

---

# **decisionengine**

***Release 1.1.1***

**Fermi Research Alliance, LLC.**

**Jun 04, 2020**



# CONTENTS

<b>1</b>	<b>Release Notes</b>	<b>3</b>
<b>2</b>	<b>Developer Documentation</b>	<b>7</b>
<b>3</b>	<b>Source code</b>	<b>11</b>
<b>4</b>	<b>Indices and tables</b>	<b>31</b>
	<b>Python Module Index</b>	<b>33</b>
	<b>Index</b>	<b>35</b>



The Decision Engine is a critical component of the HEP Cloud Facility. It provides the functionality of resource scheduling for disparate resource providers, including those which may have a cost or a restricted allocation of cycles



## RELEASE NOTES

### 1.1 Release 1.1.0

In this release:

- Fixed. [https://github.com/HEPCloud/decisionengine\\_modules/issues/108](https://github.com/HEPCloud/decisionengine_modules/issues/108) “Supply Postgres script to delete fields in main database before a certain date”
- significant code cleanup and pep8 compliance
- unit test work
- CI (GitHub actions and Travis) is introduced

commits

f894b1d : Skip unittest (#77)

632e64b : Add ipython

f681a79 : Make python 2.7 tests run on 1.1 branch

d6a32c0 : implementation of data reaper (#75)

2ad8614 : Use sparse checkout for first checkout to get .github/actions (#65)

812f032 : Cat output of pytest log Exit pylint entryptpoint with the line count of pep8 and pylint logs Deal with (detach from ...) Only tar up (S)RPMS dirs for rpm build.

6b05ec7 : Fix errors reported by run\_pylint (#62)

d9f5b66 : Setup pep8speaks

c3b8ac2 : Run github actions as non-root uid. Install packages in virtualenv and remove system rpms.

ae01f9e : Support Python 3 for Boost Python

579761c : Support Python 3 for Boost Python

044b979 : Remove unnecessary using declarations.

00f6d00 : Add extra header dependency due to Boost Python ommission.

24e0795 : Apply clang-format

17c17f9 : Remove JSON dependency.

faa0b22 : Massive cleanup.

07b555f : Updates to Github Actions to allow building with python3.6

fef6c11 : Fix errors when running pylint.sh multiple times

da6f077 : Autopep8 -i fixes

39fe5b3 : TaskManager: fix calling log\_exception with correct number of arguments and minor format changes to reduce PEP8 warnings

17396da : logicengine: get rid of compiler warnings

01dc3d1 : Only track what we need

b609d73 : Configure coveralls (and some minor cleanup)

bd9ed5e : Many C++ cleanups

2a61876 : Add Badges

c864f27 : Do not call pytest fixtures directly.

307db5f : white space fix

882b58f : fix unit tests

1da687c : Replace Boost facilities with C++ STL ones.

5a6e6b1 : Run tests on push

8404245 : Add missing Boost regex library dependency.

ceb5fe7 : Apply clang-format to files that were missed earlier.

3de9940 : Apply clang-format to C++ code.

8a8f560 : Cache venv directory instead

ad017ce : Build private boost for testing

928c64a : Test pip cache

358939a : Adjust CMakeLists.txt files to use correct Python versions

9f0ddb3 : Add pylint github action.

5e6ce4a : Remove more unused C++ files.

63717fe : Setup travis to use new cmake var

74fab2a : Use cmake argument -DPYVER=3.6 to build python3 library <https://fermicloud140.fnal.gov/reviews/r/31/>

843f30c : Minor cleanups per travis-lint

a538cac : Remove unused C++ files.

4c9d125 : Update repo where action is taken from

87fb2d9 : Update rpms installed in docker image. Update entrypoint.sh to use cmake3.

199ee87 : Find python3 libraries using cmake3 from epel rpm Also need to install python3-devel

4c79d2c : Remove unused GNUmakefiles.

94342ee : Add unit test as a Github Action

1a0e102 : more advanced travis.yml

0be413f : Add helper file for pip

7794327 : Make recursive import happy

7005c78 : Add simple target

de8b0fa : python3 compliance: replace string.join() where appropriate, handle UserDict



2662e6c : note required packages  
3b87119 : Add missing header includes.  
3e79b84 : Remove defunct code and its tests  
b1dbe1a : Ensure attribs are defined at **init**  
c4ad78a : Correct logger arguments do avoid duplicate string parse  
a8dcc67 : Remove unused imports (per pylint)  
d3502b5 : Remove obsolete CVS directories.  
d744111 : add six module to the list of required modules  
0a9b1e8 : Fix class declaration  
b83157e : Handle metaclasses  
549f33b : Add config for Travis CI  
ee71044 : Drop trailing white space  
3f82af6 : Python3 forward compatible syntax  
28bf291 : Add safe (for python 2.7) python3 compatible syntax  
1d1d76f : prepare for python3



## DEVELOPER DOCUMENTATION

First command `cd` is just to make sure that you end up in a directory that will contain two subdirectory `decisionengine` and `decisionengine_modules`. Of course this can be done in any directory, not necessarily home directory.

### 2.1 Decisionengine framework

#### 2.1.1 Prerequisites:

```
yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
yum install -y https://download.postgresql.org/pub/repos/yum/repos/EL-7-x86_64/
↳ pgdg-redhat-repo-latest.noarch.rpm
yum install -y python3 python3-pip cmake3 boost-devel python36-devel boost-python36-
↳ devel postgresql11 postgresql11-server
pip3 install pandas DBUtils psycpg2-binary tabulate mock pytest
```

#### 2.1.2 Build & test

```
cd
git clone https://github.com/HEPcloud/decisionengine

export PYTHONPATH=`pwd`

mkdir decisionengine/framework/logicengine/cxx/build
cd decisionengine/framework/logicengine/cxx/build
cmake3 .. -DPYVER=3.6
make -j <number> # say number of CPUs on your box
cd ../../
ln -s cxx/build/ErrorHandler/RE.so
ln -s cxx/build/ErrorHandler/libLogicEngine.so
export LD_LIBRARY_PATH=`pwd`
cd ../../
#pytest -v --tb=native
python3 -m pytest

===== test session starts
↳ =====
platform linux -- Python 3.6.8, pytest-5.3.5, py-1.8.1, pluggy-0.13.1
rootdir: /root/junjk/decisionengine
collected 26 items
```

(continues on next page)

(continued from previous page)

```

framework/dataspace/tests/test_Reaper.py .....
↳ [ 26%]
framework/logicengine/tests/test_cascaded_rules.py ..
↳ [ 34%]
framework/logicengine/tests/test_construction.py .....
↳ [ 53%]
framework/logicengine/tests/test_facts.py .....
↳ [ 73%]
framework/logicengine/tests/test_pandas_fact.py ..
↳ [ 80%]
framework/logicengine/tests/test_rule_with_negated_fact.py ..
↳ [ 88%]
framework/logicengine/tests/test_simple_configuration.py ..
↳ [ 96%]
framework/util/tests/test_tsort.py .
↳ [100%]

```

```

===== 26 passed in 23.86s_
↳=====

```

## 2.2 Decisionengine\_modules

### 2.2.1 Prerequisites:

In Addition to above installed packages

```

yum install condor
pip3 install htcondor boto boto3 google_auth google-api-python-client gcs-oauth2-boto-
↳plugin

```

### 2.2.2 Test

```

cd

git clone https://github.com/HEPCloud/decisionengine_modules
python3 -m pytest decisionengine_modules

```

Current status:

```

[root@fermicloud371 tmp]# python3 -m pytest decisionengine_modules
===== test session starts_
↳=====
platform linux -- Python 3.6.8, pytest-5.3.5, py-1.8.1, pluggy-0.13.1
rootdir: /root/junjk
collected 85 items

decisionengine_modules/AWS/tests/test_AWSInstancePerformance.py ..
↳ [ 2%]
decisionengine_modules/AWS/tests/test_AWSJobLimits.py ..
↳ [ 4%]
decisionengine_modules/AWS/tests/test_AWSOccupancyWithSourceProxy.py ..
↳ [ 7%]

```

(continues on next page)

