__init__.ConstrainedStrValue]] = None**

**field belief_cutoff:  float = 0.0**

**field curated_db_only:  bool = False**

**field downstream:  bool [Required]**

**field max_results:  int = 50**

**field node_blacklist:  Optional[List[str]] = None**

**field nodes:  List[str] [Required]**

**field source_filter:**
**Optional[List[indra_network_search.data_models.__init__.ConstrainedStrValue]] = None**

**field stmt_types:**
**Optional[List[indra_network_search.data_models.__init__.ConstrainedStrValue]] = None**

**field timeout: indra_network_search.data_models.__init__.ConstrainedFloatValue = 30**

**Constraints**

- **minimum** = 5.0
- **maximum** = 120.0

**pydantic model** indra_network_search.data_models.__init__.**MultiInteractorsResults**

Results post direct_multi_interactors


```
{
   "title": "MultiInteractorsResults",
   "description": "Results post direct_multi_interactors",
   "type": "object",
   "properties": {
      "targets": {
         "title": "Targets",
         "type": "array",
         "items": {
            "$ref": "#/definitions/Node"
         }
      },
      "regulators": {
         "title": "Regulators",
         "type": "array",
         "items": {
            "$ref": "#/definitions/Node"
         }
      },
      "edge_data": {
         "title": "Edge Data",
         "default": [],
         "type": "array",
         "items": {
            "$ref": "#/definitions/EdgeData"
         }
      }
   },
   "required": [
      "targets",
      "regulators"
   ],
   "definitions": {
      "Node": {
         "title": "Node",
         "description": "Data for a node",
         "type": "object",
         "properties": {
            "name": {
               "title": "Name",
               "minLength": 1,
               "type": "string"
            },
            "namespace": {
```


(continues on next page)

(continued from previous page)

```
                "title": "Namespace",
                "minLength": 1,
                "type": "string"
            },
            "identifier": {
                "title": "Identifier",
                "minLength": 1,
                "type": "string"
            },
            "lookup": {
                "title": "Lookup",
                "minLength": 1,
                "type": "string"
            },
            "sign": {
                "title": "Sign",
                "minimum": 0,
                "maximum": 1,
                "type": "integer"
            }
        },
        "required": [
            "namespace",
            "identifier"
        ]
    },
    "StmtData": {
        "title": "StmtData",
        "description": "Data for one statement supporting an edge",
        "type": "object",
        "properties": {
            "stmt_type": {
                "title": "Stmt Type",
                "type": "string"
            },
            "evidence_count": {
                "title": "Evidence Count",
                "minimum": 1,
                "type": "integer"
            },
            "stmt_hash": {
                "title": "Stmt Hash",
                "anyOf": [
                    {
                        "type": "integer"
                    },
                    {
                        "type": "string",
                        "minLength": 1,
                        "maxLength": 2083,
                        "format": "uri"
                    }
```

(continues on next page)

(continued from previous page)


```
            ]
         },
         "source_counts": {
            "title": "Source Counts",
            "type": "object",
            "additionalProperties": {
               "type": "integer"
            }
         },
         "belief": {
            "title": "Belief",
            "minimum": 0.0,
            "maximum": 1.0,
            "type": "number"
         },
         "curated": {
            "title": "Curated",
            "type": "boolean"
         },
         "english": {
            "title": "English",
            "type": "string"
         },
         "weight": {
            "title": "Weight",
            "type": "number"
         },
         "residue": {
            "title": "Residue",
            "default": "",
            "type": "string"
         },
         "position": {
            "title": "Position",
            "default": "",
            "type": "string"
         },
         "initial_sign": {
            "title": "Initial Sign",
            "minimum": 0,
            "maximum": 1,
            "type": "integer"
         },
         "db_url_hash": {
            "title": "Db Url Hash",
            "type": "string"
         }
      },
      "required": [
         "stmt_type",
         "evidence_count",
         "stmt_hash",
```


(continues on next page)

(continued from previous page)

```
            "source_counts",
            "belief",
            "curated",
            "english",
            "db_url_hash"
         ]
      },
      "StmtTypeSupport": {
         "title": "StmtTypeSupport",
         "description": "Data per statement type",
         "type": "object",
         "properties": {
            "stmt_type": {
               "title": "Stmt Type",
               "type": "string"
            },
            "source_counts": {
               "title": "Source Counts",
               "default": {},
               "type": "object",
               "additionalProperties": {
                  "type": "integer"
               }
            },
            "statements": {
               "title": "Statements",
               "type": "array",
               "items": {
                  "$ref": "#/definitions/StmtData"
               }
            }
         },
         "required": [
            "stmt_type",
            "statements"
         ]
      },
      "EdgeData": {
         "title": "EdgeData",
         "description": "Data for one single edge",
         "type": "object",
         "properties": {
            "edge": {
               "title": "Edge",
               "type": "array",
               "items": {
                  "$ref": "#/definitions/Node"
               }
            },
            "statements": {
               "title": "Statements",
               "type": "object",
```

(continues on next page)

(continued from previous page)


```
            "additionalProperties": {
               "$ref": "#/definitions/StmtTypeSupport"
            }
         },
         "belief": {
            "title": "Belief",
            "minimum": 0,
            "maximum": 1,
            "type": "number"
         },
         "weight": {
            "title": "Weight",
            "minimum": 0,
            "type": "number"
         },
         "context_weight": {
            "title": "Context Weight",
            "default": "N/A",
            "anyOf": [
               {
                  "type": "string"
               },
               {
                  "type": "number",
                  "exclusiveMinimum": 0
               },
               {
                  "enum": [
                     "N/A"
                  ],
                  "type": "string"
               }
            ]
         },
         "z_score": {
            "title": "Z Score",
            "type": "number"
         },
         "corr_weight": {
            "title": "Corr Weight",
            "exclusiveMinimum": 0.0,
            "type": "number"
         },
         "sign": {
            "title": "Sign",
            "minimum": 0,
            "maximum": 1,
            "type": "integer"
         },
         "db_url_edge": {
            "title": "Db Url Edge",
            "type": "string"
```


(continues on next page)

(continued from previous page)


```
            },
            "source_counts": {
               "title": "Source Counts",
               "default": {},
               "type": "object",
               "additionalProperties": {
                  "type": "integer"
               }
            }
         },
         "required": [
            "edge",
            "statements",
            "belief",
            "weight",
            "db_url_edge"
         ]
      }
   }
}
```


**Fields**

- *targets (List[indra_network_search.data_models.__init__.Node])*
- *regulators (List[indra_network_search.data_models.__init__.Node])*
- *edge_data (List[indra_network_search.data_models.__init__.EdgeData])*

**field edge_data: List[*indra_network_search.data_models.__init__.EdgeData*] = []**

**field regulators: List[*indra_network_search.data_models.__init__.Node*] [Required]**

**field targets: List[*indra_network_search.data_models.__init__.Node*] [Required]**

**pydantic model** indra_network_search.data_models.__init__.**NetworkSearchQuery**

The query model for network searches


```
{
   "title": "NetworkSearchQuery",
   "description": "The query model for network searches",
   "type": "object",
   "properties": {
      "source": {
         "title": "Source",
         "default": "",
         "type": "string"
      },
      "target": {
         "title": "Target",
         "default": "",
         "type": "string"
      },
      "stmt_filter": {
         "title": "Stmt Filter",
```


(continues on next page)

(continued from previous page)


```
            "default": [],
            "type": "array",
            "items": {
               "type": "string"
            }
         },
         "filter_curated": {
            "title": "Filter Curated",
            "default": true,
            "type": "boolean"
         },
         "allowed_ns": {
            "title": "Allowed Ns",
            "default": [],
            "type": "array",
            "items": {
               "type": "string"
            }
         },
         "node_blacklist": {
            "title": "Node Blacklist",
            "default": [],
            "type": "array",
            "items": {
               "type": "string"
            }
         },
         "path_length": {
            "title": "Path Length",
            "type": "integer"
         },
         "depth_limit": {
            "title": "Depth Limit",
            "default": 2,
            "type": "integer"
         },
         "sign": {
            "title": "Sign",
            "minimum": 0,
            "maximum": 1,
            "type": "integer"
         },
         "weighted": {
            "title": "Weighted",
            "default": "unweighted",
            "enum": [
               "belief",
               "context",
               "z_score",
               "unweighted"
            ],
            "type": "string"
```


(continues on next page)

(continued from previous page)


```
    },
    "belief_cutoff": {
       "title": "Belief Cutoff",
       "default": 0.0,
       "anyOf": [
          {
             "type": "number"
          },
          {
             "type": "boolean"
          }
       ]
    },
    "curated_db_only": {
       "title": "Curated Db Only",
       "default": false,
       "type": "boolean"
    },
    "fplx_expand": {
       "title": "Fplx Expand",
       "default": false,
       "type": "boolean"
    },
    "k_shortest": {
       "title": "K Shortest",
       "default": 50,
       "type": "integer"
    },
    "max_per_node": {
       "title": "Max Per Node",
       "default": 5,
       "type": "integer"
    },
    "cull_best_node": {
       "title": "Cull Best Node",
       "type": "integer"
    },
    "mesh_ids": {
       "title": "Mesh Ids",
       "default": [],
       "type": "array",
       "items": {
          "type": "string"
       }
    },
    "strict_mesh_id_filtering": {
       "title": "Strict Mesh Id Filtering",
       "default": false,
       "type": "boolean"
    },
    "const_c": {
       "title": "Const C",
```


(continues on next page)

(continued from previous page)


```
            "default": 1,
            "type": "integer"
        },
        "const_tk": {
            "title": "Const Tk",
            "default": 10,
            "type": "integer"
        },
        "user_timeout": {
            "title": "User Timeout",
            "default": 30,
            "anyOf": [
                {
                    "type": "number"
                },
                {
                    "type": "boolean"
                }
            ]
        },
        "two_way": {
            "title": "Two Way",
            "default": false,
            "type": "boolean"
        },
        "shared_regulators": {
            "title": "Shared Regulators",
            "default": false,
            "type": "boolean"
        },
        "terminal_ns": {
            "title": "Terminal Ns",
            "default": [],
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "format": {
            "title": "Format",
            "default": "json",
            "type": "string"
        }
    },
    "additionalProperties": false
}
```


**Config**

- **allow_mutation**: *bool = False*
- **extra**: *Extra = forbid*

**Fields**

- *source (indra_network_search.data_models.__init__.ConstrainedStrValue)*
- *target (indra_network_search.data_models.__init__.ConstrainedStrValue)*
- *stmt_filter (List[indra_network_search.data_models.__init__.ConstrainedStrValue])*
- *filter_curated (bool)*
- *allowed_ns (List[indra_network_search.data_models.__init__.ConstrainedStrValue])*
- *node_blacklist (List[str])*
- *path_length (Optional[int])*
- *depth_limit (int)*
- *sign (Optional[indra_network_search.data_models.__init__.ConstrainedIntValue])*
- *weighted (typing_extensions.Literal[belief, context, z_score, unweighted])*
- *belief_cutoff (Union[float, bool])*
- *curated_db_only (bool)*
- *fplx_expand (bool)*
- *k_shortest (int)*
- *max_per_node (int)*
- *cull_best_node (Optional[int])*
- *mesh_ids (List[str])*
- *strict_mesh_id_filtering (bool)*
- *const_c (int)*
- *const_tk (int)*
- *user_timeout (Union[float, bool])*
- *two_way (bool)*
- *shared_regulators (bool)*
- *terminal_ns (List[str])*
- *format (str)*

**Validators**

- *is_positive_int* » *path_length*
- *is_pos_int* » *max_per_node*
- *is_int_gt2* » *cull_best_node*

**field allowed_ns:**
**List[indra_network_search.data_models.__init__.ConstrainedStrValue] = []**

**field belief_cutoff:  Union[float, bool] = 0.0**

field const_c: int = 1

field const_tk: int = 10

field cull_best_node: Optional[int] = None

Validated by

- *is_int_gt2*

field curated_db_only: bool = False

field depth_limit: int = 2

field filter_curated: bool = True

field format: str = 'json'

field fplx_expand: bool = False

field k_shortest: int = 50

field max_per_node: int = 5

Validated by

- *is_pos_int*

field mesh_ids: List[str] = []

field node_blacklist: List[str] = []

field path_length: Optional[int] = None

Validated by

- *is_positive_int*

field shared_regulators: bool = False

field sign: Optional[indra_network_search.data_models.__init__.ConstrainedIntValue] = None

Constraints

- **minimum** = 0
- **maximum** = 1

field source: indra_network_search.data_models.__init__.ConstrainedStrValue = ''

field stmt_filter: List[indra_network_search.data_models.__init__.ConstrainedStrValue] = []

field strict_mesh_id_filtering: bool = False

field target: indra_network_search.data_models.__init__.ConstrainedStrValue = ''

field terminal_ns: List[str] = []

field two_way: bool = False

field user_timeout: Union[float, bool] = 30

field weighted: typing_extensions.Literal[belief, context, z_score, unweighted] = 'unweighted'

get_filter_options()

Returns the filter options

**Return type** *FilterOptions*

**get_hash**()
Get the corresponding query hash of the query

**get_int_sign**()
Return the integer representation of the sign

**Return type** Optional[int]

**is_context_weighted**()
Return True if this query is context weighted

**validator is_int_gt2 »** cull_best_node
Validate cull_best_node >= 2

**is_overall_weighted**()
Return True if this query is weighted

This method is used to determine if a weighted search needs to be done using either of shortest_simple_paths and open_dijkstra_search.

The exception to self.weighted not being None but still be unweighted is strict mesh id search.

