

---

**indra\_world**

**INDRA team**

**Jan 29, 2022**



# CONTENTS

<b>1</b>	<b>World Modelers INDRA service stack</b>	<b>1</b>
1.1	Using the INDRA World API	1
1.2	Setting up the INDRA World API locally	1
1.3	INDRA assemblies on S3	1
1.3.1	The corpus index	2
1.3.2	Structure of each corpus	2
<b>2</b>	<b>INDRA World Modules Reference</b>	<b>3</b>
2.1	Knowledge Sources ( <code>indra_world.sources</code> )	3
2.1.1	Eidos ( <code>indra_world.sources.eidos</code> )	3
2.1.2	Hume ( <code>indra_world.sources.hume</code> )	6
2.1.3	Sofia ( <code>indra_world.sources.sofia</code> )	7
2.1.4	CWMS ( <code>indra_world.sources.cwms</code> )	11
2.1.5	DART ( <code>indra_world.sources.dart</code> )	13
2.2	Knowledge assembly modules ( <code>indra_world.assembly</code> )	15
2.2.1	Statement preprocessing ( <code>indra_world.assembly.preprocess</code> )	15
2.2.2	Assembly operations ( <code>indra_world.assembly.operations</code> )	15
2.2.3	Matches functions ( <code>indra_world.assembly.matches</code> )	16
2.2.4	Refinement functions ( <code>indra_world.assembly.refinement</code> )	17
2.2.5	Incremental Assembler ( <code>indra_world.assembly.incremental_assembler</code> )	18
2.2.6	Statistics ( <code>indra_world.assembly.stats</code> )	20
2.3	Ontology Module ( <code>indra_world.ontology</code> )	20
2.3.1	World Ontology ( <code>indra_world.ontology.ontology</code> )	20
2.4	Belief Engine ( <code>indra_world.belief</code> )	21
2.5	Output assemblers ( <code>indra_world.assemblers</code> )	21
2.5.1	CAG Assembler ( <code>indra_world.assemblers.cag</code> )	21
2.5.2	Figaro Assembler ( <code>indra_world.assemblers.figaro</code> )	23
2.5.3	TSV Assembler ( <code>indra_world.assemblers.tsv</code> )	23
2.6	Indra World Service ( <code>indra_world.service</code> )	23
2.6.1	INDRA World Database ( <code>indra_world.service.db</code> )	23
2.6.2	Service controller ( <code>indra_world.service.controller</code> )	24
2.6.3	REST API ( <code>indra_world.service.app</code> )	24
2.6.4	Corpus manager ( <code>indra_world.service.corpus_manager</code> )	27
2.7	INDRA World Dashboard ( <code>indra_world.dashboard</code> )	28
<b>3</b>	<b>Indices and tables</b>	<b>29</b>
	<b>Python Module Index</b>	<b>31</b>
	<b>Index</b>	<b>33</b>



## WORLD MODELERS INDRA SERVICE STACK

### 1.1 Using the INDRA World API

The API is deployed and documented at [wm.indra.bio](https://wm.indra.bio).

### 1.2 Setting up the INDRA World API locally

These instructions describe setting up and using the INDRA service stack for World Modelers applications.

First, you need to build the INDRA World Docker image as follows:

```
git clone https://github.com/indralab/indra_world.git
cd indra_world/docker
docker build --tag indra_world:latest .
```

Then, in the same folder, do:

```
docker-compose up -d
```

to run the INDRA world service as well as an associated postgres container with the relational database used by the service. The *docker-compose* file reads secret configuration values for accessing various resources from two files: *indra\_world.env* and *indra\_world\_db.env*. These files are not part of the public code and need to be added manually.

### 1.3 INDRA assemblies on S3

Access to the INDRA-assembled corpora requires credentials to the shared World Modelers S3 bucket “world-modelers”. Each INDRA-assembled corpus is available within this bucket, under the “indra\_models” key base. Each corpus is identified by a string identifier.

### 1.3.1 The corpus index

The list of corpora can be obtained either using S3's list objects function or by reading the index.csv file which is maintained by INDRA. This index is a comma separated values text file which contains one row for each corpus. Each row's first element is a corpus identifier, and the second element is the UTC date-time at which the corpus was uploaded to S3. An example row in this file looks as follows

test1_newlines,2020-05-08-22-34-29
------------------------------------

where test1\_newlines is the corpus identifier and 2020-05-08-22-34-29 is the upload date-time.

### 1.3.2 Structure of each corpus

Within the world-modelers bucket, under the indra\_models key base, files for each corpus are organized under a subkey equivalent to the corpus identifier, for instance, all the files for the test1\_newlines corpus are under the indra\_models/test1\_newlines/ key base. The list of files for each corpus are as follows

- *statements.json*: a JSON dump of assembled INDRA Statements. As of May 2020, each statement's JSON representation is on a separate line in this file. Any corpus uploaded before that has a standard JSON structure. This is the main file that CauseMos needs to ingest for UI interaction.
- *raw\_statements.json*: a JSON dump of raw INDRA Statements. This file is typically not needed in downstream usage, however, the INDRA curation service needs to have access to it for internal assembly tasks.
- *metadata.json*: a JSON file containing key-value pairs that describe the corpus. The standard keys in this file are as follows:
  - *corpus\_id*: the ID of the corpus (redundant with the corresponding entry in the index).
  - *description*: a human-readable description of how the corpus was obtained.
  - *display\_name*: a human-readable display name for the corpus.
  - *readers*: a list of the names of the reading systems from which statements were obtained in the corpus.
  - *assembly*: a dictionary identifying attributes of the assembly process with the following keys:
    - \* *level*: the level of resolution used to assemble the corpus (e.g., "location\_and\_time").
    - \* *grounding\_threshold*: the threshold (if any) which was used to filter statements by grounding score (e.g., 0.7)
  - *num\_statements*: the number of assembled INDRA Statements in the corpus ( i.e., statements.json).
  - *num\_documents*: the number of documents that were read by readers to produce the statements that were assembled.

Note that any of these keys may be missing if unavailable, for instance, in the case of old uploads.

- *curations.json*: a JSON file which persists curations as collected by INDRA. This is the basis of surfacing reader-specific curations in the download\_curation endpoint (see above).

## INDRA WORLD MODULES REFERENCE

### 2.1 Knowledge Sources (`indra_world.sources`)

#### 2.1.1 Eidos (`indra_world.sources.eidos`)

##### API (`indra_world.sources.eidos.api`)

`indra_world.sources.eidos.api.process_json(json_dict, grounding_ns=None, extract_filter=None, grounding_mode=None)`

Return an EidosProcessor by processing a Eidos JSON-LD dict.

##### Parameters

- **json\_dict** (*dict*) – The JSON-LD dict to be processed.
- **grounding\_ns** (*Optional[list]*) – A list of name spaces for which INDRA should represent groundings, when given. If not specified or None, all grounding name spaces are propagated. If an empty list, no groundings are propagated. Example: ['UN', 'WM'], Default: None
- **extract\_filter** (*Optional[list]*) – A list of relation types to extract. Valid values in the list are 'influence', 'association', 'event'. If not given, all relation types are extracted. This argument can be used if, for instance, only Influence statements are of interest. Default: None
- **grounding\_mode** (*Optional[str]*) – Selects whether 'flat' or 'compositional' groundings should be extracted. Default: 'flat'.

**Returns** `ep` – A EidosProcessor containing the extracted INDRA Statements in its statements attribute.

**Return type** EidosProcessor

`indra_world.sources.eidos.api.process_json_file(file_name, grounding_ns=None, extract_filter=None, grounding_mode='flat')`

Return an EidosProcessor by processing the given Eidos JSON-LD file.

This function is useful if the output from Eidos is saved as a file and needs to be processed.

##### Parameters

- **file\_name** (*str*) – The name of the JSON-LD file to be processed.
- **grounding\_ns** (*Optional[list]*) – A list of name spaces for which INDRA should represent groundings, when given. If not specified or None, all grounding name spaces are propagated. If an empty list, no groundings are propagated. Example: ['UN', 'WM'], Default: None

- **extract\_filter** (*Optional*[*list*]) – A list of relation types to extract. Valid values in the list are ‘influence’, ‘association’, ‘event’. If not given, all relation types are extracted. This argument can be used if, for instance, only Influence statements are of interest. Default: None
- **grounding\_mode** (*Optional*[*str*]) – Selects whether ‘flat’ or ‘compositional’ groundings should be extracted. Default: ‘flat’.

**Returns** **ep** – A EidosProcessor containing the extracted INDRA Statements in its statements attribute.

**Return type** EidosProcessor

`indra_world.sources.eidos.api.process_json_str(json_str, grounding_ns=None, extract_filter=None, grounding_mode='flat')`

Return an EidosProcessor by processing the Eidos JSON-LD string.

#### Parameters

- **json\_str** (*str*) – The JSON-LD string to be processed.
- **grounding\_ns** (*Optional*[*list*]) – A list of name spaces for which INDRA should represent groundings, when given. If not specified or None, all grounding name spaces are propagated. If an empty list, no groundings are propagated. Example: [‘UN’, ‘WM’], Default: None
- **extract\_filter** (*Optional*[*list*]) – A list of relation types to extract. Valid values in the list are ‘influence’, ‘association’, ‘event’. If not given, all relation types are extracted. This argument can be used if, for instance, only Influence statements are of interest. Default: None
- **grounding\_mode** (*Optional*[*str*]) – Selects whether ‘flat’ or ‘compositional’ groundings should be extracted. Default: ‘flat’.

**Returns** **ep** – A EidosProcessor containing the extracted INDRA Statements in its statements attribute.

**Return type** EidosProcessor

`indra_world.sources.eidos.api.process_text(text, save_json='eidos_output.json', webservice=None, grounding_ns=None, extract_filter=None, grounding_mode='flat')`

Return an EidosProcessor by processing the given text.

This constructs a reader object via Java and extracts mentions from the text. It then serializes the mentions into JSON and processes the result with `process_json`.