(continued from previous page)

```

decisionengine_modules/AWS/tests/test_AWSSpotPriceWithSourceProxy.py ..
↳ [ 9%]
decisionengine_modules/AWS/tests/test_AWS_figure_of_merit_publisher.py ..
↳ [ 11%]
decisionengine_modules/AWS/tests/test_AWS_price_performance_publisher.py ..
↳ [ 14%]
decisionengine_modules/AWS/tests/test_FigureOfMerit.py ...
↳ [ 17%]
decisionengine_modules/tests/test_AwsBurnRate.py ..
↳ [ 20%]
decisionengine_modules/tests/test_GCEBillingInfo.py ..
↳ [ 22%]
decisionengine_modules/tests/test_GCEFigureOfMerit_publisher.py ..
↳ [ 24%]
decisionengine_modules/tests/test_GCEInstancePerformanceInfo.py ..
↳ [ 27%]
decisionengine_modules/tests/test_GCEPricePerformance_publisher.py ..
↳ [ 29%]
decisionengine_modules/tests/test_GCEResourceLimits.py ..
↳ [ 31%]
decisionengine_modules/tests/test_GceBurnRate.py ..
↳ [ 34%]
decisionengine_modules/tests/test_GceFigureOfMerit.py ..
↳ [ 36%]
decisionengine_modules/tests/test_GceOccupancy.py ..
↳ [ 38%]
decisionengine_modules/tests/test_NerscAllocationInfo.py ..
↳ [ 41%]
decisionengine_modules/tests/test_NerscFigureOfMerit.py ..
↳ [ 43%]
decisionengine_modules/tests/test_NerscFigureOfMerit_publisher.py ..
↳ [ 45%]
decisionengine_modules/tests/test_NerscInstancePerformance.py ..
↳ [ 48%]
decisionengine_modules/tests/test_NerscJobInfo.py ..
↳ [ 50%]
decisionengine_modules/tests/test_factory_client.py ....
↳ [ 55%]
decisionengine_modules/tests/test_factory_entries.py ....
↳ [ 60%]
decisionengine_modules/tests/test_factory_global.py ....
↳ [ 64%]
decisionengine_modules/tests/test_fomorderplugin.py ....
↳ [ 69%]
decisionengine_modules/tests/test_grid_figure_of_merit.py .
↳ [ 70%]
decisionengine_modules/tests/test_htcondor_query.py ....
↳ [ 75%]
decisionengine_modules/tests/test_job_clustering.py .....
↳ [ 81%]
decisionengine_modules/tests/test_job_clustering_publisher.py ..
↳ [ 83%]
decisionengine_modules/tests/test_job_q.py ...
↳ [ 87%]
decisionengine_modules/tests/test_slots.py ..
↳ [ 89%]
decisionengine_modules/tests/glideinwms/publishers/test_decisionenginemonitor.py ...
↳ [ 92%]

```

(continues on next page)

(continued from previous page)

```
decisionengine_modules/tests/glideinwms/publishers/test_fe_group_classads.py ...
↳ [ 96%]
decisionengine_modules/tests/glideinwms/publishers/test_glideclientglobal.py ...
↳ [100%]

===== warnings summary
↳ =====
/usr/local/lib/python3.6/site-packages/boto/plugin.py:40
  /usr/local/lib/python3.6/site-packages/boto/plugin.py:40: DeprecationWarning: the
↳ imp module is deprecated in favour of importlib; see the module's documentation for
↳ alternative uses
    import imp

-- Docs: https://docs.pytest.org/en/latest/warnings.html
===== 85 passed, 1 warning in 9.73s
↳ =====
```

## SOURCE CODE

### 3.1 Welcome to decisionengine's documentation!

#### 3.1.1 decisionengine package

##### Subpackages

##### decisionengine.framework package

##### Subpackages

##### decisionengine.framework.configmanager package

##### Submodules

##### decisionengine.framework.configmanager.ConfigManager module

```
class decisionengine.framework.configmanager.ConfigManager.ConfigManager
    Bases: object
    check_keys (channel_conf_dict)
        check that channel config has mandatory keys :type data: dict
    static create (module_name, class_name, parameters)
    get_channels ()
    get_global_config ()
    get_produces (channel_config)
    is_updated ()
    load ()
    reload ()
    validate_channel (channel)
        Validate channels :type channel: dict
```

## Module contents

### decisionengine.framework.dataspace package

#### Subpackages

### decisionengine.framework.dataspace.datasources package

#### Subpackages

### decisionengine.framework.dataspace.datasources.tests package

#### Submodules

### decisionengine.framework.dataspace.datasources.tests.test\_postgresql module

```
decisionengine.framework.dataspace.datasources.tests.test_postgresql.data()
decisionengine.framework.dataspace.datasources.tests.test_postgresql.dataprod()
decisionengine.framework.dataspace.datasources.tests.test_postgresql.datasource(postgresql,
                                         data)
decisionengine.framework.dataspace.datasources.tests.test_postgresql.header(data)
decisionengine.framework.dataspace.datasources.tests.test_postgresql.metadata(data)
decisionengine.framework.dataspace.datasources.tests.test_postgresql.taskmanager()
decisionengine.framework.dataspace.datasources.tests.test_postgresql.test_create_tables(data)
decisionengine.framework.dataspace.datasources.tests.test_postgresql.test_generate_insert_c
decisionengine.framework.dataspace.datasources.tests.test_postgresql.test_get_last_generat

decisionengine.framework.dataspace.datasources.tests.test_postgresql.test_get_taskmanager(data)
decisionengine.framework.dataspace.datasources.tests.test_postgresql.test_insert(datasource,
                                         dat-
                                         aprod-
                                         uct,
                                         header,
                                         meta-
                                         data)
decisionengine.framework.dataspace.datasources.tests.test_postgresql.test_store_taskmanager
```



## Module contents

### Submodules

#### decisionengine.framework.dataspace.datasources.postgresql module

**class** decisionengine.framework.dataspace.datasources.postgresql.**Postgresql** (*config\_dict*)  
 Bases: *decisionengine.framework.dataspace.datasource.DataSource*

Implementation of postgresql data source

**\_Postgresql\_\_query** (*query\_string, values=None, cursor\_factory=None*)

**\_abc\_cache** = <\_weakrefset.WeakSet object>

**\_abc\_negative\_cache** = <\_weakrefset.WeakSet object>

**\_abc\_negative\_cache\_version** = 185

**\_abc\_registry** = <\_weakrefset.WeakSet object>

**\_delete** (*sql\_query, values=None*)

**\_insert** (*table\_name\_or\_sql\_query, record=None*)

**\_insert\_returning\_result** (*table\_name\_or\_sql\_query, record=None*)

**\_remove** (*sql\_query, values=None*)

**\_select** (*query\_string, values=None, cursor\_factory=None*)

**\_select\_dictresult** (*sql\_query, values=None*)

**\_select\_getresult** (*sql\_query, values=None*)

**\_select\_tuple** (*sql\_query, values*)

**\_update** (*query\_string, values=None*)

**\_update\_returning\_result** (*query\_string, values=None*)

**close** ()

Close all connections to the database

**connect** ()

Create a pool of database connections

**create\_tables** ()

Create database tables

**delete\_data\_older\_than** (*days*)

Delete data older than days interval :type days: int :arg days: remove data older than days interval

**duplicate\_datablock** (*taskmanager\_id, generation\_id, new\_generation\_id*)

For the given taskmanager\_id, make a copy of the datablock with given generation\_id, set the generation\_id for the datablock copy

#### Parameters

- **taskmanager\_id** (string) – taskmanager\_id for generation to be retrieved
- **generation\_id** (int) – generation\_id of the data
- **new\_generation\_id** (int) – generation\_id of the new datablock created

**get\_connection** ()

**get\_datablock** (*taskmanager\_id*, *generation\_id*)

Return the entire datablock from the dataproduct table for the given taskmanager\_id, generation\_id

**Parameters**

- **taskmanager\_id** (*string*) – taskmanager\_id for generation to be retrieved
- **generation\_id** (*int*) – generation\_id of the data

**get\_dataproduct** (*taskmanager\_id*, *generation\_id*, *key*)

Return the data from the dataproduct table for the given taskmanager\_id, generation\_id, key

**Parameters**

- **taskmanager\_id** (*string*) – taskmanager\_id for generation to be retrieved
- **generation\_id** (*int*) – generation\_id of the data
- **key** (*string*) – key for the value

**get\_header** (*taskmanager\_id*, *generation\_id*, *key*)

Return the header from the header table for the given taskmanager\_id, generation\_id, key

**Parameters**

- **taskmanager\_id** (*string*) – taskmanager\_id for generation to be retrieved
- **generation\_id** (*int*) – generation\_id of the data
- **key** (*string*) – key for the value

**get\_last\_generation\_id** (*taskmanager\_name*, *taskmanager\_id=None*)

Return last generation id for current task manager or taskmanager w/ task\_manager\_id.

