

---

# **decisionengine**

***Release 1.1.1***

**Fermi Research Alliance, LLC.**

**Oct 02, 2020**



# CONTENTS

<b>1</b>	<b>Release Notes</b>	<b>3</b>
<b>2</b>	<b>Developer Documentation</b>	<b>9</b>
<b>3</b>	<b>Source code</b>	<b>13</b>
<b>4</b>	<b>Indices and tables</b>	<b>39</b>
	<b>Python Module Index</b>	<b>41</b>
	<b>Index</b>	<b>43</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 1.3.0

In this release:

- Introduced Jsonnet based configuration system
- Improved logging
- Improved coverage of datasource

#### 1.1.1 Full list of commits since version 1.2

239e82c : postgresql: improve SQL query (#133)

668eb1f : Update to make the code compatible with both python and JSON based config files (#129)

afd8837 : Configuration-manager fixes (#128)

571e2be : Remove pip installed system python packages

407d9ed : Update Dockerfile

1fefc69 : Implement unit tests for datablock.py (#122)

43c8d7a : Adjust global configuration to include program-option values. (#126)

2840813 : Switch to Jsonnet configuration system (#125)

5c4ae0e : logging changes: added config file and command line interface (#124)

6697f22 : Further config-manager testing and factorizations. (#123)

fa89fd0 : Insulate multiprocessing test from parent environment. (#120)

139a537 : Allow empty base directory for log file. (#119)

f14d40c : Factorize configuration-loading steps. (#118)

e00afee : Enhance testing and error reporting of ConfigManager (#117)

c3d1be3 : Python 3 upgrades. (#116)

e7399af : Header fix (#114)

0456abf : Adding editor config file, see <https://editorconfig.org/> (#115)

82112d1 : Dockerfile: fetch osg 3.5 repo rpm (#113)

97c21b1 : osg version 3.5 (#112)

33f28a8 : Introduce jsonnet dependency (#110)  
3f8b55e : improve server error handling (#108)  
f15588e : added 1.2.0 release notes  
b433325 : Remove unnecessary 'main' functionality. (#107)

## 1.2 Release 1.2.0

In this release:

- Switched to python3
- Improved coverage
- Database data retention : added reaper to remove data older than configurable number of days
- Improved logging

### 1.2.1 decisionengine

3dfe167 : Jenkins pipeline improvements (#106)  
22a7073 : pull request for review request 137 (#105)  
cafff2 : Make it possible to run code directly (for tests), and (#100)  
802e98b : replace psycog2 with psycog2-binary (#101)  
573ce8f : Jenkins pipeline improvements (#99)  
9d08835 : Run coveralls even under failed state (#97)  
bc1df4b : Add tests for PostgreSQL datasource (#71)  
c1ac391 : Fix missing py-modules.html (#96)  
8dbfdee : Setup gh-pages doc workflow (#94)  
cd4a01a : Doc (#93)  
673080d : set version to 1.2.0 (for now). Supply conf file that corresponds to (#91)  
f912225 : Db (#92)  
dc8b68a : Add reaper to the RPC (#83) (#90)  
29ade91 : adding .Jenkinsfile with Jenkins pipeline configuration (#86)  
c1dfe5c : Don't exclude E1004 from pylint, do exclude line breaks (#89)  
440f949 : Fix varname (#88)  
313d135 : Compress (#87)  
6b8dc4b : Revert "Add reaper to the RPC (#83)"  
dbea8e5 : Update utils.sh so pytest will complete.  
e848316 : Update to postgresql11  
7f4b805 : Add reaper to the RPC (#83)  
0ba2c51 : remove astpp module and dependencies it pulls in (#81)



6b8eab9 : don't track test coverage of tests (#80)  
0da18ec : made reaper.py executable  
aca24a3 : make reaper.py executable, make symbolic link to it from /usr/bin (#72)  
0202acf : Implementation of data reaper (#70)  
16b6be1 : Simple changes for Python 3 deployment (#69)  
fd2418c : Fix warnings caught by PEP-8 Speaks.  
d16359b : Python 3 (and other) simplifications.  
3c7b6b7 : Only run Github Actions for python3.6 (#68)  
453cbba : Update README.md  
b27ed53 : remove unnecessary (and atually harmful) python shebang (#66)

## 1.2.2 decisionengine\_modules

30d928b : clone version 1.2.0 of decisionengine  
ae7c5a6 : Jenkins pipeline improvements (#236)  
310befd : T198 (#235)  
a65886d : Fix import as reported in : <https://github.com/HEPCloud/decisionengin...> (#232)  
93711cc : Run coveralls even if tests fail (#229)  
03d763a : Jenkins pipeline improvements (#230)  
f48d30f : Fix/223 (#228)  
c8aa262 : github ticket 199 (#222)  
0323bda : Address : [https://github.com/HEPCloud/decisionengine\\_modules/issues/224](https://github.com/HEPCloud/decisionengine_modules/issues/224) (#226)  
62e4df6 : Add support to run CI on Jenkins (#221)  
5ab1541 : bump master version to 1.2.0 (for now) (#219)  
bc19c65 : decisionengine\_modules/NERSC: Added retry loop for NERSC API Calls (#220)  
41a50de : Sync up pep8speaks and run\_pylint.sh with decisionengine settings (#218)  
db4634f : silence pylint error (#217)  
1b95141 : Fix whitespace around operator error  
746ea38 : ignore W503  
8a8b5f4 : remove unused variable  
a6668bf : fix PEP8 warnings  
13773ee : address pep8 warnings  
6bea4ca : silence pylint error  
f589895 : Pass sort=True parameter to fix future warning (#215)  
a1d0507 : fixing pep8 warning  
a10bd17 : debugging one import error  
ec501ad : make coveralls.io links work

deab1a7 : T201 (#204)

69f2645 : Add coveragerc

6d8a5f5 : decisionengine\_modules/NERSC: Make Nersc API call backward-compatible with old config (#196)

a7e0af9 : Only run Github Actions for python3.6 (#24)

## 1.3 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 entrypoint 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 compuler 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 unnused 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/pms/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.config package