**Return type** bool

**validator is_pos_int »** max_per_node
Validate max_per_node >= 1 if given

**validator is_positive_int »** path_length
Validate path_length >= 1 if given

**reverse_search**()
Return a copy of the query with source and target switched

**pydantic model** indra_network_search.data_models.__init__.**Node**
Data for a node


```
{
   "title": "Node",
   "description": "Data for a node",
   "type": "object",
   "properties": {
      "name": {
         "title": "Name",
         "minLength": 1,
         "type": "string"
      },
      "namespace": {
         "title": "Namespace",
         "minLength": 1,
         "type": "string"
      },
      "identifier": {
         "title": "Identifier",
         "minLength": 1,
         "type": "string"
      },
      "lookup": {
         "title": "Lookup",
```


(continues on next page)

(continued from previous page)

```
            "minLength": 1,
            "type": "string"
         },
         "sign": {
            "title": "Sign",
            "minimum": 0,
            "maximum": 1,
            "type": "integer"
         }
      },
      "required": [
         "namespace",
         "identifier"
      ]
}
```

**Fields**

- *name (Optional[indra_network_search.data_models.__init__.ConstrainedStrValue])*
- *namespace (indra_network_search.data_models.__init__.ConstrainedStrValue)*
- *identifier (indra_network_search.data_models.__init__.ConstrainedStrValue)*
- *lookup (Optional[indra_network_search.data_models.__init__.ConstrainedStrValue])*
- *sign (Optional[indra_network_search.data_models.__init__.ConstrainedIntValue])*

**field identifier: indra_network_search.data_models.__init__.ConstrainedStrValue [Required]**

**Constraints**

- **minLength** = 1

**field lookup: Optional[indra_network_search.data_models.__init__.ConstrainedStrValue] [Required]**

**Constraints**

- **minLength** = 1

**field name: Optional[indra_network_search.data_models.__init__.ConstrainedStrValue] [Required]**

**Constraints**

- **minLength** = 1

**field namespace: indra_network_search.data_models.__init__.ConstrainedStrValue [Required]**

**Constraints**

- **minLength** = 1

**field sign: Optional[indra_network_search.data_models.__init__.ConstrainedIntValue] [Required]**

**Constraints**

- **minimum** = 0
- **maximum** = 1

**get_unsigned_node()**
Get unsigned version of this node instance

**signed_node_tuple()**
Get a signed node tuple of node name and node sign

**Return type** Tuple[str, int]

**Returns** A name, sign tuple

**Raises** **TypeError** – If sign is not defined, a TypeError

**pydantic model** indra_network_search.data_models.__init__.**OntologyOptions**
Arguments for indra_network_search.pathfinding.shared_parents


```
{
   "title": "OntologyOptions",
   "description": "Arguments for indra_network_search.pathfinding.shared_parents",
   "type": "object",
   "properties": {
      "source_ns": {
         "title": "Source Ns",
         "type": "string"
      },
      "source_id": {
         "title": "Source Id",
         "type": "string"
      },
      "target_ns": {
         "title": "Target Ns",
         "type": "string"
      },
      "target_id": {
         "title": "Target Id",
         "type": "string"
      },
      "max_paths": {
         "title": "Max Paths",
         "default": 50,
         "type": "integer"
      },
      "immediate_only": {
         "title": "Immediate Only",
         "default": false,
         "type": "boolean"
      },
      "is_a_part_of": {
         "title": "Is A Part Of",
         "type": "array",
```


(continues on next page)

(continued from previous page)


```
            "items": {
               "type": "string"
            },
            "uniqueItems": true
         }
      },
      "required": [
         "source_ns",
         "source_id",
         "target_ns",
         "target_id"
      ]
}
```


**Fields**

- *source_ns (str)*
- *source_id (str)*
- *target_ns (str)*
- *target_id (str)*
- *max_paths (int)*
- *immediate_only (Optional[bool])*
- *is_a_part_of (Optional[Set[str]])*

**field immediate_only: Optional[bool] = False**

**field is_a_part_of: Optional[Set[str]] = None**

**field max_paths: int = 50**

**field source_id: str [Required]**

**field source_ns: str [Required]**

**field target_id: str [Required]**

**field target_ns: str [Required]**

**pydantic model** indra_network_search.data_models.__init__.**OntologyResults**

Results for shared_parents


```
{
   "title": "OntologyResults",
   "description": "Results for shared_parents",
   "type": "object",
   "properties": {
      "source": {
         "$ref": "#/definitions/Node"
      },
      "target": {
         "$ref": "#/definitions/Node"
      },
      "parents": {
```


(continues on next page)

(continued from previous page)

```
            "title": "Parents",
            "type": "array",
            "items": {
               "$ref": "#/definitions/Node"
            }
         }
      },
      "required": [
         "source",
         "target",
         "parents"
      ],
      "definitions": {
         "Node": {
            "title": "Node",
            "description": "Data for a node",
            "type": "object",
            "properties": {
               "name": {
                  "title": "Name",
                  "minLength": 1,
                  "type": "string"
               },
               "namespace": {
                  "title": "Namespace",
                  "minLength": 1,
                  "type": "string"
               },
               "identifier": {
                  "title": "Identifier",
                  "minLength": 1,
                  "type": "string"
               },
               "lookup": {
                  "title": "Lookup",
                  "minLength": 1,
                  "type": "string"
               },
               "sign": {
                  "title": "Sign",
                  "minimum": 0,
                  "maximum": 1,
                  "type": "integer"
               }
            },
            "required": [
               "namespace",
               "identifier"
            ]
         }
      }
}
```

**Fields**

- *source (indra_network_search.data_models.__init__.Node)*
- *target (indra_network_search.data_models.__init__.Node)*
- *parents (List[indra_network_search.data_models.__init__.Node])*

**field parents:** List[*indra_network_search.data_models.__init__.Node*] [Required]

**field source:** *indra_network_search.data_models.__init__.Node* [Required]

**field target:** *indra_network_search.data_models.__init__.Node* [Required]

**is_empty**()

Return True if parents list is empty

**Return type** bool

**pydantic model** indra_network_search.data_models.__init__.**Path**

Results for a single path


```
{
   "title": "Path",
   "description": "Results for a single path",
   "type": "object",
   "properties": {
      "path": {
         "title": "Path",
         "type": "array",
         "items": {
            "$ref": "#/definitions/Node"
         }
      },
      "edge_data": {
         "title": "Edge Data",
         "type": "array",
         "items": {
            "$ref": "#/definitions/EdgeData"
         }
      }
   },
   "required": [
      "path",
      "edge_data"
   ],
   "definitions": {
      "Node": {
         "title": "Node",
         "description": "Data for a node",
         "type": "object",
         "properties": {
            "name": {
               "title": "Name",
               "minLength": 1,
               "type": "string"
            },
```


(continues on next page)

(continued from previous page)


```
            "namespace": {
               "title": "Namespace",
               "minLength": 1,
               "type": "string"
            },
            "identifier": {
               "title": "Identifier",
               "minLength": 1,
               "type": "string"
            },
            "lookup": {
               "title": "Lookup",
               "minLength": 1,
               "type": "string"
            },
            "sign": {
               "title": "Sign",
               "minimum": 0,
               "maximum": 1,
               "type": "integer"
            }
         },
         "required": [
            "namespace",
            "identifier"
         ]
      },
      "StmtData": {
         "title": "StmtData",
         "description": "Data for one statement supporting an edge",
         "type": "object",
         "properties": {
            "stmt_type": {
               "title": "Stmt Type",
               "type": "string"
            },
            "evidence_count": {
               "title": "Evidence Count",
               "minimum": 1,
               "type": "integer"
            },
            "stmt_hash": {
               "title": "Stmt Hash",
               "anyOf": [
                  {
                     "type": "integer"
                  },
                  {
                     "type": "string",
                     "minLength": 1,
                     "maxLength": 2083,
                     "format": "uri"
```


(continues on next page)

(continued from previous page)

```
                }
            ]
        },
        "source_counts": {
            "title": "Source Counts",
            "type": "object",
            "additionalProperties": {
                "type": "integer"
            }
        },
        "belief": {
            "title": "Belief",
            "minimum": 0.0,
            "maximum": 1.0,
            "type": "number"
        },
        "curated": {
            "title": "Curated",
            "type": "boolean"
        },
        "english": {
            "title": "English",
            "type": "string"
        },
        "weight": {
            "title": "Weight",
            "type": "number"
        },
        "residue": {
            "title": "Residue",
            "default": "",
            "type": "string"
        },
        "position": {
            "title": "Position",
            "default": "",
            "type": "string"
        },
        "initial_sign": {
            "title": "Initial Sign",
            "minimum": 0,
            "maximum": 1,
            "type": "integer"
        },
        "db_url_hash": {
            "title": "Db Url Hash",
            "type": "string"
        }
    },
    "required": [
        "stmt_type",
        "evidence_count",
```

(continues on next page)

(continued from previous page)

```
            "stmt_hash",
            "source_counts",
            "belief",
            "curated",
            "english",
            "db_url_hash"
         ]
      },
      "StmtTypeSupport": {
         "title": "StmtTypeSupport",
         "description": "Data per statement type",
         "type": "object",
         "properties": {
            "stmt_type": {
               "title": "Stmt Type",
               "type": "string"
            },
            "source_counts": {
               "title": "Source Counts",
               "default": {},
               "type": "object",
               "additionalProperties": {
                  "type": "integer"
               }
            },
            "statements": {
               "title": "Statements",
               "type": "array",
               "items": {
                  "$ref": "#/definitions/StmtData"
               }
            }
         },
         "required": [
            "stmt_type",
            "statements"
         ]
      },
      "EdgeData": {
         "title": "EdgeData",
         "description": "Data for one single edge",
         "type": "object",
         "properties": {
            "edge": {
               "title": "Edge",
               "type": "array",
               "items": {
                  "$ref": "#/definitions/Node"
               }
            },
            "statements": {
               "title": "Statements",
```

(continues on next page)

(continued from previous page)


```
            "type": "object",
            "additionalProperties": {
               "$ref": "#/definitions/StmtTypeSupport"
            }
         },
         "belief": {
            "title": "Belief",
            "minimum": 0,
            "maximum": 1,
            "type": "number"
         },
         "weight": {
            "title": "Weight",
            "minimum": 0,
            "type": "number"
         },
         "context_weight": {
            "title": "Context Weight",
            "default": "N/A",
            "anyOf": [
               {
                  "type": "string"
               },
               {
                  "type": "number",
                  "exclusiveMinimum": 0
               },
               {
                  "enum": [
                     "N/A"
                  ],
                  "type": "string"
               }
            ]
         },
         "z_score": {
            "title": "Z Score",
            "type": "number"
         },
         "corr_weight": {
            "title": "Corr Weight",
            "exclusiveMinimum": 0.0,
            "type": "number"
         },
         "sign": {
            "title": "Sign",
            "minimum": 0,
            "maximum": 1,
            "type": "integer"
         },
         "db_url_edge": {
            "title": "Db Url Edge",
```


(continues on next page)

(continued from previous page)


```
                "type": "string"
            },
            "source_counts": {
                "title": "Source Counts",
                "default": {},
                "type": "object",
                "additionalProperties": {
                    "type": "integer"
                }
            }
        },
        "required": [
            "edge",
            "statements",
            "belief",
            "weight",
            "db_url_edge"
        ]
    }
}
```


**Fields**

- *path (List[indra_network_search.data_models.__init__.Node])*
- *edge_data (List[indra_network_search.data_models.__init__.EdgeData])*

**field edge_data: List[*indra_network_search.data_models.__init__.EdgeData*] [Required]**

**field path: List[*indra_network_search.data_models.__init__.Node*] [Required]**

**is_empty**()
Return True if len(path) == 0 or len(edge_data) == 0

**Return type** bool

**pydantic model** indra_network_search.data_models.__init__.**PathResultData**
Results for any of the path algorithms


```
{
   "title": "PathResultData",
   "description": "Results for any of the path algorithms",
   "type": "object",
   "properties": {
      "source": {
         "$ref": "#/definitions/Node"
      },
      "target": {
         "$ref": "#/definitions/Node"
      },
      "paths": {
         "title": "Paths",
```


(continues on next page)

(continued from previous page)

```
        "type": "object",
        "additionalProperties": {
           "type": "array",
           "items": {
              "$ref": "#/definitions/Path"
           }
        }
     }
  },
  "required": [
     "paths"
  ],
  "definitions": {
     "Node": {
        "title": "Node",
        "description": "Data for a node",
        "type": "object",
        "properties": {
           "name": {
              "title": "Name",
              "minLength": 1,
              "type": "string"
           },
           "namespace": {
              "title": "Namespace",
              "minLength": 1,
              "type": "string"
           },
           "identifier": {
              "title": "Identifier",
              "minLength": 1,
              "type": "string"
           },
           "lookup": {
              "title": "Lookup",
              "minLength": 1,
              "type": "string"
           },
           "sign": {
              "title": "Sign",
              "minimum": 0,
              "maximum": 1,
              "type": "integer"
           }
        },
        "required": [
           "namespace",
           "identifier"
        ]
     },
     "StmtData": {
        "title": "StmtData",
```

(continues on next page)