#### Parameters

- **text** (*str*) – The text to be processed.
- **save\_json** (*Optional*[*str*]) – The name of a file in which to dump the JSON output of Eidos.
- **webservice** (*Optional*[*str*]) – An Eidos reader web service URL to send the request to. If None, the reading is assumed to be done with the Eidos JAR rather than via a web service. Default: None
- **grounding\_ns** (*Optional*[*list*]) – A list of name spaces for which INDRA should represent groundings, when given. If not specified or None, all grounding name spaces are propagated. If an empty list, no groundings are propagated. Example: [‘UN’, ‘WM’], Default: None



- **extract\_filter** (*Optional*[*list*]) – A list of relation types to extract. Valid values in the list are ‘influence’, ‘association’, ‘event’. If not given, all relation types are extracted. This argument can be used if, for instance, only Influence statements are of interest. Default: None
- **grounding\_mode** (*Optional*[*str*]) – Selects whether ‘flat’ or ‘compositional’ groundings should be extracted. Default: ‘flat’.

**Returns** *ep* – An EidosProcessor containing the extracted INDRA Statements in its statements attribute.

**Return type** EidosProcessor

`indra_world.sources.eidos.api.reground_texts(texts, ont_yaml, webservice=None, topk=10, filter=True, is_canonicalized=True)`

Return grounding for concept texts given an ontology.

#### Parameters

- **texts** (*list*[*str*]) – A list of concept texts to ground.
- **ont\_yaml** (*str*) – A serialized YAML string representing the ontology.
- **webservice** (*Optional*[*str*]) – The address where the Eidos web service is running, e.g., <http://localhost:9000>. If None, a local Eidos JAR is invoked via pyjnius. Default: None
- **topk** (*Optional*[*int*]) – The number of top scoring groundings to return. Default: 10
- **is\_canonicalized** (*Optional*[*bool*]) – If True, the texts are assumed to be canonicalized. If False, Eidos will canonicalize the texts which yields much better groundings but is slower. Default: False
- **filter** (*Optional*[*bool*]) – If True, Eidos filters the ontology to remove determiners from examples and other similar operations. Should typically be set to True. Default: True

**Returns** A list of the top k scored groundings for each text in the list.

**Return type** *list*[*list*]

### Client(`indra_world.sources.eidos.client`)

`indra_world.sources.eidos.client.grounding_dict_to_list(groundings)`

Transform the webservice response into a flat list.

`indra_world.sources.eidos.client.reground_texts(texts, ont_yaml, webservice, topk=10, is_canonicalized=False, filter=True, cache_path=None)`

Ground concept texts given an ontology with an Eidos web service.

#### Parameters

- **texts** (*list*[*str*]) – A list of concept texts to ground.
- **ont\_yaml** (*str*) – A serialized YAML string representing the ontology.
- **webservice** (*str*) – The address where the Eidos web service is running, e.g., <http://localhost:9000>.
- **topk** (*Optional*[*int*]) – The number of top scoring groundings to return. Default: 10
- **is\_canonicalized** (*Optional*[*bool*]) – If True, the texts are assumed to be canonicalized. If False, Eidos will canonicalize the texts which yields much better groundings but is slower. Default: False

- **filter** (*Optional[bool]*) – If True, Eidos filters the ontology to remove determiners from examples and other similar operations. Should typically be set to True. Default: True

**Returns** A JSON dict of the results from the Eidos webservice.

**Return type** `dict`

### Migration Table Processor (`indra_world.sources.eidos.migration_table_processor`)

### Processor (`indra_world.sources.eidos.processor`)

**class** `indra_world.sources.eidos.processor.EidosProcessorCompositional(json_dict, grounding_ns)`

Bases: `indra_world.sources.eidos.processor.EidosWorldProcessor`

**get\_groundings**(*entity*)

Return groundings as db\_refs for an entity.

**class** `indra_world.sources.eidos.processor.EidosWorldProcessor(json_dict, grounding_ns)`

Bases: `indra.sources.eidos.processor.EidosProcessor`

**geo\_context\_from\_ref**(*ref*)

Return a ref context object given a location reference entry.

**get\_groundings**(*entity*)

Return groundings as db\_refs for an entity.

**time\_context\_from\_ref**(*timex*)

Return a time context object given a timex reference entry.

`indra_world.sources.eidos.processor.ref_context_from_geoloc(geoloc)`

Return a RefContext object given a geoloc entry.

`indra_world.sources.eidos.processor.time_context_from_timex(timex)`

Return a TimeContext object given a timex entry.

## 2.1.2 Hume (`indra_world.sources.hume`)

Hume is a general purpose reading system developed by BBN.

Currently, INDRA can process JSON-LD files produced by Hume. When available, the API will be extended with access to the reader as a service.

### API (`indra_world.sources.hume.api`)

`indra_world.sources.hume.api.process_jsonld(jsonld, extract_filter=None, grounding_mode=None)`

Process a JSON-LD string in the new format to extract Statements.

#### Parameters

- **jsonld** (*dict*) – The JSON-LD object to be processed.
- **extract\_filter** (*Optional[list]*) – A list of relation types to extract. Valid values in the list are ‘influence’ and ‘event’. If not given, all relation types are extracted. This argument can be used if, for instance, only Influence statements are of interest. Default: None
- **grounding\_mode** (*Optional[str]*) – Selects whether ‘flat’ or ‘compositional’ groundings should be extracted. Default: ‘flat’.

**Returns** A HumeProcessor instance, which contains a list of INDRA Statements as its statements attribute.

**Return type** indra\_world.sources.hume.HumeProcessor

indra\_world.sources.hume.api.**process\_jsonld\_file**(fname, extract\_filter=None, grounding\_mode='flat')  
Process a JSON-LD file in the new format to extract Statements.

**Parameters**

- **fname** (*str*) – The path to the JSON-LD file to be processed.
- **extract\_filter** (*Optional[list]*) – A list of relation types to extract. Valid values in the list are 'influence' and 'event'. If not given, all relation types are extracted. This argument can be used if, for instance, only Influence statements are of interest. Default: None
- **grounding\_mode** (*Optional[str]*) – Selects whether 'flat' or 'compositional' groundings should be extracted. Default: 'flat'.

**Returns** A HumeProcessor instance, which contains a list of INDRA Statements as its statements attribute.

**Return type** indra\_world.sources.hume.HumeProcessor

### Processor (indra\_world.sources.hume.processor)

**class** indra\_world.sources.hume.processor.HumeJsonLdProcessor(json\_dict)

This processor extracts INDRA Statements from Hume JSON-LD output.

**Parameters** json\_dict (*dict*) – A JSON dictionary containing the Hume extractions in JSON-LD format.

**tree**

The objectpath Tree object representing the extractions.

**Type** objectpath.Tree

**statements**

A list of INDRA Statements that were extracted by the processor.

**Type** list[indra.statements.Statement]

**class** indra\_world.sources.hume.processor.HumeJsonLdProcessorCompositional(json\_dict)

### 2.1.3 Sofia (indra\_world.sources.sofia)

Sofia is a general purpose natural language processing system developed at UPitt and CMU by N. Miskov et al.

#### API (indra\_world.sources.sofia.api)

indra\_world.sources.sofia.api.**process\_json**(json\_obj, extract\_filter=None, grounding\_mode=None)

Return processor by processing a JSON object returned by Sofia.

**Parameters**

- **json\_obj** (*json*) – A JSON object containing extractions from Sofia.
- **extract\_filter** (*Optional[list]*) – A list of relation types to extract. Valid values in the list are 'influence' and 'event'. If not given, all relation types are extracted. This argument can be used if, for instance, only Influence statements are of interest. Default: None

- **grounding\_mode** (*Optional* [*str*]) – Selects whether ‘flat’ or ‘compositional’ groundings should be extracted. Default: ‘flat’.

**Returns** **sp** – A SofiaProcessor object which has a list of extracted INDRA Statements as its statements attribute.

**Return type** indra.sources.sofia.processor.SofiaProcessor

indra\_world.sources.sofia.api.**process\_json\_file**(*fname*, *extract\_filter=None*, *grounding\_mode='flat'*)

Return processor by processing a JSON file produced by Sofia.

**Parameters**

- **fname** (*str*) – The name of the JSON file to process
- **extract\_filter** (*Optional* [*list*]) – A list of relation types to extract. Valid values in the list are ‘influence’ and ‘event’. If not given, all relation types are extracted. This argument can be used if, for instance, only Influence statements are of interest. Default: None
- **grounding\_mode** (*Optional* [*str*]) – Selects whether ‘flat’ or ‘compositional’ groundings should be extracted. Default: ‘flat’.

**Returns** A SofiaProcessor object which has a list of extracted INDRA Statements as its statements attribute.

**Return type** indra.sources.sofia.processor.SofiaProcessor

indra\_world.sources.sofia.api.**process\_table**(*fname*, *extract\_filter=None*, *grounding\_mode='flat'*)

Return processor by processing a given sheet of a spreadsheet file.

**Parameters**

- **fname** (*str*) – The name of the Excel file (typically .xlsx extension) to process
- **extract\_filter** (*Optional* [*list*]) – A list of relation types to extract. Valid values in the list are ‘influence’ and ‘event’. If not given, all relation types are extracted. This argument can be used if, for instance, only Influence statements are of interest. Default: None
- **grounding\_mode** (*Optional* [*str*]) – Selects whether ‘flat’ or ‘compositional’ groundings should be extracted. Default: ‘flat’.

**Returns** **sp** – A SofiaProcessor object which has a list of extracted INDRA Statements as its statements attribute.

**Return type** indra.sources.sofia.processor.SofiaProcessor

indra\_world.sources.sofia.api.**process\_text**(*text*, *out\_file='sofia\_output.json'*, *auth=None*,  
*extract\_filter=None*, *grounding\_mode='flat'*)

Return processor by processing text given as a string.

**Parameters**

- **text** (*str*) – A string containing the text to be processed with Sofia.
- **out\_file** (*Optional* [*str*]) – The path to a file to save the reader’s output into. Default: sofia\_output.json
- **auth** (*Optional* [*list*]) – A username/password pair for the Sofia web service. If not given, the SOFIA\_USERNAME and SOFIA\_PASSWORD values are loaded from either the INDRA config or the environment.
- **extract\_filter** (*Optional* [*list*]) – A list of relation types to extract. Valid values in the list are ‘influence’ and ‘event’. If not given, all relation types are extracted. This argument can be used if, for instance, only Influence statements are of interest. Default: None

- **grounding\_mode** (*Optional[str]*) – Selects whether ‘flat’ or ‘compositional’ groundings should be extracted. Default: ‘flat’.

**Returns** `sp` – A `SofiaProcessor` object which has a list of extracted INDRA Statements as its `statements` attribute. If the API did not process the text, `None` is returned.

