# decisionengine

***Release 1.6.0rc***

**Fermilab**

**Feb 18, 2021**

# CONTENTS


1 Release Notes 3

2 Developer Documentation 13

3 Source code 17

4 Indices and tables 49

Python Module Index 51

Index 53

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

CHAPTER

ONE

# RELEASE NOTES

## 1.1 Release 1.5.0

In this release:

- Introduce data product query interface
- Cleanup of Ligic Engine code
- Improvements in error handling
- Improvements in testing and CI

### 1.1.1 Issues fixed in this release

- 217 , 218 : Add option to de-client –print-product to only print the column names in a data block and-or to print one or more records in key/value format (fe7abcf)
- 240 : Logic Engine call leads to immediate taskmanager segfault exit (d855aa0)
- 239 : implement data product browsing interface (fe9faa9)

### 1.1.2 Full list of commits since version 1.4.1

d66c54b : Add PEP-0396 metadata (#243)

bfc91a6 : More compat between psycopg2/psycopg2cffi (#248)

f5d31a6 : Cleanup Fixture FIXME (#249)

0dfaf3c : Adding docker documentation (#251)

4b166a2 : Since we are python3 only now, drop python-six compat layer (#252)

fe7abcf : Add format support to de-client (#217) (#241)

df5a3d7 : Add wheel support for easier testing (#247)

7de970d : Add place to inject env if need be (#242)

84e2930 : Fix race in test case (#250)

d855aa0 : Fix fact-lookup to support duplicate names in separate rules. (#245)

51370fb : Resolve fixture ‘quickstart’ issue (#238)

3ea9129 : Move from TravisCI to raw actions (#235)

fe9faa9 : implement data product browsing interface (#239)

cf0f3c0 : Add support to use custom base docker container to run tests (#234)

d91722f : Compat with psycopg2cffi (#233)

7d15a8c : Test failing source proxy. (#232)

b9a4bbb : Add debug logs for which threads are created #176 (#231)

6e6f4c9 : Updated Jenkins configuration documentation (#229)

2d9fd7b : Log if config passed validation #117 (#230)

60c46d3 : Self-test needs a real namespace to 'import numpy' in new python eval (#228)

a120077 : Test that the doc actually builds during CI (#227)

4b6240a : Extend timeout for coverage combine (#226)

b059696 : Update workflow per changes at github (#225)

7a71cac : Use newer compilers/runtimes (#224)

15ffd93 : Add header for strict includes (#222)

71b141a : Add special PyPy only requirement (#221)

9dbb932 : Move Python C extension to versioned .so file (#220)

ea7ade5 : Migrate from boost-python to pybind11 (#215)

e6b2eae : Add python 3.9 to testing matrix (#219)

04c8f9c : Add the option to print columns types on de-client (#216)

8815dc6 : Logic-engine cleanups (#211)

086d0d5 : fix missing back tick

54cc084 : modified release notes

24744cf : Synchronize access to the task managers (#214)

87a7fda : replde dash with underscore

743d0fd : try sphinx_rtd_theme

18c7909 : added 1.4.0 release notes

ff3d491 : force docker pull when building the docker container to make sure to use an updated base layer (#210)

## 1.2 Release 1.4.1

In this release:

- Bug fixes to 1.4.0 release

### 1.2.1 Issues fixed in this release

- 213 : de-client hangs under certain circumstances in version 1.4 and greater (race condition) (84ecfe2)

### 1.2.2 Full list of commits since version 1.4.0

9799b9a : update release version to 1.4.1

84ecfe2 : Synchronize access to the task managers (#214)

751b6b8 : Address data races; remove need to sleep in unit tests (#205)

## 1.3 Release 1.4.0

In this release:

- Improvements in error handling and client/server interactions
- Added log rotation by time
- Improvements in code coverage

### 1.3.1 Issues fixed in this release

- 153 : Have de-client –print-product return different error message if product does not exist (18a950c)
- 171 : yum update on decision engine rpm from python2 to python3 doesn't undo the symlinks (eb85c97)
- 188 : Channel debug info now leaks into startup.log (99d20a5)
- 208 : Error when trying to run reaper in version 1.4.0 (84eccf3)

### 1.3.2 Full list of commits since version 1.3

84eccf3 : Fix typo in reaper script. (#209)

d836abf : next RC

926944a : Fix coveralls reporting (#198)

b95c323 : Updating base Dockerfile (#199)

d302e31 : Help jsonnet, which doesn't understand PosixPath objects. (#204)

2d791a7 : Test configuration policies. (#197)

236e27a : Ensure items are returned in a stable order (#202)

e974f5f : add pylinit and pycodestyle (#203)

fbe7616 : Test task manager (#196)

686ca80 : require more recent version of pytest-postgresql (#195)

99d20a5 : Fix double-logging problem. (#192)

4ce3d17 : A set of fixtures to simplify unit tests (#183)

65f8052 : Fix typo (#190)

f3a4be8 : Protect against None workers (#187)

ec310fb : remove py3 from package name

7006489 : bump version to 1.4.0rc

158d835 : decisionengine/framework/modules: Fix SourceProxy retries (#184)

1356bf1 : Add support to test any branch in Jenkins (#182)

692fa8e : Add timeout support for unit test on Jenkins (#181)

e3d6e6a : Updated Jenkins documentation to take into account unit tests timeout parametr (#180)

2586a3e : Configuration redesign (#168)

fac984d : Fix error with DBUtils import. Looks like names of modules changed (#175)

7d661ee : Move postgres-specific implementation to postgres source. (#174)

eb85c97 : Rpm (#173)

10fe843 : Adding log rotation by time (#170)

a8d239b : Various improvements. (#167)

d9b92ee : Ignore vim's *.swp files (#166)

d9f72ef : Fix call to shutdown_timeout (and add sample entry to config) (#165)

3161795 : Add drops for items using tables being dropped (#164)

77d186d : Show output of test runtimes in travis (#163)

81820a4 : Allow server to start with no channels. (#161)

49879a6 : DE server and client usability improvements (#160)

de91c4f : Add tests to default and override config (#158)

14df1f6 : Use python fallthrough for options (#159)

ac64a92 : Drop python 2.7 integration tests since we are python3 only (#157)

d963301 : Update Jenkins pipeline to properly test closing PR (#156)

64248cb : Merge 'runtime' tests into running channel tests (#150)

065ad77 : Adding Jenkins pipeline documentation (#155)

18a950c : fix print-product to report non-existing product as such (#154)

6493735 : Fix invalid attribute name (#152)

d953c6a : Remove unnecessary set_start_method call (#149)

c8c9b65 : guarantee that process is killed so test never hang (#147)

f1542b6 : Channel test (#146)

7f349a8 : Fix faulty TaskManager state type (#145)

d50f1c4 : fix logging regression introduced in f5e299969e0611e3480e9fa2782052df... (#142)

becfa26 : Pass the correct type. (#144)

1a60daf : DecisionEngine: fix typo (#143)

9e7b867 : Updating Jenkins pipeline configuration (#140)

e3a6703 : fix regression introduced in f5e299969e0611e3480e9fa2782052df86d7c4ed (#141)

4900bc6 : Restore runtime test. (#139)

0823f3d : Consolidate DE server/client tests into one file. (#138)

4f84435 : A few more access fixes.

160cfd1 : Fix task manager state access.

c00d819 : A few more cleanups.

ec087e2 : Various cleanups

a309ffe : Improvements to DE client CLI.

## 1.4 Release 1.3.0

In this release:

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

### 1.4.1 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)

## 1.5 Release 1.2.0

In this release:

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

### 1.5.1 decisionengine

3dfe167 : Jenkins pipeline improvements (#106)

22a7073 : pull request for review request 137 (#105)

cafffb2 : Make it possible to run code directly (for tests), and (#100)

802e98b : replace psycog2 witt psycopg2-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 depedencies 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) simplications.