(continued from previous page)

```
        "description": "Data for one statement supporting an edge",
        "type": "object",
        "properties": {
           "stmt_type": {
              "title": "Stmt Type",
              "type": "string"
           },
           "evidence_count": {
              "title": "Evidence Count",
              "minimum": 1,
              "type": "integer"
           },
           "stmt_hash": {
              "title": "Stmt Hash",
              "anyOf": [
                 {
                    "type": "integer"
                 },
                 {
                    "type": "string",
                    "minLength": 1,
                    "maxLength": 2083,
                    "format": "uri"
                 }
              ]
           },
           "source_counts": {
              "title": "Source Counts",
              "type": "object",
              "additionalProperties": {
                 "type": "integer"
              }
           },
           "belief": {
              "title": "Belief",
              "minimum": 0.0,
              "maximum": 1.0,
              "type": "number"
           },
           "curated": {
              "title": "Curated",
              "type": "boolean"
           },
           "english": {
              "title": "English",
              "type": "string"
           },
           "weight": {
              "title": "Weight",
              "type": "number"
           },
           "residue": {
```

(continues on next page)

(continued from previous page)


```
                "title": "Residue",
                "default": "",
                "type": "string"
            },
            "position": {
                "title": "Position",
                "default": "",
                "type": "string"
            },
            "initial_sign": {
                "title": "Initial Sign",
                "minimum": 0,
                "maximum": 1,
                "type": "integer"
            },
            "db_url_hash": {
                "title": "Db Url Hash",
                "type": "string"
            }
        },
        "required": [
            "stmt_type",
            "evidence_count",
            "stmt_hash",
            "source_counts",
            "belief",
            "curated",
            "english",
            "db_url_hash"
        ]
    },
    "StmtTypeSupport": {
        "title": "StmtTypeSupport",
        "description": "Data per statement type",
        "type": "object",
        "properties": {
            "stmt_type": {
                "title": "Stmt Type",
                "type": "string"
            },
            "source_counts": {
                "title": "Source Counts",
                "default": {},
                "type": "object",
                "additionalProperties": {
                    "type": "integer"
                }
            },
            "statements": {
                "title": "Statements",
                "type": "array",
                "items": {
```


(continues on next page)

(continued from previous page)

```
                "$ref": "#/definitions/StmtData"
            }
        }
    },
    "required": [
        "stmt_type",
        "statements"
    ]
},
"EdgeData": {
    "title": "EdgeData",
    "description": "Data for one single edge",
    "type": "object",
    "properties": {
        "edge": {
            "title": "Edge",
            "type": "array",
            "items": {
                "$ref": "#/definitions/Node"
            }
        },
        "statements": {
            "title": "Statements",
            "type": "object",
            "additionalProperties": {
                "$ref": "#/definitions/StmtTypeSupport"
            }
        },
        "belief": {
            "title": "Belief",
            "minimum": 0,
            "maximum": 1,
            "type": "number"
        },
        "weight": {
            "title": "Weight",
            "minimum": 0,
            "type": "number"
        },
        "context_weight": {
            "title": "Context Weight",
            "default": "N/A",
            "anyOf": [
                {
                    "type": "string"
                },
                {
                    "type": "number",
                    "exclusiveMinimum": 0
                },
                {
                    "enum": [
```

(continues on next page)

(continued from previous page)


```
                        "N/A"
                    ],
                    "type": "string"
                }
            ]
        },
        "z_score": {
            "title": "Z Score",
            "type": "number"
        },
        "corr_weight": {
            "title": "Corr Weight",
            "exclusiveMinimum": 0.0,
            "type": "number"
        },
        "sign": {
            "title": "Sign",
            "minimum": 0,
            "maximum": 1,
            "type": "integer"
        },
        "db_url_edge": {
            "title": "Db Url Edge",
            "type": "string"
        },
        "source_counts": {
            "title": "Source Counts",
            "default": {},
            "type": "object",
            "additionalProperties": {
                "type": "integer"
            }
        }
    },
    "required": [
        "edge",
        "statements",
        "belief",
        "weight",
        "db_url_edge"
    ]
},
"Path": {
    "title": "Path",
    "description": "Results for a single path",
    "type": "object",
    "properties": {
        "path": {
            "title": "Path",
            "type": "array",
            "items": {
                "$ref": "#/definitions/Node"
```


(continues on next page)

(continued from previous page)


```
                }
            },
            "edge_data": {
                "title": "Edge Data",
                "type": "array",
                "items": {
                    "$ref": "#/definitions/EdgeData"
                }
            }
        },
        "required": [
            "path",
            "edge_data"
        ]
    }
}
}
```


**Fields**

- *source (Optional[indra_network_search.data_models.__init__.Node])*
- *target (Optional[indra_network_search.data_models.__init__.Node])*
- *paths (Dict[int, List[indra_network_search.data_models.__init__.Path]])*

**field paths: Dict[int, List[*indra_network_search.data_models.__init__.Path*]]** **[Required]**

**field source: Optional[*indra_network_search.data_models.__init__.Node*] = None**

**field target: Optional[*indra_network_search.data_models.__init__.Node*] = None**

**is_empty**()

Return True if paths list is empty

**Return type** bool

**pydantic model** indra_network_search.data_models.__init__.**Results**

The model wrapping all results from the NetworkSearchQuery


```
{
   "title": "Results",
   "description": "The model wrapping all results from the NetworkSearchQuery",
   "type": "object",
   "properties": {
      "query_hash": {
         "title": "Query Hash",
         "type": "string"
      },
      "time_limit": {
         "title": "Time Limit",
         "type": "number"
      },
```


(continues on next page)

(continued from previous page)


```
      "timed_out": {
         "title": "Timed Out",
         "type": "boolean"
      },
      "hashes": {
         "title": "Hashes",
         "default": [],
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "path_results": {
         "$ref": "#/definitions/PathResultData"
      },
      "reverse_path_results": {
         "$ref": "#/definitions/PathResultData"
      },
      "ontology_results": {
         "$ref": "#/definitions/OntologyResults"
      },
      "shared_target_results": {
         "$ref": "#/definitions/SharedInteractorsResults"
      },
      "shared_regulators_results": {
         "$ref": "#/definitions/SharedInteractorsResults"
      }
   },
   "required": [
      "query_hash",
      "time_limit",
      "timed_out"
   ],
   "definitions": {
      "Node": {
         "title": "Node",
         "description": "Data for a node",
         "type": "object",
         "properties": {
            "name": {
               "title": "Name",
               "minLength": 1,
               "type": "string"
            },
            "namespace": {
               "title": "Namespace",
               "minLength": 1,
               "type": "string"
            },
            "identifier": {
               "title": "Identifier",
               "minLength": 1,
```


(continues on next page)

(continued from previous page)


```
            "type": "string"
         },
         "lookup": {
            "title": "Lookup",
            "minLength": 1,
            "type": "string"
         },
         "sign": {
            "title": "Sign",
            "minimum": 0,
            "maximum": 1,
            "type": "integer"
         }
      },
      "required": [
         "namespace",
         "identifier"
      ]
   },
   "StmtData": {
      "title": "StmtData",
      "description": "Data for one statement supporting an edge",
      "type": "object",
      "properties": {
         "stmt_type": {
            "title": "Stmt Type",
            "type": "string"
         },
         "evidence_count": {
            "title": "Evidence Count",
            "minimum": 1,
            "type": "integer"
         },
         "stmt_hash": {
            "title": "Stmt Hash",
            "anyOf": [
               {
                  "type": "integer"
               },
               {
                  "type": "string",
                  "minLength": 1,
                  "maxLength": 2083,
                  "format": "uri"
               }
            ]
         },
         "source_counts": {
            "title": "Source Counts",
            "type": "object",
            "additionalProperties": {
               "type": "integer"
```


(continues on next page)

(continued from previous page)

```
            }
         },
         "belief": {
            "title": "Belief",
            "minimum": 0.0,
            "maximum": 1.0,
            "type": "number"
         },
         "curated": {
            "title": "Curated",
            "type": "boolean"
         },
         "english": {
            "title": "English",
            "type": "string"
         },
         "weight": {
            "title": "Weight",
            "type": "number"
         },
         "residue": {
            "title": "Residue",
            "default": "",
            "type": "string"
         },
         "position": {
            "title": "Position",
            "default": "",
            "type": "string"
         },
         "initial_sign": {
            "title": "Initial Sign",
            "minimum": 0,
            "maximum": 1,
            "type": "integer"
         },
         "db_url_hash": {
            "title": "Db Url Hash",
            "type": "string"
         }
      },
      "required": [
         "stmt_type",
         "evidence_count",
         "stmt_hash",
         "source_counts",
         "belief",
         "curated",
         "english",
         "db_url_hash"
      ]
   },
```

(continues on next page)

(continued from previous page)

```
    "StmtTypeSupport": {
       "title": "StmtTypeSupport",
       "description": "Data per statement type",
       "type": "object",
       "properties": {
          "stmt_type": {
             "title": "Stmt Type",
             "type": "string"
          },
          "source_counts": {
             "title": "Source Counts",
             "default": {},
             "type": "object",
             "additionalProperties": {
                "type": "integer"
             }
          },
          "statements": {
             "title": "Statements",
             "type": "array",
             "items": {
                "$ref": "#/definitions/StmtData"
             }
          }
       },
       "required": [
          "stmt_type",
          "statements"
       ]
    },
    "EdgeData": {
       "title": "EdgeData",
       "description": "Data for one single edge",
       "type": "object",
       "properties": {
          "edge": {
             "title": "Edge",
             "type": "array",
             "items": {
                "$ref": "#/definitions/Node"
             }
          },
          "statements": {
             "title": "Statements",
             "type": "object",
             "additionalProperties": {
                "$ref": "#/definitions/StmtTypeSupport"
             }
          },
          "belief": {
             "title": "Belief",
             "minimum": 0,
```

(continues on next page)

(continued from previous page)


```
            "maximum": 1,
            "type": "number"
        },
        "weight": {
            "title": "Weight",
            "minimum": 0,
            "type": "number"
        },
        "context_weight": {
            "title": "Context Weight",
            "default": "N/A",
            "anyOf": [
                {
                    "type": "string"
                },
                {
                    "type": "number",
                    "exclusiveMinimum": 0
                },
                {
                    "enum": [
                        "N/A"
                    ],
                    "type": "string"
                }
            ]
        },
        "z_score": {
            "title": "Z Score",
            "type": "number"
        },
        "corr_weight": {
            "title": "Corr Weight",
            "exclusiveMinimum": 0.0,
            "type": "number"
        },
        "sign": {
            "title": "Sign",
            "minimum": 0,
            "maximum": 1,
            "type": "integer"
        },
        "db_url_edge": {
            "title": "Db Url Edge",
            "type": "string"
        },
        "source_counts": {
            "title": "Source Counts",
            "default": {},
            "type": "object",
            "additionalProperties": {
                "type": "integer"
```


(continues on next page)

(continued from previous page)

```
            }
         }
      },
      "required": [
         "edge",
         "statements",
         "belief",
         "weight",
         "db_url_edge"
      ]
   },
   "Path": {
      "title": "Path",
      "description": "Results for a single path",
      "type": "object",
      "properties": {
         "path": {
            "title": "Path",
            "type": "array",
            "items": {
               "$ref": "#/definitions/Node"
            }
         },
         "edge_data": {
            "title": "Edge Data",
            "type": "array",
            "items": {
               "$ref": "#/definitions/EdgeData"
            }
         }
      },
      "required": [
         "path",
         "edge_data"
      ]
   },
   "PathResultData": {
      "title": "PathResultData",
      "description": "Results for any of the path algorithms",
      "type": "object",
      "properties": {
         "source": {
            "$ref": "#/definitions/Node"
         },
         "target": {
            "$ref": "#/definitions/Node"
         },
         "paths": {
            "title": "Paths",
            "type": "object",
            "additionalProperties": {
               "type": "array",
```

(continues on next page)

(continued from previous page)


```
                "items": {
                    "$ref": "#/definitions/Path"
                }
            }
        }
    },
    "required": [
        "paths"
    ]
},
"OntologyResults": {
    "title": "OntologyResults",
    "description": "Results for shared_parents",
    "type": "object",
    "properties": {
        "source": {
            "$ref": "#/definitions/Node"
        },
        "target": {
            "$ref": "#/definitions/Node"
        },
        "parents": {
            "title": "Parents",
            "type": "array",
            "items": {
                "$ref": "#/definitions/Node"
            }
        }
    },
    "required": [
        "source",
        "target",
        "parents"
    ]
},
"SharedInteractorsResults": {
    "title": "SharedInteractorsResults",
    "description": "Results for shared targets and shared regulators",
    "type": "object",
    "properties": {
        "source_data": {
            "title": "Source Data",
            "type": "array",
            "items": {
                "$ref": "#/definitions/EdgeData"
            }
        },
        "target_data": {
            "title": "Target Data",
            "type": "array",
            "items": {
                "$ref": "#/definitions/EdgeData"
```


(continues on next page)

(continued from previous page)

```
                }
            },
            "downstream": {
                "title": "Downstream",
                "type": "boolean"
            }
        },
        "required": [
            "source_data",
            "target_data",
            "downstream"
        ]
    }
}
}
```

**Fields**

- *query_hash (str)*
- *time_limit (float)*
- *timed_out (bool)*
- *hashes (List[str])*
- *path_results (Optional[indra_network_search.data_models.__init__.PathResultData])*
- *reverse_path_results (Optional[indra_network_search.data_models.__init__.PathResultData])*
- *ontology_results (Optional[indra_network_search.data_models.__init__.OntologyResults])*
- *shared_target_results (Optional[indra_network_search.data_models.__init__.SharedInteractorsResults])*
- *shared_regulators_results (Optional[indra_network_search.data_models.__init__.SharedInteractorsResults])*