##### Subpackages

##### decisionengine.framework.config.tests package

##### Submodules

##### decisionengine.framework.config.tests.test\_config module

```
decisionengine.framework.config.tests.test_config._channel_config_dir(relative_dir)
decisionengine.framework.config.tests.test_config._global_config_file(relative_filename)
decisionengine.framework.config.tests.test_config.load()
decisionengine.framework.config.tests.test_config.test_channel_empty_config(load,
                                                                              cap-
                                                                              sys,
                                                                              caplog)
decisionengine.framework.config.tests.test_config.test_channel_empty_dictionary(load,
                                                                                  caplog)
decisionengine.framework.config.tests.test_config.test_channel_loading(caplog)
decisionengine.framework.config.tests.test_config.test_channel_names(load)
decisionengine.framework.config.tests.test_config.test_channel_no_config_files(load)
decisionengine.framework.config.tests.test_config.test_channel_no_modules(load)
```

```

decisionengine.framework.config.tests.test_config.test_empty_config(load)
decisionengine.framework.config.tests.test_config.test_empty_dict(load)
decisionengine.framework.config.tests.test_config.test_empty_dict_with_leading_comment(load)
decisionengine.framework.config.tests.test_config.test_minimal_jsonnet_right_extension(load,
                                                                                          cap-
                                                                                          sys)
decisionengine.framework.config.tests.test_config.test_minimal_jsonnet_wrong_extension(load,
                                                                                          cap-
                                                                                          sys)
decisionengine.framework.config.tests.test_config.test_minimal_python(load,
                                                                                          cap-
                                                                                          sys)
decisionengine.framework.config.tests.test_config.test_wrong_type(load)

```

## Module contents

### Submodules

#### decisionengine.framework.config.ChannelConfigHandler module

Manager of channel configurations.

The ChannelConfigHandler manages only channel configurations and not the global decision-engine configuration. It is responsible for loading channel configuration files and validating that the channels have the correct configuration artifacts and inter-module product dependencies.

```

class decisionengine.framework.config.ChannelConfigHandler.ChannelConfigHandler(global_config,
                                                                                   chan-
                                                                                   nel_config_dir)

```

Bases: object

**\_load\_channel** (*channel\_name, path*)

**get\_channels** ()

**get\_produces** (*channel\_config*)

**load\_all\_channels** ()

Load all channel configurations inside the stored channel-configuration directory.

Any cached configurations will be dropped prior to reloading.

**load\_channel** (*channel\_name*)

Load a single configuration for a channel with the supplied name.

The behavior is to read a configuration file whose path is:

<cached channel config. dir>/{channel\_name}.jsonnet

where the cached channel-configuration directory was stored whenever the ChannelConfigHandler object was created, and {channel\_name} is the value of the supplied method argument.

**print\_channel\_config** (*channel*)

```

decisionengine.framework.config.ChannelConfigHandler._check_keys(channel_conf_dict)
    check that channel config has mandatory keys :type data: dict

```

```
decisionengine.framework.config.ChannelConfigHandler._make_logger(global_config)
decisionengine.framework.config.ChannelConfigHandler._validate(channel)
    Validate channels :type channel: dict
```

## decisionengine.framework.config.ValidConfig module

ValidConfig represents a valid JSON document.

The decision engine requires each of its configuration files to be valid JSON. This is achieved by either supplying a valid Jsonnet or JSON document upfront, or by providing a Python dictionary that can be trivially converted to a JSON document.

Vetting of a file for JSON validity happens upon construction of a ‘ValidConfig’ object. A fully constructed ‘ValidConfig’ object thus corresponds to a valid JSON document.

```
class decisionengine.framework.config.ValidConfig.ValidConfig(filename)
    Bases: collections.UserDict
```

ValidConfig represents a valid JSON configuration in the form of a dictionary.

In addition to the normal dictionary operations, users may call ‘dump()’ to print out in a string form the JSON configuration.

```
_abc_cache = <_weakrefset.WeakSet object>
_abc_negative_cache = <_weakrefset.WeakSet object>
_abc_negative_cache_version = 185
_abc_registry = <_weakrefset.WeakSet object>

dump()
    Print dictionary data to a valid JSON string.
```

```
decisionengine.framework.config.ValidConfig._config_from_file(config_file)
```

```
decisionengine.framework.config.ValidConfig._convert_to_json(config_file)
    Attempt to convert JSON non-compliant configuration into a compliant one.
```

This is a temporary facility to aid the migration of Python-based configurations to Jsonnet-based ones. Python dictionaries that are similar in structure to JSON documents are generally trivially convertible.

## decisionengine.framework.config.policies module

Decision-engine default configuration policies.

For the decision-engine process, the configuration policies are:

- The global configuration file must be named ‘decision\_engine.jsonnet’ and it must reside in (a) a directory that can be accessed through the ‘CONFIG\_PATH’ environment variable, or (b) the /etc/decisionengine directory.
- All channel configurations must reside in (a) a directory accessible through the ‘CHANNEL\_CONFIG\_PATH’ environment variable, or (b) a ‘config.d’ subdirectory of the /etc/decisionengine directory.

The utilities provided in this module provide simple means of accessing the configuration artifacts according to the policies listed above. Please consult the documentation for each function below for more detailed information.

```
decisionengine.framework.config.policies.channel_config_dir(parent_dir=None)
    Retrieve the channel configuration directory as a string.
```

This directory contains all channel configuration files. This function assumes that the directory can be accessed by using the 'CHANNEL\_CONFIG\_PATH' environment variable. If that variable has not been set, then the value of 'parent\_dir' is prepended to 'config.d', which is then assumed to be the full path to the channel-configuration directory.

If the 'parent\_dir' argument is not provided, the global configuration directory is used as the parent (see documentation for 'global\_config\_dir()').

If the final path for the channel configuration directory does not correspond to a directory, an exception is raised.

```
decisionengine.framework.config.policies.global_config_dir()
```

Retrieve global configuration dir as string.

This is the directory that houses the 'decision\_engine.jsonnet' global configuration file.

This function checks that the 'CONFIG\_PATH' variable has been set or will use /etc/decisionengine otherwise. If the path exists as a directory, then the directory path is returned as a string; otherwise an exception is raised.

```
decisionengine.framework.config.policies.global_config_file(global_config_dir=None)
```

Retrieve the path (as a string) corresponding to the global configuration.

If supplied, the 'global\_config\_dir' is assumed to be the full path corresponding to a directory containing the 'decision\_engine.jsonnet' file. If not provided, the global configuration directory is determined based on the behavior of the 'global\_config\_dir()' function.

An exception is raised if no 'decision\_engine.jsonnet' file is found.

## 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.fixtures module

pytest fixtures/constants

**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
    __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*, *taskman-*  
*ager\_id=None*,  
*genera-*  
*tion\_id=None*, *se-*  
*quence\_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

```
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.InvalidMetadataError
    Bases: Exception
```

### Errors due to invalid Metadata

[illegible]

**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

**dataproducit\_table = 'dataproducit'**

Name of the dataproducit table

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

Delete data older than 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 dataproducit 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 dataproducit 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 (*args,
                                                                    **kwargs)
```

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 (value)
    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 (global_config,
                                                                    chan-
                                                                    nel_config_loader,
                                                                    server_address)
    Bases: socketserver.ThreadingMixIn, xmlrpc.server.SimpleXMLRPCServer
    _disable_channels_with_terminated_processes ()
```

**\_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** ()

**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\_set\_channel\_log\_level** (*channel, log\_level*)

Assumes log\_level is a string corresponding to the supported logging-module levels.