3c7b6b7 : Only run Github Actions for python3.6 (#68)

453cbba : Update README.md

b27ed53 : remove unnecessary (and atually harmful) python shebang (#66)

### 1.5.2 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 (#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)

## 1.6 Release 1.1.0

In this release:

- Fixed. 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 argumement -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

CHAPTER

TWO

# 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/reporpms/EL-7-x86_64/
↪pgdg-redhat-repo-latest.noarch.rpm
yum install -y python3 python3-pip python36-devel postgresql11 postgresql11-server
pip3 install pandas DBUtils psycopg2-binary tabulate mock pytest toposort
```

### 2.1.2 Build & test

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

export PYTHONPATH=`pwd`

python3 -m pytest

==================================== test session starts␣
↪====================================
platform linux -- Python 3.6.8, pytest-6.0.1, py-1.9.0, pluggy-0.13.1
rootdir: /cloud/login/knoepfel/de-devel/decisionengine
plugins: timeout-1.4.2, postgresql-2.5.1, profiling-1.7.0
collected 89 items

framework/config/tests/test_config.py .............                                ␣
↪[ 14%]
framework/config/tests/test_policies.py ....                                       ␣
↪[ 19%]
framework/dataspace/datasources/tests/test_postgresql.py ......                    ␣
↪[ 25%]
framework/dataspace/tests/test_Reaper.py .......                                   ␣
↪[ 33%]
framework/dataspace/tests/test_datablock.py ..............                         ␣
↪[ 49%]
```

(continues on next page)

(continued from previous page)

```
framework/engine/tests/test_client_only.py ..                                  ␣
↪[ 51%]
framework/engine/tests/test_startup.py ..                                      ␣
↪[ 53%]
framework/logicengine/tests/test_cascaded_rules.py ..                          ␣
↪[ 56%]
framework/logicengine/tests/test_construction.py .....                         ␣
↪[ 61%]
framework/logicengine/tests/test_facts.py .....                                ␣
↪[ 67%]
framework/logicengine/tests/test_pandas_fact.py ..                             ␣
↪[ 69%]
framework/logicengine/tests/test_rule_with_negated_fact.py ..                  ␣
↪[ 71%]
framework/logicengine/tests/test_simple_configuration.py ..                    ␣
↪[ 74%]
framework/taskmanager/tests/test_processing_state.py .....                     ␣
↪[ 79%]
framework/taskmanager/tests/test_task_manager.py ...                           ␣
↪[ 83%]
framework/tests/test_defaults.py ...                                           ␣
↪[ 86%]
framework/tests/test_reaper.py ...                                             ␣
↪[ 89%]
framework/tests/test_restart_channel.py .                                      ␣
↪[ 91%]
framework/tests/test_sample_config.py ...                                      ␣
↪[ 94%]
framework/tests/test_start_with_no_channels.py .                               ␣
↪[ 95%]
framework/util/tests/test_fs.py ...                                            ␣
↪[ 98%]
framework/util/tests/test_tsort.py .                                           ␣
↪[100%]

============================== 89 passed in 90.01s (0:01:30)␣
↪==============================
```

## 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%]
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%]
```

(continues on next page)

(continued from previous page)