**field hashes: List[str] = []**

**field ontology_results:**
**Optional[*indra_network_search.data_models.__init__.OntologyResults*] = None**

**field path_results:**
**Optional[*indra_network_search.data_models.__init__.PathResultData*] = None**

**field query_hash: str [Required]**

**field reverse_path_results:**
**Optional[*indra_network_search.data_models.__init__.PathResultData*] = None**

**field shared_regulators_results:**
**Optional[*indra_network_search.data_models.__init__.SharedInteractorsResults*] = None**

**field shared_target_results:**
**Optional[*indra_network_search.data_models.__init__.SharedInteractorsResults*] = None**

**field time_limit:** *float* **[Required]**

**field timed_out:** *bool* **[Required]**

**pydantic model** indra_network_search.data_models.__init__.**SharedInteractorsOptions**

Arguments for indra_network_search.pathfinding.shared_interactors


```
{
   "title": "SharedInteractorsOptions",
   "description": "Arguments for indra_network_search.pathfinding.shared_interactors
↪",
   "type": "object",
   "properties": {
      "source": {
         "title": "Source",
         "anyOf": [
            {
               "type": "string"
            },
            {
               "type": "array",
               "items": [
                  {
                     "type": "string"
                  },
                  {
                     "type": "integer"
                  }
               ]
            }
         ]
      },
      "target": {
         "title": "Target",
         "anyOf": [
            {
               "type": "string"
            },
            {
               "type": "array",
               "items": [
                  {
                     "type": "string"
                  },
                  {
                     "type": "integer"
                  }
               ]
            }
         ]
      },
      "allowed_ns": {
         "title": "Allowed Ns",
         "type": "array",
```


(continues on next page)

(continued from previous page)


```
            "items": {
               "type": "string"
            }
         },
         "stmt_types": {
            "title": "Stmt Types",
            "type": "array",
            "items": {
               "type": "string"
            }
         },
         "source_filter": {
            "title": "Source Filter",
            "type": "array",
            "items": {
               "type": "string"
            }
         },
         "max_results": {
            "title": "Max Results",
            "default": 50,
            "type": "integer"
         },
         "regulators": {
            "title": "Regulators",
            "default": false,
            "type": "boolean"
         },
         "sign": {
            "title": "Sign",
            "type": "integer"
         }
      },
      "required": [
         "source",
         "target"
      ]
}
```


**Fields**

- *source (Union[str, Tuple[str, int]])*
- *target (Union[str, Tuple[str, int]])*
- *allowed_ns (Optional[List[str]])*
- *stmt_types (Optional[List[str]])*
- *source_filter (Optional[List[str]])*
- *max_results (Optional[int])*
- *regulators (Optional[bool])*
- *sign (Optional[int])*

**field allowed_ns:** Optional[List[str]] = None

**field max_results:** Optional[int] = 50

**field regulators:** Optional[bool] = False

**field sign:** Optional[int] = None

**field source:** Union[str, Tuple[str, int]] [Required]

**field source_filter:** Optional[List[str]] = None

**field stmt_types:** Optional[List[str]] = None

**field target:** Union[str, Tuple[str, int]] [Required]

**pydantic model** indra_network_search.data_models.__init__.**SharedInteractorsResults**

Results for shared targets and shared regulators


```
{
   "title": "SharedInteractorsResults",
   "description": "Results for shared targets and shared regulators",
   "type": "object",
   "properties": {
      "source_data": {
         "title": "Source Data",
         "type": "array",
         "items": {
            "$ref": "#/definitions/EdgeData"
         }
      },
      "target_data": {
         "title": "Target Data",
         "type": "array",
         "items": {
            "$ref": "#/definitions/EdgeData"
         }
      },
      "downstream": {
         "title": "Downstream",
         "type": "boolean"
      }
   },
   "required": [
      "source_data",
      "target_data",
      "downstream"
   ],
   "definitions": {
      "Node": {
         "title": "Node",
         "description": "Data for a node",
         "type": "object",
         "properties": {
            "name": {
               "title": "Name",
               "minLength": 1,
```


(continues on next page)

(continued from previous page)

```
            "type": "string"
         },
         "namespace": {
            "title": "Namespace",
            "minLength": 1,
            "type": "string"
         },
         "identifier": {
            "title": "Identifier",
            "minLength": 1,
            "type": "string"
         },
         "lookup": {
            "title": "Lookup",
            "minLength": 1,
            "type": "string"
         },
         "sign": {
            "title": "Sign",
            "minimum": 0,
            "maximum": 1,
            "type": "integer"
         }
      },
      "required": [
         "namespace",
         "identifier"
      ]
   },
   "StmtData": {
      "title": "StmtData",
      "description": "Data for one statement supporting an edge",
      "type": "object",
      "properties": {
         "stmt_type": {
            "title": "Stmt Type",
            "type": "string"
         },
         "evidence_count": {
            "title": "Evidence Count",
            "minimum": 1,
            "type": "integer"
         },
         "stmt_hash": {
            "title": "Stmt Hash",
            "anyOf": [
               {
                  "type": "integer"
               },
               {
                  "type": "string",
                  "minLength": 1,
```

(continues on next page)

(continued from previous page)


```
                    "maxLength": 2083,
                    "format": "uri"
                }
            ]
        },
        "source_counts": {
            "title": "Source Counts",
            "type": "object",
            "additionalProperties": {
                "type": "integer"
            }
        },
        "belief": {
            "title": "Belief",
            "minimum": 0.0,
            "maximum": 1.0,
            "type": "number"
        },
        "curated": {
            "title": "Curated",
            "type": "boolean"
        },
        "english": {
            "title": "English",
            "type": "string"
        },
        "weight": {
            "title": "Weight",
            "type": "number"
        },
        "residue": {
            "title": "Residue",
            "default": "",
            "type": "string"
        },
        "position": {
            "title": "Position",
            "default": "",
            "type": "string"
        },
        "initial_sign": {
            "title": "Initial Sign",
            "minimum": 0,
            "maximum": 1,
            "type": "integer"
        },
        "db_url_hash": {
            "title": "Db Url Hash",
            "type": "string"
        }
    },
    "required": [
```


(continues on next page)

(continued from previous page)

```
            "stmt_type",
            "evidence_count",
            "stmt_hash",
            "source_counts",
            "belief",
            "curated",
            "english",
            "db_url_hash"
         ]
      },
      "StmtTypeSupport": {
         "title": "StmtTypeSupport",
         "description": "Data per statement type",
         "type": "object",
         "properties": {
            "stmt_type": {
               "title": "Stmt Type",
               "type": "string"
            },
            "source_counts": {
               "title": "Source Counts",
               "default": {},
               "type": "object",
               "additionalProperties": {
                  "type": "integer"
               }
            },
            "statements": {
               "title": "Statements",
               "type": "array",
               "items": {
                  "$ref": "#/definitions/StmtData"
               }
            }
         },
         "required": [
            "stmt_type",
            "statements"
         ]
      },
      "EdgeData": {
         "title": "EdgeData",
         "description": "Data for one single edge",
         "type": "object",
         "properties": {
            "edge": {
               "title": "Edge",
               "type": "array",
               "items": {
                  "$ref": "#/definitions/Node"
               }
            },
```

(continues on next page)

(continued from previous page)

```
            "statements": {
               "title": "Statements",
               "type": "object",
               "additionalProperties": {
                  "$ref": "#/definitions/StmtTypeSupport"
               }
            },
            "belief": {
               "title": "Belief",
               "minimum": 0,
               "maximum": 1,
               "type": "number"
            },
            "weight": {
               "title": "Weight",
               "minimum": 0,
               "type": "number"
            },
            "context_weight": {
               "title": "Context Weight",
               "default": "N/A",
               "anyOf": [
                  {
                     "type": "string"
                  },
                  {
                     "type": "number",
                     "exclusiveMinimum": 0
                  },
                  {
                     "enum": [
                        "N/A"
                     ],
                     "type": "string"
                  }
               ]
            },
            "z_score": {
               "title": "Z Score",
               "type": "number"
            },
            "corr_weight": {
               "title": "Corr Weight",
               "exclusiveMinimum": 0.0,
               "type": "number"
            },
            "sign": {
               "title": "Sign",
               "minimum": 0,
               "maximum": 1,
               "type": "integer"
            },
```

(continues on next page)

(continued from previous page)


```
            "db_url_edge": {
               "title": "Db Url Edge",
               "type": "string"
            },
            "source_counts": {
               "title": "Source Counts",
               "default": {},
               "type": "object",
               "additionalProperties": {
                  "type": "integer"
               }
            }
         },
         "required": [
            "edge",
            "statements",
            "belief",
            "weight",
            "db_url_edge"
         ]
      }
   }
}
```


**Fields**

- *source_data (List[indra_network_search.data_models.__init__.EdgeData])*
- *target_data (List[indra_network_search.data_models.__init__.EdgeData])*
- *downstream (bool)*

**field downstream: bool [Required]**

**field source_data: List[*indra_network_search.data_models.__init__.EdgeData*] [Required]**

**field target_data: List[*indra_network_search.data_models.__init__.EdgeData*] [Required]**

**is_empty**()
Return True if both source and target data is empty

**pydantic model** indra_network_search.data_models.__init__.**ShortestSimplePathOptions**
Arguments for indra.explanation.pathfinding.shortest_simple_paths


```
{
   "title": "ShortestSimplePathOptions",
   "description": "Arguments for indra.explanation.pathfinding.shortest_simple_paths
↪",
   "type": "object",
   "properties": {
      "source": {
```


(continues on next page)

(continued from previous page)


```
            "title": "Source",
            "anyOf": [
               {
                  "type": "string"
               },
               {
                  "type": "array",
                  "items": [
                     {
                        "type": "string"
                     },
                     {
                        "type": "integer"
                     }
                  ]
               }
            ]
         },
         "target": {
            "title": "Target",
            "anyOf": [
               {
                  "type": "string"
               },
               {
                  "type": "array",
                  "items": [
                     {
                        "type": "string"
                     },
                     {
                        "type": "integer"
                     }
                  ]
               }
            ]
         },
         "weight": {
            "title": "Weight",
            "type": "string"
         },
         "ignore_nodes": {
            "title": "Ignore Nodes",
            "type": "array",
            "items": {
               "type": "string"
            },
            "uniqueItems": true
         },
         "ignore_edges": {
            "title": "Ignore Edges",
            "type": "array",
```


(continues on next page)

(continued from previous page)


```
            "items": {
               "type": "array",
               "items": [
                  {
                     "type": "string"
                  },
                  {
                     "type": "string"
                  }
               ]
            },
            "uniqueItems": true
         },
         "hashes": {
            "title": "Hashes",
            "type": "array",
            "items": {
               "type": "integer"
            }
         },
         "strict_mesh_id_filtering": {
            "title": "Strict Mesh Id Filtering",
            "default": false,
            "type": "boolean"
         },
         "const_c": {
            "title": "Const C",
            "default": 1,
            "type": "integer"
         },
         "const_tk": {
            "title": "Const Tk",
            "default": 10,
            "type": "integer"
         }
      },
      "required": [
         "source",
         "target"
      ]
}
```


**Fields**

- *source (Union[str, Tuple[str, int]])*
- *target (Union[str, Tuple[str, int]])*
- *weight (Optional[str])*
- *ignore_nodes (Optional[Set[str]])*
- *ignore_edges (Optional[Set[Tuple[str, str]]])*
- *hashes (Optional[List[int]])*

- *ref_counts_function (Optional[Callable])*
- *strict_mesh_id_filtering (Optional[bool])*
- *const_c (Optional[int])*
- *const_tk (Optional[int])*

**field const_c: Optional[int] = 1**

**field const_tk: Optional[int] = 10**

**field hashes: Optional[List[int]] = None**

**field ignore_edges: Optional[Set[Tuple[str, str]]] = None**

**field ignore_nodes: Optional[Set[str]] = None**

**field ref_counts_function: Optional[Callable] = None**

**field source: Union[str, Tuple[str, int]] [Required]**

**field strict_mesh_id_filtering: Optional[bool] = False**

**field target: Union[str, Tuple[str, int]] [Required]**

**field weight: Optional[str] = None**

**pydantic model** indra_network_search.data_models.__init__.**StmtData**

Data for one statement supporting an edge


```
{
   "title": "StmtData",
   "description": "Data for one statement supporting an edge",
   "type": "object",
   "properties": {
      "stmt_type": {
         "title": "Stmt Type",
         "type": "string"
      },
      "evidence_count": {
         "title": "Evidence Count",
         "minimum": 1,
         "type": "integer"
      },
      "stmt_hash": {
         "title": "Stmt Hash",
         "anyOf": [
            {
               "type": "integer"
            },
            {
               "type": "string",
               "minLength": 1,
               "maxLength": 2083,
               "format": "uri"
            }
         ]
      },
      "source_counts": {
```


(continues on next page)

(continued from previous page)


```
            "title": "Source Counts",
            "type": "object",
            "additionalProperties": {
               "type": "integer"
            }
        },
        "belief": {
            "title": "Belief",
            "minimum": 0.0,
            "maximum": 1.0,
            "type": "number"
        },
        "curated": {
            "title": "Curated",
            "type": "boolean"
        },
        "english": {
            "title": "English",
            "type": "string"
        },
        "weight": {
            "title": "Weight",
            "type": "number"
        },
        "residue": {
            "title": "Residue",
            "default": "",
            "type": "string"
        },
        "position": {
            "title": "Position",
            "default": "",
            "type": "string"
        },
        "initial_sign": {
            "title": "Initial Sign",
            "minimum": 0,
            "maximum": 1,
            "type": "integer"
        },
        "db_url_hash": {
            "title": "Db Url Hash",
            "type": "string"
        }
    },
    "required": [
        "stmt_type",
        "evidence_count",
        "stmt_hash",
        "source_counts",
        "belief",
        "curated",
```


