decisionengine

Release 1.1.1

Fermi Research Alliance, LLC.

Sep 10, 2020

CONTENTS

1 Release Notes 3

2 Developer Documentation 9

3 Source code 13

4 Indices and tables 35

Python Module Index 37

Index 39

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 1.3.0

In this release:

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

1.1.1 Full list of commits since version 1.2

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

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

afd8837 : Configuration-manager fixes (#128)

571e2be : Remove pip installed system python packages

407d9ed : Update Dockerfile

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

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

2840813 : Switch to Jsonnet configuration system (#125)

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

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

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

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

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

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

c3d1be3 : Python 3 upgrades. (#116)

e7399af : Header fix (#114)

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

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

97c21b1 : osg version 3.5 (#112)

33f28a8 : Introduce jsonnet dependency (#110)

3f8b55e : improve server error handling (#108)

f15588e : added 1.2.0 release notes

b433325 : Remove unnecessary 'main' functionality. (#107)

1.2 Release 1.2.0

In this release:

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

1.2.1 decisionengine

3dfe167 : Jenkins pipeline improvements (#106)

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

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)

1.2.2 decisionengine_modules

30d928b : clone version 1.2.0 of decisionengine

ae7c5a6 : Jenkins pipeline improvements (#236)

310befd : T198 (#235)

a65886d : Fix import as reported in : https://github.com/HEPCloud/decisionengin... (#232)

93711cc : Run coveralls even if tests fail (#229)

03d763a : Jenkins pipeline improvements (#230)

f48d30f : Fix/223 (#228)

c8aa262 : github ticket 199 (#222)

0323bda : Address : https://github.com/HEPCloud/decisionengine_modules/issues/224 (#226)

62e4df6 : Add support to run CI on Jenkins (#221)

5ab1541 : bump master version to 1.2.0 (for now) (#219)

bc19c65 : decisionengine_modules/NERSC: Added retry loop for NERSC API Calls (#220)

41a50de : Sync up pep8speaks and run_pylint.sh with decisionengine settings (#218)

db4634f : silence pylint error (#217)

1b95141 : Fix whitespace around operator error

746ea38 : ignore W503

8a8b5f4 : remove unused variable

a6668bf : fix PEP8 warnings

13773ee : address pep8 warnings

6bea4ca : silence pylint error

f589895 : Pass sort=True parameter to fix future warning (#215)

a1d0507 : fixing pep8 warning

a10bd17 : debugging one import error

ec501ad : make coveralls.io links work

deab1a7 : T201 (#204)

69f2645 : Add coveragerc

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

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

1.3 Release 1.1.0

In this release:

- Fixed. https://github.com/HEPCloud/decisionengine_modules/issues/108 "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 cmake3 boost-devel python36-devel boost-python36-
↪devel postgresql11 postgresql11-server
pip3 install pandas DBUtils psycopg2-binary tabulate mock pytest
```

2.1.2 Build & test

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

export PYTHONPATH=`pwd`

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

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

(continues on next page)

(continued from previous page)

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

================================== 26 passed in 23.86s␣
↪==================================
```

2.2 Decisionengine_modules

2.2.1 Prerequisites:

In Addition to above installed packages

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

2.2.2 Test

```
cd

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

Current status:

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

decisionengine_modules/AWS/tests/test_AWSInstancePerformance.py ..               ␣
↪ [  2%]
decisionengine_modules/AWS/tests/test_AWSJobLimits.py ..                         ␣
↪ [  4%]
decisionengine_modules/AWS/tests/test_AWSOccupancyWithSourceProxy.py ..          ␣
↪ [  7%]
```

(continues on next page)

(continued from previous page)

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

(continues on next page)

(continued from previous page)

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

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

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

CHAPTER

THREE

SOURCE CODE

3.1 Welcome to decisionengine's documentation!

3.1.1 decisionengine package

Subpackages

decisionengine.framework package

Subpackages

decisionengine.framework.configmanager package

Subpackages

decisionengine.framework.configmanager.tests package

Submodules

decisionengine.framework.configmanager.tests.test_configmanager module

decisionengine.framework.configmanager.tests.test_configmanager.**load**(*monkeypatch*)

decisionengine.framework.configmanager.tests.test_configmanager.**test_channel_empty_config**(*l*
c
s
c

decisionengine.framework.configmanager.tests.test_configmanager.**test_channel_empty_dictiona**

decisionengine.framework.configmanager.tests.test_configmanager.**test_channel_names**(*load*)

decisionengine.framework.configmanager.tests.test_configmanager.**test_channel_no_config_file**

decisionengine.framework.configmanager.tests.test_configmanager.**test_channel_no_modules**(*load*

decisionengine.framework.configmanager.tests.test_configmanager.**test_empty_config**(*load*)