**rpc\_show\_config** (*channel*)

Show the configuration for a channel.

**rpc\_show\_de\_config** ()

**rpc\_start\_channel** (*channel\_name*)

**rpc\_start\_channels** ()

**rpc\_status** ()

**rpc\_stop** ()

**rpc\_stop\_channel** (*channel*)

**rpc\_stop\_channels** ()

**service\_actions** ()

Called by the serve\_forever() loop.

May be overridden by a subclass / Mixin to implement any code that needs to be run during the loop.

**start\_channel** (*channel\_name, channel\_config*)

**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.Worker(task_manager, log-
                                                            ger_config)

    Bases: multiprocessing.context.Process

    get_state_name()

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

class decisionengine.framework.engine.DecisionEngine.WorkerInErrorState(task_manager_id)

    Bases: object

    get_state_name()

    is_alive()

decisionengine.framework.engine.DecisionEngine._channel_preamble(name)

decisionengine.framework.engine.DecisionEngine._get_de_conf_manager(global_config_dir,
                                                                    chan-
                                                                    nel_config_dir,
                                                                    options)

decisionengine.framework.engine.DecisionEngine._get_global_config(config_file,
                                                                    options)

decisionengine.framework.engine.DecisionEngine._start_de_server(global_config,
                                                                    chan-
                                                                    nel_config_loader)

    start the DE server with the passed global configuration and config manager

decisionengine.framework.engine.DecisionEngine.main(args=None)
    If args is None, sys.argv will be used instead If args is a list, it will be used instead of sys.argv (for unit testing)

decisionengine.framework.engine.DecisionEngine.parse_program_options(args=None)
    If args is a list, it will be used instead of sys.argv
```

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

```
decisionengine.framework.engine.de_client.create_parser()

decisionengine.framework.engine.de_client.execute_command_from_args(argsparsed,
                                                                    de_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_parameters()  
    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()  
    post_create(global_config)  
    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')  
    post_create (global_config)  
    produces ()  
        Assumes that self.datakeys has the following structure or  
decisionengine.framework.modules.SourceProxy.main ()  
    Call this 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

Logger 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_level, file_rotate_by,
                                                         rotation_time_unit,
                                                         rotation_interval,
                                                         max_backup_count,
                                                         max_file_size=200000000,
                                                         log_file_name='/tmp/decision_engine_logs/decision_
```

### Parameters

- **log\_level** (str) – log level
- **file\_rotate\_by** – files rotation by size or by time
- **rotation\_time\_unit** (str) – unit of time for file rotation
- **rotation\_interval** (int) – time in rotation\_time\_units between file rotations
- **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.State(value)
```

Bases: enum.Enum

An enumeration.

**BOOT** = 0

**OFFLINE** = 2

**SHUTDOWN** = 4

**SHUTTINGDOWN** = 3

**STEADY** = 1

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

Bases: object

Task Manager

**\_take\_offline** (*current\_data\_block*)  
offline and stop 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** ()

**get\_state\_name** ()

**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*)  
Assumes log\_level is a string corresponding to the supported logging-module levels.

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

**start\_sources** (*data\_block=None*)  
Start sources, each in a separate thread

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

**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 an events to synchronise execution

decisionengine.framework.taskmanager.TaskManager.**\_\_create\_worker** (*module\_name*,  
*class\_name*,  
*parameters*)

Create instance of dynamically loaded module

decisionengine.framework.taskmanager.TaskManager.**\_\_make\_workers\_for** (*configs*)

## Module contents

### decisionengine.framework.tests package

#### Submodules

#### decisionengine.framework.tests.PublisherNOP module

**class** decisionengine.framework.tests.PublisherNOP.**PublisherNOP** (*config*)  
Bases: *decisionengine.framework.modules.Publisher.Publisher*

**consumes** (*name\_list=None*)

**publish** (*data\_block=None*)

### decisionengine.framework.tests.SourceNOP module

```
class decisionengine.framework.tests.SourceNOP.SourceNOP (config)
    Bases: decisionengine.framework.modules.Source.Source

    acquire ()

    produces ()
```

### decisionengine.framework.tests.TransformNOP module

```
class decisionengine.framework.tests.TransformNOP.TransformNOP (config)
    Bases: decisionengine.framework.modules.Transform.Transform

    consumes (name_list=None)

    produces (name_schema_id_list=None)

    transform (data_block)
```

### decisionengine.framework.tests.fixtures module

defaults for pytest

```
decisionengine.framework.tests.fixtures.DEServer (conf_path=None,
                                                    conf_override=None,      chan-
                                                    nel_conf_path=None,      chan-
                                                    nel_conf_override=None,
                                                    host='127.0.0.1',      port=None,
                                                    pg_prog_name='PG_PROG',
                                                    pg_db_conn_name='DE_DB')
```

A DE Server using our private database

```
decisionengine.framework.tests.fixtures.DE_DB (request: _pytest.fixtures.FixtureRequest)
    → psycopg2.extensions.connection
```

Fixture factory for PostgreSQL.

**Parameters** **request** (*FixtureRequest*) – fixture request object

**Returns** postgresql client

```
decisionengine.framework.tests.fixtures.PG_PROG (request:
    _pytest.fixtures.FixtureRequest,
    tmpdir_factory:
    _pytest.tmpdir.TempdirFactory) →
    pytest_postgresql.executor.PostgreSQLExecutor
```

Process fixture for PostgreSQL.

**Parameters** **request** (*FixtureRequest*) – fixture request object

**Return type** *pytest\_dbfixtures.executors.TCPExecutor*

**Returns** tcp executor

### decisionengine.framework.tests.test\_defaults module

Fixture based DE Server tests of the sample config

```
decisionengine.framework.tests.test_defaults.test_client_can_get_de_server_show_channel_log_level
    Verify unknown channel has NOTSET
```

```
decisionengine.framework.tests.test_defaults.test_client_de_config_is_json(deserver)
    Verify config can be fetched in json format
```

```
decisionengine.framework.tests.test_defaults.test_global_channel_log_level_in_config(deserver)
    Verify global_channel_log_level setting exists
```

### decisionengine.framework.tests.test\_reaper module

Fixture based DE Server for the reaper tests

```
decisionengine.framework.tests.test_reaper.test_client_can_get_de_server_reaper_start_delay
    Verify reaper can start with delay
```

```
decisionengine.framework.tests.test_reaper.test_client_can_get_de_server_reaper_status(deserver)
    Verify reaper status
```

```
decisionengine.framework.tests.test_reaper.test_client_can_get_de_server_reaper_stop(deserver)
    Verify reaper can stop
```

### decisionengine.framework.tests.test\_sample\_config module

Fixture based DE Server tests of the defaults

```
decisionengine.framework.tests.test_sample_config.test_client_can_get_de_server_show_config
    Verify config has expected items
```

```
decisionengine.framework.tests.test_sample_config.test_client_can_get_de_server_show_log_level
    Verify can fetch log level
```

```
decisionengine.framework.tests.test_sample_config.test_client_can_get_de_server_status(deserver)
    Verify channel enters stable state
```

### decisionengine.framework.tests.test\_start\_with\_no\_channels module

Fixture based DE Server tests of the server without channels, then with them

```
decisionengine.framework.tests.test_start_with_no_channels.cleanup_tmpdir()
```

```
decisionengine.framework.tests.test_start_with_no_channels.test_start_from_nothing(deserver)
```

## Module contents

### decisionengine.framework.util package

## Submodules

### decisionengine.framework.util.fs module

`decisionengine.framework.util.fs.files_with_extensions` (*dir\_path*, *\*extensions*)

Return all files in *dir\_path* that match the provided extensions.