**Return type** `indra.sources.sofia.processor.SofiaProcessor`

## Processor (`indra_world.sources.sofia.processor`)

**class** `indra_world.sources.sofia.processor.SofiaExcelProcessor`(*relation\_rows, event\_rows, entity\_rows, \*\*kwargs*)

Bases: `indra_world.sources.sofia.processor.SofiaProcessor`

An Excel processor extracting statements from reading done by Sofia

**extract\_events**(*event\_rows, relation\_rows*)

Extract Event statements of a Sofia document in Excel format

**Parameters**

- **event\_rows** (*Iterator[Tuple[Cell, ...]]*) – The extracted event data from an Excel document
- **relation\_rows** (*Iterator[Tuple[Cell, ...]]*) – The extracted relation data from an Excel document

**Return type** `None`

**extract\_relations**(*relation\_rows*)

Extract Influence statements from relation events

**Parameters** **relation\_rows** (*Iterator[Tuple[Cell, ...]]*) – The extracted relation data from an Excel document

**Return type** `None`

**process\_events**(*event\_rows*)

Process the events of Sofia document extractions in Excel format

**Parameters** **event\_rows** (*Iterator[Tuple[Cell, ...]]*) – The extracted event data from an Excel document

**Returns** A dict of event keyed by their event index

**Return type** `processed_event_dict`

**class** `indra_world.sources.sofia.processor.SofiaJsonProcessor`(*jd, \*\*kwargs*)

Bases: `indra_world.sources.sofia.processor.SofiaProcessor`

A JSON processor extracting statements from reading done by Sofia

**extract\_events**(*jd*)

Extract Event statements from a Sofia document extraction

**Parameters** **jd** (*Dict[str, str]*) – A dictionary with document extractions

**Return type** `None`

**extract\_relations**(*jd*)

Extract Influence statements from a Sofia document extraction

**Parameters** **jd** (*Dict[str, Any]*) – A dictionary with document extractions

**Return type** `None`

**process\_entities(*jd*)**

Process the entities of a Sofia document extraction

**Parameters** **jd** (**Dict**[**str**, **Any**]) – The extracted data from a document

**Returns** A dictionary of processed entities keyed by their entity index

**Return type** **ent\_dict**

**process\_events(*jd*)**

Process the event of a Sofia document extraction

**Parameters** **jd** (**Dict**[**str**, **Any**]) – The extracted data from a document

**Returns** A dictionary of processed events keyed by their event index

**Return type** **processed\_event\_dict**

**class** **indra\_world.sources.sofia.processor.SofiaProcessor**(*score\_cutoff=None*,  
*grounding\_mode='flat'*)

Bases: **object**

A processor extracting statements from reading done by Sofia

**get\_compositional\_grounding(*event\_entry*)**

Get the compositional grounding for an event

**Parameters** **event\_entry** (**Dict**[**str**, **str**]) – The event to get the compositional grounding for

**Returns** The name of the grounding and a tuple of representing the compositional grounding

**Return type** **grounding**

**get\_event(*event\_entry*)**

Get an Event with the pre-set grounding mode

The grounding mode is set at initialization of the class and is stored in the attribute *grounding\_mode*.

**Parameters** **event\_entry** (**Dict**[**str**, **str**]) – The event to process

**Returns** An Event statement

**Return type** **event**

**get\_event\_compositional(*event\_entry*)**

Get an Event with compositional grounding

**Parameters** **event\_entry** (**Dict**[**str**, **str**]) – The event to process

**Returns** An Event statement

**Return type** **event**

**get\_event\_flat(*event\_entry*)**

Get an Event with flattened grounding

**Parameters** **event\_entry** (**Dict**[**str**, **str**]) – The event to process

**Returns** An Event statement

**Return type** **event**

**get\_meaningful\_events(*raw\_event\_dict*)**

Process events by extracting polarity

**Parameters** **raw\_event\_dict** (**Dict**[**str**, **Any**]) – A dict of events to process

**Returns** A dict of event data

**Return type** processed\_event\_dict

**get\_relation\_events**(*rel\_dict*)

Get a list of the event indices associated with a causal entry

**Parameters** *rel\_dict* (*Dict[str, str]*) – A causal entry to extract event indices from

**Returns** A list of event indices

**Return type** relation\_events

## 2.1.4 CWMS (indra\_world.sources.cwms)

CWMS is a variant of the TRIPS system. It is a general purpose natural language understanding system with applications in world modeling. For more information, see: <http://trips.ihmc.us/parser/cgi/cwmsreader>

### API (indra\_world.sources.cwms.api)

*indra\_world.sources.cwms.api.process\_ekb*(*ekb\_str*, *extract\_filter=None*, *grounding\_mode='flat'*)

Processes an EKB string produced by CWMS.

#### Parameters

- **ekb\_str** (*str*) – EKB string to process
- **extract\_filter** (*Optional[list]*) – A list of relation types to extract. Valid values in the list are ‘influence’, ‘association’, ‘event’ and ‘migration’. If not given, only Influences are extracted since processing other relation types can be time consuming. This argument can be used if the extraction of other relation types such as Events are also of interest.
- **grounding\_mode** (*Optional[str]*) – Selects whether ‘flat’ or ‘compositional’ groundings should be extracted. Default: ‘flat’.

**Returns** *cp* – A CWMSProcessor, which contains a list of INDRA statements in its statements attribute.

**Return type** *indra.sources.cwms.CWMSProcessor*

*indra\_world.sources.cwms.api.process\_ekb\_file*(*fname*, *extract\_filter=None*, *grounding\_mode='flat'*)

Processes an EKB file produced by CWMS.

#### Parameters

- **fname** (*str*) – Path to the EKB file to process.
- **extract\_filter** (*Optional[list]*) – A list of relation types to extract. Valid values in the list are ‘influence’, ‘association’, ‘event’ and ‘migration’. If not given, only Influences are extracted since processing other relation types can be time consuming. This argument can be used if the extraction of other relation types such as Events are also of interest.
- **grounding\_mode** (*Optional[str]*) – Selects whether ‘flat’ or ‘compositional’ groundings should be extracted. Default: ‘flat’.

**Returns** *cp* – A CWMSProcessor, which contains a list of INDRA statements in its statements attribute.

**Return type** *indra.sources.cwms.CWMSProcessor*

*indra\_world.sources.cwms.api.process\_text*(*text*, *save\_xml='cwms\_output.xml'*, *extract\_filter=None*, *grounding\_mode='flat'*)

Processes text using the CWMS web service.

**Parameters**

- **text** (*str*) – Text to process
- **save\_xml** (*Optional[str]*) – A file name in which to dump the output from CWMS. Default: `cwms_output.xml`
- **extract\_filter** (*Optional[list]*) – A list of relation types to extract. Valid values in the list are ‘influence’, ‘association’, ‘event’ and ‘migration’. If not given, only Influences are extracted since processing other relation types can be time consuming. This argument can be used if the extraction of other relation types such as Events are also of interest.
- **grounding\_mode** (*Optional[str]*) – Selects whether ‘flat’ or ‘compositional’ groundings should be extracted. Default: ‘flat’.

**Returns** `cp` – A CWMSProcessor, which contains a list of INDRA statements in its `statements` attribute.

**Return type** `indra.sources.cwms.CWMSProcessor`

**Processor** (`indra_world.sources.cwms.processor`)

**exception** `indra_world.sources.cwms.processor.CWMSError`

Bases: `Exception`

**class** `indra_world.sources.cwms.processor.CWMSProcessor(xml_string)`

Bases: `object`

The CWMSProcessor currently extracts causal relationships between terms (nouns) in EKB. In the future, this processor can be extended to extract other types of relations, or to extract relations involving events.

For more details on the TRIPS EKB XML format, see <http://trips.ihmc.us/parser/cgi/drum>

**Parameters** `xml_string` (*str*) – A TRIPS extraction knowledge base (EKB) in XML format as a string.

**tree**

An ElementTree object representation of the TRIPS EKB XML.

**Type** `xml.etree.ElementTree.Element`

**doc\_id**

Document ID

**Type** `str`

**statements**

A list of INDRA Statements that were extracted from the EKB.

**Type** `list[indra.statements.Statement]`

**sentences**

The list of all sentences in the EKB with their IDs

**Type** `dict[str: str]`

**paragraphs**

The list of all paragraphs in the EKB with their IDs

**Type** `dict[str: str]`

**par\_to\_sec**

A map from paragraph IDs to their associated section types



```

    Type dict[str: str]

event_from_event(event_term)
    Return an Event from an EVENT element in the EKB.

extract_causal_relations()
    Extract Influence Statements from the EKB.

extract_events()
    Extract standalone Events from the EKB.

influence_from_event(event)
    Return an Influence from an EVENT element in the EKB.

influence_from_relation(relation)
    Return an Influence from a CC element in the EKB.

migration_from_event(event_term)
    Return a Migration event from an EVENT element in the EKB.

class indra_world.sources.cwms.processor.CWMSProcessorCompositional(xml_string)
    Bases: indra\_world.sources.cwms.processor.CWMSProcessor

```

## 2.1.5 DART ([indra\\_world.sources.dart](#))

**API** ([indra\\_world.sources.dart.api](#))

**Client** ([indra\\_world.sources.dart.client](#))

A client for accessing reader output from the DART system.

```

class indra_world.sources.dart.client.DartClient(storage_mode='web', dart_url=None,
                                                dart_uname=None, dart_pwd=None,
                                                local_storage=None)

```

A client for the DART web service with optional local storage.

### Parameters

- **storage\_mode** (*Optional[str]*) – If *web*, the configured DART URL and credentials are used to communicate with the DART web service. If *local*, a local storage is used to access and store reader outputs.
- **dart\_url** (*Optional[str]*) – The DART service URL. If given, it overrides the DART\_WM\_URL configuration value.
- **dart\_uname** (*Optional[str]*) – The DART service user name. If given, it overrides the DART\_WM\_USERNAME configuration value.
- **dart\_pwd** (*Optional[str]*) – The DART service password. If given, it overrides the DART\_WM\_PASSWORD configuration value.
- **local\_storage** (*Optional[str]*) – A path that points to a folder for local storage. If the storage\_mode is *web*, this local\_storage is used as a local cache. If the storage\_mode is *local*, it is used as the primary location to access reader outputs. If given, it overrides the INDRA\_WM\_CACHE configuration value.

```

cache_record(record, overwrite=False)
    Download and cache a given record in local storage.

```

**Parameters** **record** (*dict*) – A DART record.

**cache\_records**(*records*, *overwrite=False*)

Download and cache a list of records in local storage.