**Parameters**

- **name** (*string*) – task manager name
- **taskmanager\_id** (*string*) – task manager id

**get\_metadata** (*taskmanager\_id*, *generation\_id*, *key*)

Return the metadata from the metadata table for the given taskmanager\_id, generation\_id, key

**Parameters**

- **taskmanager\_id** (*string*) – taskmanager\_id for generation to be retrieved
- **generation\_id** (*int*) – generation\_id of the data
- **key** (*string*) – key for the value

**get\_schema** (*table=None*)

Given the table name return it's schema

**Parameters** **table** (*string*) – Name of the table

**get\_taskmanager** (*taskmanager\_name*, *taskmanager\_id=None*)

Retrieve TaskManager :type taskmanager\_name: *string* :arg taskmanager\_name: name of taskmanager to retrieve :type taskmanager\_id: *string* :arg taskmanager\_id: id of taskmanager to retrieve

**insert** (*taskmanager\_id*, *generation\_id*, *key*, *value*, *header*, *metadata*)

Insert data into respective tables for the given taskmanager\_id, generation\_id, key

**Parameters**

- **taskmanager\_id** (*string*) – taskmanager\_id for generation to be retrieved
- **generation\_id** (*int*) – generation\_id of the data

- **key** (string) – key for the value
- **value** (object) – Value can be an object or dict
- **header** (Header) – Header for the value
- **header** – Metadata for the value

**store\_taskmanager** (name, taskmanager\_id)

Store TaskManager :type taskmanager\_name: string :arg taskmanager\_name: name of taskmanager to

retrieve :type taskmanager\_id: string :arg taskmanager\_id: id of taskmanager to retrieve

**tables** = {'dataprodukt': ['taskmanager\_id TEXT', 'generation\_id INT', 'key TEXT', 'va

**update** (taskmanager\_id, generation\_id, key, value, header, metadata)

Update the data in respective tables for the given taskmanager\_id, generation\_id, key

#### Parameters

- **taskmanager\_id** (string) – taskmanager\_id for generation to be retrieved
- **generation\_id** (int) – generation\_id of the data
- **key** (string) – key for the value
- **value** (object) – Value can be an object or dict
- **header** (Header) – Header for the value
- **header** – Metadata for the value

decisionengine.framework.dataspace.datasources.postgresql.**generate\_insert\_query** (table\_name, keys)

Generate insert query given table name and list of fields

#### Parameters

- **table\_name** (str) – Name of the table to insert into
- **keys** – List of column names

**Keys** list

**Return type** str - insert query

## Module contents

### Submodules

#### decisionengine.framework.dataspace.datablock module

**class** decisionengine.framework.dataspace.datablock.**DataBlock** (dataspace, name, taskmanager\_id=None, generation\_id=None, sequence\_id=None)

Bases: object

**\_insert** (key, value, header, metadata)

Insert a new product into database with header and metadata

**\_\_setitem** (*key, value, header, metadata=None*)  
put a product in the database with header and metadata

**\_\_update** (*key, value, header, metadata*)  
Update an existing product in the database with header and metadata

**duplicate** ()  
Duplicate the datablock and return this new DataBlock. The intent is that at the point the duplication occurs there is only information from the sources in the DataBlock. This also increments the generation\_id of this DataBlock.

TODO: Also update the header and the metadata information TODO: Make this threadsafe

**Return type** *DataBlock*

**get** (*key, default=None*)  
Return the value associated with the key in the database

**Return type** *dict*

**get\_header** (*key*)  
Return the Header associated with the key in the database

**Return type** *Header*

**get\_metadata** (*key*)  
Return the metadata associated with the key in the database

**Return type** *Metadata*

**get\_taskmanager** (*taskmanager\_name, taskmanager\_id=None*)  
Retrieve TaskManager :type taskmanager\_name: *string* :arg taskmanager\_name: name of taskmanager to retrieve :type taskmanager\_id: *string* :arg taskmanager\_id: id of taskmanager to retrieve :rtype: :obj: *dict*

The dictionary returned looks like : {'datestamp': datetime.datetime(2017, 12, 20, 17, 37, 17, 503210, tzinfo=psycpg2.tz.FixedOffsetTimezone(offset=-360, name=None)),  
'sequence\_id': 135L, 'name': 'AWS\_Calculations', 'taskmanager\_id': '77B16EB5-C79E-45B0-B1B1-37E846692E1D'}

**is\_expired** (*key=None*)  
Check if the dataproduct for a given key or any key is expired

**keys** ()

**mark\_expired** (*expiration\_time*)  
Set the expiration\_time for the current generation of the dataproduct and mark it as expired if expiration\_time <= current time

**put** (*key, value, header, metadata=None*)  
Put data into the DataBlock

**store\_taskmanager** (*taskmanager\_name, taskmanager\_id*)  
Persist TaskManager, returns sequence number :type taskmanager\_name: *string* :type taskmanager\_id: :obj: *string* :rtype: *int*

**exception** decisionengine.framework.dataspace.datablock.**ExpiredDataError**

Bases: Exception

Errors due to invalid Metadata

**class** decisionengine.framework.dataspace.datablock.**Header** (*taskmanager\_id, create\_time=None, expiration\_time=None, scheduled\_create\_time=None, creator='module', schema\_id=None*)

Bases: collections.UserDict

**\_abc\_cache** = <\_weakrefset.WeakSet object>

**\_abc\_negative\_cache** = <\_weakrefset.WeakSet object>

**\_abc\_negative\_cache\_version** = 185

**\_abc\_registry** = <\_weakrefset.WeakSet object>

**default\_data\_lifetime** = 1800

**is\_valid()**

Check if the Header has minimum required information

**required\_keys** = {'create\_time', 'creator', 'expiration\_time', 'scheduled\_create\_time',

**exception** decisionengine.framework.dataspace.datablock.**InvalidHeaderError**

Bases: Exception

Errors due to invalid Metadata

**exception** decisionengine.framework.dataspace.datablock.**InvalidMetadataError**

Bases: Exception

Errors due to invalid Metadata

**exception** decisionengine.framework.dataspace.datablock.**KeyNotFoundError**

Bases: Exception

Errors due to invalid Metadata

**class** decisionengine.framework.dataspace.datablock.**Metadata** (*taskmanager\_id, state='NEW', generation\_id=None, generation\_time=None, missed\_update\_count=0*)

Bases: collections.UserDict

**\_abc\_cache** = <\_weakrefset.WeakSet object>

**\_abc\_negative\_cache** = <\_weakrefset.WeakSet object>

**\_abc\_negative\_cache\_version** = 185

**\_abc\_registry** = <\_weakrefset.WeakSet object>

**required\_keys** = {'generation\_id', 'generation\_time', 'missed\_update\_count', 'state',

**set\_state** (*state*)

Set the state for the Metadata

**valid\_states** = {'END\_CYCLE', 'METADATA\_UPDATE', 'NEW', 'START\_BACKUP'}

`decisionengine.framework.dataspace.datablock.compress(obj)`

Compress python object :param obj: python object :return: compressed object

`decisionengine.framework.dataspace.datablock.decompress(zbytes)`

Decompress zipped byte stream, convert to string. :param zbytes: byte stream :return: uncompressed string

`decisionengine.framework.dataspace.datablock.zdumps(obj)`

Pickle and compress :param obj: a python object :return: compressed string

`decisionengine.framework.dataspace.datablock.zloads(zbytes)`

Decompress and unpickle If input is not compressed attempts to just unpickle it

**Parameters** `zbytes` – compressed bytes

**Returns** returns python object

### `decisionengine.framework.dataspace.datasource` module

**class** `decisionengine.framework.dataspace.datasource.DataSource(config)`

Bases: object

`_abc_cache = <_weakrefset.WeakSet object>`

`_abc_negative_cache = <_weakrefset.WeakSet object>`

`_abc_negative_cache_version = 185`

`_abc_registry = <_weakrefset.WeakSet object>`

**abstract** `close()`

Close all connections to the database

**abstract** `connect()`

Create a pool of database connections

**abstract** `create_tables()`

Create database tables

`dataprodut_table = 'dataprodut'`

Name of the dataprodut table

**abstract** `delete_data_older_than(days)`

Delete data older that interval :type days: long :arg days: remove data older than interval

**abstract** `duplicate_datablock(taskmanager_id, generation_id, new_generation_id)`

For the given taskmanager\_id, make a copy of the datablock with given generation\_id, set the generation\_id for the datablock copy

#### **Parameters**

- `taskmanager_id`(string) – taskmanager\_id for generation to be retrieved
- `generation_id`(int) – generation\_id of the data
- `new_generation_id`(int) – generation\_id of the new datablock created

**abstract** `get_datablock(taskmanager_id, generation_id)`

Return the entire datablock from the dataprodut table for the given taskmanager\_id, generation\_id

#### **Parameters**

- `taskmanager_id`(string) – taskmanager\_id for generation to be retrieved
- `generation_id`(int) – generation\_id of the data

**abstract get\_dataproduct** (*taskmanager\_id, generation\_id, key*)

Return the data from the dataproduct table for the given taskmanager\_id, generation\_id, key

**Parameters**

- **taskmanager\_id** (*string*) – taskmanager\_id for generation to be retrieved
- **generation\_id** (*int*) – generation\_id of the data
- **key** (*string*) – key for the value

**abstract get\_header** (*taskmanager\_id, generation\_id, key*)

Return the header from the header table for the given taskmanager\_id, generation\_id, key

**Parameters**

- **taskmanager\_id** (*string*) – taskmanager\_id for generation to be retrieved
- **generation\_id** (*int*) – generation\_id of the data
- **key** (*string*) – key for the value

**abstract get\_last\_generation\_id** (*name, taskmanager\_id=None*)

Return last generation id for current task manager or taskmanager w/ task\_manager\_id.