If no extensions are given, then all files in *dir\_path* are returned.

### decisionengine.framework.util.sockets module

`decisionengine.framework.util.sockets.get_random_port` ()

### 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, 36  
 decisionengine.framework, 36  
 decisionengine.framework.config, 16  
 decisionengine.framework.config.ChannelConfigHandler, 14  
 decisionengine.framework.config.policies, 15  
 decisionengine.framework.config.tests, 14  
 decisionengine.framework.config.tests.test\_config, 13  
 decisionengine.framework.config.ValidConfig, 15  
 decisionengine.framework.dataspace, 26  
 decisionengine.framework.dataspace.datablock, 20  
 decisionengine.framework.dataspace.dataspace, 22  
 decisionengine.framework.dataspace.datasources, 20  
 decisionengine.framework.dataspace.datasources.postgresql, 17  
 decisionengine.framework.dataspace.datasources.tests, 17  
 decisionengine.framework.dataspace.datasources.tests.fixtures, 16  
 decisionengine.framework.dataspace.datasources.tests.test\_postgresql, 17  
 decisionengine.framework.dataspace.dataspace, 25  
 decisionengine.framework.engine, 29  
 decisionengine.framework.engine.de\_client, 28  
 decisionengine.framework.engine.DecisionEngine, 26  
 decisionengine.framework.modules, 31  
 decisionengine.framework.modules.de\_logger, 31  
 decisionengine.framework.modules.LogicEngine, 29  
 decisionengine.framework.modules.Module, 29  
 decisionengine.framework.modules.Publisher, 29  
 decisionengine.framework.modules.Source, 29  
 decisionengine.framework.modules.SourceProxy, 30  
 decisionengine.framework.modules.Transform, 30  
 decisionengine.framework.taskmanager, 33  
 decisionengine.framework.taskmanager.TaskManager, 31  
 decisionengine.framework.tests, 36  
 decisionengine.framework.tests.fixtures, 34  
 decisionengine.framework.tests.PublisherNOP, 33  
 decisionengine.framework.tests.SourceNOP, 34  
 decisionengine.framework.tests.test\_defaults, 35  
 decisionengine.framework.tests.test\_reaper, 35  
 decisionengine.framework.tests.test\_sample\_config, 35  
 decisionengine.framework.tests.test\_start\_with\_no\_c, 35  
 decisionengine.framework.tests.TransformNOP, 34  
 decisionengine.framework.util, 36  
 decisionengine.framework.util.fs, 36  
 decisionengine.framework.util.sockets, 36  
 decisionengine.framework.util.tsort, 36



# INDEX

## Symbols

<code>__query()</code> ( <i>decisionengine.framework.dataspace.datasources.postgresql.Postgresql</i> method), 17	<code>_abc_negative_cache_version</code> ( <i>decisionengine.framework.dataspace.datasource.DataSource</i> attribute), 22
<code>_abc_cache</code> ( <i>decisionengine.framework.config.ValidConfig.ValidConfig</i> attribute), 15	<code>_abc_negative_cache_version</code> ( <i>decisionengine.framework.dataspace.datasources.postgresql.Postgresql</i> attribute), 17
<code>_abc_cache</code> ( <i>decisionengine.framework.dataspace.datablock.Header</i> attribute), 21	<code>_abc_registry</code> ( <i>decisionengine.framework.config.ValidConfig.ValidConfig</i> attribute), 15
<code>_abc_cache</code> ( <i>decisionengine.framework.dataspace.datablock.Metadata</i> attribute), 22	<code>_abc_registry</code> ( <i>decisionengine.framework.dataspace.datablock.Header</i> attribute), 21
<code>_abc_cache</code> ( <i>decisionengine.framework.dataspace.datasource.DataSource</i> attribute), 22	<code>_abc_registry</code> ( <i>decisionengine.framework.dataspace.datablock.Metadata</i> attribute), 22
<code>_abc_cache</code> ( <i>decisionengine.framework.dataspace.datasources.postgresql.Postgresql</i> attribute), 17	<code>_abc_registry</code> ( <i>decisionengine.framework.dataspace.datasource.DataSource</i> attribute), 22
<code>_abc_negative_cache</code> ( <i>decisionengine.framework.config.ValidConfig.ValidConfig</i> attribute), 15	<code>_abc_registry</code> ( <i>decisionengine.framework.dataspace.datasources.postgresql.Postgresql</i> attribute), 17
<code>_abc_negative_cache</code> ( <i>decisionengine.framework.dataspace.datablock.Header</i> attribute), 21	<code>_channel_config_dir()</code> (in module <i>decisionengine.framework.config.tests.test_config</i> ), 13
<code>_abc_negative_cache</code> ( <i>decisionengine.framework.dataspace.datablock.Metadata</i> attribute), 22	<code>_channel_preamble()</code> (in module <i>decisionengine.framework.engine.DecisionEngine</i> ), 28
<code>_abc_negative_cache</code> ( <i>decisionengine.framework.dataspace.datasource.DataSource</i> attribute), 22	<code>_check_keys()</code> (in module <i>decisionengine.framework.config.ChannelConfigHandler</i> ), 14
<code>_abc_negative_cache</code> ( <i>decisionengine.framework.dataspace.datasources.postgresql.Postgresql</i> attribute), 17	<code>_config_from_file()</code> (in module <i>decisionengine.framework.config.ValidConfig</i> ), 15
<code>_abc_negative_cache_version</code> ( <i>decisionengine.framework.config.ValidConfig.ValidConfig</i> attribute), 15	<code>_convert_to_json()</code> (in module <i>decisionengine.framework.config.ValidConfig</i> ), 15
<code>_abc_negative_cache_version</code> ( <i>decisionengine.framework.dataspace.datablock.Header</i> attribute), 21	<code>create_worker()</code> (in module <i>decisionengine.framework.taskmanager.TaskManager</i> ), 33
<code>_abc_negative_cache_version</code> ( <i>decisionengine.framework.dataspace.datablock.Metadata</i> attribute), 22	<code>delete()</code> ( <i>decisionengine.framework.dataspace.datasources.postgresql.Postgresql</i> method), 17
	<code>disable_channels_with_terminated_processes()</code> ( <i>decisionengine.framework.engine.DecisionEngine.DecisionEngine</i> method), 26