**Parameters** **records** (*list[dict]*) – A list of DART records.

**download\_output**(*storage\_key*)

Return content from the DART web service based on its storage key.

**Parameters** **storage\_key** (*str*) – A DART storage key.

**Returns** The content corresponding to the storage key.

**Return type** *str*

**get\_local\_storage\_path**(*record*)

Return the local storage path for a DART record.

**get\_output\_from\_record**(*record*)

Return reader output corresponding to a single record.

**Parameters** **record** (*dict*) – A single DART record.

**Returns** The reader output corresponding to the given record.

**Return type** *str*

**get\_outputs\_from\_records**(*records*)

Return reader outputs corresponding to a list of records.

**Parameters** **records** (*list of dict*) – A list of records returned from the reader output query.

**Returns** A two-level dict of reader output keyed by reader and then document id.

**Return type** *dict(str, dict)*

**get\_reader\_output\_records**(*readers=None*, *versions=None*, *document\_ids=None*, *timestamp=None*)

Return reader output metadata records by querying the DART API

**Query json structure:** {"readers": ["MyAwesomeTool", "SomeOtherAwesomeTool"], "versions": ["3.1.4", "1.3.3.7"], "document\_ids": ["qwerty1234", "poiuyt0987"], "timestamp": {"before": "yyyy-mm-dd"|"yyyy-mm-dd hh:mm:ss", "after": "yyyy-mm-dd"|"yyyy-mm-dd hh:mm:ss", "on": "yyyy-mm-dd"}}

**Parameters**

- **readers** (*list*) – A list of reader names
- **versions** (*list*) – A list of versions to match with the reader name(s)
- **document\_ids** (*list*) – A list of document identifiers
- **timestamp** (*dict("on"/"before"/"after", str)*) – The timestamp string must of format "yyyy-mm-dd" or "yyyy-mm-dd hh:mm:ss" (only for "before" and "after").

**Returns** The JSON payload of the response from the DART API

**Return type** *dict*

**get\_reader\_versions**(*reader*)

Return the available versions for a given reader.

**indra\_world.sources.dart.client.prioritize\_records**(*records*, *priorities=None*)

Return unique records per reader and document prioritizing by version.

**Parameters**

- **records** (*list of dict*) – A list of records returned from the reader output query.
- **priorities** (*dict of list*) – A dict keyed by reader names (e.g., cwms, eidos) with values representing reader versions in decreasing order of priority.

**Returns records** – A list of records that are unique per reader and document, picked by version priority when multiple records exist for the same reader and document.

**Return type** list of dict

## 2.2 Knowledge assembly modules (indra\_world.assembly)

### 2.2.1 Statement preprocessing (indra\_world.assembly.preprocess)

indra\_world.assembly.preprocess.**preprocess\_statements**(*raw\_statements, steps*)

Run a preprocessing pipeline on raw statements.

**Parameters**

- **raw\_statements** (*List[Statement]*) – A list of INDRA Statements to preprocess.
- **steps** (*List[Dict[str, Any]]*) – A list of AssemblyPipeline steps that define the steps of preprocessing.

**Returns** A list of preprocessed INDRA Statements.

**Return type** preprocessed\_statements

### 2.2.2 Assembly operations (indra\_world.assembly.operations)

class indra\_world.assembly.operations.**CompositionalRefinementFilter**(*ontology, nproc=None*)  
**extend**(*stmts\_by\_hash*)

Extend the initial data structures with a set of new statements.

**Parameters** **stmts\_by\_hash** (*dict[int, indra.statements.Statement]*) – A dict of statements keyed by their hashes.

**get\_related**(*stmt, possibly\_related=None, direction='less\_specific'*)

Return a set of statement hashes that a given statement is potentially related to.

**Parameters**

- **stmt** (*indra.statements.Statement*) – The INDRA statement whose potential relations we want to filter.
- **possibly\_related** (*set or None*) – A set of statement hashes that this statement is potentially related to, as determined by some other filter. If this parameter is a set (including an empty set), this function should return a subset of it (intuitively, this filter can only further eliminate some of the potentially related hashes that were previously determined to be potential relations). If this argument is None, the function must assume that no previous filter was run before, and should therefore return all the possible relations that it determines.
- **direction** (*str*) – One of 'less\_specific' or 'more\_specific'. Since refinements are directed relations, this function can operate in two different directions: it can either find less specific potentially related statements, or it can find more specific potentially related statements, as determined by this argument.

**Returns** A set of INDRA Statement hashes that are potentially related to the given statement.

**Return type** set of int

**initialize**(*stmts\_by\_hash*)

Initialize the filter class with a set of statements.

The filter can build up some useful data structures in this function before being applied to any specific statements.

**Parameters** *stmts\_by\_hash* (*dict*[*int*, *indra.statements.Statement*]) – A dict of statements keyed by their hashes.

*indra\_world.assembly.operations.get\_expanded\_events\_influences*(*stmts*)

Return a list of all standalone events from a list of statements.

*indra\_world.assembly.operations.location\_matches\_compositional*(*stmt*)

Return a matches\_key which takes geo-location into account.

*indra\_world.assembly.operations.location\_refinement\_compositional*(*st1*, *st2*, *ontology*,  
*entities\_refined=True*)

Return True if there is a location-aware refinement between stmts.

*indra\_world.assembly.operations.make\_display\_name*(*comp\_grounding*)

Return display name from a compositional grounding with 'of' linkers.

*indra\_world.assembly.operations.make\_display\_name\_linear*(*comp\_grounding*)

Return display name from compositional grounding with linear joining.

*indra\_world.assembly.operations.merge\_deltas*(*stmts\_in*)

Gather and merge original Influence delta information from evidence.

This function is only applicable to Influence Statements that have subj and obj deltas. All other statement types are passed through unchanged. Polarities and adjectives for subjects and objects respectively are collected and merged by traversing all evidences of a Statement.

**Parameters** *stmts\_in* (*list*[*indra.statements.Statement*]) – A list of INDRA Statements whose influence deltas should be merged. These Statements are meant to have been preassembled and potentially have multiple pieces of evidence.

**Returns** *stmts\_out* – The list of Statements now with deltas merged at the Statement level.

**Return type** *list*[*indra.statements.Statement*]

*indra\_world.assembly.operations.remove\_namespaces*(*stmts*, *namespaces*)

Remove unnecessary namespaces from Concept grounding.

*indra\_world.assembly.operations.remove\_raw\_grounding*(*stmts*)

Remove the raw\_grounding annotation to decrease output size.

### 2.2.3 Matches functions (*indra\_world.assembly.matches*)

*indra\_world.assembly.matches.event\_location\_time\_matches*(*event*)

Return Event matches key which takes location and time into account.

*indra\_world.assembly.matches.get\_location*(*stmt*)

Return the grounded geo-location context associated with a Statement.

*indra\_world.assembly.matches.get\_location\_from\_object*(*loc\_obj*)

Return geo-location from a RefContext location object.

*indra\_world.assembly.matches.get\_time*(*stmt*)

Return the time context associated with a Statement.

`indra_world.assembly.matches.has_location(stmt)`  
Return True if a Statement has grounded geo-location context.

`indra_world.assembly.matches.has_time(stmt)`  
Return True if a Statement has time context.

`indra_world.assembly.matches.location_matches(stmt)`  
Return a matches\_key which takes geo-location into account.

`indra_world.assembly.matches.location_matches_compositional(stmt)`  
Return a matches\_key which takes geo-location into account.

## 2.2.4 Refinement functions (`indra_world.assembly.refinement`)

**class** `indra_world.assembly.refinement.CompositionalRefinementFilter(ontology, nproc=None)`  
**extend**(*stmts\_by\_hash*)

Extend the initial data structures with a set of new statements.

**Parameters** *stmts\_by\_hash* (*dict*[*int*, *indra.statements.Statement*]) – A dict of statements keyed by their hashes.

**get\_related**(*stmt*, *possibly\_related=None*, *direction='less\_specific'*)

Return a set of statement hashes that a given statement is potentially related to.

### Parameters

- **stmt** (*indra.statements.Statement*) – The INDRA statement whose potential relations we want to filter.
- **possibly\_related** (*set* or *None*) – A set of statement hashes that this statement is potentially related to, as determined by some other filter. If this parameter is a set (including an empty set), this function should return a subset of it (intuitively, this filter can only further eliminate some of the potentially related hashes that were previously determined to be potential relations). If this argument is *None*, the function must assume that no previous filter was run before, and should therefore return all the possible relations that it determines.
- **direction** (*str*) – One of 'less\_specific' or 'more\_specific'. Since refinements are directed relations, this function can operate in two different directions: it can either find less specific potentially related statements, or it can find more specific potentially related statements, as determined by this argument.

**Returns** A set of INDRA Statement hashes that are potentially related to the given statement.

**Return type** set of int

**initialize**(*stmts\_by\_hash*)

Initialize the filter class with a set of statements.

The filter can build up some useful data structures in this function before being applied to any specific statements.

**Parameters** *stmts\_by\_hash* (*dict*[*int*, *indra.statements.Statement*]) – A dict of statements keyed by their hashes.

`indra_world.assembly.refinement.event_location_refinement(st1, st2, ontology, entities_refined, ignore_polarity=False)`

Return True if there is a location-aware refinement between Events.

`indra_world.assembly.refinement.event_location_time_refinement(st1, st2, ontology, entities_refined)`

Return True if there is a location/time refinement between Events.

`indra_world.assembly.refinement.get_agent_key(agent, comp_idx)`

Return a key for an Agent for use in refinement finding.

**Parameters** `agent` (*indra.statements.Agent* or *None*) – An INDRA Agent whose key should be returned.

**Returns** The key that maps the given agent to the ontology, with special handling for ungrounded and None Agents.

**Return type** `tuple` or *None*

`indra_world.assembly.refinement.location_refinement(st1, st2, ontology, entities_refined)`

Return True if there is a location-aware refinement between stmts.

`indra_world.assembly.refinement.location_refinement_compositional(st1, st2, ontology, entities_refined=True)`

Return True if there is a location-aware refinement between stmts.

`indra_world.assembly.refinement.location_time_refinement(st1, st2, ontology, entities_refined)`

Return True if there is a location/time refinement between stmts.

## 2.2.5 Incremental Assembler (`indra_world.assembly.incremental_assembler`)

`class indra_world.assembly.incremental_assembler.AssemblyDelta(new_stmts, new_evidences, new_refinements, beliefs, matches_fun=None)`

Represents changes to the assembly structure as a result of new statements added to a set of existing statements.