**Parameters**

- **name** (*string*) – task manager name
- **taskmanager\_id** (*string*) – task manager id

**abstract get\_metadata** (*taskmanager\_id, generation\_id, key*)

Return the metadata from the metadata table for the given taskmanager\_id, generation\_id, key

**Parameters**

- **taskmanager\_id** (*string*) – taskmanager\_id for generation to be retrieved
- **generation\_id** (*int*) – generation\_id of the data
- **key** (*string*) – key for the value

**abstract get\_schema** (*table=None*)

Given the table name return it's schema

**Parameters** **table** (*string*) – Name of the table

**abstract get\_taskmanager** (*taskmanager\_name, taskmanager\_id*)

Retrieve TaskManager :type taskmanager\_name: *string* :arg taskmanager\_name: name of taskmanager to retrieve :type taskmanager\_id: *string* :arg taskmanager\_id: id of taskmanager to retrieve

**header\_table = 'header'**

Name of the header table

**abstract insert** (*taskmanager\_id, generation\_id, key, value, header, metadata*)

Insert data into respective tables for the given taskmanager\_id, generation\_id, key

**Parameters**

- **taskmanager\_id** (*string*) – taskmanager\_id for generation to be retrieved
- **generation\_id** (*int*) – generation\_id of the data
- **key** (*string*) – key for the value
- **value** (*object*) – Value can be an object or dict
- **header** (*Header*) – Header for the value

- **header** – Metadata for the value

**metadata\_table** = 'metadata'

Name of the metadata table

**abstract store\_taskmanager** (*taskmanager\_name, taskmanager\_id*)

Store TaskManager :type taskmanager\_name: string :arg taskmanager\_name: name of taskmanager to

retrieve :type taskmanager\_id: string :arg taskmanager\_id: id of taskmanager to retrieve

**taskmanager\_table** = 'taskmanager'

Name of the taskmanager table

**abstract update** (*taskmanager\_id, generation\_id, key, value, header, metadata*)

Update the data in respective tables for the given taskmanager\_id, generation\_id, key

#### Parameters

- **taskmanager\_id** (string) – taskmanager\_id for generation to be retrieved
- **generation\_id** (int) – generation\_id of the data
- **key** (string) – key for the value
- **value** (object) – Value can be an object or dict
- **header** (Header) – Header for the value
- **header** – Metadata for the value

### decisionengine.framework.dataspace.dataspace module

**class** decisionengine.framework.dataspace.dataspace.**DataSourceLoader**

Bases: object

**\_ds** = None

**static create\_datasource** (*module\_name, class\_name, config*)

**class** decisionengine.framework.dataspace.dataspace.**DataSpace** (*config*)

Bases: object

DataSpace class is collection of datablocks and provides interface to the database used to store the actual data

**\_tables\_created** = False

Description of tables and their columns

**close** ()

**delete** (*taskmanager\_id, all\_generations=False*)

**duplicate\_datablock** (*taskmanager\_id, generation\_id, new\_generation\_id*)

**get\_dataproduct** (*taskmanager\_id, generation\_id, key*)

**get\_header** (*taskmanager\_id, generation\_id, key*)

**get\_last\_generation\_id** (*taskmanager\_name, taskmanager\_id=None*)

**get\_metadata** (*taskmanager\_id, generation\_id, key*)

**get\_taskmanager** (*taskmanager\_name, taskmanager\_id=None*)

**insert** (*taskmanager\_id, generation\_id, key, value, header, metadata*)

**mark\_demented** (*taskmanager\_id, keys, generation\_id=None*)



```

mark_expired (taskmanager_id, generation_id, key, expiry_time)

store_taskmanager (name, id)

update (taskmanager_id, generation_id, key, value, header, metadata)

exception decisionengine.framework.dataspace.dataspace.DataSpaceConfigurationError
    Bases: Exception
    Errors related to database access

exception decisionengine.framework.dataspace.dataspace.DataSpaceConnectionError
    Bases: Exception
    Errors related to database access

exception decisionengine.framework.dataspace.dataspace.DataSpaceError
    Bases: Exception
    Errors related to database access

exception decisionengine.framework.dataspace.dataspace.DataSpaceExistsError
    Bases: Exception
    Errors related to database access

class decisionengine.framework.dataspace.dataspace.Reaper (config)
    Bases: object
    Reaper provides functionality of periodic deletion of data older than retention_interval in days

    _reaper_loop (delay)
    _set_state (value)
    get_retention_interval ()
    get_state ()
    reap ()
    set_retention_interval (interval)
    start (delay=0)
        Start thread with an optional delay to start the thread in X seconds
    stop ()

class decisionengine.framework.dataspace.dataspace.Singleton
    Bases: type
    Singleton pattern using Metaclass http://stackoverflow.com/questions/6760685/creating-a-singleton-in-python
    _instances = {}

class decisionengine.framework.dataspace.dataspace.State
    Bases: enum.Enum
    An enumeration.

    ERROR = 7
    IDLE = 1
    RUNNING = 3
    SLEEPING = 4
    STARTING = 2

```

```
STOPPED = 6
STOPPING = 5
```

## Module contents

### decisionengine.framework.engine package

#### Submodules

#### decisionengine.framework.engine.DecisionEngine module

Main loop for Decision Engine. The following environment variable points to decision engine configuration file: `DECISION_ENGINE_CONFIG_FILE` if this environment variable is not defined the `DE-Config.py` file from the `../tests/etc/` directory will be used.

```
class decisionengine.framework.engine.DecisionEngine.DecisionEngine(cfg,
                                                                    server_address,
                                                                    Re-
                                                                    questHandler-
                                                                    Class)
```

Bases: `socketserver.ThreadingMixIn`, `xmlrpc.server.SimpleXMLRPCServer`

```
_dispatch (method, params)
```

Dispatches the XML-RPC method.

XML-RPC calls are forwarded to a registered function that matches the called XML-RPC method name. If no such function exists then the call is forwarded to the registered instance, if available.

If the registered instance has a `_dispatch` method then that method will be called with the name of the XML-RPC method and its parameters as a tuple e.g. `instance._dispatch('add',(2,3))`

If the registered instance does not have a `_dispatch` method then the instance will be searched to find a matching method and, if found, will be called.

Methods beginning with an `'_'` are considered private and will not be called.

```
get_logger ()
```

```
handle_sighup (signum, frame)
```

```
reaper_start (delay)
```

```
reaper_status ()
```

```
reaper_stop ()
```

```
reload_config ()
```

```
rpc_get_channel_log_level (channel)
```

```
rpc_get_log_level ()
```

```
rpc_print_product (product, columns=None, query=None)
```

```
rpc_print_products ()
```

```
rpc_reaper_start (delay=0)
```

Start the reaper process after `'delay'` seconds. Default 0 seconds delay. :type delay: int

```
rpc_reaper_status ()
```

```

rpc_reaper_stop()
rpc_reload_config()
rpc_set_channel_log_level(channel, log_level)
rpc_show_config(channel=None)

rpc_start_channel(channel)
rpc_start_channels()
rpc_status()
rpc_stop()
rpc_stop_channel(channel)
rpc_stop_channels()
start_channel(channel)
start_channels()
stop_channel(channel)
stop_channels()

class decisionengine.framework.engine.DecisionEngine.RequestHandler(request,
                                                                    client_address,
                                                                    server)

    Bases: xmlrpc.server.SimpleXMLRPCRequestHandler

    rpc_paths = ('/RPC2',)

class decisionengine.framework.engine.DecisionEngine.RpcServer(server_address,
                                                                Re-
                                                                questHandler-
                                                                Class)

    Bases: socketserver.ThreadingMixIn, xmlrpc.server.SimpleXMLRPCServer

class decisionengine.framework.engine.DecisionEngine.Worker(task_manager, con-
                                                                fig)

    Bases: multiprocessing.context.Process

    run()
        Method to be run in sub-process; can be overridden in sub-class

decisionengine.framework.engine.DecisionEngine.main(args_to_parse=None)
    If you pass a list of args, they will be used instead of sys.argv

decisionengine.framework.engine.DecisionEngine.main_create_parser()