```
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%]
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␣
↪==============================
```

CHAPTER

THREE

# 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*, *capsys*, *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*, *capsys*)

decisionengine.framework.config.tests.test_config.**test_minimal_jsonnet_wrong_extension**(*load*, *capsys*)

decisionengine.framework.config.tests.test_config.**test_minimal_python**(*load*, *capsys*)

decisionengine.framework.config.tests.test_config.**test_wrong_type**(*load*)

### decisionengine.framework.config.tests.test_policies module

decisionengine.framework.config.tests.test_policies.**test_channel_config_dir**(*tmp_path*, *monkeypatch*)

decisionengine.framework.config.tests.test_policies.**test_global_config_dir**(*tmp_path*, *monkeypatch*)

decisionengine.framework.config.tests.test_policies.**test_global_config_file**(*tmp_path*, *monkeypatch*)

decisionengine.framework.config.tests.test_policies.**test_valid_dir**(*tmp_path*)

### 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*, *channel_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 my call 'dump()' to print out in a string form the JSON configuration.

**_abc_impl = <_abc._abc_data 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 pathlib.Path object.

This function returns a path object according to the following precedence rules:

1. If the 'parent_dir' argument is provided, the returned path object will correspond to '{parent_dir}/config.d'.
2. If the 'CHANNEL_CONFIG_PATH' environment variable has been set, the returned path object will correspond to ${CHANNEL_CONFIG_PATH}.
3. If neither 1 or 2 apply, the returned path object corresponds to '{global_config_dir()}/config.d' (see documentation for 'global_config_dir()').

Regardless of the precedence rule used, the returned path object must be a valid directory or an exception will be raised–i.e. if the 'parent_dir' argument is supplied, and the resulting path object is not a valid directory, the function will exit with an exception and not attempt rule 2 or 3.

decisionengine.framework.config.policies.**global_config_dir**()
Retrieve global configuration dir as pathlib.Path object.

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**(*parent_dir=None*)
Return the pathlib.Path object corresponding to the global configuration.

If supplied, the 'parent_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.

decisionengine.framework.config.policies.**valid_dir**(*path*, *scope*)
Throws if the supplied path object is not a directory, otherwise returns the path object.

### 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.fixtures.**mock_data_block**()
This fixture replaces the standard datablock implementation.

The current DataBlock implementation does not own any data products but forwards them immediately to a backend datasource. The only implemented datasource requires Postgres, which is overkill when needing to test simple data-product communication between modules.

This mock datablock class directly owns the data products, thus avoiding the need for a datasource backend. It is anticipated that a future design of the DataBlock will own the data products, thus making this mock class unnecessary.

### decisionengine.framework.dataspace.datasources.tests.test_postgresql module

decisionengine.framework.dataspace.datasources.tests.test_postgresql.**data**()

decisionengine.framework.dataspace.datasources.tests.test_postgresql.**dataproduct**()

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**(*dat*

decisionengine.framework.dataspace.datasources.tests.test_postgresql.**test_generate_insert_**

decisionengine.framework.dataspace.datasources.tests.test_postgresql.**test_get_last_generat**

decisionengine.framework.dataspace.datasources.tests.test_postgresql.**test_get_taskmanager**(

decisionengine.framework.dataspace.datasources.tests.test_postgresql.**test_insert**(*datasource*, *dataproduct*, *header*, *metadata*)

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_impl = <_abc._abc_data 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 that 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_dataproducts**(*taskmanager_id*)
Return list of all data products associated with with taskmanager_id

**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

**get_taskmanagers**(*taskmanager_name=None*, *start_time=None*, *end_time=None*)
Retrieve TaskManagers :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 = {'dataproduct': ['taskmanager_id TEXT', 'generation_id INT', 'key TEXT', 'va**

**update**(*taskmanager_id*, *generation_id*, *key*, *value*, *header*, *metadata*)
Update the data in respective tables for the given taskmanager_id, generation_id, key

**Parameters**

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

decisionengine.framework.dataspace.datasources.postgresql.**generate_insert_query**(*table_name*, *keys*)
Generate insert query given table name and list of fields

**Parameters**

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

**Keys** list

**Return type** str - insert query

### Module contents

### Submodules

### decisionengine.framework.dataspace.datablock module

**class** decisionengine.framework.dataspace.datablock.**DataBlock**(*dataspace*, *name*, *taskmanager_id=None*, *generation_id=None*, *sequence_id=None*)

Bases: `object`

**_insert**(*key*, *value*, *header*, *metadata*)
Insert a new product into database with header and metadata

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

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

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

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

**Return type** *DataBlock*

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

**Return type** `dict`

**get_dataproducts**()

**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=psycopg2.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_impl = <_abc._abc_data 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

**class** decisionengine.framework.dataspace.datablock.**Metadata**(*taskmanager_id*, *state='NEW'*, *generation_id=None*, *generation_time=None*, *missed_update_count=0*)
Bases: `collections.UserDict`

**_abc_impl = <_abc._abc_data 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_impl = <_abc._abc_data object>**

**abstract close**()
Close all connections to the database

**abstract connect**()
Create a pool of database connections

**abstract create_tables**()
Create database tables

**dataproduct_table = 'dataproduct'**
Name of the dataproduct table

**abstract delete_data_older_than**(*days*)
Delete data older that interval :type days: long :arg days: remove data older than interval

**abstract duplicate_datablock**(*taskmanager_id*, *generation_id*, *new_generation_id*)
For the given taskmanager_id, make a copy of the datablock with given generation_id, set the generation_id for the datablock copy

**Parameters**

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

**abstract get_datablock**(*taskmanager_id*, *generation_id*)
Return the entire datablock from the 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

**abstract get_dataproduct**(*taskmanager_id*, *generation_id*, *key*)
Return the data from the dataproduct table for the given taskmanager_id, generation_id, key

**Parameters**

- **taskmanager_id** (string) – taskmanager_id for generation to be retrieved
- **generation_id** (int) – generation_id of the data

- **key** (string) – key for the value

**abstract get_dataproducts**(*taskmanager_id*)
Return list of all data products associated with with taskmanager_id

**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

**abstract get_taskmanagers**(*taskmanager_name=None*, *start_time=None*, *end_time=None*)
Retrieve TaskManagers :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_dataproducts**(*taskmanager_id*)

**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*)

**get_taskmanagers**(*taskmanager_name=None*, *start_time=None*, *end_time=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

**__has_state_no_lock**(*this_state*)
During startup we check state, but we don't want to lock and prevent the thread from changing the state. That is the condition we are activly looking for!

**_reaper_loop**(*delay*)
The first thing this loop does should be to set the state to State.STARTING so the caller can validate the thread is in fact running and doing things.

**_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*, *channel_config_loader*, *server_address*)
Bases: socketserver.ThreadingMixIn, xmlrpc.server.SimpleXMLRPCServer

**_dispatch**(*method*, *params*)
Dispatches the XML-RPC method.

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

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

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

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

**block_until**(*state*)

**block_while**(*state*)

**get_logger**()

**handle_sighup**(*signum*, *frame*)

**reaper_start**(*delay*)

**reaper_status**()

**reaper_stop**()

**rpc_block_while**(*state_str*)

**rpc_get_channel_log_level**(*channel*)

**rpc_get_log_level**()

**rpc_print_product**(*product*, *columns=None*, *query=None*, *types=False*, *format=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**()

**start_channel**(*channel_name*, *channel_config*)

**start_channels**()

**stop_channel**(*channel*)

**stop_channels**()

**stop_worker**(*worker*)

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

Bases: xmlrpc.server.SimpleXMLRPCRequestHandler

**rpc_paths = ('/RPC2',)**

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

decisionengine.framework.engine.DecisionEngine.**_create_de_server**(*global_config*, *channel_config_loader*)

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

decisionengine.framework.engine.DecisionEngine.**_get_de_conf_manager**(*global_config_dir*, *channel_config_dir*, *options*)

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

decisionengine.framework.engine.DecisionEngine.**_start_de_server**(*global_config*, *channel_config_loader*)
Create and 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.Workers module

**class** decisionengine.framework.engine.Workers.**Worker**(*task_manager*, *logger_config*)
Bases: multiprocessing.context.Process

Class that encapsulates a channel's task manager as a separate process.

This class' run function is called whenever the process is started. If the process is abruptly terminated–e.g. the run method is pre-empted by a signal or an os._exit(n) call–the Worker object will still exist even if the operating-system process no longer does.

To determine the exit code of this process, use the Worker.exitcode value, provided by the multiprocessing.Process base class.

**get_state_name**()

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

**wait_until**(*state*)

**wait_while**(*state*)

**class** decisionengine.framework.engine.Workers.**Workers**
Bases: object