(continues on next page)

(continued from previous page)

```
      "english",
      "db_url_hash"
   ]
}
```

**Fields**

- *stmt_type (str)*
- *evidence_count (indra_network_search.data_models.__init__.ConstrainedIntValue)*
- *stmt_hash (Union[int, pydantic.networks.HttpUrl])*
- *source_counts (Dict[str, int])*
- *belief (indra_network_search.data_models.__init__.ConstrainedFloatValue)*
- *curated (bool)*
- *english (str)*
- *weight (Optional[float])*
- *residue (Optional[str])*
- *position (Optional[str])*
- *initial_sign (Optional[indra_network_search.data_models.__init__.ConstrainedIntValue])*
- *db_url_hash (str)*

**field belief: indra_network_search.data_models.__init__.ConstrainedFloatValue [Required]**

**Constraints**

- **minimum** = 0.0
- **maximum** = 1.0

**field curated: bool [Required]**

**field db_url_hash: str [Required]**

**field english: str [Required]**

**field evidence_count: indra_network_search.data_models.__init__.ConstrainedIntValue [Required]**

**Constraints**

- **minimum** = 1

**field initial_sign: Optional[indra_network_search.data_models.__init__.ConstrainedIntValue] = None**

**Constraints**

- **minimum** = 0
- **maximum** = 1

field position: Optional[str] = ''

field residue: Optional[str] = ''

field source_counts: Dict[str, int] [Required]

field stmt_hash: Union[int, pydantic.networks.HttpUrl] [Required]

field stmt_type: str [Required]

field weight: Optional[float] = None

pydantic model indra_network_search.data_models.__init__.StmtTypeSupport

Data per statement type


```
{
   "title": "StmtTypeSupport",
   "description": "Data per statement type",
   "type": "object",
   "properties": {
      "stmt_type": {
         "title": "Stmt Type",
         "type": "string"
      },
      "source_counts": {
         "title": "Source Counts",
         "default": {},
         "type": "object",
         "additionalProperties": {
            "type": "integer"
         }
      },
      "statements": {
         "title": "Statements",
         "type": "array",
         "items": {
            "$ref": "#/definitions/StmtData"
         }
      }
   },
   "required": [
      "stmt_type",
      "statements"
   ],
   "definitions": {
      "StmtData": {
         "title": "StmtData",
         "description": "Data for one statement supporting an edge",
         "type": "object",
         "properties": {
            "stmt_type": {
               "title": "Stmt Type",
               "type": "string"
            },
            "evidence_count": {
               "title": "Evidence Count",
```


(continues on next page)

(continued from previous page)


```
            "minimum": 1,
            "type": "integer"
        },
        "stmt_hash": {
            "title": "Stmt Hash",
            "anyOf": [
                {
                    "type": "integer"
                },
                {
                    "type": "string",
                    "minLength": 1,
                    "maxLength": 2083,
                    "format": "uri"
                }
            ]
        },
        "source_counts": {
            "title": "Source Counts",
            "type": "object",
            "additionalProperties": {
                "type": "integer"
            }
        },
        "belief": {
            "title": "Belief",
            "minimum": 0.0,
            "maximum": 1.0,
            "type": "number"
        },
        "curated": {
            "title": "Curated",
            "type": "boolean"
        },
        "english": {
            "title": "English",
            "type": "string"
        },
        "weight": {
            "title": "Weight",
            "type": "number"
        },
        "residue": {
            "title": "Residue",
            "default": "",
            "type": "string"
        },
        "position": {
            "title": "Position",
            "default": "",
            "type": "string"
        },
```


(continues on next page)

(continued from previous page)


```
            "initial_sign": {
               "title": "Initial Sign",
               "minimum": 0,
               "maximum": 1,
               "type": "integer"
            },
            "db_url_hash": {
               "title": "Db Url Hash",
               "type": "string"
            }
         },
         "required": [
            "stmt_type",
            "evidence_count",
            "stmt_hash",
            "source_counts",
            "belief",
            "curated",
            "english",
            "db_url_hash"
         ]
      }
   }
}
```


**Fields**

- *stmt_type (str)*
- *source_counts (Dict[str, int])*
- *statements (List[indra_network_search.data_models.__init__.StmtData])*

**field source_counts: Dict[str, int] = {}**

**field statements: List[*indra_network_search.data_models.__init__.StmtData*] [Required]**

**field stmt_type: str [Required]**

**set_source_counts()**
Updates the source count field from the set statement data

**pydantic model** indra_network_search.data_models.__init__.**SubgraphOptions**
Argument for indra_network_search.pathfinding.get_subgraph_edges


```
{
   "title": "SubgraphOptions",
   "description": "Argument for indra_network_search.pathfinding.get_subgraph_edges
↪",
   "type": "object",
   "properties": {
      "nodes": {
         "title": "Nodes",
         "type": "array",
```


(continues on next page)

(continued from previous page)


```
            "items": {
               "$ref": "#/definitions/Node"
            }
         }
      },
      "required": [
         "nodes"
      ],
      "definitions": {
         "Node": {
            "title": "Node",
            "description": "Data for a node",
            "type": "object",
            "properties": {
               "name": {
                  "title": "Name",
                  "minLength": 1,
                  "type": "string"
               },
               "namespace": {
                  "title": "Namespace",
                  "minLength": 1,
                  "type": "string"
               },
               "identifier": {
                  "title": "Identifier",
                  "minLength": 1,
                  "type": "string"
               },
               "lookup": {
                  "title": "Lookup",
                  "minLength": 1,
                  "type": "string"
               },
               "sign": {
                  "title": "Sign",
                  "minimum": 0,
                  "maximum": 1,
                  "type": "integer"
               }
            },
            "required": [
               "namespace",
               "identifier"
            ]
         }
      }
}
```


**Fields**

- *nodes (List[indra_network_search.data_models.__init__.Node])*

**field nodes:** List[*indra_network_search.data_models.__init__.Node*] **[Required]**

**pydantic model** indra_network_search.data_models.__init__.**SubgraphRestQuery**

Subgraph query


```
{
   "title": "SubgraphRestQuery",
   "description": "Subgraph query",
   "type": "object",
   "properties": {
      "nodes": {
         "title": "Nodes",
         "minItems": 1,
         "maxItems": 4000,
         "type": "array",
         "items": {
            "$ref": "#/definitions/Node"
         }
      }
   },
   "required": [
      "nodes"
   ],
   "definitions": {
      "Node": {
         "title": "Node",
         "description": "Data for a node",
         "type": "object",
         "properties": {
            "name": {
               "title": "Name",
               "minLength": 1,
               "type": "string"
            },
            "namespace": {
               "title": "Namespace",
               "minLength": 1,
               "type": "string"
            },
            "identifier": {
               "title": "Identifier",
               "minLength": 1,
               "type": "string"
            },
            "lookup": {
               "title": "Lookup",
               "minLength": 1,
               "type": "string"
            },
            "sign": {
               "title": "Sign",
               "minimum": 0,
               "maximum": 1,
               "type": "integer"
```


(continues on next page)

(continued from previous page)

```
            }
         },
         "required": [
            "namespace",
            "identifier"
         ]
      }
   }
}
```

**Fields**

- *nodes (types.ConstrainedListValue[indra_network_search.data_models.__init__.Node])*

**field nodes:**
**types.ConstrainedListValue[*indra_network_search.data_models.__init__.Node*]**
**[Required]**

**Constraints**

- **minItems** = 1
- **maxItems** = 4000

**pydantic model** indra_network_search.data_models.__init__.**SubgraphResults**

Results for get_subgraph_edges

```
{
   "title": "SubgraphResults",
   "description": "Results for get_subgraph_edges",
   "type": "object",
   "properties": {
      "input_nodes": {
         "title": "Input Nodes",
         "type": "array",
         "items": {
            "$ref": "#/definitions/Node"
         }
      },
      "not_in_graph": {
         "title": "Not In Graph",
         "type": "array",
         "items": {
            "$ref": "#/definitions/Node"
         }
      },
      "available_nodes": {
         "title": "Available Nodes",
         "type": "array",
         "items": {
            "$ref": "#/definitions/Node"
         }
      },
```

(continues on next page)

(continued from previous page)


```
      "edges": {
         "title": "Edges",
         "type": "array",
         "items": {
            "$ref": "#/definitions/EdgeDataByHash"
         }
      }
   },
   "required": [
      "input_nodes",
      "not_in_graph",
      "available_nodes",
      "edges"
   ],
   "definitions": {
      "Node": {
         "title": "Node",
         "description": "Data for a node",
         "type": "object",
         "properties": {
            "name": {
               "title": "Name",
               "minLength": 1,
               "type": "string"
            },
            "namespace": {
               "title": "Namespace",
               "minLength": 1,
               "type": "string"
            },
            "identifier": {
               "title": "Identifier",
               "minLength": 1,
               "type": "string"
            },
            "lookup": {
               "title": "Lookup",
               "minLength": 1,
               "type": "string"
            },
            "sign": {
               "title": "Sign",
               "minimum": 0,
               "maximum": 1,
               "type": "integer"
            }
         },
         "required": [
            "namespace",
            "identifier"
         ]
      },
```


(continues on next page)

(continued from previous page)

```
    "StmtData": {
       "title": "StmtData",
       "description": "Data for one statement supporting an edge",
       "type": "object",
       "properties": {
          "stmt_type": {
             "title": "Stmt Type",
             "type": "string"
          },
          "evidence_count": {
             "title": "Evidence Count",
             "minimum": 1,
             "type": "integer"
          },
          "stmt_hash": {
             "title": "Stmt Hash",
             "anyOf": [
                {
                   "type": "integer"
                },
                {
                   "type": "string",
                   "minLength": 1,
                   "maxLength": 2083,
                   "format": "uri"
                }
             ]
          },
          "source_counts": {
             "title": "Source Counts",
             "type": "object",
             "additionalProperties": {
                "type": "integer"
             }
          },
          "belief": {
             "title": "Belief",
             "minimum": 0.0,
             "maximum": 1.0,
             "type": "number"
          },
          "curated": {
             "title": "Curated",
             "type": "boolean"
          },
          "english": {
             "title": "English",
             "type": "string"
          },
          "weight": {
             "title": "Weight",
             "type": "number"
```

(continues on next page)

(continued from previous page)


```
            },
            "residue": {
               "title": "Residue",
               "default": "",
               "type": "string"
            },
            "position": {
               "title": "Position",
               "default": "",
               "type": "string"
            },
            "initial_sign": {
               "title": "Initial Sign",
               "minimum": 0,
               "maximum": 1,
               "type": "integer"
            },
            "db_url_hash": {
               "title": "Db Url Hash",
               "type": "string"
            }
         },
         "required": [
            "stmt_type",
            "evidence_count",
            "stmt_hash",
            "source_counts",
            "belief",
            "curated",
            "english",
            "db_url_hash"
         ]
      },
      "EdgeDataByHash": {
         "title": "EdgeDataByHash",
         "description": "Data for one single edge, with data keyed by hash",
         "type": "object",
         "properties": {
            "edge": {
               "title": "Edge",
               "type": "array",
               "items": {
                  "$ref": "#/definitions/Node"
               }
            },
            "stmts": {
               "title": "Stmts",
               "type": "object",
               "additionalProperties": {
                  "$ref": "#/definitions/StmtData"
               }
            },
```


(continues on next page)

(continued from previous page)


```
            "belief": {
               "title": "Belief",
               "type": "number"
            },
            "weight": {
               "title": "Weight",
               "type": "number"
            },
            "db_url_edge": {
               "title": "Db Url Edge",
               "type": "string"
            },
            "url_by_type": {
               "title": "Url By Type",
               "type": "object",
               "additionalProperties": {
                  "type": "string"
               }
            }
         },
         "required": [
            "edge",
            "stmts",
            "belief",
            "weight",
            "db_url_edge",
            "url_by_type"
         ]
      }
   }
}
```


**Fields**

- *input_nodes (List[indra_network_search.data_models.__init__.Node])*
- *not_in_graph (List[indra_network_search.data_models.__init__.Node])*
- *available_nodes (List[indra_network_search.data_models.__init__.Node])*
- *edges (List[indra_network_search.data_models.__init__.EdgeDataByHash])*

**field available_nodes: List[*indra_network_search.data_models.__init__.Node*] [Required]**

**field edges: List[*indra_network_search.data_models.__init__.EdgeDataByHash*] [Required]**

**field input_nodes: List[*indra_network_search.data_models.__init__.Node*] [Required]**

**field not_in_graph: List[*indra_network_search.data_models.__init__.Node*] [Required]**

indra_network_search.data_models.__init__.**basemodel_in_iterable**(*basemodel*, *iterable*, *any_item*, *exclude=None*)

Test if a basemodel object is part of a collection

**Parameters**

- **basemodel** (BaseModel) – A BaseModel to test membership in iterable for
- **iterable** (Iterable) – An iterable that contains objects to test for equality with basemodel
- **any_item** (bool) – If True, use any() when testing collections for equality, otherwise use all(), i.e. the collections must match exactly
- **exclude** (Optional[Set[str]]) – A set of field names to exclude from the basemodels

**Return type** bool

**Returns** True if basemodel is found in the collection

indra_network_search.data_models.__init__.**basemodels_equal**(*basemodel*, *other_basemodel*, *any_item*, *exclude=None*)