```

## decisionengine.framework.engine.de\_client module

```
decisionengine.framework.engine.de_client.build_xmlrpc_connection(host, port)
decisionengine.framework.engine.de_client.create_parser()
decisionengine.framework.engine.de_client.execute_command_from_args(argsparsed,
                                                                    xmlrpc-
                                                                    socket)

    argsparsed should be from create_parser in this file
decisionengine.framework.engine.de_client.main(args_to_parse=None)
    If you pass a list of args, they will be used instead of sys.argv
```

## Module contents

### decisionengine.framework.modules package

#### Submodules

### decisionengine.framework.modules.LogicEngine module

```
class decisionengine.framework.modules.LogicEngine.LogicEngine(set_of_parameters)
    Bases: decisionengine.framework.modules.Module.Module

    evaluate(data_block)
```

### decisionengine.framework.modules.Module module

```
class decisionengine.framework.modules.Module.Module(set_of_parameters)
    Bases: object

    get_data_block()
    get_paramaters()
    set_data_block(data_block)
```

### decisionengine.framework.modules.Publisher module

```
class decisionengine.framework.modules.Publisher.Publisher(set_of_parameters)
    Bases: decisionengine.framework.modules.Module.Module

    consumes(name_list)
    publish(data_block=None)
```

**decisionengine.framework.modules.Source module**

```
class decisionengine.framework.modules.Source.Source (set_of_parameters)
    Bases: decisionengine.framework.modules.Module.Module

    acquire ()

    produces (name_schema_id_list)
```

**decisionengine.framework.modules.SourceProxy module**

Fill in data from another channel data block

```
class decisionengine.framework.modules.SourceProxy.SourceProxy (*args,
                                                                **kwargs)
    Bases: decisionengine.framework.modules.Source.Source
```

Source Proxy Channel configuration using source proxy must have in parameters 'channel\_name', defining foreign channel name and 'Dataproducts', defining foreign (and optionally local) data keys. See consumes() doc. Example of source proxy configuration:

```
    "AWSJobLimits": { "module": "modules.source_proxy", "name": "SourceProxy", "parameters":
    {"channel_name": "channel_aws_config_data",
        "Dataproducts": [("aws_instance_limits",    "Job_Limits")],    "retries":    3,
        "retry_timeout": 20,
    },
    "schedule": 360,
},
_get_data (data_block, key)

acquire ()
    Overrides Source class method

consumes ()
    Assumes that self.datakeys has the following structure: is a list of tuples or singletons: [
        (data_product_name, data_product_name_translation), .... ] or [ data_product_name, ....
    ]

must_have = ('channel_name', 'Dataproducts')

produces ()
```

Assumes that self.datakeys has the following structure or

```
decisionengine.framework.modules.SourceProxy.main ()
    Call this a a test unit or use as CLI of this module
```

```
decisionengine.framework.modules.SourceProxy.module_config_info ()
    print this module configuration information
```

```
decisionengine.framework.modules.SourceProxy.module_config_template ()
    print a template for this module configuration data
```

## decisionengine.framework.modules.Transform module

```
class decisionengine.framework.modules.Transform.Transform(set_of_parameters)
    Bases: decisionengine.framework.modules.Module.Module

    consumes (name_list)

    produces (name_schema_id_list)

    transform()
```

## decisionengine.framework.modules.de\_logger module

Looger to use in all modules

```
decisionengine.framework.modules.de_logger.get_logger()
    get default logger - "decision_engine":rtype: logging.Logger - rotating file logger
```

```
decisionengine.framework.modules.de_logger.set_logging(log_file_name='/tmp/decision_engine_logs/decision_
                                                         max_file_size=200000000,
                                                         max_backup_count=6,
                                                         log_level='DEBUG')
```

### Parameters

- **log\_file\_name** (str) – log file name
- **max\_file\_size** (int) – maximal size of log file. If reached save and start new log.
- **max\_backup\_count** (int) – start rotaion after this number is reached

**Return type** logging.Logger - rotating file logger

```
decisionengine.framework.modules.de_logger.set_stream_logging(logger_name="")
    This is for debugging. Set stream logging for logger.
```

**Parameters** **logger\_name** (str) – logger name

**Return type** logging.Logger

## Module contents

### decisionengine.framework.taskmanager package

#### Submodules

### decisionengine.framework.taskmanager.TaskManager module

Task Manager

```
class decisionengine.framework.taskmanager.TaskManager.Channel(channel_dict)
    Bases: object

    Decision Channel. Instantiates workers according to channel configuration
```

```
class decisionengine.framework.taskmanager.TaskManager.TaskManager (name,  
task_manager_id,  
genera-  
tion_id,  
chan-  
nel_dict,  
global_config)
```

Bases: object

Task Manager

**data\_block\_put** (*data, header, data\_block*)

Put data into data block

**Parameters**

- **data** (dict) – key, value pairs
- **header** (Header) – data header
- **data\_block** (DataBlock) – data block

**decision\_cycle** ()

Decision cycle to be run periodically (by trigger)

**do\_backup** ()

Duplicate current data block and return its copy

**Return type** DataBlock

**get\_loglevel** ()

**get\_state** ()

**offline\_task\_manager** (*current\_data\_block*)

offline and stop task manager

**run** ()

Task Manager main loop

**run\_logic\_engine** (*data\_block=None*)

Run Logic Engine.

**Parameters** **data\_block** (DataBlock) – data block

**run\_publishers** (*actions, facts, data\_block=None*)

Run Publishers in main process.

**Parameters** **data\_block** (DataBlock) – data block

**run\_source** (*src*)

Get the data from source and put it into the data block

**Parameters** **src** (*Worker*) – source Worker

**run\_transform** (*transform, data\_block*)

Run a transform

**Parameters**

- **transform** (*Worker*) – source Worker
- **data\_block** (DataBlock) – data block

**run\_transforms** (*data\_block=None*)

Run transforms. So far in main process.

**Parameters** `data_block` (`DataBlock`) – data block

**set\_loglevel** (`log_level`)

**set\_state** (`state`)

**start\_sources** (`data_block=None`)

Start sources, each in a separate thread

**Parameters** `data_block` (`DataBlock`) – data block

**stop\_task\_manager** ()

signal task manager to stop

**wait\_for\_all** (`events_done`)

Wait for all sources or transforms to finish

**Parameters** `events_done` (`list`) – list of events to wait for

**wait\_for\_any** (`events_done`)

Wait for any sources to finish

**Parameters** `events_done` (`list`) – list of events to wait for

**class** `decisionengine.framework.taskmanager.TaskManager.Worker` (`conf_dict`)

Bases: `object`

Provides interface to loadable modules and events to synchronise execution

**DEFAULT\_SCHEDULE** = 300

`decisionengine.framework.taskmanager.TaskManager.log_exception` (`logger`,  
`header_message`)

## Module contents

### decisionengine.framework.util package

#### Submodules

#### decisionengine.framework.util.tsort module

See:

[https://en.wikipedia.org/wiki/Topological\\_sorting](https://en.wikipedia.org/wiki/Topological_sorting)

Kahn's topological sorting algorithm

L Empty list that will contain the sorted elements S Set of all nodes with no incoming edge while S is non-empty do

    remove a node n from S add n to tail of L for each node m with an edge e from n to m do

        remove edge e from the graph if m has no other incoming edges then

            insert m into S

**if graph has edges then** return error (graph has at least one cycle)

**else** return L (a topologically sorted order)

`decisionengine.framework.util.tsort.tsort` (`graph`)

Function implementing Kahn's topological sorting algorithm returns two lists : sorted list and cyclic lost (if graph is acyclic second list is always None)



**Return type** `list`

**Module contents**

**Module contents**

**Module contents**

## 3.2 Indices and tables

- [genindex](#)
- [modindex](#)
- [search](#)