**new\_stmts**

A dict of new statement keyed by hash.

**Type** `dict[str, indra.statements.Statement]`

**new\_evidences**

A dict of new evidences for existing or new statements keyed by statement hash.

**Type** `dict[str, indra.statements.Evidence]`

**new\_refinements**

A list of statement hash pairs representing new refinement links.

**Type** `list[tuple]`

**beliefs**

A dict of belief scores keyed by all statement hashes (both old and new).

**Type** `dict[str, float]`

**matches\_fun**

An optional custom matches function. When using a custom matches function for assembly, providing it here is necessary to get correct JSON serialization.

**Type** `Optional[Callable[[Statement], str]]`

**to\_json()**

Return a JSON representation of the assembly delta.

```
class indra_world.assembly.incremental_assembler.IncrementalAssembler(prepared_stmts,
                                                                    refinement_filters=None,
                                                                    matches_fun=<function
                                                                    loca-
                                                                    tion_matches_compositional>,
                                                                    curations=None,
                                                                    post_processing_steps=None,
                                                                    ontol-
                                                                    ogy=<indra_world.ontology.ontology.World
                                                                    object>)
```

Assemble a set of prepared statements and allow incremental extensions.

#### Parameters

- **prepared\_stmts** (*list[indra.statements.Statement]*) – A list of prepared INDRA Statements.
- **refinement\_filters** (*Optional[list[indra.preassembler.refinement.RefinementFilter]]*) – A list of refinement filter classes to be used for refinement finding. Default: the standard set of compositional refinement filters.
- **matches\_fun** (*Optional[function]*) – A custom matches function for determining matching statements and calculating hashes. Default: matches function that takes compositional grounding and location into account.
- **curations** (*list[dict]*) – A list of user curations to be integrated into the assembly results.
- **post\_processing\_steps** (*list[dict]*) – Steps that can be used in an INDRA AssemblyPipeline to do post-processing on statements.

#### refinement\_edges

A set of tuples of statement hashes representing refinement links between statements.

Type *set*

#### add\_statements(stmts)

Add new statements for incremental assembly.

**Parameters** *stmts* (*list[indra.statements.Statement]*) – A list of new prepared statements to be incrementally assembled into the set of existing statements.

**Returns** An AssemblyDelta object representing the changes to the assembly as a result of the new added statements.

**Return type** *AssemblyDelta*

#### static annotate\_evidences(stmt)

Add annotations to evidences of a given statement.

#### apply\_curations()

Apply the set of curations to the de-duplicated statements.

#### static build\_refinements\_graph(stmts\_by\_hash, refinement\_edges)

Return a refinements graph based on statements and refinement edges.

#### deduplicate()

Build hash-based statement and evidence data structures to deduplicate.

#### get\_all\_supporting\_evidence(sh)

Return direct and indirect evidence for a statement hash.

**get\_beliefs()**

Calculate and return beliefs for all statements.

**get\_curation\_effect**(*old\_hash*, *curation*)

Return changed matches hash as a result of curation.

**get\_refinements()**

Calculate refinement relationships between de-duplicated statements.

**get\_statements()**

Return a flat list of statements with their evidences.

**indra\_world.assembly.incremental\_assembler.parse\_factor\_grounding\_curation**(*cur*)

Parse details from a curation that changes a concept's grounding.

**indra\_world.assembly.incremental\_assembler.parse\_factor\_polarity\_curation**(*cur*)

Parse details from a curation that changes an event's polarity.

## 2.2.6 Statistics (**indra\_world.assembly.stats**)

## 2.3 Ontology Module (**indra\_world.ontology**)

Module containing the implementation of an IndraOntology for the World Modelers use case.

### 2.3.1 World Ontology (**indra\_world.ontology.ontology**)

**class** **indra\_world.ontology.ontology.WorldOntology**(*url*)

Represents the ontology used for World Modelers applications.

**Parameters** **url** (*str*) – The URL or file path pointing to a World Modelers ontology YAML.

**url**

The URL or file path pointing to a World Modelers ontology YAML.

**Type** *str*

**yml**

The ontology YAML as loaded by the yml package from the URL.

**Type** *list*

**add\_entry**(*entry*, *examples=None*, *neg\_examples=None*)

Add a new ontology entry with examples.

This works by adding the entry to the yml attribute first and then reloading the entire yml to build a new graph.

**Parameters**

- **entry** (*str*) – The new entry.
- **examples** (*Optional[list of str]*) – Examples for the new entry.
- **neg\_examples** (*Optional[list of str]*) – Negative examples for the new entry.

**build\_relations**(*node*, *tree*, *prefix*)

Build relations for the classic ontology format <= v3.0

**build\_relations\_new\_format**(*node*, *prefix*)

Build relations for the new ontology format > v3.0



**dump\_yaml\_str()**

Return a string-serialized form of the loaded YAML

**Returns** The YAML string of the ontology.

**Return type** `str`

**initialize()**

Load the World Modelers ontology from the web and build the graph.

`indra_world.ontology.ontology.load_world_ontology(url=None, default_type='compositional')`

Load the world ontology from a given URL or file path.

`indra_world.ontology.ontology.load_yaml_from_path(path)`

Return a YAML object loaded from a YAML file URL.

## 2.4 Belief Engine (`indra_world.belief`)

`indra_world.belief.get_eidos_bayesian_scorer(prior_counts=None)`

Return a BayesianScorer based on Eidos curation counts.

**Returns** A BayesianScorer belief scorer instance.

**Return type** `scorer`

`indra_world.belief.get_eidos_scorer()`

Return a SimpleScorer based on Eidos curated precision estimates.

**Returns** A SimpleScorer instance loaded with default prior probabilities as well as prior probabilities derived from curation-based counts.

**Return type** `scorer`

`indra_world.belief.load_eidos_curation_table()`

Return a pandas table of Eidos curation data.

**Returns** A pandas dataframe of the curation data.

**Return type** `table`

## 2.5 Output assemblers (`indra_world.assemblers`)

As opposed to INDRA, the importance of output/model assemblers in INDRA World is minor, since other systems such as Delphi (<https://github.com/ml4ai/delphi>) and DySE (<https://dl.acm.org/doi/10.1145/3359115.3359123>) take on the role of converting assembled INDRA Statements into probabilistic and logical dynamical models, respectively.

### 2.5.1 CAG Assembler (`indra_world.assemblers.cag`)

Assemble simple graphs of assembled INDRA Statements that can be embedded into websites or notebooks.

**Assembler (indra\_world.assemblers.cag.assembler)**

**class** indra\_world.assemblers.cag.assembler.CAGAssembler(*stmts=None*)

Assembles a causal analysis graph from INDRA Statements.

**Parameters** *stmts* (*Optional*[*list*[indra.statement.Statements]]) – A list of INDRA Statements to be assembled. Currently supports Influence Statements.

**statements**

A list of INDRA Statements to be assembled.

**Type** *list*[indra.statements.Statement]

**CAG**

A networkx MultiDiGraph object representing the causal analysis graph.

**Type** nx.MultiDiGraph

**add\_statements**(*stmts*)

Add a list of Statements to the assembler.

**export\_to\_cytoscapejs**()

Return CAG in format readable by CytoscapeJS.

**Returns** A JSON-like dict representing the graph for use with CytoscapeJS.

**Return type** *dict*

**generate\_jupyter\_js**(*cyjs\_style=None, cyjs\_layout=None*)

Generate Javascript from a template to run in Jupyter notebooks.

**Parameters**

- **cyjs\_style** (*Optional*[*dict*]) – A dict that sets CytoscapeJS style as specified in <https://github.com/cytoscape/cytoscape.js/blob/master/documentation/md/style.md>.
- **cyjs\_layout** (*Optional*[*dict*]) – A dict that sets CytoscapeJS layout parameters.

**Returns** A Javascript string to be rendered in a Jupyter notebook cell.

**Return type** *str*

**make\_model**(*grounding\_ontology='UN', grounding\_threshold=None*)

Return a networkx MultiDiGraph representing a causal analysis graph.

**Parameters**

- **grounding\_ontology** (*Optional*[*str*]) – The ontology from which the grounding should be taken (e.g. UN, FAO)
- **grounding\_threshold** (*Optional*[*float*]) – Minimum threshold score for Eidos grounding.

**Returns** The assembled CAG.

**Return type** nx.MultiDiGraph

## 2.5.2 Figaro Assembler (`indra_world.assemblers.figaro`)

A proof-of-concept assembler for INDRA Statements into probabilistic programs in the Figaro (<https://github.com/p2t2/figaro>) framework.

**Assembler** (`indra_world.assemblers.figaro.assembler`)

## 2.5.3 TSV Assembler (`indra_world.assemblers.tsv`)

Assemble tab separated spreadsheets of assembled INDRA Statements for curation purposes.

## 2.6 Indra World Service (`indra_world.service`)

*Note:* This is the documentation of the codebase used in the INDRA World service. Documentation of the service API can be found [here](#).

### 2.6.1 INDRA World Database (`indra_world.service.db`)

**Database Manager** (`indra_world.service.db.manager`)

**class** `indra_world.service.db.manager.DbManager(url)`

Manages transactions with the assembly database and exposes an API for various operations.

**add\_curation\_for\_project**(*project\_id*, *stmt\_hash*, *curation*)

Add curations for a given project.

**add\_dart\_record**(*reader*, *reader\_version*, *document\_id*, *storage\_key*, *date*, *output\_version=None*,  
*labels=None*, *tenants=None*)

Insert a DART record into the database.

**add\_project**(*project\_id*, *name*)

Add a new project.

#### Parameters

- **project\_id** (*str*) – The project ID.
- **name** (*str*) – The project name

**add\_records\_for\_project**(*project\_id*, *record\_keys*)

Add document IDs for a project with the given ID.

**add\_statements\_for\_record**(*record\_key*, *stmts*, *indra\_version*)

Add a set of prepared statements for a given document.

**create\_all**()

Create all the database tables in the schema.

**execute**(*operation*)

Execute an insert operation on the current session and return results.

**get\_curations\_for\_project**(*project\_id*)

Return curations for a given project

**get\_dart\_records**(*reader=None, document\_id=None, reader\_version=None, output\_version=None, labels=None, tenants=None*)  
Return storage keys for DART records given constraints.