This class manages and provides access to the task-manager workers.

The intention is that the decision engine never directly interacts with the workers but refers to them via a context manager:

**with workers.access() as ws:** # Access to ws now protected ws['new_channel'] = Worker(...)

In cases where the decision engine's block_while or block_until methods must be called (e.g. during tests), one should used the unguarded access:

**with workers.unguarded_access() as ws:** # Access to ws is unprotected ws['new_channel'].wait_until(...)

Calling a blocking method while using the protected context manager (i.e. workers.access()) will likely result in a deadlock.

**class Access**(*workers*, *lock*)
Bases: object

**_update_channel_states**()

**access**()

**unguarded_access**()

### 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.logicengine package

### Subpackages

### decisionengine.framework.logicengine.tests package

### Submodules

### decisionengine.framework.logicengine.tests.test_cascaded_rules module

decisionengine.framework.logicengine.tests.test_cascaded_rules.**myengine**()

decisionengine.framework.logicengine.tests.test_cascaded_rules.**test_rule_that_does_not_fire**

decisionengine.framework.logicengine.tests.test_cascaded_rules.**test_rule_that_fires**(*myengine*

### decisionengine.framework.logicengine.tests.test_construction module

decisionengine.framework.logicengine.tests.test_construction.**test_configuration_with_fact_u**

decisionengine.framework.logicengine.tests.test_construction.**test_configuration_with_numy_f**

decisionengine.framework.logicengine.tests.test_construction.**test_default_construction**()

LogicEngine is not default constructible.

decisionengine.framework.logicengine.tests.test_construction.**test_trivial_configuration**()

Logic engine constructed with trivial rules and facts.

decisionengine.framework.logicengine.tests.test_construction.**test_wrong_configuration**()

LogicEngine construction requires rules and facts; if we don't supply them it is an error.

### decisionengine.framework.logicengine.tests.test_duplicate_fact_names module

decisionengine.framework.logicengine.tests.test_duplicate_fact_names.**test_duplicate_fact_na**

### decisionengine.framework.logicengine.tests.test_facts module

decisionengine.framework.logicengine.tests.test_facts.**make_db**(*maximum*)

decisionengine.framework.logicengine.tests.test_facts.**test_compound_fact**()

decisionengine.framework.logicengine.tests.test_facts.**test_fact_using_numpy_array**()

decisionengine.framework.logicengine.tests.test_facts.**test_fact_using_numpy_function**()

decisionengine.framework.logicengine.tests.test_facts.**test_fact_with_nested_names**()

decisionengine.framework.logicengine.tests.test_facts.**test_simple_fact**()

### decisionengine.framework.logicengine.tests.test_pandas_fact module

decisionengine.framework.logicengine.tests.test_pandas_fact.**mydata**(*y*)
Return a 'datablock' surrogate carrying a Pandas DataFrame, and a parameter named 'y' with value y.

decisionengine.framework.logicengine.tests.test_pandas_fact.**myengine**()

decisionengine.framework.logicengine.tests.test_pandas_fact.**test_rule_that_does_not_fire**(*m*
Rules that do not fire do not create entries in the returned actions and newfacts.

decisionengine.framework.logicengine.tests.test_pandas_fact.**test_rule_that_fires**(*myengine*)

### decisionengine.framework.logicengine.tests.test_rule_with_negated_fact module

decisionengine.framework.logicengine.tests.test_rule_with_negated_fact.**myengine**()

decisionengine.framework.logicengine.tests.test_rule_with_negated_fact.**test_rule_that_does_**
Rules that do not fire do not create entries in the returned actions and newfacts.

decisionengine.framework.logicengine.tests.test_rule_with_negated_fact.**test_rule_that_fires**

### decisionengine.framework.logicengine.tests.test_simple_configuration module

decisionengine.framework.logicengine.tests.test_simple_configuration.**myengine**()

decisionengine.framework.logicengine.tests.test_simple_configuration.**test_rule_that_does_no**
Rules that do not fire do not create entries in the returned actions and newfacts.

decisionengine.framework.logicengine.tests.test_simple_configuration.**test_rule_that_fires**(*m*

### Module contents

### Submodules

#### decisionengine.framework.logicengine.BooleanExpression module

**class** decisionengine.framework.logicengine.BooleanExpression.**BooleanExpression**(*expr*)
Bases: object

**evaluate**(*d*)
Return the evaluated Boolen value of this expression in the context of the given data 'd'.

**exception** decisionengine.framework.logicengine.BooleanExpression.**LogicError**
Bases: TypeError

decisionengine.framework.logicengine.BooleanExpression.**function_name_from_call**(*callnode*)

#### decisionengine.framework.logicengine.FactLookup module

**class** decisionengine.framework.logicengine.FactLookup.**FactLookup**(*fact_names*, *rules_cfg*)
Bases: object

Establishes a policy for looking up a fact based on the given name.

To wit, the first fact with a given name is the one that is used in the evaluation of all subsequent facts.

As an example, consider the following configuration:

**facts: {** should_publish: "(True)",

}, rules: {

**publish_1: {** expression: "should_publish", facts: ["should_publish"]

}, publish_2: {

expression: "should_publish", actions: ["go_to_press"] facts: ["should_publish"]

} retract: {

expression: "not should_publish", facts: ["should_retract"]

}

In the above, the first fact to be evaluated will always be the top-level facts (i.e. those not encapsulated by the 'rules' table). The rules labeled 'publish_1' and 'publish_2' both rely on the 'should_publish' fact in their expressions, and they in turn create their own facts with the same name. FactLookup ensures that 'publish_1' and 'publish_2' will both use the evaluated fact from the top-level 'facts' table.

**rule_for**(*fact_name*)
Selects rule required to evaluate fact with the supplied name.

**Parameters** **fact_name** (*str*) – Name of fact for which rule will be selected.

**Return type** str

**Returns** Rule name

**sorted_rules**(*rules_cfg*)
Rules sorted according to rule dependencies.

**Parameters** **rules_cfg** (*dict*) – rules as specified in logic-engine configuration

**Return type** list

**Returns** Rules to be evaluated by the rule engine.

### decisionengine.framework.logicengine.LogicEngine module

**class** decisionengine.framework.logicengine.LogicEngine.**LogicEngine**(*cfg*)
Bases: *decisionengine.framework.modules.Module.Module*

**_create_facts_dataframe**(*newfacts*)
Convert newfacts dict in format below to dataframe with columns ['rule_name', 'fact_name', fact_value']

facts dict format: 'newfacts': {

**'publish_glidein_requests': {** 'allow_hpc_new': True, 'allow_foo': True

}, 'dummy_rule': {

'dummy_new_fact': True

}

}

**consumes**()
Return the names of all the items that must be in the DataBlock for the rules to be evaluated.

**evaluate**(*db*)
Evaluate our facts and rules, in the context of the given data. db can be any mappable, in particular a DataBlock or dictionary.

**Parameters** **db** (DataBlock) – Products used to evaluate facts.

**evaluate_facts**(*db*)

**Parameters** **db** (DataBlock) – Products used to evaluate facts.

**Return type** dict

**Returns** Evaluated fact values (e.g. True or False) for each fact name.

**produces**()

### decisionengine.framework.logicengine.Rule module

**class** decisionengine.framework.logicengine.Rule.**Rule**(*rule_name*, *rule_cfg*)
Bases: object

In-memory representation of logic-engine rule, relying on parsing utilities in BooleanExpression.

**evaluate**(*evaluated_facts*)
Evaluates a compiled expression given the supplied facts.

**Parameters** **evaluated_facts** (*dict*) – Initial fact values (e.g. True or False) for each fact name.

**Return type** bool

### decisionengine.framework.logicengine.RuleEngine module

**class** decisionengine.framework.logicengine.RuleEngine.**RuleEngine**(*fact_names*, *rules_cfg*)

Bases: object

Engine responsible for evaluating logic-engine rules.

This class is responsible for (a) forming a sorted set of rules that supports dependencies between them, and (b) evaluating the rules according to a specified fact-lookup policy.

**execute**(*evaluated_facts*)

Evaluates all rules given the supplied facts.

**Parameters** **evaluated_facts** (*dict*) – Initial fact values (e.g. True or False) for each fact name.

**Return type** tuple

**Returns** Actions to be taken based on rule evaluation; new facts produced during that evaluation.

### 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_bock**()

**get_paramaters**()

**set_data_bock**(*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*)

**shutdown**()

### 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 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.ProcessingState module

The ProcessingState class can represent any of the following task-manager states:

BOOT STEADY OFFLINE SHUTTINGDOWN SHUTDOWN ERROR

In addition, the class supports 'wait_until(state)' and 'wait_while(state)' methods, which, when called from a different process, block until the state has been entered or exited, respectively.

**class** decisionengine.framework.taskmanager.ProcessingState.**ProcessingState**(*state=<State.BOOT: 0>*)

Bases: object

**get**()

**has_value**(*state*)

**inactive**()

**set**(*state*)

**should_stop**()

**wait_until**(*state*)

**wait_while**(*state*)

**class** decisionengine.framework.taskmanager.ProcessingState.**State**(*value*)

Bases: enum.Enum

An enumeration.

**BOOT = 0**

**ERROR = 5**

**OFFLINE = 4**

**SHUTDOWN = 3**

**SHUTTINGDOWN = 2**

**STEADY = 1**

### decisionengine.framework.taskmanager.TaskManager module

Task Manager

**class** decisionengine.framework.taskmanager.TaskManager.**Channel**(*channel_dict*)

Bases: object

Decision Channel. Instantiates workers according to channel configuration

**class** decisionengine.framework.taskmanager.TaskManager.**TaskManager**(*name*, *generation_id*, *channel_dict*, *global_config*)

Bases: object

Task Manager

**data_block_put**(*data*, *header*, *data_block*)

Put data into data block

**Parameters**

- **data** (dict) – key, value pairs

- **header** (Header) – data header
- **data_block** (DataBlock) – data block

**decision_cycle**()
Decision cycle to be run periodically (by trigger)

**do_backup**()
Duplicate current data block and return its copy

**Return type** DataBlock

**get_loglevel**()

**get_state**()

**get_state_name**()

**get_state_value**()

**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.

**start_sources**(*data_block=None*)
Start sources, each in a separate thread

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

**take_offline**(*current_data_block*)
offline and stop task manager