decisionengine.framework.configmanager.tests.test_configmanager.**test_empty_dict**(*load*)

decisionengine.framework.configmanager.tests.test_configmanager.**test_empty_dict_with_leadi**

decisionengine.framework.configmanager.tests.test_configmanager.**test_minimal_jsonnet_right_**

decisionengine.framework.configmanager.tests.test_configmanager.**test_minimal_jsonnet_wrong_**

decisionengine.framework.configmanager.tests.test_configmanager.**test_minimal_python**(*load*, *capsys*)

decisionengine.framework.configmanager.tests.test_configmanager.**test_minimal_python_default**

decisionengine.framework.configmanager.tests.test_configmanager.**test_program_options_defaul**

decisionengine.framework.configmanager.tests.test_configmanager.**test_program_options_update**

decisionengine.framework.configmanager.tests.test_configmanager.**test_wrong_type**(*load*)

Module contents

Submodules

decisionengine.framework.configmanager.ConfigManager module

class decisionengine.framework.configmanager.ConfigManager.**ConfigManager**(*program_options=None*)
Bases: object

_load_channels()

static create(*module_name*, *class_name*, *parameters*)
Factory method: create instance of dynamically loaded module

get_channels()

get_global_config()

get_produces(*channel_config*)

is_updated()

load(*config_file_name=None*)

reload(*config_file_name=None*)

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

decisionengine.framework.configmanager.ConfigManager.**_config_from_file**(*config_file*)

decisionengine.framework.configmanager.ConfigManager.**_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.configmanager.ConfigManager.**_make_logger**(*global_config*)

decisionengine.framework.configmanager.ConfigManager.**_validate_channel**(*channel*)
Validate channels :type channel: dict

Module contents

decisionengine.framework.dataspace package

Subpackages

decisionengine.framework.dataspace.datasources package

Subpackages

decisionengine.framework.dataspace.datasources.tests package

Submodules

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

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

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

Module contents

Submodules

decisionengine.framework.dataspace.datasources.postgresql module

class decisionengine.framework.dataspace.datasources.postgresql.**Postgresql**(*config_dict*)

Bases: *decisionengine.framework.dataspace.datasource.DataSource*

Implementation of postgresql data source

__query(*query_string*, *values=None*, *cursor_factory=None*)

_abc_cache = <_weakrefset.WeakSet object>

_abc_negative_cache = <_weakrefset.WeakSet object>

_abc_negative_cache_version = 185

_abc_registry = <_weakrefset.WeakSet object>

_delete(*sql_query*, *values=None*)

_insert(*table_name_or_sql_query*, *record=None*)

_insert_returning_result(*table_name_or_sql_query*, *record=None*)

_remove(*sql_query*, *values=None*)

_select(*query_string*, *values=None*, *cursor_factory=None*)

_select_dictresult(*sql_query*, *values=None*)

_select_getresult(*sql_query*, *values=None*)

_select_tuple(*sql_query*, *values*)

_update(*query_string*, *values=None*)

_update_returning_result(*query_string*, *values=None*)

close()
Close all connections to the database

connect()
Create a pool of database connections

create_tables()
Create database tables

delete_data_older_than(*days*)
Delete data older 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_header(*taskmanager_id*, *generation_id*, *key*)
Return the header from the header table for the given taskmanager_id, generation_id, key

Parameters

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

get_last_generation_id(*taskmanager_name*, *taskmanager_id=None*)
Return last generation id for current task manager or taskmanager w/ task_manager_id.

Parameters

- **name** (string) – task manager name
- **taskmanager_id** (string) – task manager id

get_metadata(*taskmanager_id*, *generation_id*, *key*)
Return the metadata from the metadata table for the given taskmanager_id, generation_id, key

Parameters

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

get_schema(*table=None*)
Given the table name return it's schema

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

get_taskmanager(*taskmanager_name*, *taskmanager_id=None*)
Retrieve TaskManager :type taskmanager_name: string :arg taskmanager_name: name of taskmanager to retrieve :type taskmanager_id: string :arg taskmanager_id: id of taskmanager to retrieve

insert(*taskmanager_id*, *generation_id*, *key*, *value*, *header*, *metadata*)
Insert data into respective tables for the given taskmanager_id, generation_id, key

Parameters

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

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

store_taskmanager(*name*, *taskmanager_id*)
Store TaskManager :type taskmanager_name: string :arg taskmanager_name: name of taskmanager to retrieve :type taskmanager_id: string :arg taskmanager_id: id of taskmanager to retrieve

tables = {'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_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_cache = <_weakrefset.WeakSet object>

_abc_negative_cache = <_weakrefset.WeakSet object>

_abc_negative_cache_version = 185

_abc_registry = <_weakrefset.WeakSet object>

default_data_lifetime = 1800

is_valid()