**get\_full\_dart\_records**(*reader=None, document\_id=None, reader\_version=None, output\_version=None, labels=None, tenants=None*)  
Return full DART records given constraints.

**get\_session**()  
Return the current active session or create one if not available.

**get\_statements**()  
Return all prepared statements in the DB.

**get\_statements\_for\_document**(*document\_id, reader=None, reader\_version=None, indra\_version=None*)  
Return prepared statements for a given document.

**get\_statements\_for\_record**(*record\_key*)  
Return prepared statements for given record.

**query**(\**query\_args*)  
Run and return results of a generic query.

**sql\_query**(*query\_str*)  
Run and return results of a generic SQL query.

#### Database Schema (`indra_world.service.db.Schema`)

```
class indra_world.service.db.schema.Corpora(**kwargs)
class indra_world.service.db.schema.CorporusRecords(**kwargs)
class indra_world.service.db.schema.Curations(**kwargs)
class indra_world.service.db.schema.DartRecords(**kwargs)
class indra_world.service.db.schema.Ontologies(**kwargs)
class indra_world.service.db.schema.PreparedStatements(**kwargs)
class indra_world.service.db.schema.ProjectRecords(**kwargs)
class indra_world.service.db.schema.Projects(**kwargs)
```

### 2.6.2 Service controller (`indra_world.service.controller`)

#### 2.6.3 REST API (`indra_world.service.app`)

```
class indra_world.service.app.AddProjectRecords(api=None, *args, **kwargs)
    post()
```

Add project records and assemble them.

##### Parameters

- **project\_id** (*str*) – ID of a project to add records.
- **records** (*list[dict]*) – A list of records to add, each should have a ‘storage\_key’.

**Returns** `delta_json` – A JSON representation of AssemblyDelta.

**Return type** `json`

```
class indra_world.service.app.CwmsProcessText(api=None, *args, **kwargs)
    post()
```

Process text with CWMS and return INDRA Statements.

**Parameters** **text** (*str*) – Text to process

**Returns** **statements** – A list of extracted INDRA Statements.

**Return type** *list*[indra.statements.Statement.to\_json()]

```
class indra_world.service.app.EidosProcessJsonld(api=None, *args, **kwargs)
    post()
```

Process an EIDOS JSON-LD and return INDRA Statements.

**Parameters**

- **jsonld** (*str*) – The JSON-LD string to be processed.
- **grounding\_ns** (*Optional[list]*) – A list of name spaces for which INDRA should represent groundings, when given. If not specified or None, all grounding name spaces are propagated. If an empty list, no groundings are propagated. Example: ['UN', 'WM'], Default: None
- **extract\_filter** (*Optional[list]*) – A list of relation types to extract. Valid values in the list are 'influence', 'association', 'event'. If not given, all relation types are extracted. This argument can be used if, for instance, only Influence statements are of interest. Default: None
- **grounding\_mode** (*Optional[str]*) – Selects whether 'flat' or 'compositional' groundings should be extracted. Default: 'flat'.

**Returns** **statements** – A list of extracted INDRA Statements.

**Return type** *list*[indra.statements.Statement.to\_json()]

```
class indra_world.service.app.EidosProcessText(api=None, *args, **kwargs)
    post()
```

Process text with EIDOS and return INDRA Statements.

**Parameters**

- **text** (*str*) – The text to be processed.
- **webservice** (*Optional[str]*) – An Eidos reader web service URL to send the request to. If None, the reading is assumed to be done with the Eidos JAR rather than via a web service. Default: None
- **grounding\_ns** (*Optional[list]*) – A list of name spaces for which INDRA should represent groundings, when given. If not specified or None, all grounding name spaces are propagated. If an empty list, no groundings are propagated. Example: ['UN', 'WM'], Default: None
- **extract\_filter** (*Optional[list]*) – A list of relation types to extract. Valid values in the list are 'influence', 'association', 'event'. If not given, all relation types are extracted. This argument can be used if, for instance, only Influence statements are of interest. Default: None
- **grounding\_mode** (*Optional[str]*) – Selects whether 'flat' or 'compositional' groundings should be extracted. Default: 'flat'.

**Returns** **statements** – A list of extracted INDRA Statements.

**Return type** *list*[indra.statements.Statement.to\_json()]

```
class indra_world.service.app.GetAllRecords(api=None, *args, **kwargs)
```

**get()**

Get all DART records captured by the service.

**Returns** **records** – A list of records.

**Return type** `list[dict]`

**class** `indra_world.service.app.GetProjectCurations(api=None, *args, **kwargs)`

**get()**

Get project curations.

**Parameters** **project\_id** (*str*) – ID of a project.

**Returns** **curations** – A list of curations for the project.

**Return type** `list[dict]`

**class** `indra_world.service.app.GetProjectRecords(api=None, *args, **kwargs)`

**get()**

Get records for a project.

**Parameters** **project\_id** (*str*) – ID of a project.

**Returns** **records** – A list of records for the project.

**Return type** `list[dict]`

**class** `indra_world.service.app.GetProjects(api=None, *args, **kwargs)`

**get()**

Get a list of all projects.

**class** `indra_world.service.app.Health(api=None, *args, **kwargs)`

**class** `indra_world.service.app.HumeProcessJsonld(api=None, *args, **kwargs)`

**post()**

Process Hume JSON-LD and return INDRA Statements.

**Parameters** **jsonld** (*str*) – The JSON-LD string to be processed.

**Returns** **statements** – A list of extracted INDRA Statements.

**Return type** `list[indra.statements.Statement.to_json()]`

**class** `indra_world.service.app.NewProject(api=None, *args, **kwargs)`

**post()**

Create new project.

**Parameters**

- **project\_id** (*str*) – ID of a new project.
- **project\_name** (*str*) – Name of a new project.
- **corpus\_id** (*str*) – ID of a corpus.

**class** `indra_world.service.app.Notify(api=None, *args, **kwargs)`

**post()**

Add and process DART record.

**Parameters**

- **identity** (*str*) – Name of the reader.
- **version** (*str*) – Reader version.
- **document\_id** (*str*) – ID of a document to process.

- **storage\_key** (*str*) – Key to store the record with.
- **output\_version** (*str*) – The output version (typically ontology version).
- **labels** (*list of str*) – A list of labels for the output.
- **tenants** (*list of str*) – A list of tenants for the output.

```
class indra_world.service.app.SofiaProcessJson(api=None, *args, **kwargs)
    post()
```

Process a Sofia JSON and return INDRA Statements.

#### Parameters

- **json** (*str*) – The JSON string to be processed.
- **extract\_filter** (*Optional[list]*) – A list of relation types to extract. Valid values in the list are ‘influence’, ‘association’, ‘event’. If not given, all relation types are extracted. This argument can be used if, for instance, only Influence statements are of interest. Default: None
- **grounding\_mode** (*Optional[str]*) – Selects whether ‘flat’ or ‘compositional’ groundings should be extracted. Default: ‘flat’.

**Returns** **statements** – A list of extracted INDRA Statements.

**Return type** *list[indra.statements.Statement.to\_json()]*

```
class indra_world.service.app.SubmitCurations(api=None, *args, **kwargs)
    post()
```

Submit curations.

#### Parameters

- **project\_id** (*str*) – ID of a project.
- **curations** (*list[dict]*) – A list of curations to submit.

**Returns** **mappings** – For any statement matches hashes that have changed due to the curations submitted here, the new hash (after applying the curation) is given. Statements whose hash didn’t change, or if a curation for some reason couldn’t be applied, the given statement is not added to the return value.

**Return type** *dict*

## 2.6.4 Corpus manager (*indra\_world.service.corpus\_manager*)

This module allows running one-off assembly on a set of DART records (i.e., reader outputs) into a ‘seed corpus’ that can be dumped on S3 for loading into CauseMos.

```
class indra_world.service.corpus_manager.CorporusManager(db_url, dart_records, corpus_id, metadata,
                                                         dart_client=None)
```

Corpus manager class allowing running assembly on a set of DART records.

```
assemble()
```

Run assembly on the prepared statements.

This function loads all the prepared statements associated with the corpus and then runs assembly on them.

```
dump_local(base_folder)
```

Dump assembled corpus into local files.

```
dump_s3()
```

Dump assembled corpus onto S3.

**prepare**(*records\_exist=False*)

Run the preprocessing pipeline on statements.

This function adds the new corpus to the DB, adds records to the new corpus, then processes the reader outputs for those records into statements, preprocesses the statements, and then stores these prepared statements in the DB.

`indra_world.service.corpus_manager.download_corpus(corpus_id, fname)`

Download a given corpus of assembled statements from S3.

**Parameters**

- **corpus\_id** (*str*) – The ID of the corpus.
- **fname** (*str*) – The file in which the downloaded corpus should be written.

**Return type** *None*

`indra_world.service.corpus_manager.get_corpus_index()`

Return the corpus index as a list of tuples with corpus IDs and dates.

## 2.7 INDRA World Dashboard (`indra_world.dashboard`)

The dashboard provides a simple web-based interface to run INDRA World assembly by searching for reader outputs under some constraints, specify assembly configurations, and parameters for the output (name, description, etc.).