\_dispatch() (decisionengine.framework.engine.DecisionEngine), 26  
 \_ds (decisionengine.framework.dataspace.dataspace.DataSourceLoader), 20  
 \_get\_data() (decisionengine.framework.modules.SourceProxy.SourceProxy), 30  
 \_get\_de\_conf\_manager() (in module decisionengine.framework.engine.DecisionEngine), 28  
 \_get\_global\_config() (in module decisionengine.framework.engine.DecisionEngine), 28  
 \_global\_config\_file() (in module decisionengine.framework.config.tests.test\_config), 13  
 \_insert() (decisionengine.framework.dataspace.datablock.DataBlock), 20  
 \_insert() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql), 18  
 \_insert\_returning\_result() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql), 18  
 \_instances (decisionengine.framework.dataspace.dataspace.Singleton), 26  
 \_load\_channel() (decisionengine.framework.config.ChannelConfigHandler.ChannelConfigHandler), 14  
 \_make\_logger() (in module decisionengine.framework.config.ChannelConfigHandler), 14  
 \_make\_workers\_for() (in module decisionengine.framework.taskmanager.TaskManager), 33  
 \_reaper\_loop() (decisionengine.framework.dataspace.dataspace.Reaper), 25  
 \_remove() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql), 18  
 \_select() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql), 18  
 \_select\_dictresult() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql), 18  
 \_select\_getresult() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql), 18  
 \_select\_tuple() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql), 18  
 \_set\_state() (decisionengine.framework.dataspace.dataspace.Reaper), 25  
 \_start\_de\_server() (in module decisionengine.framework.engine.DecisionEngine), 28  
 \_tables\_created (decisionengine.framework.dataspace.dataspace.DataSpace), 25  
 \_take\_offline() (decisionengine.framework.taskmanager.TaskManager.TaskManager), 32  
 \_update() (decisionengine.framework.dataspace.datablock.DataBlock), 20  
 \_update() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql), 18  
 \_validate() (in module decisionengine.framework.config.ChannelConfigHandler), 15  
 acquire() (decisionengine.framework.modules.Source.Source), 29  
 acquire() (decisionengine.framework.modules.SourceProxy.SourceProxy), 29  
 acquire() (decisionengine.framework.tests.SourceNOP.SourceNOP), 34  
 BOOT (decisionengine.framework.taskmanager.TaskManager.State), 31  
 Channel (class in decisionengine.framework.taskmanager.TaskManager), 31  
 channel\_config\_dir() (in module decisionengine.framework.config.policies), 15  
 ChannelConfigHandler (class in decisionengine.framework.config.ChannelConfigHandler), 14  
 cleanup\_tmpdir() (in module decisionengine.framework.tests.test\_start\_with\_no\_channels), 33  
 close() (decisionengine.framework.dataspace.datasource.DataSource), 22  
 close() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql), 18  
 close() (decisionengine.framework.dataspace.dataspace.DataSpace), 25



Method	Module	Package	Line
compress()	(in module decisionengine.framework.dataspace.datablock),	22	17
connect()	(decisionengine.framework.dataspace.datasource.DataSourceLoader (class in decisionengine.framework.dataspace.dataspace),	method), 22	25
connect()	(decisionengine.framework.dataspace.datasource.DataSourceLoader (class in decisionengine.framework.dataspace.dataspace),	method), 18	25
consumes()	(decisionengine.framework.modules.Publisher.Publisher (class in decisionengine.framework.modules.Publisher.Publisher),	method), 29	25
consumes()	(decisionengine.framework.modules.SourceProxy.SourceProxy (class in decisionengine.framework.modules.SourceProxy.SourceProxy),	method), 30	25
consumes()	(decisionengine.framework.modules.Transform.Transform (class in decisionengine.framework.modules.Transform.Transform),	method), 30	34
consumes()	(decisionengine.framework.tests.PublisherNOP.PublisherNOP (class in decisionengine.framework.tests.PublisherNOP.PublisherNOP),	method), 33	32
consumes()	(decisionengine.framework.tests.TransformNOP.TransformNOP (class in decisionengine.framework.tests.TransformNOP.TransformNOP),	method), 34	26
create_datasource()	(decisionengine.framework.dataspace.dataspace.DataSourceLoader (class in decisionengine.framework.dataspace.dataspace),	static method), 25	36
create_parser()	(in module decisionengine.framework.engine.de_client),	28	16
create_tables()	(decisionengine.framework.dataspace.datasource.DataSourceLoader (class in decisionengine.framework.dataspace.dataspace),	method), 22	14
create_tables()	(decisionengine.framework.dataspace.datasources.postgresql.PostgreSQL (class in decisionengine.framework.dataspace.datasources.postgresql),	method), 18	14
data()	(in module decisionengine.framework.dataspace.datasources.tests.test_postgresql),	17	15
data_block_put()	(decisionengine.framework.taskmanager.TaskManager.TaskManager (class in decisionengine.framework.taskmanager.TaskManager),	method), 32	20
DataBlock	(class in decisionengine.framework.dataspace.datablock),	20	22
dataprodut()	(in module decisionengine.framework.dataspace.datasources.postgresql.PostgreSQL),	17	17
dataprodut_table	(decisionengine.framework.dataspace.datasource.DataSourceLoader (class in decisionengine.framework.dataspace.dataspace),	attribute), 23	16
DataSource	(class in decisionengine.framework.dataspace.datasource),	22	17
datasource()	(in module decisionengine.framework.dataspace.dataspace),	25	25