Wrapper to test two basemodels for equality, can exclude irrelevant keys

**Parameters**

- **basemodel** (BaseModel) – BaseModel to test against other_basemodel
- **other_basemodel** (BaseModel) – BaseModel to test against basemodel
- **any_item** (bool) – If True, use any() when testing collections for equality, otherwise use all(), i.e. the collections must match exactly
- **exclude** (Optional[Set[str]]) – A set of field names to exclude from the basemodels

**Return type** bool

**Returns** True if the two models are equal

## 2.3 Rest Models (indra_network_search.data_models.rest_models)

Contains return models from the rest api

**pydantic model** indra_network_search.data_models.rest_models.**Health**

Health status


```
{
   "title": "Health",
   "description": "Health status",
   "type": "object",
   "properties": {
      "status": {
         "title": "Status",
         "enum": [
            "booting",
            "available"
         ],
         "type": "string"
      }
   },
   "required": [
```


(continues on next page)

(continued from previous page)

```
      "status"
   ]
}
```

**Fields**

- *status (typing_extensions.Literal[booting, available])*

**field status: typing_extensions.Literal[booting, available] [Required]**

**pydantic model** indra_network_search.data_models.rest_models.**ServerStatus**

Status with more detail than health

```
{
   "title": "ServerStatus",
   "description": "Status with more detail than health",
   "type": "object",
   "properties": {
      "unsigned_nodes": {
         "title": "Unsigned Nodes",
         "type": "integer"
      },
      "signed_nodes": {
         "title": "Signed Nodes",
         "type": "integer"
      },
      "unsigned_edges": {
         "title": "Unsigned Edges",
         "type": "integer"
      },
      "signed_edges": {
         "title": "Signed Edges",
         "type": "integer"
      },
      "graph_date": {
         "title": "Graph Date",
         "type": "string",
         "format": "date"
      },
      "status": {
         "title": "Status",
         "enum": [
            "booting",
            "available"
         ],
         "type": "string"
      }
   },
   "required": [
      "status"
   ]
}
```

**Fields**

- *unsigned_nodes (Optional[int])*
- *signed_nodes (Optional[int])*
- *unsigned_edges (Optional[int])*
- *signed_edges (Optional[int])*
- *graph_date (Optional[datetime.date])*
- *status (typing_extensions.Literal[booting, available])*

**field graph_date: Optional[datetime.date] = None**

**field signed_edges: Optional[int] = None**

**field signed_nodes: Optional[int] = None**

**field status: typing_extensions.Literal[booting, available] [Required]**

**field unsigned_edges: Optional[int] = None**

**field unsigned_nodes: Optional[int] = None**

## 2.4 Pathfinding (indra_network_search.pathfinding.pathfinding)

Pathfinding algorithms local to this repository

indra_network_search.pathfinding.pathfinding.**direct_multi_interactors**(*graph*, *nodes*, *downstream*, *allowed_ns=None*, *stmt_types=None*, *source_filter=None*, *max_results=50*, *hash_blacklist=None*, *node_blacklist=None*, *belief_cutoff=0.0*, *curated_db_only=False*)

Find up- or downstream common nodes from a list of nodes

**Parameters**

- **graph** (DiGraph) – The graph to look in
- **nodes** (List[Union[str, Tuple[str, int]]]) – The starting nodes
- **downstream** (bool) – If True, look for shared targets, otherwise look for shared regulators
- **allowed_ns** (Optional[List[str]]) – A list of allowed node namespaces
- **stmt_types** (Optional[List[str]]) – A list of allowed statemtent types
- **source_filter** (Optional[List[str]]) – A list of valid sources
- **max_results** (int) – The maximum number of results to return
- **hash_blacklist** (Optional[Set[int]]) – A list of hashes to blacklist
- **node_blacklist** (Optional[List[str]]) – A list of node names to blacklist
- **belief_cutoff** (float) – If set, an edge will not be allowed if all its supporting statements have belief scores below this value

- **curated_db_only** (bool) – If True, only allow edge that have support from curated databases

**Return type** Iterator[Union[str, Tuple[str, int]]]

**Returns** An Iterator of the resulting nodes

indra_network_search.pathfinding.pathfinding.**get_subgraph_edges**(*graph*, *nodes*)

Get the subgraph connecting the provided nodes

**Parameters**

- **graph** (DiGraph) – Graph to look for in and out edges in
- **nodes** (List[Dict[str, str]]) – List of dicts of Node instances to look for neighbors in

**Return type** Iterator[Tuple[str, str]]

**Returns** A dict keyed by each of the input node names that were present in the graph. For each node, two lists are provided for in-edges and out-edges respectively

indra_network_search.pathfinding.pathfinding.**shared_interactors**(*graph*, *source*, *target*, *allowed_ns=None*, *stmt_types=None*, *source_filter=None*, *max_results=50*, *regulators=False*, *sign=None*, *hash_blacklist=None*, *node_blacklist=None*, *belief_cutoff=0.0*, *curated_db_only=False*)

Get shared regulators or targets and filter them based on sign

Closely resembles get_st and get_sr from depmap_analysis.scripts.depmap_script_expl_funcs

**Parameters**

- **graph** (DiGraph) – The graph to perform the search in
- **source** (Union[str, Tuple[str, int]]) – Node to look for common up- or downstream neighbors from with target
- **target** (Union[str, Tuple[str, int]]) – Node to look for common up- or downstream neighbors from with source
- **allowed_ns** (Optional[List[str]]) – If provided, filter common nodes to these namespaces
- **stmt_types** (Optional[List[str]]) – If provided, filter the statements in the supporting edges to these statement types
- **source_filter** (Optional[List[str]]) – If provided, filter the statements in the supporting edges to those with these sources
- **max_results** (int) – The maximum number of results to return
- **regulators** (bool) – If True, do shared regulator search (upstream), otherwise do shared target search (downstream). Default False.
- **sign** (Optional[int]) –

  **If provided, match edges to sign:**

  – positive: edges must have same sign

- negative: edges must have opposite sign

- **hash_blacklist** (Optional[Set[str]]) – A list of hashes to exclude from the edges
- **node_blacklist** (Optional[List[str]]) – A list of node names to exclude
- **belief_cutoff** (float) – Exclude statements that are below the cutoff. Default: 0.0 (no cutoff)
- **curated_db_only** (bool) – If True, exclude statements in edge support that only have readers in their sources. Default: False.

**Return type** Iterator[Tuple[List[Union[str, Tuple[str, int]]], List[Union[str, Tuple[str, int]]]]]

**Returns** An iterator of regulators or targets to source and target as edges

indra_network_search.pathfinding.pathfinding.**shared_parents**(*source_ns*, *source_id*, *target_ns*, *target_id*, *immediate_only=False*, *is_a_part_of=None*, *max_paths=50*)

Get shared ontological parents of source and target

**Parameters**

- **source_ns** (str) – Namespace of source
- **source_id** (str) – Identifier of source
- **target_ns** (str) – Namespace of target
- **target_id** (str) – Identifier of target
- **immediate_only** (bool) – Determines if all or just the immediate parents should be returned. Default: False, i.e. all parents.
- **is_a_part_of** (Optional[Set[str]]) – If provided, the parents must be in this set of ids. The set is assumed to be valid ontology labels (see ontology.label()).
- **max_paths** (int) – Maximum number of results to return. Default: 50.

**Return type** Iterator[Tuple[str, Any, Any, str]]

**Returns** An iterator of parents to source and target, each parent being a tuple of (name, namespace, id, lookup URL)

## 2.5 Utility Functions (indra_network_search.util)

### 2.5.1 Util (indra_network_search.util.curation_cache)

class indra_network_search.util.curation_cache.**CurationCache**

Gathers curations of wrong statements from the indra DB

**get_all_hashes**()

Get all hashes present in the cache as a set

**Return type** Set[int]

**Returns** The set of all hashes stored in the cache

## 2.6 Query Classes (indra_network_search.query)

This file contains the Query classes mapping incoming rest queries to different algorithms used in the search api.

**class** indra_network_search.query.**BreadthFirstSearchQuery**(*query*, *hash_blacklist=None*)

Check queries that will use the bfs_search algorithm

**alg_options**()

Match arguments of bfs_search from query

**Return type** Dict[str, Any]

**Returns** The argument to provide bfs_search

**mesh_options**(*graph=None*)

Get mesh options for bfs_search

**Parameters** **graph** (Optional[DiGraph]) – The graph

**Return type** Dict[str, Union[Set, bool, Callable]]

**Returns** The mesh option for bfs_search

**options**

alias of indra_network_search.data_models.BreadthFirstSearchOptions


```
{
   "title": "BreadthFirstSearchOptions",
   "description": "Arguments for indra.explanation.pathfinding.bfs_search",
   "type": "object",
   "properties": {
      "source_node": {
         "title": "Source Node",
         "anyOf": [
            {
               "type": "string"
            },
            {
               "type": "array",
               "items": [
                  {
                     "type": "string"
                  },
                  {
                     "type": "integer"
                  }
               ]
            }
         ]
      },
      "reverse": {
         "title": "Reverse",
         "default": false,
         "type": "boolean"
      },
      "depth_limit": {
         "title": "Depth Limit",
```


(continues on next page)

(continued from previous page)


```
            "default": 2,
            "type": "integer"
        },
        "path_limit": {
            "title": "Path Limit",
            "type": "integer"
        },
        "max_per_node": {
            "title": "Max Per Node",
            "default": 5,
            "type": "integer"
        },
        "node_filter": {
            "title": "Node Filter",
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "node_blacklist": {
            "title": "Node Blacklist",
            "type": "array",
            "items": {
                "type": "string"
            },
            "uniqueItems": true
        },
        "terminal_ns": {
            "title": "Terminal Ns",
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "sign": {
            "title": "Sign",
            "type": "integer"
        },
        "max_memory": {
            "title": "Max Memory",
            "default": 536870912,
            "type": "integer"
        },
        "hashes": {
            "title": "Hashes",
            "type": "array",
            "items": {
                "type": "integer"
            }
        },
        "strict_mesh_id_filtering": {
            "title": "Strict Mesh Id Filtering",
```


(continues on next page)

(continued from previous page)

```
            "default": false,
            "type": "boolean"
        }
    },
    "required": [
        "source_node"
    ]
}
```

**Fields**

- source_node (Union[str, Tuple[str, int]])
- reverse (Optional[bool])
- depth_limit (Optional[int])
- path_limit (Optional[int])
- max_per_node (Optional[int])
- node_filter (Optional[List[str]])
- node_blacklist (Optional[Set[str]])
- terminal_ns (Optional[List[str]])
- sign (Optional[int])
- max_memory (Optional[int])
- hashes (Optional[List[int]])
- allow_edge (Optional[Callable[[networkx.classes.digraph.DiGraph, Union[str, Tuple[str, int]], Union[str, Tuple[str, int]]], bool]])
- edge_filter (Optional[Callable[[networkx.classes.digraph.DiGraph, Union[str, Tuple[str, int]], Union[str, Tuple[str, int]]], bool]])
- strict_mesh_id_filtering (Optional[bool])

**class** indra_network_search.query.**DijkstraQuery**(*query*, *hash_blacklist=None*)

Check queries that will use the open_dijkstra_search algorithm

**alg_options**()

Match arguments of open_dijkstra_search from query

**Return type** Dict[str, Any]

**Returns** A dict with arguments for open_dijkstra_search

**mesh_options**(*graph=None*)

Produces mesh arguments matching open_dijkstra_search from query

**Return type** Dict[str, Union[Set, bool, Callable]]

**Returns** The mesh options for open_dijkstra_query

**options**

alias of indra_network_search.data_models.DijkstraOptions

```
{
   "title": "DijkstraOptions",
   "description": "Arguments for open_dijkstra_search",
   "type": "object",
   "properties": {
      "start": {
         "title": "Start",
         "anyOf": [
            {
               "type": "string"
            },
            {
               "type": "array",
               "items": [
                  {
                     "type": "string"
                  },
                  {
                     "type": "integer"
                  }
               ]
            }
         ]
      },
      "reverse": {
         "title": "Reverse",
         "default": false,
         "type": "boolean"
      },
      "path_limit": {
         "title": "Path Limit",
         "type": "integer"
      },
      "hashes": {
         "title": "Hashes",
         "type": "array",
         "items": {
            "type": "integer"
         }
      },
      "ignore_nodes": {
         "title": "Ignore Nodes",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "ignore_edges": {
         "title": "Ignore Edges",
         "type": "array",
         "items": {
            "type": "array",
            "items": [
```


(continues on next page)

(continued from previous page)


```
                {
                   "type": "string"
                },
                {
                   "type": "string"
                }
             ]
          }
       },
       "terminal_ns": {
          "title": "Terminal Ns",
          "type": "array",
          "items": {
             "type": "string"
          }
       },
       "weight": {
          "title": "Weight",
          "type": "string"
       },
       "const_c": {
          "title": "Const C",
          "default": 1,
          "type": "integer"
       },
       "const_tk": {
          "title": "Const Tk",
          "default": 10,
          "type": "integer"
       }
    },
    "required": [
       "start"
    ]
}
```


**Fields**

- start (Union[str, Tuple[str, int]])
- reverse (Optional[bool])
- path_limit (Optional[int])
- hashes (Optional[List[int]])
- ignore_nodes (Optional[List[str]])
- ignore_edges (Optional[List[Tuple[str, str]]])
- terminal_ns (Optional[List[str]])
- weight (Optional[str])
- ref_counts_function (Optional[Callable])
- const_c (Optional[int])

- const_tk (Optional[int])

class indra_network_search.query.**MultiInteractorsQuery**(*rest_query*)