## INDICES AND TABLES

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



## PYTHON MODULE INDEX

### d

`decisionengine`, [29](#)  
`decisionengine.framework`, [29](#)  
`decisionengine.framework.configmanager`,  
    [12](#)  
`decisionengine.framework.configmanager.ConfigManager`,  
    [11](#)  
`decisionengine.framework.dataspace`, [22](#)  
`decisionengine.framework.dataspace.datablock`,  
    [15](#)  
`decisionengine.framework.dataspace.datasource`,  
    [18](#)  
`decisionengine.framework.dataspace.datasources`,  
    [15](#)  
`decisionengine.framework.dataspace.datasources.postgresql`,  
    [13](#)  
`decisionengine.framework.dataspace.datasources.tests`,  
    [13](#)  
`decisionengine.framework.dataspace.datasources.tests.test_postgresql`,  
    [12](#)  
`decisionengine.framework.dataspace.dataspace`,  
    [20](#)  
`decisionengine.framework.engine`, [24](#)  
`decisionengine.framework.engine.de_client`,  
    [24](#)  
`decisionengine.framework.engine.DecisionEngine`,  
    [22](#)  
`decisionengine.framework.modules`, [26](#)  
`decisionengine.framework.modules.de_logger`,  
    [26](#)  
`decisionengine.framework.modules.LogicEngine`,  
    [24](#)  
`decisionengine.framework.modules.Module`,  
    [24](#)  
`decisionengine.framework.modules.Publisher`,  
    [24](#)  
`decisionengine.framework.modules.Source`,  
    [25](#)  
`decisionengine.framework.modules.SourceProxy`,  
    [25](#)  
`decisionengine.framework.modules.Transform`,  
    [26](#)  
`decisionengine.framework.taskmanager`,  
    [28](#)  
`decisionengine.framework.taskmanager.TaskManager`,  
    [26](#)  
`decisionengine.framework.util`, [29](#)  
`decisionengine.framework.util.tsort`, [28](#)



# INDEX

## Symbols

<code>_Postgresql__query()</code>	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 13	<code>_abc_registry</code>	(decisionengine.framework.dataspace.datablock.Metadata attribute), 17
<code>_abc_cache</code>	(decisionengine.framework.dataspace.datablock.Header attribute), 17	<code>_abc_registry</code>	(decisionengine.framework.dataspace.datasource.DataSource attribute), 18
<code>_abc_cache</code>	(decisionengine.framework.dataspace.datablock.Metadata attribute), 17	<code>_abc_registry</code>	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql attribute), 13
<code>_abc_cache</code>	(decisionengine.framework.dataspace.datasource.DataSource attribute), 18	<code>_delete()</code>	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 13
<code>_abc_cache</code>	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql attribute), 13	<code>_dispatch()</code>	(decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 22
<code>_abc_negative_cache</code>	(decisionengine.framework.dataspace.datablock.Header attribute), 17	<code>_ds</code>	(decisionengine.framework.dataspace.datasource.Loader attribute), 20
<code>_abc_negative_cache</code>	(decisionengine.framework.dataspace.datablock.Metadata attribute), 17	<code>_get_data()</code>	(decisionengine.framework.modules.SourceProxy.SourceProxy method), 25
<code>_abc_negative_cache</code>	(decisionengine.framework.dataspace.datablock.Metadata attribute), 17	<code>_insert()</code>	(decisionengine.framework.dataspace.datablock.DataBlock method), 15
<code>_abc_negative_cache</code>	(decisionengine.framework.dataspace.datasource.DataSource attribute), 18	<code>_insert()</code>	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 13
<code>_abc_negative_cache</code>	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql attribute), 13	<code>_insert_returning_result()</code>	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 13
<code>_abc_negative_cache_version</code>	(decisionengine.framework.dataspace.datablock.Header attribute), 17	<code>_instances</code>	(decisionengine.framework.dataspace.datasource.Singleton attribute), 21
<code>_abc_negative_cache_version</code>	(decisionengine.framework.dataspace.datablock.Metadata attribute), 17	<code>_reaper_loop()</code>	(decisionengine.framework.dataspace.datasource.Reaper method), 21
<code>_abc_negative_cache_version</code>	(decisionengine.framework.dataspace.datasource.DataSource attribute), 18	<code>_remove()</code>	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 13
<code>_abc_negative_cache_version</code>	(decisionengine.framework.dataspace.datasource.DataSource attribute), 18	<code>_select()</code>	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 13
<code>_abc_negative_cache_version</code>	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql attribute), 13	<code>_select_dictresult()</code>	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 13
<code>_abc_registry</code>	(decisionengine.framework.dataspace.datablock.Header attribute), 17	<code>_select_getresult()</code>	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 13

**A**

```

method), 13
_select_tuple() (decisionengine.framework.modules.Publisher.Publisher
nengine.framework.dataspace.datasources.postgresql.Postgresql), 24
method), 13
consumes() (decisionengine.framework.modules.SourceProxy.SourceProxy
method), 25
_set_state() (decisionengine.framework.modules.Transform.Transform
method), 26
nengine.framework.dataspace.datablock.DataBlock
method), 15
create() (decisionengine.framework.configmanager.ConfigManager.ConfigManager
static method), 11
_nengine.framework.dataspace.dataspace.DataSpace.create_datasource() (decisionengine.framework.dataspace.dataspace.DataSourceLoader
attribute), 20
static method), 20
_update() (decisionengine.framework.dataspace.datablock.DataBlock
method), 16
create_parser() (in module decisionengine.framework.engine.de_client), 24
_update() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql
method), 13
create_tables() (decisionengine.framework.dataspace.datasource.DataSource
method), 18
_update_returning_result() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql
method), 13
create_tables() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql
method), 13

```

**B**

```

acquire() (decisionengine.framework.modules.Source.Source
method), 25
acquire() (decisionengine.framework.modules.SourceProxy.SourceProxy (in module decisionengine.framework.dataspace.datasources.tests.test_postgresql),
method), 25
12

```

**C**

```

Channel (class in decisionengine.framework.taskmanager.TaskManager), 26
check_keys() (decisionengine.framework.configmanager.ConfigManager.ConfigManager
method), 11
close() (decisionengine.framework.dataspace.datasource.DataSource
method), 18
close() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql
method), 13
close() (decisionengine.framework.dataspace.dataspace.DataSpace
method), 20
compress() (in module decisionengine.framework.dataspace.datablock), 17
ConfigManager (class in decisionengine.framework.configmanager.ConfigManager), 11
connect() (decisionengine.framework.dataspace.datasource.DataSource
method), 18
connect() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql
method), 13

```

**D**

```

data_block_put() (decisionengine.framework.taskmanager.TaskManager.TaskManager
method), 27
DataBlock (class in decisionengine.framework.dataspace.datablock), 15
dataprodut() (in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 12
dataprodut_table (decisionengine.framework.dataspace.datasource.DataSource
attribute), 18
DataSource (class in decisionengine.framework.dataspace.datasource), 18
DataSource() (in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 12
DataSourceLoader (class in decisionengine.framework.dataspace.dataspace), 20
DataSpace (class in decisionengine.framework.dataspace.dataspace), 20
DataSpaceConfigurationError, 21
DataSpaceConnectionError, 21
DataSpaceError, 21

```



DataSpaceExistsError, 21  
 decision\_cycle() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 27  
 decisionengine module, 29  
 DecisionEngine (class in decisionengine.framework.engine.DecisionEngine), 22  
 decisionengine.framework module, 29  
 decisionengine.framework.configmanager module, 12  
 decisionengine.framework.configmanager.DEFAULT\_MANAGER\_SCHEDULE (decisionengine.framework.taskmanager.TaskManager.Worker attribute), 28  
 decisionengine.framework.dataspace module, 22  
 decisionengine.framework.dataspace.datablock module, 15  
 decisionengine.framework.dataspace.datasource module, 18  
 decisionengine.framework.dataspace.datasources module, 13  
 decisionengine.framework.dataspace.datasources.postgresql module, 13  
 decisionengine.framework.dataspace.datasources.postgresql.test\_postgresql module, 12  
 decisionengine.framework.dataspace.dataspace module, 20  
 decisionengine.framework.engine module, 24  
 decisionengine.framework.engine.de\_client module, 24  
 decisionengine.framework.engine.DecisionEngine module, 22  
 decisionengine.framework.modules module, 26  
 decisionengine.framework.modules.de\_logger module, 26  
 decisionengine.framework.modules.LogicEngine module, 24  
 decisionengine.framework.modules.Module module, 24  
 decisionengine.framework.modules.Publisher module, 24  
 decisionengine.framework.modules.Source module, 25  
 decisionengine.framework.modules.SourceProxy module, 25  
 decisionengine.framework.modules.Transform module, 26  
 decisionengine.framework.taskmanager module, 28  
 decisionengine.framework.taskmanager.TaskManager module, 26  
 decisionengine.framework.util module, 29  
 decisionengine.framework.util.tsort module, 28  
 decompress() (in module decisionengine.framework.dataspace.datablock), 18  
 default\_data\_lifetime (decisionengine.framework.dataspace.datablock.Header attribute), 17  
 delete() (decisionengine.framework.dataspace.dataspace.DataSpace method), 20  
 delete\_data\_older\_than() (decisionengine.framework.dataspace.datasource.DataSource method), 18  
 delete\_data\_older\_than() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 13  
 do\_backup() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 27  
 duplicate\_data\_sources.test\_postgresql (decisionengine.framework.dataspace.datablock.DataBlock method), 16  
 duplicate\_datablock() (decisionengine.framework.dataspace.datasource.DataSource method), 18  
 duplicate\_datablock() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 13  
 duplicate\_datablock() (decisionengine.framework.dataspace.dataspace.DataSpace method), 20  
 ERROR (decisionengine.framework.dataspace.dataspace.State attribute), 21  
 evaluate() (decisionengine.framework.modules.LogicEngine.LogicEngine method), 24  
 execute\_command\_from\_args() (in module decisionengine.framework.engine.de\_client), 24  
 ExpiredDataError, 16  
 generate\_insert\_query() (in module decisionengine.framework.dataspace.datasources.postgresql), 15