decisionengine.framework.engine	22	
module, 29	default_data_lifetime	(decisionengine.framework.dataspace.datablock.Header attribute), 21
decisionengine.framework.engine.de_client		
module, 28	delete_data_older_than()	(decisionengine.framework.dataspace.dataspace.DataSpace method), 25
decisionengine.framework.engine.DecisionEngine		
module, 26	delete_data_older_than()	(decisionengine.framework.dataspace.datasource.DataSource method), 23
decisionengine.framework.modules		
module, 31	delete_data_older_than()	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 18
decisionengine.framework.modules.de_logger		
module, 31	DEServer()	(in module decisionengine.framework.tests.fixtures), 34
decisionengine.framework.modules.LogicEngine		
module, 29	do_backup()	(decisionengine.framework.taskmanager.TaskManager.TaskManager method), 32
decisionengine.framework.modules.Module		
module, 29	dump()	(decisionengine.framework.config.ValidConfig.ValidConfig method), 15
decisionengine.framework.modules.Published		
module, 29	duplicate()	(decisionengine.framework.dataspace.datablock.DataBlock method), 20
decisionengine.framework.modules.Source		
module, 29	duplicate_datablock()	(decisionengine.framework.dataspace.datasource.DataSource method), 23
decisionengine.framework.modules.SourceProxy		
module, 30	duplicate_datablock()	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 18
decisionengine.framework.modules.Transform		
module, 30	duplicate_datablock()	(decisionengine.framework.dataspace.dataspace.DataSpace method), 25
decisionengine.framework.taskmanager		
module, 33		
decisionengine.framework.taskmanager.TaskManager		
module, 31		
decisionengine.framework.tests		
module, 36		
decisionengine.framework.tests.fixtures		
module, 34		
decisionengine.framework.tests.PublisherNOP		
module, 33		
decisionengine.framework.tests.SourceNOP		
module, 34		
decisionengine.framework.tests.test_defaults		
module, 35		
decisionengine.framework.tests.test_reaper		
module, 35		
decisionengine.framework.tests.test_sample_config		
module, 35		
decisionengine.framework.tests.test_start_with_no_channels		
module, 35		
decisionengine.framework.tests.TransformNOP		
module, 34		
decisionengine.framework.util		
module, 36		
decisionengine.framework.util.fs		
module, 36		
decisionengine.framework.util.sockets		
module, 36		
decisionengine.framework.util.tsort		
module, 36		
decompress()	(in module decisionengine.framework.dataspace.datablock),	

## E

ERROR (decisionengine.framework.dataspace.dataspace.State attribute), 26

evaluate() (decisionengine.framework.modules.LogicEngine.LogicEngine method), 29

extract\_config\_command\_from\_args() (in module decisionengine.framework.engine.de\_client), 28

## F

files\_with\_extensions() (in module decisionengine.framework.util.fs), 36

## G

generate\_insert\_query() (in module decisionengine.framework.dataspace.datasources.postgresql), 20

get() (decisionengine.framework.dataspace.datablock.DataBlock method), 20

get\_channels() (decisionengine.framework.config.ChannelConfigHandler.ChannelConfig method), 14

get_connection()	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 18	get_metadata()	(decisionengine.framework.dataspace.datasource.DataSource method), 23
get_data_block()	(decisionengine.framework.modules.Module.Module method), 29	get_metadata()	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 19
get_datablock()	(decisionengine.framework.dataspace.datasource.DataSource method), 23	get_metadata()	(decisionengine.framework.dataspace.dataspace.DataSpace method), 25
get_datablock()	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 18	get_paramaters()	(decisionengine.framework.modules.Module.Module method), 29
get_dataproduct()	(decisionengine.framework.dataspace.datasource.DataSource method), 23	get_produces()	(decisionengine.framework.config.ChannelConfigHandler.ChannelConfig method), 14
get_dataproduct()	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 18	get_random_port()	(in module decisionengine.framework.util.sockets), 36
get_dataproduct()	(decisionengine.framework.dataspace.dataspace.DataSpace method), 25	get_retention_interval()	(decisionengine.framework.dataspace.dataspace.Reaper method), 26
get_header()	(decisionengine.framework.dataspace.datablock.DataBlock method), 21	get_schema()	(decisionengine.framework.dataspace.datasource.DataSource method), 24
get_header()	(decisionengine.framework.dataspace.datasource.DataSource method), 23	get_schema()	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 19
get_header()	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 18	get_state()	(decisionengine.framework.dataspace.dataspace.Reaper method), 26
get_header()	(decisionengine.framework.dataspace.dataspace.DataSpace method), 25	get_state()	(decisionengine.framework.taskmanager.TaskManager.TaskManager method), 32
get_last_generation_id()	(decisionengine.framework.dataspace.datasource.DataSource method), 23	get_state_name()	(decisionengine.framework.engine.DecisionEngine.Worker method), 28
get_last_generation_id()	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 19	get_state_name()	(decisionengine.framework.engine.DecisionEngine.WorkerInErrorState method), 28
get_last_generation_id()	(decisionengine.framework.dataspace.dataspace.DataSpace method), 25	get_state_name()	(decisionengine.framework.taskmanager.TaskManager.TaskManager method), 32
get_logger()	(decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 27	get_taskmanager()	(decisionengine.framework.dataspace.datablock.DataBlock method), 21
get_logger()	(in module decisionengine.framework.modules.de_logger), 31	get_taskmanager()	(decisionengine.framework.dataspace.datasource.DataSource method), 24
get_loglevel()	(decisionengine.framework.taskmanager.TaskManager.TaskManager method), 32	get_taskmanager()	(decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 19
get_metadata()	(decisionengine.framework.dataspace.datablock.DataBlock method), 21	get_taskmanager()	(decisionengine.framework.dataspace.dataspace.DataSpace method), 25
		global_config_dir()	(in module decisionengine.framework.config.ChannelConfigHandler.ChannelConfig method), 14

`nengine.framework.config.policies)`, 16  
`global_config_file()` (in module `decisionengine.framework.config.policies)`, 16

## H

`handle_sighup()` (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 27  
Header (class in decisionengine.framework.dataspace.datablock), 21  
`header()` (in module decisionengine.framework.dataspace.datasources.tests.test\_postgresql), 17  
`header_table` (decisionengine.framework.dataspace.datasource.DataSource attribute), 24

## I

IDLE (decisionengine.framework.dataspace.dataspace.State attribute), 26  
`insert()` (decisionengine.framework.dataspace.datasource.DataSource method), 24  
`insert()` (decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 19  
`insert()` (decisionengine.framework.dataspace.dataspace.DataSource metadata), 25  
InvalidMetadataError, 21  
`is_alive()` (decisionengine.framework.engine.DecisionEngine.WorkerInErrorState method), 28  
`is_expired()` (decisionengine.framework.dataspace.datablock.DataBlock method), 21  
`is_valid()` (decisionengine.framework.dataspace.datablock.Header method), 21

## K

`keys()` (decisionengine.framework.dataspace.datablock.DataBlock method), 21

## L

`load()` (in module decisionengine.framework.config.tests.test\_config), 13  
`load_all_channels()` (decisionengine.framework.config.ChannelConfigHandler.ChannelConfigHandler method), 14  
`load_channel()` (decisionengine.framework.config.ChannelConfigHandler.ChannelConfigHandler method), 14

LogicEngine (class in decisionengine.framework.modules.LogicEngine), 29

## M

`main()` (in module decisionengine.framework.engine.de\_client), 28  
`main()` (in module decisionengine.framework.engine.DecisionEngine), 28  
`main()` (in module decisionengine.framework.modules.SourceProxy), 30