**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 sycronise 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.FailingPublisher module

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

**consumes**(*name_list*)

**publish**(*data_block*)

### decisionengine.framework.tests.FailingSourceProxy module

**class** decisionengine.framework.tests.FailingSourceProxy.**FailingSourceProxy**(*config*)
Bases: *decisionengine.framework.modules.SourceProxy.SourceProxy*

**acquire**()
Overrides Source class method

**produces**()
Assumes that self.datakeys has the following structure data_keys[key1] = (data_product_name, data_product_name_translation) .... or

### 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*, *channel_conf_path=None*, *channel_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 a 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_client_server module

Fixture based DE Server for the de-client tests

decisionengine.framework.tests.test_client_server.**test_client_print_product**(*deserver*)

### 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**
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_failing_source_proxy module

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

decisionengine.framework.tests.test_failing_source_proxy.**test_stop_failing_source_proxy**(*des*

### 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**(*deser*
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_restart_channel module

decisionengine.framework.tests.test_restart_channel.**deserver_mock_data_block**(*mock_data_block*)

decisionengine.framework.tests.test_restart_channel.**test_restart_channel**(*deserver_mock_data_bloc*

### 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_confi**
Verify config has expected items

decisionengine.framework.tests.test_sample_config.**test_client_can_get_de_server_show_logge**
Verify can fetch log level

decisionengine.framework.tests.test_sample_config.**test_client_can_get_de_server_status**(*dese*
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.**deserver_mock_data_block**(*mock_dat*

decisionengine.framework.tests.test_start_with_no_channels.**test_start_from_nothing**(*deserver_m*

### 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.

Results are sorted by channel name to ensure stable output.

### 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

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

#### Submodules

#### decisionengine.framework.about module

PEP-0396 provides instructions for providing module versions While we are at it, add a few other useful bits

#### Module contents

#### Module contents

## 3.2 Indices and tables

- genindex
- modindex
- search

CHAPTER

FOUR

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## d


decisionengine, 47
decisionengine.framework, 47
decisionengine.framework.about, 47
decisionengine.framework.config, 21
decisionengine.framework.config.ChannelConfigHandler, 18
decisionengine.framework.config.policies, 20
decisionengine.framework.config.tests, 18
decisionengine.framework.config.tests.test_config, 17
decisionengine.framework.config.tests.test_policies, 18
decisionengine.framework.config.ValidConfig, 19
decisionengine.framework.dataspace, 31
decisionengine.framework.dataspace.datablock, 25
decisionengine.framework.dataspace.datasource, 27
decisionengine.framework.dataspace.datasources, 25
decisionengine.framework.dataspace.datasources.postgresql, 22
decisionengine.framework.dataspace.datasources.tests, 22
decisionengine.framework.dataspace.datasources.tests.fixtures, 21
decisionengine.framework.dataspace.datasources.tests.test_postgresql, 21
decisionengine.framework.dataspace.dataspace, 29
decisionengine.framework.engine, 34
decisionengine.framework.engine.de_client, 34
decisionengine.framework.engine.DecisionEngine, 31
decisionengine.framework.engine.Workers, 33
decisionengine.framework.logicengine, 38
decisionengine.framework.logicengine.BooleanExpress 36
decisionengine.framework.logicengine.FactLookup, 36
decisionengine.framework.logicengine.LogicEngine, 37
decisionengine.framework.logicengine.Rule, 37
decisionengine.framework.logicengine.RuleEngine, 38
decisionengine.framework.logicengine.tests, 36
decisionengine.framework.logicengine.tests.test_cas 34
decisionengine.framework.logicengine.tests.test_cor 34
decisionengine.framework.logicengine.tests.test_dup 35
decisionengine.framework.logicengine.tests.test_fac 35
decisionengine.framework.logicengine.tests.test_par 35
decisionengine.framework.logicengine.tests.test_rul 35
decisionengine.framework.logicengine.tests.test_sin 35
decisionengine.framework.modules, 40
decisionengine.framework.modules.de_logger, 40
decisionengine.framework.modules.LogicEngine, 38
decisionengine.framework.modules.Module, 38
decisionengine.framework.modules.Publisher, 38
decisionengine.framework.modules.Source, 39
decisionengine.framework.modules.SourceProxy, 39
decisionengine.framework.modules.Transform, 40

decisionengine.framework.taskmanager,
43
decisionengine.framework.taskmanager.ProcessingState,
40
decisionengine.framework.taskmanager.TaskManager,
41
decisionengine.framework.tests, 46
decisionengine.framework.tests.FailingPublisher,
43
decisionengine.framework.tests.FailingSourceProxy,
43
decisionengine.framework.tests.fixtures,
44
decisionengine.framework.tests.PublisherNOP,
43
decisionengine.framework.tests.SourceNOP,
44
decisionengine.framework.tests.test_client_server,
45
decisionengine.framework.tests.test_defaults,
45
decisionengine.framework.tests.test_failing_source_proxy,
45
decisionengine.framework.tests.test_reaper,
45
decisionengine.framework.tests.test_restart_channel,
45
decisionengine.framework.tests.test_sample_config,
46
decisionengine.framework.tests.test_start_with_no_channels,
46
decisionengine.framework.tests.TransformNOP,
44
decisionengine.framework.util, 47
decisionengine.framework.util.fs, 46
decisionengine.framework.util.sockets,
46
decisionengine.framework.util.tsort, 46

# INDEX

## Symbols


__has_state_no_lock() (*decisionengine.framework.dataspace.dataspace.Reaper method*), 30
__query() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 22
_abc_impl (*decisionengine.framework.config.ValidConfig.ValidConfig attribute*), 19
_abc_impl (*decisionengine.framework.dataspace.datablock.Header attribute*), 26
_abc_impl (*decisionengine.framework.dataspace.datablock.Metadata attribute*), 26
_abc_impl (*decisionengine.framework.dataspace.datasource.DataSource attribute*), 27
_abc_impl (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql attribute*), 22
_channel_config_dir() (*in module decisionengine.framework.config.tests.test_config*), 17
_channel_preamble() (*in module decisionengine.framework.engine.DecisionEngine*), 32
_check_keys() (*in module decisionengine.framework.config.ChannelConfigHandler*), 19
_config_from_file() (*in module decisionengine.framework.config.ValidConfig*), 19
_convert_to_json() (*in module decisionengine.framework.config.ValidConfig*), 19
_create_de_server() (*in module decisionengine.framework.engine.DecisionEngine*), 32
_create_facts_dataframe() (*decisionengine.framework.logicengine.LogicEngine.LogicEngine method*), 37
_create_worker() (*in module decisionengine.framework.taskmanager.TaskManager*), 43
_delete() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 22
_dispatch() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 31
_ds (*decisionengine.framework.dataspace.dataspace.DataSourceLoader attribute*), 29
_get_data() (*decisionengine.framework.modules.SourceProxy.SourceProxy method*), 39
_get_de_conf_manager() (*in module decisionengine.framework.engine.DecisionEngine*), 32
_get_global_config() (*in module decisionengine.framework.engine.DecisionEngine*), 33
_global_config_file() (*in module decisionengine.framework.config.tests.test_config*), 17
_insert() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 25
_insert() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 22
_insert_returning_result() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 22
_instances (*decisionengine.framework.dataspace.dataspace.Singleton attribute*), 30
_load_channel() (*decisionengine.framework.config.ChannelConfigHandler.ChannelConfig method*), 18
_make_logger() (*in module decisionengine.framework.config.ChannelConfigHandler*), 19
_make_workers_for() (*in module decisionengine.framework.taskmanager.TaskManager*), 43
_reaper_loop() (*decisionengine.framework.dataspace.dataspace.Reaper method*), 30
_remove() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 22
_select() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 22
_select_dictresult() (*decisio-*

*nengine.framework.dataspace.datasources.postgresql.Postgresql method*), 22
_select_getresult() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 22
_select_tuple() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 22
_set_state() (*decisionengine.framework.dataspace.dataspace.Reaper method*), 30
_setitem() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 25
_start_de_server() (*in module decisionengine.framework.engine.DecisionEngine*), 33
_tables_created (*decisionengine.framework.dataspace.dataspace.DataSpace attribute*), 29
_update() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 25
_update() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 22
_update_channel_states() (*decisionengine.framework.engine.Workers.Workers method*), 33
_update_returning_result() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 22
_validate() (*in module decisionengine.framework.config.ChannelConfigHandler*), 19

## A

access() (*decisionengine.framework.engine.Workers.Workers method*), 33
acquire() (*decisionengine.framework.modules.Source.Source method*), 39
acquire() (*decisionengine.framework.modules.SourceProxy.SourceProxy method*), 39
acquire() (*decisionengine.framework.tests.FailingSourceProxy.FailingSourceProxy method*), 43
acquire() (*decisionengine.framework.tests.SourceNOP.SourceNOP method*), 44

## B

block_until() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 31
block_while() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 31
BooleanExpression (*class in decisionengine.framework.logicengine.BooleanExpression*), 36
BOOT (*decisionengine.framework.taskmanager.ProcessingState.State attribute*), 41

## C

Channel (*class in decisionengine.framework.taskmanager.TaskManager*), 41
channel_config_dir() (*in module decisionengine.framework.config.policies*), 20
ChannelConfigHandler (*class in decisionengine.framework.config.ChannelConfigHandler*), 18
close() (*decisionengine.framework.dataspace.datasource.DataSource method*), 27
close() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 22
close() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 29
compress() (*in module decisionengine.framework.dataspace.datablock*), 26
connect() (*decisionengine.framework.dataspace.datasource.DataSource method*), 27
connect() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 22
consumes() (*decisionengine.framework.logicengine.LogicEngine.LogicEngine method*), 37