`get ()` (*decisionengine.framework.dataspace.datablock.DataBlock* *nengine.framework.modules.de\_logger*,  
*method*), 16 26  
`get_channels ()` (*decisionengine.framework.configmanager.ConfigManager.ConfigManager* *nengine.framework.taskmanager.TaskManager.TaskManager*  
*method*), 11 *method*), 27  
`get_connection ()` (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql* *nengine.framework.dataspace.datablock.DataBlock*  
*method*), 13 *method*), 16  
`get_data_block ()` (*decisionengine.framework.modules.Module.Module* *nengine.framework.dataspace.datasource.DataSource*  
*method*), 24 *method*), 19  
`get_datablock ()` (*decisionengine.framework.dataspace.datasource.DataSource* *nengine.framework.dataspace.datasources.postgresql.Postgresql*  
*method*), 18 *method*), 14  
`get_datablock ()` (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql* *nengine.framework.dataspace.dataspace.DataSpace*  
*method*), 13 *method*), 20  
`get_dataproduct ()` (*decisionengine.framework.dataspace.datasource.DataSource* *nengine.framework.modules.Module.Module*  
*method*), 18 *method*), 24  
`get_dataproduct ()` (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql* *nengine.framework.configmanager.ConfigManager.ConfigManager*  
*method*), 14 *method*), 11  
`get_dataproduct ()` (*decisionengine.framework.dataspace.dataspace.DataSpace* *nengine.framework.dataspace.dataspace.Reaper*  
*method*), 20 *method*), 21  
`get_global_config ()` (*decisionengine.framework.configmanager.ConfigManager.ConfigManager* *nengine.framework.dataspace.datasource.DataSource*  
*method*), 11 *method*), 19  
`get_header ()` (*decisionengine.framework.dataspace.datablock.DataBlock* *nengine.framework.dataspace.datasources.postgresql.Postgresql*  
*method*), 16 *method*), 14  
`get_header ()` (*decisionengine.framework.dataspace.datasource.DataSource* *nengine.framework.dataspace.dataspace.Reaper*  
*method*), 19 *method*), 21  
`get_header ()` (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql* *nengine.framework.taskmanager.TaskManager.TaskManager*  
*method*), 14 *method*), 27  
`get_header ()` (*decisionengine.framework.dataspace.dataspace.DataSpace* *nengine.framework.dataspace.datablock.DataBlock*  
*method*), 20 *method*), 16  
`get_last_generation_id ()` (*decisionengine.framework.dataspace.datasource.DataSource* *nengine.framework.dataspace.datasource.DataSource*  
*method*), 19 *method*), 19  
`get_last_generation_id ()` (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql* *nengine.framework.dataspace.datasources.postgresql.Postgresql*  
*method*), 14 *method*), 14  
`get_last_generation_id ()` (*decisionengine.framework.dataspace.dataspace.DataSpace* *nengine.framework.dataspace.dataspace.DataSpace*  
*method*), 20 *method*), 20  
`get_logger ()` (*decisionengine.framework.engine.DecisionEngine.DecisionEngine* *handle\_sighup ()*  
*method*), 22 *method*), 22  
`get_logger ()` (*in module decisionengine.framework.engine.DecisionEngine.DecisionEngine*

method), 22  
 Header (class in decisionengine.framework.dataspace.datablock), 17  
 header() (in module decisionengine.framework.dataspace.datasources.tests.test\_postgresql), 12  
 header\_table (decisionengine.framework.dataspace.dataspace.DataSource attribute), 19  
**I**  
 IDLE (decisionengine.framework.dataspace.dataspace.DataSource attribute), 21  
 insert() (decisionengine.framework.dataspace.datasource.DataSource method), 19  
 insert() (decisionengine.framework.dataspace.datasources.postgresql.PostgreSQL method), 14  
 insert() (decisionengine.framework.dataspace.dataspace.DataSource method), 20  
 InvalidHeaderError, 17  
 InvalidMetadataError, 17  
 is\_expired() (decisionengine.framework.dataspace.datablock.DataBlock method), 16  
 is\_updated() (decisionengine.framework.configmanager.ConfigManager.ConfigManager method), 11  
 is\_valid() (decisionengine.framework.dataspace.datablock.Header method), 17  
**K**  
 KeyNotFoundError, 17  
 keys() (decisionengine.framework.dataspace.datablock.DataBlock method), 16  
**L**  
 load() (decisionengine.framework.configmanager.ConfigManager.ConfigManager method), 11  
 log\_exception() (in module decisionengine.framework.taskmanager.TaskManager), 28  
 LogicEngine (class in decisionengine.framework.modules.LogicEngine), 24  
**M**  
 main() (in module decisionengine.framework.engine.de\_client), 24  
 main() (in module decisionengine.framework.engine.DecisionEngine), 23  
 main() (in module decisionengine.framework.modules.SourceProxy), 25  
 main\_create\_parser() (in module decisionengine.framework.engine.DecisionEngine), 23  
 mark\_demented() (decisionengine.framework.dataspace.dataspace.DataSpace method), 20  
 mark\_expired() (decisionengine.framework.dataspace.datablock.DataBlock method), 16  
 mark\_expired() (decisionengine.framework.dataspace.dataspace.DataSpace method), 20  
 Metadata (class in decisionengine.framework.dataspace.datablock), 17  
 metadata() (in module decisionengine.framework.dataspace.datasources.tests.test\_postgresql), 12  
 metadata\_table (decisionengine.framework.datasource.DataSource attribute), 20  
 module  
 decisionengine, 29  
 decisionengine.framework, 29  
 decisionengine.framework.configmanager, 12  
 decisionengine.framework.configmanager.ConfigManager, 11  
 decisionengine.framework.dataspace, 22  
 decisionengine.framework.dataspace.datablock, 15  
 decisionengine.framework.dataspace.datasource, 18  
 decisionengine.framework.dataspace.datasources, 15  
 decisionengine.framework.dataspace.datasources.postgresql, 13  
 decisionengine.framework.dataspace.datasources.postgresql.PostgreSQL, 13  
 decisionengine.framework.dataspace.datasources.postgresql.PostgreSQLPostgreSQL, 12  
 decisionengine.framework.dataspace.dataspace, 20  
 decisionengine.framework.engine, 24  
 decisionengine.framework.engine.de\_client, 24  
 decisionengine.framework.engine.DecisionEngine, 22  
 decisionengine.framework.modules, 26  
 decisionengine.framework.modules.de\_logger,