`mark_demented()` (decisionengine.framework.dataspace.dataspace.DataSpace method), 25  
`mark_expired()` (decisionengine.framework.dataspace.datablock.DataBlock method), 21  
`mark_expired()` (decisionengine.framework.dataspace.dataspace.DataSpace method), 25  
Metadata (class in decisionengine.framework.dataspace.datablock), 22  
`metadata()` (in module decisionengine.framework.dataspace.datasources.tests.test\_postgresql), 17  
`metadata_table` (decisionengine.framework.dataspace.datasource.DataSource attribute), 24  
decisionengine, 36  
decisionengine.framework, 36  
decisionengine.framework.config, 16  
decisionengine.framework.config.ChannelConfigHandler, 14  
decisionengine.framework.config.policies, 15  
decisionengine.framework.config.tests, 14  
decisionengine.framework.config.tests.test\_config, 13  
decisionengine.framework.config.ValidConfig, 15  
decisionengine.framework.dataspace, 26  
decisionengine.framework.dataspace.datablock, 20  
decisionengine.framework.dataspace.datasource, 22  
decisionengine.framework.dataspace.datasources, 20

decisionengine.framework.dataspace.datasources.postgresql,  
 17 decisionengine.framework.util.tsort,  
 decisionengine.framework.dataspace.datasources.tests,  
 17 Module (class in decision-  
 decisionengine.framework.dataspace.datasources.engine.framework.modules.Module), 29  
 16 module\_config\_info() (in module decisio-  
 decisionengine.framework.dataspace.datasources.engine.framework.modules.SourceProxy),  
 17 30  
 decisionengine.framework.dataspace.dataspace\_config\_template() (in module decisio-  
 25 engine.framework.modules.SourceProxy), 30  
 decisionengine.framework.engine, 29 must\_have (decisionengine.framework.modules.SourceProxy.SourceProxy  
 decisionengine.framework.engine.de\_client, attribute), 30  
 28  
 decisionengine.framework.engine.DecisionEngine,  
 26 OFFLINE (decisionengine.framework.taskmanager.TaskManager.State  
 decisionengine.framework.modules, 31 attribute), 31  
 decisionengine.framework.modules.de\_logger,  
 31  
 decisionengine.framework.modules.LogicEngine,  
 29 parse\_program\_options() (in module decisio-  
 decisionengine.framework.modules.Module, 28 engine.framework.engine.DecisionEngine),  
 29 PG\_PROG() (in module decisio-  
 decisionengine.framework.modules.Publisher, engine.framework.tests.fixtures), 34  
 29 post\_create() (decision-  
 decisionengine.framework.modules.Source, engine.framework.modules.Source.Source  
 29 method), 29  
 decisionengine.framework.modules.SourceProxy,  
 30 post\_create() (decision-  
 decisionengine.framework.modules.Transform, engine.framework.modules.SourceProxy.SourceProxy  
 30 method), 30  
 decisionengine.framework.taskmanager, Postgresql (class in decisio-  
 33 engine.framework.dataspace.datasources.postgresql),  
 17  
 decisionengine.framework.taskmanager.TaskManager,  
 31 print\_channel\_config() (decision-  
 decisionengine.framework.tests, 36 engine.framework.config.ChannelConfigHandler.ChannelConfig  
 decisionengine.framework.tests.fixtures, method), 14  
 34 produces() (decision-  
 decisionengine.framework.tests.PublisherNOP, engine.framework.modules.Source.Source  
 33 method), 29  
 decisionengine.framework.tests.SourceNOP, produces() (decision-  
 34 engine.framework.modules.SourceProxy.SourceProxy  
 method), 30  
 decisionengine.framework.tests.test\_defaults, produces() (decision-  
 35 engine.framework.modules.Transform.Transform  
 decisionengine.framework.tests.test\_reaper, method), 30  
 35 produces() (decision-  
 decisionengine.framework.tests.test\_sample\_config, engine.framework.tests.SourceNOP.SourceNOP  
 35 method), 34  
 decisionengine.framework.tests.test\_start\_with\_no\_channels, produces() (decision-  
 35 engine.framework.tests.TransformNOP.TransformNOP  
 decisionengine.framework.tests.TransformNOP, method), 34  
 34 publish() (decisionengine.framework.modules.Publisher.Publisher  
 decisionengine.framework.util, 36 method), 29  
 decisionengine.framework.util.fs, 36 publish() (decisionengine.framework.tests.PublisherNOP.PublisherNOP  
 decisionengine.framework.util.sockets, method), 33



Publisher (class in decisionengine.framework.modules.Publisher),  
 29  
 PublisherNOP (class in decisionengine.framework.tests.PublisherNOP),  
 33  
 put () (decisionengine.framework.dataspace.datablock.DataBlock  
 method), 21  
**R**  
 reap () (decisionengine.framework.dataspace.dataspace.Reaper  
 method), 26  
 Reaper (class in decisionengine.framework.dataspace.dataspace),  
 25  
 reaper\_start () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 reaper\_status () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 reaper\_stop () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 RequestHandler (class in decisionengine.framework.engine.DecisionEngine),  
 28  
 required\_keys (decisionengine.framework.dataspace.datablock.Header  
 attribute), 21  
 required\_keys (decisionengine.framework.dataspace.datablock.Metadata  
 attribute), 22  
 rpc\_get\_channel\_log\_level () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 rpc\_get\_log\_level () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 rpc\_paths (decisionengine.framework.engine.DecisionEngine.RequestHandler  
 attribute), 28  
 rpc\_print\_product () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 rpc\_print\_products () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 rpc\_reaper\_start () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 rpc\_reaper\_status () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 rpc\_reaper\_stop () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 rpc\_set\_channel\_log\_level () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 rpc\_show\_config () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 rpc\_show\_de\_config () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 rpc\_start\_channel () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 rpc\_start\_channels () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 rpc\_status () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 rpc\_stop () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 rpc\_stop\_channel () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 rpc\_stop\_channels () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27  
 run () (decisionengine.framework.engine.DecisionEngine.Worker  
 method), 28  
 run () (decisionengine.framework.taskmanager.TaskManager.TaskManager  
 method), 32  
 run\_logic\_engine () (decisionengine.framework.taskmanager.TaskManager.TaskManager  
 method), 32  
 run\_publishers () (decisionengine.framework.taskmanager.TaskManager.TaskManager  
 method), 32  
 run\_request\_handler () (decisionengine.framework.taskmanager.TaskManager.TaskManager  
 method), 32  
 run\_transform () (decisionengine.framework.taskmanager.TaskManager.TaskManager  
 method), 32  
 run\_transforms () (decisionengine.framework.taskmanager.TaskManager.TaskManager  
 method), 33  
 RUNNING (decisionengine.framework.dataspace.dataspace.State  
 attribute), 26  
**S**  
 service\_actions () (decisionengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 27