consumes() (*decisionengine.framework.modules.Publisher.Publisher method*), 38
consumes() (*decisionengine.framework.modules.SourceProxy.SourceProxy method*), 39
consumes() (*decisionengine.framework.modules.Transform.Transform method*), 40
consumes() (*decisionengine.framework.tests.FailingPublisher.FailingPublisher method*), 43
consumes() (*decisionengine.framework.tests.PublisherNOP.PublisherNOP method*), 43
consumes() (*decisionengine.framework.tests.TransformNOP.TransformNOP method*), 44
create_datasource() (*decisionengine.framework.dataspace.dataspace.DataSourceLoader static method*), 29
create_parser() (*in module decisionengine.framework.engine.de_client*), 34

create_tables() (*decisionengine.framework.dataspace.datasource.DataSource method*), 27
create_tables() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 22

## D

data() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 21
data_block_put() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 41
DataBlock (*class in decisionengine.framework.dataspace.datablock*), 25
dataproduct() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 21
dataproduct_table (*decisionengine.framework.dataspace.datasource.DataSource attribute*), 27
DataSource (*class in decisionengine.framework.dataspace.datasource*), 27
datasource() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 21
DataSourceLoader (*class in decisionengine.framework.dataspace.dataspace*), 29
DataSpace (*class in decisionengine.framework.dataspace.dataspace*), 29
DataSpaceConfigurationError, 30
DataSpaceConnectionError, 30
DataSpaceError, 30
DataSpaceExistsError, 30
DE_DB() (*in module decisionengine.framework.tests.fixtures*), 44
decision_cycle() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 42
decisionengine
    module, 47
DecisionEngine (*class in decisionengine.framework.engine.DecisionEngine*), 31
decisionengine.framework
    module, 47
decisionengine.framework.about
    module, 47
decisionengine.framework.config
    module, 21
decisionengine.framework.config.ChannelConfigHandle
    module, 18
decisionengine.framework.config.policies
    module, 20
decisionengine.framework.config.tests
    module, 18
decisionengine.framework.config.tests.test_config
    module, 17
decisionengine.framework.config.tests.test_policies
    module, 18
decisionengine.framework.config.ValidConfig
    module, 19
decisionengine.framework.dataspace
    module, 31
decisionengine.framework.dataspace.datablock
    module, 25
decisionengine.framework.dataspace.datasource
    module, 27
decisionengine.framework.dataspace.datasources
    module, 25
decisionengine.framework.dataspace.datasources.post
    module, 22
decisionengine.framework.dataspace.datasources.test
    module, 22
decisionengine.framework.dataspace.datasources.test
    module, 21
decisionengine.framework.dataspace.datasources.test
    module, 21
decisionengine.framework.dataspace.dataspace
    module, 29
decisionengine.framework.engine
    module, 34
decisionengine.framework.engine.de_client
    module, 34
decisionengine.framework.engine.DecisionEngine
    module, 31
decisionengine.framework.engine.Workers
    module, 33
decisionengine.framework.logicengine
    module, 38
decisionengine.framework.logicengine.BooleanExpress
    module, 36
decisionengine.framework.logicengine.FactLookup
    module, 36
decisionengine.framework.logicengine.LogicEngine
    module, 37
decisionengine.framework.logicengine.Rule
    module, 37
decisionengine.framework.logicengine.RuleEngine
    module, 38
decisionengine.framework.logicengine.tests
    module, 36
decisionengine.framework.logicengine.tests.test_cas

module, 34
decisionengine.framework.logicengine.tests
module, 34
decisionengine.framework.logicengine.tests.fixtures
module, 35
decisionengine.framework.logicengine.tests.test_engine
module, 35
decisionengine.framework.logicengine.tests.test_engine_fail
module, 35
decisionengine.framework.logicengine.tests.test_engine_with_new_framework
module, 35
decisionengine.framework.modules
module, 40
decisionengine.framework.modules.de_logger
module, 40
decisionengine.framework.modules.LogicEngine
module, 38
decisionengine.framework.modules.Module
module, 38
decisionengine.framework.modules.Publisher
module, 38
decisionengine.framework.modules.Source
module, 39
decisionengine.framework.modules.SourceProxy
module, 39
decisionengine.framework.modules.Transform
module, 40
decisionengine.framework.taskmanager
module, 43
decisionengine.framework.taskmanager.ProcessState
module, 40
decisionengine.framework.taskmanager.TaskManager
module, 41
decisionengine.framework.tests
module, 46
decisionengine.framework.tests.FailingPublisher
module, 43
decisionengine.framework.tests.FailingSourceProxy
module, 43
decisionengine.framework.tests.fixtures
module, 44
decisionengine.framework.tests.PublisherNOP
module, 43
decisionengine.framework.tests.SourceNOP
module, 44
decisionengine.framework.tests.test_client_server
module, 45
decisionengine.framework.tests.test_default_config
module, 45
decisionengine.framework.tests.test_failing_sources
module, 45
decisionengine.framework.tests.test_reaper
module, 45
decisionengine.framework.tests.test_restart_channel
module, 45
decisionengine.framework.tests.test_sample_config
module, 46
decisionengine.framework.tests.test_start_with_no_channels
module, 46
decisionengine.framework.tests.TransformNOP
module, 44
decisionengine.framework.util
module, 47
decisionengine.framework.util.fs
module, 46
decisionengine.framework.util.sockets
module, 46
decisionengine.framework.util.tsort
module, 46
decompress() (*in module decisionengine.framework.dataspace.datablock*), 26
default_data_lifetime (*decisionengine.framework.dataspace.datablock.Header attribute*), 26
delete() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 29
delete_data_older_than() (*decisionengine.framework.dataspace.datasource.DataSource method*), 27
delete_data_older_than() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 22
DESTROYED (*in module decisionengine.framework.tests.fixtures*), 44
destroy_mock_data_block() (*in module decisionengine.framework.tests.test_restart_channel*), 45
destroy_mock_data_block() (*in module decisionengine.framework.tests.test_start_with_no_channels*), 46
do_backup() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 42
dump() (*decisionengine.framework.config.ValidConfig.ValidConfig method*), 19
duplicate() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 25
duplicate_datablock() (*decisionengine.framework.dataspace.datasource.DataSource method*), 27
duplicate_datablock() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql*

*method*), 22
duplicate_datablock() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 29

## E

ERROR (*decisionengine.framework.dataspace.dataspace.State attribute*), 31
ERROR (*decisionengine.framework.taskmanager.ProcessingState.State attribute*), 41
evaluate() (*decisionengine.framework.logicengine.BooleanExpression.BooleanExpression method*), 36
evaluate() (*decisionengine.framework.logicengine.LogicEngine.LogicEngine method*), 37
evaluate() (*decisionengine.framework.logicengine.Rule.Rule method*), 37
evaluate() (*decisionengine.framework.modules.LogicEngine.LogicEngine method*), 38
evaluate_facts() (*decisionengine.framework.logicengine.LogicEngine.LogicEngine method*), 37
execute() (*decisionengine.framework.logicengine.RuleEngine.RuleEngine method*), 38
execute_command_from_args() (*in module decisionengine.framework.engine.de_client*), 34

## F

FactLookup (*class in decisionengine.framework.logicengine.FactLookup*), 36
FailingPublisher (*class in decisionengine.framework.tests.FailingPublisher*), 43
FailingSourceProxy (*class in decisionengine.framework.tests.FailingSourceProxy*), 43
files_with_extensions() (*in module decisionengine.framework.util.fs*), 46
function_name_from_call() (*in module decisionengine.framework.logicengine.BooleanExpression*), 36

## G

generate_insert_query() (*in module decisionengine.framework.dataspace.datasources.postgresql*), 24
get() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 25
get() (*decisionengine.framework.taskmanager.ProcessingState.ProcessingState method*), 41
get_channels() (*decisionengine.framework.config.ChannelConfigHandler.ChannelConfig method*), 18
get_connection() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 23
get_data_bock() (*decisionengine.framework.modules.Module.Module method*), 38
get_datablock() (*decisionengine.framework.dataspace.datasource.DataSource method*), 27
get_datablock() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 23
get_dataproduct() (*decisionengine.framework.dataspace.datasource.DataSource method*), 27
get_dataproduct() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 23
get_dataproduct() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 29
get_dataproducts() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 25
get_dataproducts() (*decisionengine.framework.dataspace.datasource.DataSource method*), 28
get_dataproducts() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 23
get_dataproducts() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 29
get_header() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 25
get_header() (*decisionengine.framework.dataspace.datasource.DataSource method*), 28
get_header() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 23
get_header() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 29
get_last_generation_id() (*decisionengine.framework.dataspace.datasource.DataSource method*), 28
get_last_generation_id() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 23

get_last_generation_id() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 29
get_logger() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 31
get_logger() (*in module decisionengine.framework.modules.de_logger*), 40
get_loglevel() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 42
get_metadata() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 25
get_metadata() (*decisionengine.framework.dataspace.datasource.DataSource method*), 28
get_metadata() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 23
get_metadata() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 29
get_paramaters() (*decisionengine.framework.modules.Module.Module method*), 38
get_produces() (*decisionengine.framework.config.ChannelConfigHandler.ChannelConfigHandler method*), 18
get_random_port() (*in module decisionengine.framework.util.sockets*), 46
get_retention_interval() (*decisionengine.framework.dataspace.dataspace.Reaper method*), 30
get_schema() (*decisionengine.framework.dataspace.datasource.DataSource method*), 28
get_schema() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 23
get_state() (*decisionengine.framework.dataspace.dataspace.Reaper method*), 30
get_state() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 42
get_state_name() (*decisionengine.framework.engine.Workers.Worker method*), 33
get_state_name() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 42