26 put () (*decisionengine.framework.dataspace.datablock.DataBlock*  
*decisionengine.framework.modules.LogicEngine* method), 16  
 24  
 24 **R**  
*decisionengine.framework.modules.Module*,  
 24 reap () (*decisionengine.framework.dataspace.dataspace.Reaper*  
*decisionengine.framework.modules.Publisher*, method), 21  
 24 Reaper (class in *decision-*  
*decisionengine.framework.modules.Source*, *engine.framework.dataspace.dataspace*),  
 25 21  
*decisionengine.framework.modules.SourceProxy*,  
 25 reaper\_start () (*decision-*  
*engine.framework.engine.DecisionEngine.DecisionEngine*  
*decisionengine.framework.modules.Transform*, method), 22  
 26 reaper\_status () (*decision-*  
*decisionengine.framework.taskmanager*, *engine.framework.engine.DecisionEngine.DecisionEngine*  
 28 method), 22  
*decisionengine.framework.taskmanager.TaskManager*,  
 26 reaper\_stop () (*decision-*  
*engine.framework.engine.DecisionEngine.DecisionEngine*  
*decisionengine.framework.util*, 29 method), 22  
*decisionengine.framework.util.tsort*, reload () (*decisionengine.framework.configmanager.ConfigManager.ConfigManager*  
 28 method), 11  
Module (class in *decision-* reload\_config () (*decision-*  
*engine.framework.modules.Module*), 24 *engine.framework.engine.DecisionEngine.DecisionEngine*  
module\_config\_info () (in module *decision-* method), 22  
*engine.framework.modules.SourceProxy*),  
 25 RequestHandler (class in *decision-*  
*engine.framework.engine.DecisionEngine*),  
module\_config\_template () (in module *decision-* 23  
*engine.framework.modules.SourceProxy*), 25 required\_keys (*decision-*  
*must\_have* (*decisionengine.framework.modules.SourceProxy.SourceProxy*, *engine.framework.dataspace.datablock.Header*  
attribute), 25 attribute), 17  
**O** required\_keys (*decision-*  
*engine.framework.dataspace.datablock.Metadata*  
offline\_task\_manager () (*decision-* attribute), 17  
*engine.framework.taskmanager.TaskManager.TaskManager*,  
method), 27 *channel\_log\_level* () (*decision-*  
*engine.framework.engine.DecisionEngine.DecisionEngine*  
method), 22  
**P** rpc\_get\_log\_level () (*decision-*  
*engine.framework.engine.DecisionEngine.DecisionEngine*  
Postgresql (class in *decision-* method), 22  
*engine.framework.dataspace.datasources.postgresql*),  
 13 rpc\_paths (*decisionengine.framework.engine.DecisionEngine.RequestH*  
produces () (*decision-* attribute), 23  
*engine.framework.modules.Source.Source* rpc\_print\_product () (*decision-*  
method), 25 *engine.framework.engine.DecisionEngine.DecisionEngine*  
method), 22  
produces () (*decision-* rpc\_print\_products () (*decision-*  
*engine.framework.modules.SourceProxy.SourceProxy*, *engine.framework.engine.DecisionEngine.DecisionEngine*  
method), 25 method), 22  
produces () (*decision-* rpc\_reaper\_start () (*decision-*  
*engine.framework.modules.Transform.Transform* *engine.framework.engine.DecisionEngine.DecisionEngine*  
method), 26 method), 22  
publish () (*decisionengine.framework.modules.Publisher.Publisher* method), 22  
method), 24 rpc\_reaper\_status () (*decision-*  
*engine.framework.engine.DecisionEngine.DecisionEngine*  
Publisher (class in *decision-* method), 22  
*engine.framework.modules.Publisher*),  
 24 rpc\_reaper\_stop () (*decision-*  
*engine.framework.engine.DecisionEngine.DecisionEngine*

## S

*method*), 22  
 rpc\_reload\_config() (decisionengine.framework.engine.DecisionEngine.*DecisionEngine* *set\_data\_block()* (decisionengine.framework.modules.Module.*Module* *method*), 23  
 rpc\_set\_channel\_log\_level() (decisionengine.framework.engine.DecisionEngine.*DecisionEngine* *set\_logging()* (in module decisionengine.framework.modules.de\_logger), *method*), 23  
 rpc\_show\_config() (decisionengine.framework.engine.DecisionEngine.*DecisionEngine* *set\_loglevel()* (decisionengine.framework.taskmanager.TaskManager.*TaskManager* *method*), 23  
 rpc\_start\_channel() (decisionengine.framework.engine.DecisionEngine.*DecisionEngine* *set\_retention\_interval()* (decisionengine.framework.dataspace.dataspace.Reaper *method*), 23  
 rpc\_start\_channels() (decisionengine.framework.engine.DecisionEngine.*DecisionEngine* *set\_state()* (decisionengine.framework.dataspace.datablock.Metadata *method*), 23  
 rpc\_status() (decisionengine.framework.engine.DecisionEngine.*DecisionEngine* *set\_state()* (decisionengine.framework.taskmanager.TaskManager.*TaskManager* *method*), 23  
 rpc\_stop() (decisionengine.framework.engine.DecisionEngine.*DecisionEngine* *set\_stream\_logging()* (in module decisionengine.framework.modules.de\_logger), *method*), 23  
 rpc\_stop\_channel() (decisionengine.framework.engine.DecisionEngine.*DecisionEngine* *Singleton* (class in decisionengine.framework.dataspace.dataspace), *method*), 23  
 rpc\_stop\_channels() (decisionengine.framework.engine.DecisionEngine.*DecisionEngine* *SLEEPING* (decisionengine.framework.dataspace.dataspace.State *attribute*), 21  
 RpcServer (class in decisionengine.framework.engine.DecisionEngine), *SourceProxy* (class in decisionengine.framework.modules.Source), 25  
 run() (decisionengine.framework.engine.DecisionEngine.*Worker* *start()* (decisionengine.framework.dataspace.dataspace.Reaper *method*), 23  
 run() (decisionengine.framework.taskmanager.TaskManager.*TaskManager* *start\_channel()* (decisionengine.framework.engine.DecisionEngine.*DecisionEngine* *method*), 27  
 run\_logic\_engine() (decisionengine.framework.taskmanager.TaskManager.*TaskManager* *start\_channels()* (decisionengine.framework.engine.DecisionEngine.*DecisionEngine* *method*), 27  
 run\_publishers() (decisionengine.framework.taskmanager.TaskManager.*TaskManager* *start\_sources()* (decisionengine.framework.taskmanager.TaskManager.*TaskManager* *method*), 27  
 run\_source() (decisionengine.framework.taskmanager.TaskManager.*TaskManager* *STARTING* (decisionengine.framework.dataspace.dataspace.State *attribute*), 21  
 run\_transform() (decisionengine.framework.taskmanager.TaskManager.*TaskManager* *State* (class in decisionengine.framework.dataspace.dataspace), *method*), 27  
 run\_transforms() (decisionengine.framework.taskmanager.TaskManager.*TaskManager* *stop()* (decisionengine.framework.dataspace.dataspace.Reaper *method*), 27  
 RUNNING (decisionengine.framework.dataspace.dataspace.State *attribute*), 21  
 stop\_channel() (decisionengine.framework.engine.DecisionEngine.*DecisionEngine* *method*), 23  
 stop\_channels() (decisionengine.framework.engine.DecisionEngine.*DecisionEngine* *method*), 23



method), 23

stop\_task\_manager() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 26

stop\_task\_manager() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 28

STOPPED (decisionengine.framework.dataspace.dataspace.State attribute), 21

STOPPING (decisionengine.framework.dataspace.dataspace.State attribute), 22

store\_taskmanager() (decisionengine.framework.dataspace.datablock.DataBlock method), 16

store\_taskmanager() (decisionengine.framework.dataspace.datasource.DataSource method), 20

store\_taskmanager() (decisionengine.framework.dataspace.datasource.DataSource method), 20

store\_taskmanager() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 15

store\_taskmanager() (decisionengine.framework.dataspace.dataspace.DataSpace method), 21

**T**

tables (decisionengine.framework.dataspace.datasources.postgresql.Postgresql attribute), 15

TaskManager (class in decisionengine.framework.taskmanager.TaskManager), 26

taskmanager() (in module decisionengine.framework.dataspace.datasources.tests.test\_postgresql), 12

taskmanager\_table (decisionengine.framework.dataspace.datasource.DataSource attribute), 20

test\_create\_tables() (in module decisionengine.framework.dataspace.datasources.tests.test\_postgresql), 12

test\_generate\_insert\_query() (in module decisionengine.framework.dataspace.datasources.tests.test\_postgresql), 12

test\_get\_last\_generation\_id() (in module decisionengine.framework.dataspace.datasources.tests.test\_postgresql), 12

test\_get\_taskmanager() (in module decisionengine.framework.dataspace.datasources.tests.test\_postgresql), 12

test\_insert() (in module decisionengine.framework.dataspace.datasources.tests.test\_postgresql), 12

test\_store\_taskmanager() (in module decisionengine.framework.dataspace.datasources.tests.test\_postgresql), 12

Transform (class in decisionengine.framework.modules.Transform), 26

transform() (decisionengine.framework.modules.Transform.Transform method), 26

tsort() (in module decisionengine.framework.util.tsort), 28

**U**

update() (decisionengine.framework.dataspace.datasource.DataSource method), 20

update() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 15

update() (decisionengine.framework.dataspace.dataspace.DataSpace method), 21

**V**

valid\_states (decisionengine.framework.dataspace.datablock.Metadata attribute), 17

validate\_channel() (decisionengine.framework.configmanager.ConfigManager.ConfigManager method), 15

**W**

wait\_for\_all() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 28

wait\_for\_any() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 28

Worker (class in decisionengine.framework.engine.DecisionEngine), 23

Worker (class in decisionengine.framework.taskmanager.TaskManager), 28

z.dumps() (in module decisionengine.framework.dataspace.datablock), 18

z.loads() (in module decisionengine.framework.dataspace.datablock), 18