method), 27  
 set\_data\_block() (decisionengine.framework.modules.Module.Module method), 29  
 set\_logging() (in module decisionengine.framework.modules.de\_logger), 31  
 set\_loglevel() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 33  
 set\_retention\_interval() (decisionengine.framework.dataspace.dataspace.Reaper method), 26  
 set\_state() (decisionengine.framework.dataspace.datablock.Metadata method), 22  
 set\_state() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 33  
 set\_stream\_logging() (in module decisionengine.framework.modules.de\_logger), 31  
 SHUTDOWN (decisionengine.framework.taskmanager.TaskManager.State attribute), 31  
 SHUTTINGDOWN (decisionengine.framework.taskmanager.TaskManager.State attribute), 32  
 Singleton (class in decisionengine.framework.dataspace.dataspace), 26  
 SLEEPING (decisionengine.framework.dataspace.dataspace.State attribute), 26  
 Source (class in decisionengine.framework.modules.Source), 29  
 SourceNOP (class in decisionengine.framework.tests.SourceNOP), 34  
 SourceProxy (class in decisionengine.framework.modules.SourceProxy), 30  
 start() (decisionengine.framework.dataspace.dataspace.Reaper method), 26  
 start\_channel() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 27  
 start\_channels() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 27  
 start\_sources() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 33  
 STARTING (decisionengine.framework.dataspace.dataspace.State attribute), 26  
 State (class in decisionengine.framework.dataspace.dataspace), 26  
 State (class in decisionengine.framework.taskmanager.TaskManager), 31  
 STEADY (decisionengine.framework.taskmanager.TaskManager.State attribute), 32  
 stop() (decisionengine.framework.dataspace.dataspace.Reaper method), 26  
 stop\_channel() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 27  
 stop\_channels() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 28  
 STOPPED (decisionengine.framework.dataspace.dataspace.State attribute), 26  
 STOPPING (decisionengine.framework.dataspace.dataspace.State attribute), 26  
 store\_taskmanager() (decisionengine.framework.dataspace.datablock.DataBlock method), 21  
 store\_taskmanager() (decisionengine.framework.dataspace.datasource.DataSource method), 24  
 store\_taskmanager() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 19  
 store\_taskmanager() (decisionengine.framework.dataspace.dataspace.DataSpace method), 25  
 tables (decisionengine.framework.dataspace.datasources.postgresql.Postgresql attribute), 19  
 TaskManager (class in decisionengine.framework.taskmanager.TaskManager), 32  
 taskmanager() (in module decisionengine.framework.dataspace.datasources.tests.test\_postgresql), 17  
 taskmanager\_table (decisionengine.framework.dataspace.datasource.DataSource attribute), 24  
 test\_channel\_empty\_config() (in module decisionengine.framework.config.tests.test\_config), 13  
 test\_channel\_empty\_dictionary() (in module decisionengine.framework.config.tests.test\_config), 13  
 test\_channel\_loading() (in module decisionengine.framework.config.tests.test\_config), 13  
 test\_channel\_names() (in module decisionengine.framework.config.tests.test\_config), 13

test_channel_no_config_files() (in module decisionengine.framework.config.tests.test_config), 13	test_get_taskmanager() (in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 17
test_channel_no_modules() (in module decisionengine.framework.config.tests.test_config), 13	test_global_channel_log_level_in_config() (in module decisionengine.framework.tests.test_defaults), 35
test_client_can_get_de_server_reaper_status_delay() (in module decisionengine.framework.tests.test_reaper), 35	test_insert() (in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 17
test_client_can_get_de_server_reaper_status() (in module decisionengine.framework.tests.test_reaper), 35	test_minimal_jsonnet_right_extension() (in module decisionengine.framework.config.tests.test_config), 14
test_client_can_get_de_server_reaper_stop() (in module decisionengine.framework.tests.test_reaper), 35	test_minimal_jsonnet_wrong_extension() (in module decisionengine.framework.config.tests.test_config), 14
test_client_can_get_de_server_show_channel_log_level() (in module decisionengine.framework.tests.test_defaults), 35	test_minimal_python() (in module decisionengine.framework.config.tests.test_config), 14
test_client_can_get_de_server_show_config() (in module decisionengine.framework.tests.test_sample_config), 35	test_start_from_nothing() (in module decisionengine.framework.tests.test_start_with_no_channels), 36
test_client_can_get_de_server_show_logger_level() (in module decisionengine.framework.tests.test_sample_config), 35	test_store_taskmanager() (in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 17
test_client_can_get_de_server_status() (in module decisionengine.framework.tests.test_sample_config), 35	test_wrong_type() (in module decisionengine.framework.config.tests.test_config), 14
test_client_de_config_is_json() (in module decisionengine.framework.tests.test_defaults), 35	Transform (class in decisionengine.framework.modules.Transform), 30
test_create_tables() (in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 17	transform() (decisionengine.framework.modules.Transform.Transform method), 30
test_empty_config() (in module decisionengine.framework.config.tests.test_config), 13	transform() (decisionengine.framework.tests.TransformNOP.TransformNOP method), 34
test_empty_dict() (in module decisionengine.framework.config.tests.test_config), 14	TransformNOP (class in decisionengine.framework.tests.TransformNOP), 34
test_empty_dict_with_leading_comment() (in module decisionengine.framework.config.tests.test_config), 14	tsort() (in module decisionengine.framework.util.tsort), 36
test_generate_insert_query() (in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 17	<b>U</b>
test_get_last_generation_id() (in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 17	update() (decisionengine.framework.dataspace.datasource.DataSource method), 24
	update() (decisionengine.framework.dataspace.datasources.postgresql. method), 19
	update() (decisionengine.framework.dataspace.dataspace.DataSpace method), 25
	valid_states (decision-



*nengine.framework.dataspace.datablock.Metadata*  
*attribute*), 22

*ValidConfig* (class in *decisionengine.framework.config.ValidConfig*), 15

## W

*wait\_for\_all()* (*decisionengine.framework.taskmanager.TaskManager.TaskManager*  
*method*), 33

*wait\_for\_any()* (*decisionengine.framework.taskmanager.TaskManager.TaskManager*  
*method*), 33

*Worker* (class in *decisionengine.framework.engine.DecisionEngine*),  
 28

*Worker* (class in *decisionengine.framework.taskmanager.TaskManager*),  
 33

*WorkerInErrorState* (class in *decisionengine.framework.engine.DecisionEngine*),  
 28

## Z

*zdumps()* (in module *decisionengine.framework.dataspace.datablock*),  
 22

*zloads()* (in module *decisionengine.framework.dataspace.datablock*),  
 22