Check queries that will use pathfinding.direct_multi_interactors

**options**

alias of indra_network_search.data_models.MultiInteractorsOptions


```
{
   "title": "MultiInteractorsOptions",
   "description": "Multi interactors options",
   "type": "object",
   "properties": {
      "nodes": {
         "title": "Nodes",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "downstream": {
         "title": "Downstream",
         "type": "boolean"
      },
      "allowed_ns": {
         "title": "Allowed Ns",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "stmt_types": {
         "title": "Stmt Types",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "source_filter": {
         "title": "Source Filter",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "max_results": {
         "title": "Max Results",
         "default": 50,
         "type": "integer"
      },
      "hash_blacklist": {
         "title": "Hash Blacklist",
         "type": "array",
         "items": {
```


(continues on next page)

(continued from previous page)


```
            "type": "integer"
         },
         "uniqueItems": true
      },
      "node_blacklist": {
         "title": "Node Blacklist",
         "type": "array",
         "items": {
            "type": "string"
         }
      },
      "belief_cutoff": {
         "title": "Belief Cutoff",
         "default": 0.0,
         "type": "number"
      },
      "curated_db_only": {
         "title": "Curated Db Only",
         "default": false,
         "type": "boolean"
      }
   },
   "required": [
      "nodes",
      "downstream"
   ]
}
```


**Fields**

- nodes (List[str])
- downstream (bool)
- allowed_ns (Optional[List[str]])
- stmt_types (Optional[List[str]])
- source_filter (Optional[List[str]])
- max_results (int)
- hash_blacklist (Optional[Set[int]])
- node_blacklist (Optional[List[str]])
- belief_cutoff (float)
- curated_db_only (bool)

**result_options**()

Return a dict with options for the MultiInteractorsResultManager

**Return type** Dict[str, Any]

**Returns** A dict with the options for the MultiInteractorsResultManager

**run_options**()

Return options needed for direct_multi_interactors

**Return type** Dict[str, Any]

**Returns** The options needed for direct_multi_interactors

**class** indra_network_search.query.**OntologyQuery**(*query*)

Check queries that will use shared_parents

**alg_options**()

Match arguments of shared_parents from query

**Return type** Dict[str, Any]

**Returns** A dict with arguments for shared_parents

**options**

alias of indra_network_search.data_models.OntologyOptions


```
{
   "title": "OntologyOptions",
   "description": "Arguments for indra_network_search.pathfinding.shared_parents
↪",
   "type": "object",
   "properties": {
      "source_ns": {
         "title": "Source Ns",
         "type": "string"
      },
      "source_id": {
         "title": "Source Id",
         "type": "string"
      },
      "target_ns": {
         "title": "Target Ns",
         "type": "string"
      },
      "target_id": {
         "title": "Target Id",
         "type": "string"
      },
      "max_paths": {
         "title": "Max Paths",
         "default": 50,
         "type": "integer"
      },
      "immediate_only": {
         "title": "Immediate Only",
         "default": false,
         "type": "boolean"
      },
      "is_a_part_of": {
         "title": "Is A Part Of",
         "type": "array",
         "items": {
            "type": "string"
         },
         "uniqueItems": true
```


(continues on next page)

(continued from previous page)

```
        }
    },
    "required": [
        "source_ns",
        "source_id",
        "target_ns",
        "target_id"
    ]
}
```

**Fields**

- source_ns (str)
- source_id (str)
- target_ns (str)
- target_id (str)
- max_paths (int)
- immediate_only (Optional[bool])
- is_a_part_of (Optional[Set[str]])

**result_options**()

Provide args to OntologyResultManager in result_handler

**Return type** Dict

**Returns** The arguments for the Ontology Result manger

**run_options**(*graph=None*)

Check query options and return them for shared_parents

**Parameters** **graph** (Optional[DiGraph]) – The graph used in the search. Must contains node attributes 'ns' and 'id' for each node.

**Return type** Dict[str, Any]

**Returns** The options for shared_parents

**class** indra_network_search.query.**PathQuery**(*query*, *hash_blacklist*)

Parent Class for ShortestSimplePaths, Dijkstra and BreadthFirstSearch

**alg_options**()

Returns the options for the algorithm used, excl mesh options

**Return type** Dict[str, Any]

**mesh_options**(*graph=None*)

Return algorithm specific mesh options

**Return type** Dict[str, Any]

**result_options**()

Provide args to corresponding result class in result_handler

**Return type** Dict

**Returns** Options for the PathResult class

**run_options**(*graph=None*)
Combines all options to one dict that can be sent to algorithm

**Return type** Dict[str, Any]

**class** indra_network_search.query.**Query**(*query*)
Parent class to all Query classes

The Query classes are helpers that make sure the methods of the IndraNetworkSearchAPI receive the data needed from a NetworkSearchQuery or other Rest query.

**alg_options**()
Returns the options for the algorithm used

**Return type** Dict[str, Any]

**api_options**()
These options are used when IndraNetworkSearchAPI handles the query

The options here impact decisions on which extra search algorithms to include and which graph to pick

**Return type** Dict[str, Any]

**Returns** A dict of ApiOptions

**result_options**()
Provide args to corresponding result class in result_handler

**Return type** Dict

**run_options**(*graph=None*)
Combines all options to one dict that can be sent to algorithm

**Return type** Dict[str, Any]

**class** indra_network_search.query.**SharedRegulatorsQuery**(*query*, *hash_blacklist=None*)
Check queries that will use shared_interactors(regulators=True)

**class** indra_network_search.query.**SharedTargetsQuery**(*query*, *hash_blacklist=None*)
Check queries that will use shared_interactors(regulators=False)

**class** indra_network_search.query.**ShortestSimplePathsQuery**(*query*, *hash_blacklist=None*)
Check queries that will use the shortest_simple_paths algorithm

**alg_options**()
Match arguments of shortest_simple_paths from query

**Return type** Dict[str, Any]

**Returns** A dict with arguments for shortest_simple_paths

**mesh_options**(*graph=None*)
Match input to shortest_simple_paths

**Return type** Dict[str, Union[Set, int, bool, Callable]]

**Returns** The mesh options for shortest_simple_paths

**options**
alias of indra_network_search.data_models.ShortestSimplePathOptions

```
{
   "title": "ShortestSimplePathOptions",
   "description": "Arguments for indra.explanation.pathfinding.shortest_simple_
↪paths",
```

(continues on next page)

(continued from previous page)


```
   "type": "object",
   "properties": {
      "source": {
         "title": "Source",
         "anyOf": [
            {
               "type": "string"
            },
            {
               "type": "array",
               "items": [
                  {
                     "type": "string"
                  },
                  {
                     "type": "integer"
                  }
               ]
            }
         ]
      },
      "target": {
         "title": "Target",
         "anyOf": [
            {
               "type": "string"
            },
            {
               "type": "array",
               "items": [
                  {
                     "type": "string"
                  },
                  {
                     "type": "integer"
                  }
               ]
            }
         ]
      },
      "weight": {
         "title": "Weight",
         "type": "string"
      },
      "ignore_nodes": {
         "title": "Ignore Nodes",
         "type": "array",
         "items": {
            "type": "string"
         },
         "uniqueItems": true
      },
```


(continues on next page)

(continued from previous page)


```
      "ignore_edges": {
         "title": "Ignore Edges",
         "type": "array",
         "items": {
            "type": "array",
            "items": [
               {
                  "type": "string"
               },
               {
                  "type": "string"
               }
            ]
         },
         "uniqueItems": true
      },
      "hashes": {
         "title": "Hashes",
         "type": "array",
         "items": {
            "type": "integer"
         }
      },
      "strict_mesh_id_filtering": {
         "title": "Strict Mesh Id Filtering",
         "default": false,
         "type": "boolean"
      },
      "const_c": {
         "title": "Const C",
         "default": 1,
         "type": "integer"
      },
      "const_tk": {
         "title": "Const Tk",
         "default": 10,
         "type": "integer"
      }
   },
   "required": [
      "source",
      "target"
   ]
}
```


**Fields**

- source (Union[str, Tuple[str, int]])
- target (Union[str, Tuple[str, int]])
- weight (Optional[str])
- ignore_nodes (Optional[Set[str]])

- ignore_edges (Optional[Set[Tuple[str, str]]])
- hashes (Optional[List[int]])
- ref_counts_function (Optional[Callable])
- strict_mesh_id_filtering (Optional[bool])
- const_c (Optional[int])
- const_tk (Optional[int])

**class** indra_network_search.query.**SubgraphQuery**(*query*)

Check queries that gets the subgraph

**alg_options**(*graph*)

Match arguments of get_subgraph_edges

**Parameters** **graph** (DiGraph) – The graph the search will be performed with

**Return type** Dict[str, List[Node]]

**Returns** A dict with the arguments for get_subgraph_edges

**options**

alias of indra_network_search.data_models.SubgraphOptions


```
{
   "title": "SubgraphOptions",
   "description": "Argument for indra_network_search.pathfinding.get_subgraph_
↪edges",
   "type": "object",
   "properties": {
      "nodes": {
         "title": "Nodes",
         "type": "array",
         "items": {
            "$ref": "#/definitions/Node"
         }
      }
   },
   "required": [
      "nodes"
   ],
   "definitions": {
      "Node": {
         "title": "Node",
         "description": "Data for a node",
         "type": "object",
         "properties": {
            "name": {
               "title": "Name",
               "minLength": 1,
               "type": "string"
            },
            "namespace": {
               "title": "Namespace",
               "minLength": 1,
```


(continues on next page)

(continued from previous page)

```
                "type": "string"
            },
            "identifier": {
                "title": "Identifier",
                "minLength": 1,
                "type": "string"
            },
            "lookup": {
                "title": "Lookup",
                "minLength": 1,
                "type": "string"
            },
            "sign": {
                "title": "Sign",
                "minimum": 0,
                "maximum": 1,
                "type": "integer"
            }
        },
        "required": [
            "namespace",
            "identifier"
        ]
    }
}
```

**Fields**

- nodes (List[indra_network_search.data_models.Node])

**result_options**()

Return options needed for SubgraphResultManager

**Return type** Dict[str, Any]

**Returns** A dict with options for the SubgraphResultManager

**run_options**(*graph*)

Return options needed for get_subgraph_edges

**Parameters** **graph** (DiGraph) – The graph the search will be performed with

**Return type** Dict[str, Any]

**Returns** A dict with the arguments for get_subgraph_edges

**class** indra_network_search.query.**UIQuery**(*query*, *hash_blacklist=None*)

Parent Class for all possible queries that come from the web UI

**alg_options**()

Returns the options for the algorithm used

**Return type** Dict[str, Any]

**result_options**()

Provide args to corresponding result class in result_handler

**Return type** Dict

**run_options**(*graph=None*)
Combines all options to one dict that can be sent to algorithm

**Return type** Dict[str, Any]

## 2.7 Query Handler (indra_network_search.query_handler)

The QueryHandler's job is to act as a middle layer between the network search functionalities and the REST API that receives queries.

**class** indra_network_search.query_handler.**QueryHandler**(*rest_query*, *curation_cache=<indra_network_search.util.curation_cache.Curatio object>*)
Maps queries from the REST API to a method of the IndraNetworkSearchAPI

The QueryHandler's job is to act as a middle layer between the methods of the IndraNetworkSearchAPI and the REST API. It figures out which queries are eligible from the input query.

**get_queries**()
Returns a dict of all eligible queries

**Return type** Dict[str, *UIQuery*]

**Returns** A dict containing the instances of the eligible Query classes as {query name: Query instance}

## 2.8 Rest API (indra_network_search.rest_api)

This documentation is here for completeness and for code documentation purposes. For a better description of the Rest API, read the API docs.

The IndraNetworkSearch REST API

indra_network_search.rest_api.**get_prefix_autocomplete**(*prefix=Query(Ellipsis)*, *max_res=Query(100)*)
Get the case-insensitive node names with (ns, id) starting in prefix

**Parameters**

- **prefix** (str) – The prefix of a node name to search for. Note: for prefixes of 1 and 2 characters, only exact matches are returned. For 3+ characters, prefix matching is done. If the prefix contains ':', an namespace:id search is done.
- **max_res** (int) – The top ranked (by node degree) results will be returned, cut off at this many results.

**Return type** List[Tuple[str, str, str]]

**Returns** A list of tuples of (node name, namespace, identifier)

indra_network_search.rest_api.**get_xrefs**(*ns*, *id*)
Get all cross-refs given a namespace and ID

**Parameters**

- **ns** (str) – The namespace of the entity to find cross-refs for
- **id** (str) – The identifier of the entity to find cross-regs for

**Return type** List[List[str]]

**Returns** A list of tuples containing namespace, identifier, lookup url to identifiers.org

async indra_network_search.rest_api.**health**()
Returns health status

**Returns**

**Return type** *Health*

indra_network_search.rest_api.**node_id_in_graph**(*db_name=Query(Ellipsis)*, *db_id=Query(Ellipsis)*)
Check if a node by provided db name and db id exists

**Parameters**

- **db_name** (str) – The database name, e.g. hgnc, chebi or up
- **db_id** (str) – The identifier for the entity in the given database, e.g. 11018

**Return type** Optional[Node]

**Returns** When a match is found, the full information of the node is returned

indra_network_search.rest_api.**node_name_in_graph**(*node_name=Query(Ellipsis)*)
Check if node by provided name (case sensitive) exists in graph

**Parameters** **node_name** (str) – The name of the node to check

**Return type** Optional[Node]

**Returns** When a match is found, the full information of the node is returned

indra_network_search.rest_api.**query**(*search_query*, *background_tasks*)
Interface with IndraNetworkSearchAPI.handle_query

**Parameters** **search_query** (NetworkSearchQuery) – Query to the NetworkSearchQuery

**Returns**

**Return type** *Results*

async indra_network_search.rest_api.**server_status**()
Returns the status of the server and some info about the loaded graphs

**Returns**

indra_network_search.rest_api.**sub_graph**(*search_query*)
Interface with IndraNetworkSearchAPI.handle_subgraph_query

**Parameters** **search_query** (SubgraphRestQuery) – Query to for IndraNetworkSearchAPI.handle_subgraph_query

**Returns**

**Return type** *SubgraphResults*

## 2.9 Rest API Utilities (indra_network_search.rest_util)

Utility functions for the Network Search API and Rest API

indra_network_search.rest_util.**check_existence_and_date_s3**(*query_hash*)
Check if a query hash has corresponding result and query json on S3

**Parameters** **query_hash** (Union[int, str]) – The query hash to check

**Return type** Dict[str, str]

**Returns** Dict with S3 key for query and corresponding result, if they exist

indra_network_search.rest_util.**dump_query_json_to_s3**(*query_hash*, *json_obj*, *get_url=False*)
Dump a query json to S3

**Parameters**

- **query_hash** (Union[str, int]) – The query hash associated with the query
- **json_obj** (Dict) – The json object to upload
- **get_url** (bool) – If True return the S3 url of the object. Default: False.