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### i

`indra_world.sources.sofia.processor`, 9

- `indra_world.assemblers`, 21
- `indra_world.assemblers.cag`, 21
- `indra_world.assemblers.cag.assembler`, 22
- `indra_world.assemblers.figaro`, 23
- `indra_world.assemblers.figaro.assembler`, 23
- `indra_world.assemblers.tsv`, 23
- `indra_world.assembly`, 15
- `indra_world.assembly.incremental_assembler`, 18
- `indra_world.assembly.matches`, 16
- `indra_world.assembly.operations`, 15
- `indra_world.assembly.preprocess`, 15
- `indra_world.assembly.refinement`, 17
- `indra_world.assembly.stats`, 20
- `indra_world.belief`, 21
- `indra_world.dashboard`, 28
- `indra_world.ontology`, 20
- `indra_world.ontology.ontology`, 20
- `indra_world.service.app`, 24
- `indra_world.service.controller`, 24
- `indra_world.service.corpus_manager`, 27
- `indra_world.service.db`, 23
- `indra_world.service.db.manager`, 23
- `indra_world.service.db.schema`, 24
- `indra_world.sources.cwms`, 11
- `indra_world.sources.cwms.api`, 11
- `indra_world.sources.cwms.processor`, 12
- `indra_world.sources.dart`, 13
- `indra_world.sources.dart.api`, 13
- `indra_world.sources.dart.client`, 13
- `indra_world.sources.eidos`, 3
- `indra_world.sources.eidos.api`, 3
- `indra_world.sources.eidos.client`, 5
- `indra_world.sources.eidos.migration_table_processor`, 6
- `indra_world.sources.eidos.processor`, 6
- `indra_world.sources.hume`, 6
- `indra_world.sources.hume.api`, 6
- `indra_world.sources.hume.processor`, 7
- `indra_world.sources.sofia`, 7
- `indra_world.sources.sofia.api`, 7



## INDEX

### A

**add\_curation\_for\_project()** (in- *dra\_world.service.db.manager.DbManager* method), 23  
**add\_dart\_record()** (in- *dra\_world.service.db.manager.DbManager* method), 23  
**add\_entry()** (*indra\_world.ontology.ontology.WorldOntology* method), 20  
**add\_project()** (*indra\_world.service.db.manager.DbManager* method), 23  
**add\_records\_for\_project()** (in- *dra\_world.service.db.manager.DbManager* method), 23  
**add\_statements()** (in- *dra\_world.assemblers.cag.assembler.CAGAssembler* method), 22  
**add\_statements()** (in- *dra\_world.assembly.incremental\_assembler.IncrementalAssembler* method), 19  
**add\_statements\_for\_record()** (in- *dra\_world.service.db.manager.DbManager* method), 23  
**AddProjectRecords** (class in *indra\_world.service.app*), 24  
**annotate\_evidences()** (in- *dra\_world.assembly.incremental\_assembler.IncrementalAssembler* static method), 19  
**apply\_curations()** (in- *dra\_world.assembly.incremental\_assembler.IncrementalAssembler* method), 19  
**assemble()** (*indra\_world.service.corpus\_manager.CorporaManager* method), 27  
**AssemblyDelta** (class in *indra\_world.assembly.incremental\_assembler*), 18

### B

**beliefs** (*indra\_world.assembly.incremental\_assembler.AssemblyDelta* attribute), 18

**build\_refinements\_graph()** (in- *dra\_world.assembly.incremental\_assembler.IncrementalAssembler*

static method), 19

**build\_relations()** (in- *dra\_world.ontology.ontology.WorldOntology* method), 20

**build\_relations\_new\_format()** (in- *dra\_world.ontology.ontology.WorldOntology* method), 20

### C

**cache\_record()** (*indra\_world.sources.dart.client.DartClient* method), 13

**cache\_records()** (in- *dra\_world.sources.dart.client.DartClient* method), 13

**CAG** (*indra\_world.assemblers.cag.assembler.CAGAssembler* attribute), 22

**CAGAssembler** (class in *indra\_world.assemblers.cag.assembler*), 22

**CompositionalRefinementFilter** (class in *indra\_world.assembly.operations*), 15

**CompositionalRefinementFilter** (class in *indra\_world.assembly.refinement*), 17

**Corpora** (class in *indra\_world.service.db.schema*), 24

**CorpusManager** (class in *indra\_world.service.corpus\_manager*), 27

**CorpusRecords** (class in *indra\_world.service.db.schema*), 24

**create\_all()** (*indra\_world.service.db.manager.DbManager* method), 23

**Curations** (class in *indra\_world.service.db.schema*), 24

**CWMSProcessor** (class in *indra\_world.sources.cwms.processor*), 12

**CWMSProcessorCompositional** (class in *indra\_world.sources.cwms.processor*), 13

**CwmsProcessText** (class in *indra\_world.service.app*), 24

### D

**DartClient** (class in *indra\_world.sources.dart.client*),

13

DartRecords (class in *indra\_world.service.db.schema*), 24  
 DbManager (class in *indra\_world.service.db.manager*), 23  
 deduplicate() (*indra\_world.assembly.incremental\_assembler.IncrementalAssembler* method), 19  
 doc\_id(*indra\_world.sources.cwms.processor.CWMSProcessor* attribute), 12  
 download\_corpus() (in module *indra\_world.service.corpus\_manager*), 28  
 download\_output() (*indra\_world.sources.dart.client.DartClient* method), 14  
 dump\_local() (*indra\_world.service.corpus\_manager.CorporaManager* method), 27  
 dump\_s3() (*indra\_world.service.corpus\_manager.CorporaManager* method), 27  
 dump\_yaml\_str() (*indra\_world.ontology.ontology.WorldOntology* method), 20  
**E**  
 EidosProcessJsonld (class in *indra\_world.service.app*), 25  
 EidosProcessorCompositional (class in *indra\_world.sources.eidos.processor*), 6  
 EidosProcessText (class in *indra\_world.service.app*), 25  
 EidosWorldProcessor (class in *indra\_world.sources.eidos.processor*), 6  
 event\_from\_event() (*indra\_world.sources.cwms.processor.CWMSProcessor* method), 13  
 event\_location\_refinement() (in module *indra\_world.assembly.refinement*), 17  
 event\_location\_time\_matches() (in module *indra\_world.assembly.matches*), 16  
 event\_location\_time\_refinement() (in module *indra\_world.assembly.refinement*), 17  
 execute() (*indra\_world.service.db.manager.DbManager* method), 23  
 export\_to\_cytoscapejs() (*indra\_world.assemblers.cag.assembler.CAGAssembler* method), 22  
 extend() (*indra\_world.assembly.operations.CompositionalRefinementFileRefinement* method), 15  
 extend() (*indra\_world.assembly.refinement.CompositionalRefinementFileRefinement* method), 17  
 extract\_causal\_relations() (*indra\_world.sources.cwms.processor.CWMSProcessor* method), 13  
 extract\_events() (*indra\_world.sources.cwms.processor.CWMSProcessor* method), 13  
 extract\_events() (*indra\_world.sources.sofia.processor.SofiaExcelProcessor* method), 9  
 extract\_events() (*indra\_world.sources.sofia.processor.SofiaJsonProcessor* method), 9  
 extract\_relations() (*indra\_world.sources.sofia.processor.SofiaExcelProcessor* method), 9  
 extract\_relations() (*indra\_world.sources.sofia.processor.SofiaJsonProcessor* method), 9  
**G**  
 generate\_jupyter\_js() (*indra\_world.assemblers.cag.assembler.CAGAssembler* method), 22  
 get\_context\_from\_ref() (*indra\_world.sources.eidos.processor.EidosWorldProcessor* method), 6  
 get() (*indra\_world.service.app.GetAllRecords* method), 25  
 get() (*indra\_world.service.app.GetProjectCurations* method), 26  
 get() (*indra\_world.service.app.GetProjectRecords* method), 26  
 get() (*indra\_world.service.app.GetProjects* method), 26  
 get\_agent\_key() (in module *indra\_world.assembly.refinement*), 17  
 get\_all\_supporting\_evidence() (*indra\_world.assembly.incremental\_assembler.IncrementalAssembler* method), 19  
 get\_beliefs() (*indra\_world.assembly.incremental\_assembler.IncrementalAssembler* method), 19  
 get\_compositional\_grounding() (*indra\_world.sources.sofia.processor.SofiaProcessor* method), 10  
 get\_corpus\_index() (in module *indra\_world.service.corpus\_manager*), 28  
 get\_curation\_effect() (*indra\_world.assembly.incremental\_assembler.IncrementalAssembler* method), 20  
 get\_curations\_for\_project() (*indra\_world.service.db.manager.DbManager* method), 23  
 get\_dart\_records() (*indra\_world.service.db.manager.DbManager* method), 23  
 get\_eidos\_bayesian\_scorer() (in module *indra\_world.belief*), 21  
 get\_eidos\_scorer() (in module *indra\_world.belief*), 21  
 get\_event() (*indra\_world.sources.sofia.processor.SofiaProcessor* method), 10

get_event_compositional()	(in- dra_world.sources.sofia.processor.SofiaProcessor method), 10	get_statements()	(in- dra_world.service.db.manager.DbManager method), 24
get_event_flat()	(in- dra_world.sources.sofia.processor.SofiaProcessor method), 10	get_statements_for_document()	(in- dra_world.service.db.manager.DbManager method), 24
get_expanded_events_influences()	(in module in- dra_world.assembly.operations), 16	get_statements_for_record()	(in- dra_world.service.db.manager.DbManager method), 24
get_full_dart_records()	(in- dra_world.service.db.manager.DbManager method), 24	get_time()	(in module indra_world.assembly.matches), 16
get_groundings()	(in- dra_world.sources.eidos.processor.EidosProcessor method), 6	GetAllRecords	(class in indra_world.service.app), 26
get_groundings()	(in- dra_world.sources.eidos.processor.EidosWorldProcessor method), 6	GetProjectRecords	(class in indra_world.service.app), 26
get_local_storage_path()	(in- dra_world.sources.dart.client.DartClient method), 14	GetProjectRecords	(class in indra_world.service.app), 26
get_location()	(in module in- dra_world.assembly.matches), 16	GetProjects	(class in indra_world.service.app), 26
get_location_from_object()	(in module in- dra_world.assembly.matches), 16	grounding_dict_to_list()	(in module in- dra_world.sources.eidos.client), 5
get_meaningful_events()	(in- dra_world.sources.sofia.processor.SofiaProcessor method), 10		
get_output_from_record()	(in- dra_world.sources.dart.client.DartClient method), 14		
get_outputs_from_records()	(in- dra_world.sources.dart.client.DartClient method), 14		
get_reader_output_records()	(in- dra_world.sources.dart.client.DartClient method), 14		
get_reader_versions()	(in- dra_world.sources.dart.client.DartClient method), 14		
get_refinements()	(in- dra_world.assembly.incremental_assembler.IncrementalAssembler method), 20		
get_related()	(indra_world.assembly.operations.CompositionalRefinementFilter method), 15		
get_related()	(indra_world.assembly.refinement.CompositionalRefinementFilter method), 17		
get_relation_events()	(in- dra_world.sources.sofia.processor.SofiaProcessor method), 11		
get_session()	(indra_world.service.db.manager.DbManager method), 24		
get_statements()	(in- dra_world.assembly.incremental_assembler.IncrementalAssembler method), 20		