get_state_value() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 42
get_taskmanager() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 25
get_taskmanager() (*decisionengine.framework.dataspace.datasource.DataSource method*), 28
get_taskmanager() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 24
get_taskmanager() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 29
get_taskmanagers() (*decisionengine.framework.dataspace.datasource.DataSource method*), 28
get_taskmanagers() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 24
get_taskmanagers() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 29
global_config_dir() (*in module decisionengine.framework.config.policies*), 20
global_config_file() (*in module decisionengine.framework.config.policies*), 20

## H

handle_sighup() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 31
has_value() (*decisionengine.framework.taskmanager.ProcessingState.ProcessingState method*), 41
Header (*class in decisionengine.framework.dataspace.datablock*), 26
header() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 21
header_table (*decisionengine.framework.dataspace.datasource.DataSource attribute*), 28

## I

IDLE (*decisionengine.framework.dataspace.dataspace.State attribute*), 31
inactive() (*decisionengine.framework.taskmanager.ProcessingState.ProcessingState method*), 41
insert() (*decisionengine.framework.dataspace.datasource.DataSource method*), 28

insert() (*decisionengine.framework.dataspace.datasources.postgresql.PostgreSQL method*), 24
insert() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 29
InvalidMetadataError, 26
is_expired() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 26
is_valid() (*decisionengine.framework.dataspace.datablock.Header method*), 26

## K

keys() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 26

## L

load() (*in module decisionengine.framework.config.tests.test_config*), 17
load_all_channels() (*decisionengine.framework.config.ChannelConfigHandler.ChannelConfigHandler method*), 19
load_channel() (*decisionengine.framework.config.ChannelConfigHandler.ChannelConfigHandler method*), 19
LogicEngine (*class in decisionengine.framework.logicengine.LogicEngine*), 37
LogicEngine (*class in decisionengine.framework.modules.LogicEngine*), 38
LogicError, 36

## M

main() (*in module decisionengine.framework.engine.de_client*), 34
main() (*in module decisionengine.framework.engine.DecisionEngine*), 33
main() (*in module decisionengine.framework.modules.SourceProxy*), 39
make_db() (*in module decisionengine.framework.logicengine.tests.test_facts*), 35
mark_demented() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 30
mark_expired() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 26
max_postgresql_port() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 30
Metadata (*class in decisionengine.framework.dataspace.datablock*), 26
metadata() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 21
metadata_table (*decisionengine.framework.dataspace.datasource.DataSource attribute*), 29
mock_data_block() (*in module decisionengine.framework.dataspace.datasources.tests.fixtures*), 21
module
decisionengine, 47
decisionengine.framework, 47
decisionengine.framework.about, 47
decisionengine.framework.config, 21
decisionengine.framework.config.ChannelConfigHa 18
decisionengine.framework.config.policies, 20
decisionengine.framework.config.tests, 18
decisionengine.framework.config.tests.test_conf 17
decisionengine.framework.config.tests.test_poli 18
decisionengine.framework.config.ValidConfig, 19
decisionengine.framework.dataspace, 31
decisionengine.framework.dataspace.datablock, 25
decisionengine.framework.dataspace.datasource, 27
decisionengine.framework.dataspace.datasources, 25
decisionengine.framework.dataspace.datasources. 22
decisionengine.framework.dataspace.datasources. 22
decisionengine.framework.dataspace.datasources. 21
decisionengine.framework.dataspace.datasources. 21
decisionengine.framework.dataspace.dataspace, 29
decisionengine.framework.engine, 34
decisionengine.framework.engine.de_client, 34
decisionengine.framework.engine.DecisionEngine,

31
decisionengine.framework.engine.Workers, 33
decisionengine.framework.logicengine, 38
decisionengine.framework.logicengine.BoolearExpression, 36
decisionengine.framework.logicengine.FactLookup, 36
decisionengine.framework.logicengine.LogicEngine, 37
decisionengine.framework.logicengine.Rule, 37
decisionengine.framework.logicengine.RuleEngine, 38
decisionengine.framework.logicengine.tests, 36
decisionengine.framework.logicengine.tests.test_cascaded_rules, 34
decisionengine.framework.logicengine.tests.test_engine_frame, 34
decisionengine.framework.logicengine.tests.test_engine_framework.test, 35
decisionengine.framework.logicengine.tests.test_facts, 35
decisionengine.framework.logicengine.tests.test_engine_frame, 35
decisionengine.framework.logicengine.tests.test_engine_with_negated_fact, 35
decisionengine.framework.logicengine.tests.test_simple_configuration, 35
decisionengine.framework.modules, 40
decisionengine.framework.modules.de_logger, 40
decisionengine.framework.modules.LogicEngine, 38
decisionengine.framework.modules.Module, 38
decisionengine.framework.modules.Publisher, 38
decisionengine.framework.modules.Source, 39
decisionengine.framework.modules.SourceProxy, 39
decisionengine.framework.modules.Transform, 40
decisionengine.framework.taskmanager, 43
decisionengine.framework.taskmanager.ProcessingState, 40
decisionengine.framework.taskmanager.TaskManager, 41
decisionengine.framework.tests, 46
decisionengine.framework.tests.FailingPublisher, 43
decisionengine.framework.tests.FailingSourcePro 43
decisionengine.framework.tests.fixtures, 44
decisionengine.framework.tests.PublisherNOP, 43
decisionengine.framework.tests.SourceNOP, 44
decisionengine.framework.tests.test_client_serv 45
decisionengine.framework.tests.test_defaults, 45
decisionengine.framework.tests.test_failing_sou 45
decisionengine.framework.tests.test_reaper, 45
decisionengine.framework.tests.test_restart_cha 45
decisionengine.framework.tests.test_sample_conf 46
decisionengine.framework.tests.test_start_with_ 46
decisionengine.framework.tests.TransformNOP, 44
decisionengine.framework.util, 47
decisionengine.framework.util.fs, 46
decisionengine.framework.util.sockets, 46
decisionengine.framework.util.tsort, 46
Module (*class in decisionengine.framework.modules.Module*), 38
module_config_info() (*in module decisionengine.framework.modules.SourceProxy*), 39
module_config_template() (*in module decisionengine.framework.modules.SourceProxy*), 39
must_have (*decisionengine.framework.modules.SourceProxy.SourceProx attribute*), 39
my_data() (*in module decisionengine.framework.logicengine.tests.test_pandas_fact*), 35
myengine() (*in module decisionengine.framework.logicengine.tests.test_cascaded_rules*), 34
myengine() (*in module decisionengine.framework.logicengine.tests.test_pandas_fact*), 35
myengine() (*in module decisionengine.framework.logicengine.tests.test_rule_with_negated_fact 35
myengine() (*in module decisionengine.framework.logicengine.tests.test_simple_configuration*),

35

## O

OFFLINE (*decisionengine.framework.taskmanager.ProcessingState.State attribute*), 41

## P

parse_program_options() (*in module decisionengine.framework.engine.DecisionEngine*), 33
PG_PROG() (*in module decisionengine.framework.tests.fixtures*), 44
post_create() (*decisionengine.framework.modules.Source.Source method*), 39
post_create() (*decisionengine.framework.modules.SourceProxy.SourceProxy method*), 39
Postgresql (*class in decisionengine.framework.dataspace.datasources.postgresql*), 22
print_channel_config() (*decisionengine.framework.config.ChannelConfigHandler.ChannelConfigHandler method*), 19
ProcessingState (*class in decisionengine.framework.taskmanager.ProcessingState*), 41
produces() (*decisionengine.framework.logicengine.LogicEngine.LogicEngine method*), 37
produces() (*decisionengine.framework.modules.Source.Source method*), 39
produces() (*decisionengine.framework.modules.SourceProxy.SourceProxy method*), 39
produces() (*decisionengine.framework.modules.Transform.Transform method*), 40
produces() (*decisionengine.framework.tests.FailingSourceProxy.FailingSourceProxy method*), 43
produces() (*decisionengine.framework.tests.SourceNOP.SourceNOP method*), 44
produces() (*decisionengine.framework.tests.TransformNOP.TransformNOP method*), 44
publish() (*decisionengine.framework.modules.Publisher.Publisher method*), 38
publish() (*decisionengine.framework.tests.FailingPublisher.FailingPublisher method*), 43
publish() (*decisionengine.framework.tests.PublisherNOP.PublisherNOP method*), 43
Publisher (*class in decisionengine.framework.modules.Publisher*), 38
PublisherNOP (*class in decisionengine.framework.tests.PublisherNOP*), 43
put() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 26

## R

reap() (*decisionengine.framework.dataspace.dataspace.Reaper method*), 30
Reaper (*class in decisionengine.framework.dataspace.dataspace*), 30
reaper_start() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 31
reaper_status() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 31
reaper_stop() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 31
RequestHandler (*class in decisionengine.framework.engine.DecisionEngine*), 32
required_keys (*decisionengine.framework.dataspace.datablock.Header attribute*), 26
required_keys (*decisionengine.framework.dataspace.datablock.Metadata attribute*), 26
rpc_block_while() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
rpc_get_channel_log_level() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
rpc_get_log_level() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
rpc_paths (*decisionengine.framework.engine.DecisionEngine.RequestH attribute*), 32
rpc_print_product() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
rpc_print_products() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
rpc_reaper_start() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
rpc_reaper_status() (*decisio-*

*nengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
rpc_reaper_stop() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
rpc_set_channel_log_level() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
rpc_show_config() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
rpc_show_de_config() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
rpc_start_channel() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
rpc_start_channels() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
rpc_status() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
rpc_stop() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
rpc_stop_channel() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
rpc_stop_channels() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
Rule (*class in decisionengine.framework.logicengine.Rule*), 37
rule_for() (*decisionengine.framework.logicengine.FactLookup.FactLookup method*), 36
RuleEngine (*class in decisionengine.framework.logicengine.RuleEngine*), 38