**Return type** Optional[str]

**Returns** Optionally return the S3 url of the json file

indra_network_search.rest_util.**dump_query_result_to_s3**(*filename*, *json_obj*, *get_url=False*)
Dump a result or query json from the network search to S3

**Parameters**

- **filename** (str) – The filename to use
- **json_obj** (Dict) – The json object to upload
- **get_url** (bool) – If True return the S3 url of the object. Default: False.

**Return type** Optional[str]

**Returns** Optionally return the S3 url of the json file

indra_network_search.rest_util.**dump_result_json_to_s3**(*query_hash*, *json_obj*, *get_url=False*)
Dump a result json to S3

**Parameters**

- **query_hash** (Union[str, int]) – The query hash associated with the result
- **json_obj** (Dict) – The json object to upload
- **get_url** (bool) – If True return the S3 url of the object. Default: False.

**Return type** Optional[str]

**Returns** Optionally return the S3 url of the json file

indra_network_search.rest_util.**get_default_args**(*func*)
Returns the default args of a function as a dictionary

Returns a dictionary of {arg: default} of the arguments that have default values. Arguments without default values and ***kwargs* type arguments are excluded.

Code copied from: https://stackoverflow.com/a/12627202/10478812

**Parameters** **func** (Callable) – Function to find default arguments for

**Return type** Dict[str, Any]

**Returns** A dictionary with the default values keyed by argument name

indra_network_search.rest_util.**get_mandatory_args**(*func*)
Returns the mandatory args for a function as a set

Returns the set of arguments names of a functions that are mandatory, i.e. does not have a default value. ***kwargs* type arguments are ignored.

**Parameters** **func** (Callable) – Function to find mandatory arguments for

**Return type** Set[str]

**Returns** The of mandatory arguments

indra_network_search.rest_util.**get_query_hash**(*query_json*, *ignore_keys=None*)
Create an FNV-1a 32-bit hash from the query json

**Parameters**

- **query_json** (Dict) – A json compatible query dict
- **ignore_keys** (Union[Set, List, None]) – A list or set of keys to ignore in the query_json. By default, no keys are ignored. Default: None.

**Return type** int

**Returns** An FNV-1a 32-bit hash of the query json ignoring the keys in ignore_keys

indra_network_search.rest_util.**get_queryable_stmt_types**()
Return Statement class names that can be used for querying.

indra_network_search.rest_util.**get_s3_client**(*unsigned=True*)
Return a boto3 S3 client with optional unsigned config.

**Parameters** **unsigned** (*Optional[bool]*) – If True, the client will be using unsigned mode in which public resources can be accessed without credentials. Default: True

**Returns** A client object to AWS S3.

**Return type** botocore.client.S3

indra_network_search.rest_util.**is_context_weighted**(*mesh_id_list*, *strict_filtering*)
Return True if context weighted

**Parameters**

- **mesh_id_list** (List[str]) – A list of mesh ids
- **strict_filtering** (bool) – whether to run strict context filtering or not

**Return type** bool

**Returns** True for the combination of mesh ids being present and unstrict filtering, otherwise False

indra_network_search.rest_util.**is_weighted**(*weighted*, *mesh_ids*, *strict_mesh_filtering*)
Return True if the combination is either weighted or context weighted

**Parameters**

- **weighted** (bool) – If a query is weighted or not
- **mesh_ids** (List[str]) – A list of mesh ids
- **strict_mesh_filtering** (*bool*) – whether to run strict context filtering or not

**Return type** bool

**Returns** True if the combination is either weighted or context weighted

indra_network_search.rest_util.**list_chunk_gen**(*lst*, *size=1000*)
Given list, generate chunks <= size

indra_network_search.rest_util.**load_indra_graph**(*unsigned_graph=True*, *unsigned_multi_graph=False*, *sign_edge_graph=False*, *sign_node_graph=True*, *use_cache=False*)
Return a tuple of graphs to be used in the network search API

**Parameters**

- **unsigned_graph** (bool) – Load the latest unsigned graph. Default: True.
- **unsigned_multi_graph** (bool) – Load the latest unsigned multi graph. Default: False.
- **sign_node_graph** (bool) – Load the latest signed node graph. Default: True.
- **sign_edge_graph** (bool) – Load the latest signed edge graph. Default: False.
- **use_cache** (bool) – If True, try to load files from the designated local cache

**Returns**

**Returns, as a tuple:**

- unsigned graph
- unsigned multi graph
- signed edge graph
- signed node graph

If a graph was not chosen to be loaded or wasn't found, None will be returned in its place in the tuple.

**Return type** Tuple[nx.DiGraph, nx.MultiDiGraph, nx.MultiDiGraph, nx.DiGraph]

## 2.10 Result Handlers (indra_network_search.result_handler)

Handles the aggregation of results from the IndraNetworkSearchAPI

The result manager deals with things like:

- Stopping path iteration when timeout is reached
- Keeping count of number of paths returned
- Filtering results when it's not done in the algorithm

**class** indra_network_search.result_handler.**BreadthFirstSearchResultManager**(*path_generator*, *graph*, *filter_options*, *source*, *target*, *reverse*, *timeout=30*)
Handles results from bfs_search

class indra_network_search.result_handler.**DijkstraResultManager**(*path_generator*, *graph*, *filter_options*, *source*, *target*, *reverse*, *timeout=30*, *hash_blacklist=None*)

Handles results from open_dijkstra_search

class indra_network_search.result_handler.**MultiInteractorsResultManager**(*path_generator*, *graph*, *input_nodes*, *filter_options*, *downstream*, *timeout=30*)

Handles results from *pathfinding.direct_multi_interactors*

**get_results**()

Execute the result assembly with the loaded generator

**Return type** MultiInteractorsResults

**Returns** Results for direct_multi_interactors as a BaseModel

class indra_network_search.result_handler.**OntologyResultManager**(*path_generator*, *graph*, *filter_options*, *source*, *target*)

Handles results from shared_parents

**get_results**()

Execute the result assembly with the loaded generator

**Return type** BaseModel

**Returns** Results for shared_parents as a BaseModel

class indra_network_search.result_handler.**SharedInteractorsResultManager**(*path_generator*, *filter_options*, *graph*, *source*, *target*, *is_targets_query*)

Handles results from shared_interactors, both up and downstream

downstream is True for shared targets and False for shared regulators

**get_results**()

Execute the result assembly with the loaded generator

**Return type** SharedInteractorsResults

**Returns** Results for shared_interactors as a BaseModel

class indra_network_search.result_handler.**ShortestSimplePathsResultManager**(*path_generator*, *graph*, *filter_options*, *source*, *target*, *timeout=30*, *hash_blacklist=None*)

Handles results from shortest_simple_paths

class indra_network_search.result_handler.**SubgraphResultManager**(*path_generator*, *graph*, *filter_options*, *original_nodes*, *nodes_in_graph*, *not_in_graph*, *ev_limit=10*)

Handles results from get_subgraph_edges

**get_results**()

Execute the result assembly with the loaded generator

**Return type** SubgraphResults

**Returns** Results for get_subgraph_edges as a BaseModel

## 2.11 Search API (indra_network_search.search_api)

The INDRA Network Search API

This class represents an API that executes search queries

Queries for specific searches are found in indra_network_search.query

**class** indra_network_search.search_api.**IndraNetworkSearchAPI**(*unsigned_graph*, *signed_node_graph*)
The search API class

**breadth_first_search**(*breadth_first_search_query*, *is_signed*)
Get results from running bfs_search

**Parameters**

- **breadth_first_search_query** (*BreadthFirstSearchQuery*) – The input query holding the options to the algorithm
- **is_signed** (bool) – Whether the query is signed or not

**Return type** *BreadthFirstSearchResultManager*

**Returns** An instance of the BreadthFirstSearchResultManager, holding results from running bfs_search

**dijkstra**(*dijkstra_query*, *is_signed*)
Get results from running open_dijkstra_search

**Parameters**

- **dijkstra_query** (*DijkstraQuery*) – The input query holding options for open_dijkstra_search and DijkstraResultManager
- **is_signed** (bool) – Whether the query is signed or not

**Return type** *DijkstraResultManager*

**Returns** An instance of the DijkstraResultManager, holding results from running open_dijkstra_search

**get_graph**(*signed=False*)
Returns the graph used for pathfinding

**Return type** DiGraph

**get_node**(*node_name*)
Returns an instance of a Node matching the input name, if it exists

**Parameters** **node_name** (str) – Name of node to look up

**Return type** Optional[Node]

**Returns** An instance of a node corresponding to the input name

**handle_multi_interactors_query**(*multi_interactors_rest_query*)
Interface with pathfinding.direct_multi_interactors

**Parameters** **multi_interactors_rest_query** (MultiInteractorsRestQuery) – The input query holding options for direct multi interactors

**Return type** MultiInteractorsResults

**Returns** Results holding node and edge data

**handle_query**(*rest_query*)
Handle a NetworkSearchQuery and return the corresponding results

**Parameters** **rest_query** (NetworkSearchQuery) – A query from the rest api with all relevant information to execute path queries and other related queries. See available queries in indra_network_search.query

**Return type** Results

**Returns** A model containing all results from the query. For more information about the data structure, see indra_network_search.data_models

**handle_subgraph_query**(*subgraph_rest_query*)
Interface for handling queries to get_subgraph_edges

**Parameters** **subgraph_rest_query** (SubgraphRestQuery) – A rest query containing the list of nodes needed for get_subgraph_edges

**Return type** SubgraphResults

**Returns** The data put together from the results of get_subgraph_edges

**multi_interactors_query**(*query*)
Run direct_multi_interactors and return the result manager

**Parameters** **query** (*MultiInteractorsQuery*) – An instance of MultiInteractorsQuery, that interfaces with the algorithm and the result manager

**Return type** *MultiInteractorsResultManager*

**Returns** A MultiInteractorsResultManager holding the results of running direct_multi_interactors

**path_query**(*path_query*, *is_signed*)
Wrapper for the mutually exclusive path queries

**Parameters**

- **path_query** (Union[*Query*, *PathQuery*]) – An instance of a Query or PathQuery
- **is_signed** (bool) – Signifies if the path query is signed or not

**Return type** ResultManager

**Returns** A ResultManager with the path generator loaded before get_results() have been executed for the first time

**shared_parents**(*ontology_query*)
Get results from running shared_parents

**Parameters** **ontology_query** (*OntologyQuery*) – The input query holding options for shared_parents

**Return type** *OntologyResultManager*

**Returns** An instance of the OntologyResultManager, holding results from running shared_parents

**shared_regulators**(*shared_regulators_query*, *is_signed*)
Get results from running shared_interactors looking for regulators

**Parameters**

- **shared_regulators_query** (*SharedRegulatorsQuery*) – The input query holding options for shared_interactors
- **is_signed** (bool) – Whether the query is signed or not

**Return type** *SharedInteractorsResultManager*

**Returns** An instance of the SharedInteractorsResultManager, holding results from running shared_interactors looking for regulators

**shared_targets**(*shared_targets_query*, *is_signed*)
Get results from running shared_interactors looking for targets

**Parameters**

- **shared_targets_query** (*SharedTargetsQuery*) – The input query holding options for shared_interactors
- **is_signed** (bool) – Whether the query is signed or not

**Return type** *SharedInteractorsResultManager*

**Returns** An instance of the SharedInteractorsResultManager, holding results from running shared_interactors looking for targets

**shortest_simple_paths**(*shortest_simple_paths_query*, *is_signed*)
Get results from running shortest_simple_paths

**Parameters**

- **shortest_simple_paths_query** (*ShortestSimplePathsQuery*) – The input query holding the options to the algorithm
- **is_signed** (bool) – Whether the query is signed or not

**Return type** *ShortestSimplePathsResultManager*

**Returns** An instance of the ShortestSimplePathsResultManager, holding results from running shortest_simple_paths_query

**subgraph_query**(*query*)
Get results from running get_subgraph_edges

**Parameters** **query** (*SubgraphQuery*) – The input query holding the options for get_subgraph_edges

**Return type** *SubgraphResultManager*

**Returns** An instance of the SubgraphResultManager, holding results from running get_subgraph_edges

CHAPTER

THREE

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## i

# INDEX

## H



## I

## O



## P



## Q



## R

## S

## T



## U

## W



## Z