- module, 16
- indra\_world.assembly.operations
  - module, 15
- indra\_world.assembly.preprocess
  - module, 15
- indra\_world.assembly.refinement
  - module, 17
- indra\_world.assembly.stats
  - module, 20
- indra\_world.belief
  - module, 21
- indra\_world.dashboard
  - module, 28
- indra\_world.ontology
  - module, 20
- indra\_world.ontology.ontology
  - module, 20
- indra\_world.service.app
  - module, 24
- indra\_world.service.controller
  - module, 24
- indra\_world.service.corpus\_manager
  - module, 27
- indra\_world.service.db
  - module, 23
- indra\_world.service.db.manager
  - module, 23
- indra\_world.service.db.schema
  - module, 24
- indra\_world.sources.cwms
  - module, 11
- indra\_world.sources.cwms.api
  - module, 11
- indra\_world.sources.cwms.processor
  - module, 12
- indra\_world.sources.dart
  - module, 13
- indra\_world.sources.dart.api
  - module, 13
- indra\_world.sources.dart.client
  - module, 13
- indra\_world.sources.eidos
  - module, 3
- indra\_world.sources.eidos.api
  - module, 3
- indra\_world.sources.eidos.client
  - module, 5
- indra\_world.sources.eidos.migration\_table\_processor
  - module, 6
- indra\_world.sources.eidos.processor
  - module, 6
- indra\_world.sources.hume
  - module, 6
- indra\_world.sources.hume.api

- module, 6
- indra\_world.sources.hume.processor
  - module, 7
- indra\_world.sources.sofia
  - module, 7
- indra\_world.sources.sofia.api
  - module, 7
- indra\_world.sources.sofia.processor
  - module, 9
- influence\_from\_event() (in *indra\_world.sources.cwms.processor.CWMSProcessor* method), 13
- influence\_from\_relation() (in *indra\_world.sources.cwms.processor.CWMSProcessor* method), 13
- initialize() (*indra\_world.assembly.operations.CompositionalRefinement* method), 16
- initialize() (*indra\_world.assembly.refinement.CompositionalRefinement* method), 17
- initialize() (*indra\_world.ontology.ontology.WorldOntology* method), 21

## L

- load\_eidos\_curation\_table() (in module *indra\_world.belief*), 21
- load\_world\_ontology() (in module *indra\_world.ontology.ontology*), 21
- load\_yaml\_from\_path() (in module *indra\_world.ontology.ontology*), 21
- location\_matches() (in module *indra\_world.assembly.matches*), 17
- location\_matches\_compositional() (in module *indra\_world.assembly.matches*), 17
- location\_matches\_compositional() (in module *indra\_world.assembly.operations*), 16
- location\_refinement() (in module *indra\_world.assembly.refinement*), 18
- location\_refinement\_compositional() (in module *indra\_world.assembly.operations*), 16
- location\_refinement\_compositional() (in module *indra\_world.assembly.refinement*), 18
- location\_time\_refinement() (in module *indra\_world.assembly.refinement*), 18

## M

- make\_display\_name() (in module *indra\_world.assembly.operations*), 16
- make\_display\_name\_linear() (in module *indra\_world.assembly.operations*), 16
- make\_model() (*indra\_world.assemblers.cag.assembler.CAGAssembler* method), 22
- matches\_fun(*indra\_world.assembly.incremental\_assembler.AssemblyDelta* attribute), 18



merge\_deltas() (in module *indra\_world.assembly.operations*), 16

migration\_from\_event() (in module *indra\_world.sources.cwms.processor.CWMSProcessor*), 13

module

- indra\_world.assemblers*, 21
- indra\_world.assemblers.cag*, 21
- indra\_world.assemblers.cag.assembler*, 22
- indra\_world.assemblers.figaro*, 23
- indra\_world.assemblers.figaro.assembler*, 23
- indra\_world.assemblers.tsv*, 23
- indra\_world.assembly*, 15
- indra\_world.assembly.incremental\_assembler*, 18
- indra\_world.assembly.matches*, 16
- indra\_world.assembly.operations*, 15
- indra\_world.assembly.preprocess*, 15
- indra\_world.assembly.refinement*, 17
- indra\_world.assembly.stats*, 20
- indra\_world.belief*, 21
- indra\_world.dashboard*, 28
- indra\_world.ontology*, 20
- indra\_world.ontology.ontology*, 20
- indra\_world.service.app*, 24
- indra\_world.service.controller*, 24
- indra\_world.service.corpus\_manager*, 27
- indra\_world.service.db*, 23
- indra\_world.service.db.manager*, 23
- indra\_world.service.db.schema*, 24
- indra\_world.sources.cwms*, 11
- indra\_world.sources.cwms.api*, 11
- indra\_world.sources.cwms.processor*, 12
- indra\_world.sources.dart*, 13
- indra\_world.sources.dart.api*, 13
- indra\_world.sources.dart.client*, 13
- indra\_world.sources.eidos*, 3
- indra\_world.sources.eidos.api*, 3
- indra\_world.sources.eidos.client*, 5
- indra\_world.sources.eidos.migration\_table*, 6
- indra\_world.sources.eidos.processor*, 6
- indra\_world.sources.hume*, 6
- indra\_world.sources.hume.api*, 6
- indra\_world.sources.hume.processor*, 7
- indra\_world.sources.sofia*, 7
- indra\_world.sources.sofia.api*, 7
- indra\_world.sources.sofia.processor*, 9

**N**

new\_evidences (*indra\_world.assembly.incremental\_assembler.AssemblyDelta* attribute), 18

new\_refinements (in module *indra\_world.assembly.incremental\_assembler.AssemblyDelta* attribute), 18

new\_stmts (*indra\_world.assembly.incremental\_assembler.AssemblyDelta* attribute), 18

NewProject (class in *indra\_world.service.app*), 26

Notify (class in *indra\_world.service.app*), 26

**O**

Ontologies (class in *indra\_world.service.db.schema*), 24

**P**

par\_to\_sec (*indra\_world.sources.cwms.processor.CWMSProcessor* attribute), 12

paragraphs (*indra\_world.sources.cwms.processor.CWMSProcessor* attribute), 12

parse\_factor\_grounding\_curation() (in module *indra\_world.assembly.incremental\_assembler*), 20

parse\_factor\_polarity\_curation() (in module *indra\_world.assembly.incremental\_assembler*), 20

post() (*indra\_world.service.app.AddProjectRecords* method), 24

post() (*indra\_world.service.app.CwmsProcessText* method), 25

post() (*indra\_world.service.app.EidosProcessJsonld* method), 25

post() (*indra\_world.service.app.EidosProcessText* method), 25

post() (*indra\_world.service.app.HumeProcessJsonld* method), 26

post() (*indra\_world.service.app.NewProject* method), 26

post() (*indra\_world.service.app.Notify* method), 26

post() (*indra\_world.service.app.SofiaProcessJsonld* method), 27

post() (*indra\_world.service.app.SubmitCurations* method), 27

prepare() (*indra\_world.service.corpus\_manager.CorporaManager* method), 28

PreparedStatements (class in *indra\_world.service.db.schema*), 24

preprocess\_statements() (in module *indra\_world.assembly.preprocess*), 15

prioritize\_records() (in module *indra\_world.sources.dart.client*), 14

process\_ekb() (in module *indra\_world.sources.cwms.api*), 11

process\_ekb\_file() (in module *indra\_world.sources.cwms.api*), 11

process\_entities() (in- S  
dra\_world.sources.sofia.processor.SofiaJsonProcessor  
method), 9 sentences (indra\_world.sources.cwms.processor.CWMSProcessor  
attribute), 12

process\_events() (in- SofiaExcelProcessor (class in in-  
dra\_world.sources.sofia.processor.SofiaExcelProcessor dra\_world.sources.sofia.processor), 9  
method), 9

process\_events() (in- SofiaJsonProcessor (class in in-  
dra\_world.sources.sofia.processor.SofiaJsonProcessor dra\_world.sources.sofia.processor), 9  
method), 10 SofiaProcessJson (class in indra\_world.service.app),  
27

process\_json() (in module in- SofiaProcessor (class in in-  
dra\_world.sources.eidos.api), 3 dra\_world.sources.sofia.processor), 10

process\_json() (in module in- sql\_query() (indra\_world.service.db.manager.DbManager  
dra\_world.sources.sofia.api), 7 method), 24

process\_json\_file() (in module in- statements (indra\_world.assemblers.cag.assembler.CAGAssembler  
dra\_world.sources.eidos.api), 3 attribute), 22

process\_json\_file() (in module in- statements (indra\_world.sources.cwms.processor.CWMSProcessor  
dra\_world.sources.sofia.api), 8 attribute), 12

process\_json\_str() (in module in- statements (indra\_world.sources.hume.processor.HumeJsonLdProcessor  
dra\_world.sources.eidos.api), 4 attribute), 7

process\_jsonld() (in module in- SubmitCurations (class in indra\_world.service.app),  
dra\_world.sources.hume.api), 6 27

process\_jsonld\_file() (in module in- T  
dra\_world.sources.hume.api), 7

process\_table() (in module in- time\_context\_from\_ref() (in-  
dra\_world.sources.sofia.api), 8 dra\_world.sources.eidos.processor.EidosWorldProcessor  
method), 6

process\_text() (in module in- time\_context\_from\_timex() (in module in-  
dra\_world.sources.cwms.api), 11 dra\_world.sources.eidos.processor), 6

process\_text() (in module in- to\_json() (indra\_world.assembly.incremental\_assembler.AssemblyDelta  
dra\_world.sources.eidos.api), 4 method), 18

process\_text() (in module in- tree (indra\_world.sources.cwms.processor.CWMSProcessor  
dra\_world.sources.sofia.api), 8 attribute), 12

ProjectRecords (class in in- tree (indra\_world.sources.hume.processor.HumeJsonLdProcessor  
dra\_world.service.db.schema), 24 attribute), 7

Projects (class in indra\_world.service.db.schema), 24

Q U  
query() (indra\_world.service.db.manager.DbManager url (indra\_world.ontology.ontology.WorldOntology at-  
method), 24 tribute), 20

R W  
ref\_context\_from\_geoloc() (in module in- WorldOntology (class in in-  
dra\_world.sources.eidos.processor), 6 dra\_world.ontology.ontology), 20

refinement\_edges (in- IncrementalAssembler  
dra\_world.assembly.incremental\_assembler.IncrementalAssembler  
attribute), 19

reground\_texts() (in module in- yml (indra\_world.ontology.ontology.WorldOntology at-  
dra\_world.sources.eidos.api), 5 tribute), 20

reground\_texts() (in module in-  
dra\_world.sources.eidos.client), 5

remove\_namespaces() (in module in-  
dra\_world.assembly.operations), 16

remove\_raw\_grounding() (in module in-  
dra\_world.assembly.operations), 16