run() (*decisionengine.framework.engine.Workers.Worker method*), 33
run() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 42
run_logic_engine() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 42
run_publishers() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 42
run_source() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 42
run_transform() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 42
run_transforms() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 42
RUNNING (*decisionengine.framework.dataspace.dataspace.State attribute*), 31

## S

see() (*decisionengine.framework.taskmanager.ProcessingState.ProcessingState method*), 41
set_data_bock() (*decisionengine.framework.modules.Module.Module method*), 38
set_logging() (*in module decisionengine.framework.modules.de_logger*), 40
set_loglevel() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 42
set_retention_interval() (*decisionengine.framework.dataspace.dataspace.Reaper method*), 30
set_state() (*decisionengine.framework.dataspace.datablock.Metadata method*), 26
set_stream_logging() (*in module decisionengine.framework.modules.de_logger*), 40
should_stop() (*decisionengine.framework.taskmanager.ProcessingState.ProcessingState method*), 41
SHUTDOWN (*decisionengine.framework.taskmanager.ProcessingState.State attribute*), 41
shutdown() (*decisionengine.framework.modules.Publisher.Publisher method*), 38
SHUTTINGDOWN (*decisionengine.framework.taskmanager.ProcessingState.State attribute*), 41
Singleton (*class in decisionengine.framework.dataspace.dataspace*), 30
SLEEPING (*decisionengine.framework.dataspace.dataspace.State attribute*), 31
sorted_rules() (*decisionengine.framework.logicengine.FactLookup.FactLookup method*), 36
Source (*class in decisionengine.framework.modules.Source*), 39
SourceNOP (*class in decisionengine.framework.tests.SourceNOP*), 44
SourceProxy (*class in decisionengine.framework.modules.SourceProxy*), 39

start() (*decisionengine.framework.dataspace.dataspace.Reaper method*), 30
start_channel() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
start_channels() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
start_sources() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 42
STARTING (*decisionengine.framework.dataspace.dataspace.State attribute*), 31
State (*class in decisionengine.framework.dataspace.dataspace*), 30
State (*class in decisionengine.framework.taskmanager.ProcessingState*), 41
STEADY (*decisionengine.framework.taskmanager.ProcessingState.State attribute*), 41
stop() (*decisionengine.framework.dataspace.dataspace.Reaper method*), 30
stop_channel() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
stop_channels() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
stop_worker() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 32
STOPPED (*decisionengine.framework.dataspace.dataspace.State attribute*), 31
STOPPING (*decisionengine.framework.dataspace.dataspace.State attribute*), 31
store_taskmanager() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 26
store_taskmanager() (*decisionengine.framework.dataspace.datasource.DataSource method*), 29
store_taskmanager() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 24
store_taskmanager() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 30

## T

tables (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql attribute*), 24
take_offline() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 42
TaskManager (*class in decisionengine.framework.taskmanager.TaskManager*), 41
taskmanager() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 21
taskmanager_table (*decisionengine.framework.dataspace.datasource.DataSource attribute*), 29
test_channel_config_dir() (*in module decisionengine.framework.config.tests.test_policies*), 18
test_channel_empty_config() (*in module decisionengine.framework.config.tests.test_config*), 17
test_channel_empty_dictionary() (*in module decisionengine.framework.config.tests.test_config*), 17
test_channel_loading() (*in module decisionengine.framework.config.tests.test_config*), 17
test_channel_names() (*in module decisionengine.framework.config.tests.test_config*), 17
test_channel_no_config_files() (*in module decisionengine.framework.config.tests.test_config*), 17
test_channel_no_modules() (*in module decisionengine.framework.config.tests.test_config*), 17
test_client_can_get_de_server_reaper_start_delay() (*in module decisionengine.framework.tests.test_reaper*), 45
test_client_can_get_de_server_reaper_status() (*in module decisionengine.framework.tests.test_reaper*), 45
test_client_can_get_de_server_reaper_stop() (*in module decisionengine.framework.tests.test_reaper*), 45
test_client_can_get_de_server_show_channel_logger_l (*in module decisionengine.framework.tests.test_defaults*), 45
test_client_can_get_de_server_show_config() (*in module decisionengine.framework.tests.test_sample_config*), 46
test_client_can_get_de_server_show_logger_level() (*in module decisionengine.framework.tests.test_sample_config*), 46
test_client_can_get_de_server_status() (*in module decisionengine.framework.tests.test_sample_config*),

46
test_client_de_config_is_json() (in module decisionengine.framework.tests.test_defaults), 45
test_client_print_product() (in module decisionengine.framework.tests.test_client_server), 45
test_compound_fact() (in module decisionengine.framework.logicengine.tests.test_facts), 35
test_configuration_with_fact_using_function() (in module decisionengine.framework.logicengine.tests.test_construction), 34
test_configuration_with_numy_facts() (in module decisionengine.framework.logicengine.tests.test_construction), 34
test_create_tables() (in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 21
test_default_construction() (in module decisionengine.framework.logicengine.tests.test_construction), 34
test_duplicate_fact_names() (in module decisionengine.framework.logicengine.tests.test_duplicate_fact_names), 35
test_empty_config() (in module decisionengine.framework.config.tests.test_config), 17
test_empty_dict() (in module decisionengine.framework.config.tests.test_config), 18
test_empty_dict_with_leading_comment() (in module decisionengine.framework.config.tests.test_config), 18
test_fact_using_numpy_array() (in module decisionengine.framework.logicengine.tests.test_facts), 35
test_fact_using_numpy_function() (in module decisionengine.framework.logicengine.tests.test_facts), 35
test_fact_with_nested_names() (in module decisionengine.framework.logicengine.tests.test_facts), 35
test_generate_insert_query() (in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 21
test_get_last_generation_id() (in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 21
test_get_taskmanager() (in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 21
test_global_channel_log_level_in_config() (in module decisionengine.framework.tests.test_defaults), 45
test_global_config_dir() (in module decisionengine.framework.config.tests.test_policies), 18
test_global_config_file() (in module decisionengine.framework.config.tests.test_policies), 18
test_insert() (in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 21
test_minimal_jsonnet_right_extension() (in module decisionengine.framework.config.tests.test_config), 18
test_minimal_jsonnet_wrong_extension() (in module decisionengine.framework.config.tests.test_config), 18
test_minimal_python() (in module decisionengine.framework.config.tests.test_config), 18
test_restart_channel() (in module decisionengine.framework.tests.test_restart_channel), 45
test_rule_that_does_not_fire() (in module decisionengine.framework.logicengine.tests.test_cascaded_rules), 34
test_rule_that_does_not_fire() (in module decisionengine.framework.logicengine.tests.test_pandas_fact), 35
test_rule_that_does_not_fire() (in module decisionengine.framework.logicengine.tests.test_rule_with_negated_fact), 35
test_rule_that_does_not_fire() (in module decisionengine.framework.logicengine.tests.test_simple_configuration), 35
test_rule_that_fires() (in module decisionengine.framework.logicengine.tests.test_cascaded_rules), 34
test_rule_that_fires() (in module decisio-

*nengine.framework.logicengine.tests.test_pandas_facts*), 35
test_rule_that_fires() (*in module decisionengine.framework.logicengine.tests.test_rule_with_negated_fact*), 35
test_rule_that_fires() (*in module decisionengine.framework.logicengine.tests.test_simple_configuration*), 35
test_simple_fact() (*in module decisionengine.framework.logicengine.tests.test_facts*), 35
test_start_from_nothing() (*in module decisionengine.framework.tests.test_start_with_no_channels*), 46
test_stop_failing_source_proxy() (*in module decisionengine.framework.tests.test_failing_source_proxy*), 45
test_store_taskmanager() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 22
test_trivial_configuration() (*in module decisionengine.framework.logicengine.tests.test_construction*), 34
test_valid_dir() (*in module decisionengine.framework.config.tests.test_policies*), 18
test_wrong_configuration() (*in module decisionengine.framework.logicengine.tests.test_construction*), 34
test_wrong_type() (*in module decisionengine.framework.config.tests.test_config*), 18
Transform (*class in decisionengine.framework.modules.Transform*), 40
transform() (*decisionengine.framework.modules.Transform.Transform method*), 40
transform() (*decisionengine.framework.tests.TransformNOP.TransformNOP method*), 44
TransformNOP (*class in decisionengine.framework.tests.TransformNOP*), 44
tsort() (*in module decisionengine.framework.util.tsort*), 47

## U

unguarded_access() (*decisionengine.framework.engine.Workers.Workers method*), 33
update() (*decisionengine.framework.dataspace.datasource.DataSource method*), 29
update() (*decisionengine.framework.dataspace.datasources.postgresql.P method*), 24
update() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 30

## V

valid_dir() (*in module decisionengine.framework.config.policies*), 20
valid_states (*decisionengine.framework.dataspace.datablock.Metadata attribute*), 26
ValidConfig (*class in decisionengine.framework.config.ValidConfig*), 19

## W

wait_for_all() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 42
wait_for_any() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 42
wait_until() (*decisionengine.framework.engine.Workers.Worker method*), 33
wait_until() (*decisionengine.framework.taskmanager.ProcessingState.ProcessingState method*), 41
wait_while() (*decisionengine.framework.engine.Workers.Worker method*), 33
wait_while() (*decisionengine.framework.taskmanager.ProcessingState.ProcessingState method*), 41
Worker (*class in decisionengine.framework.engine.Workers*), 33
Worker (*class in decisionengine.framework.taskmanager.TaskManager*), 43
Workers (*class in decisionengine.framework.engine.Workers*), 33
Workers.Access (*class in decisionengine.framework.engine.Workers*), 33

## Z

zdumps() (*in module decisionengine.framework.dataspace.datablock*), 27
zloads() (*in module decisionengine.framework.dataspace.datablock*), 27