Check if the Header has minimum required information

required_keys = {'create_time', 'creator', 'expiration_time', 'scheduled_create_time',

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

Bases: Exception

Errors due to invalid Metadata

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

Bases: collections.UserDict

_abc_cache = <_weakrefset.WeakSet object>

_abc_negative_cache = <_weakrefset.WeakSet object>

_abc_negative_cache_version = 185

_abc_registry = <_weakrefset.WeakSet object>

required_keys = {'generation_id', 'generation_time', 'missed_update_count', 'state', '

set_state(*state*)

Set the state for the Metadata

valid_states = {'END_CYCLE', 'METADATA_UPDATE', 'NEW', 'START_BACKUP'}

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

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

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

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

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

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

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

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

Parameters **zbytes** – compressed bytes

Returns returns python object

decisionengine.framework.dataspace.datasource module

class decisionengine.framework.dataspace.datasource.**DataSource**(*config*)
Bases: object

_abc_cache = <_weakrefset.WeakSet object>

_abc_negative_cache = <_weakrefset.WeakSet object>

_abc_negative_cache_version = 185

_abc_registry = <_weakrefset.WeakSet object>

abstract close()
Close all connections to the database

abstract connect()
Create a pool of database connections

abstract create_tables()
Create database tables

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_header(*taskmanager_id*, *generation_id*, *key*)
Return the header from the header table for the given taskmanager_id, generation_id, key

Parameters

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

abstract get_last_generation_id(*name*, *taskmanager_id=None*)
Return last generation id for current task manager or taskmanager w/ task_manager_id.

Parameters

- **name** (string) – task manager name
- **taskmanager_id** (string) – task manager id

abstract get_metadata(*taskmanager_id*, *generation_id*, *key*)
Return the metadata from the metadata table for the given taskmanager_id, generation_id, key

Parameters

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

abstract get_schema(*table=None*)
Given the table name return it's schema

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

abstract get_taskmanager(*taskmanager_name*, *taskmanager_id*)
Retrieve TaskManager :type taskmanager_name: string :arg taskmanager_name: name of taskmanager to retrieve :type taskmanager_id: string :arg taskmanager_id: id of taskmanager to retrieve

header_table = 'header'
Name of the header table

abstract insert(*taskmanager_id*, *generation_id*, *key*, *value*, *header*, *metadata*)
Insert data into respective tables for the given taskmanager_id, generation_id, key

Parameters

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

metadata_table = 'metadata'
Name of the metadata table

abstract store_taskmanager(*taskmanager_name*, *taskmanager_id*)
Store TaskManager :type taskmanager_name: string :arg taskmanager_name: name of taskmanager to retrieve :type taskmanager_id: string :arg taskmanager_id: id of taskmanager to retrieve

taskmanager_table = 'taskmanager'
Name of the taskmanager table

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

Parameters

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

decisionengine.framework.dataspace.dataspace module

class decisionengine.framework.dataspace.dataspace.**DataSourceLoader**(**args*, ***kwargs*)
Bases: object

_ds = None

static create_datasource(*module_name*, *class_name*, *config*)

class decisionengine.framework.dataspace.dataspace.**DataSpace**(*config*)
Bases: object

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

_tables_created = False
Description of tables and their columns

close()

delete(*taskmanager_id*, *all_generations=False*)

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

get_dataproduct(*taskmanager_id*, *generation_id*, *key*)

get_header(*taskmanager_id*, *generation_id*, *key*)

get_last_generation_id(*taskmanager_name*, *taskmanager_id=None*)

get_metadata(*taskmanager_id*, *generation_id*, *key*)

get_taskmanager(*taskmanager_name*, *taskmanager_id=None*)

insert(*taskmanager_id*, *generation_id*, *key*, *value*, *header*, *metadata*)

mark_demented(*taskmanager_id*, *keys*, *generation_id=None*)

mark_expired(*taskmanager_id*, *generation_id*, *key*, *expiry_time*)

store_taskmanager(*name*, *id*)

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

exception decisionengine.framework.dataspace.dataspace.**DataSpaceConfigurationError**
Bases: Exception

Errors related to database access

exception decisionengine.framework.dataspace.dataspace.**DataSpaceConnectionError**
Bases: Exception

Errors related to database access

exception decisionengine.framework.dataspace.dataspace.**DataSpaceError**
Bases: Exception

Errors related to database access

exception decisionengine.framework.dataspace.dataspace.**DataSpaceExistsError**
Bases: Exception

Errors related to database access

class decisionengine.framework.dataspace.dataspace.**Reaper**(*config*)
Bases: object

Reaper provides functionality of periodic deletion of data older than retention_interval in days

_reaper_loop(*delay*)

_set_state(*value*)

get_retention_interval()

get_state()

reap()

set_retention_interval(*interval*)

start(*delay=0*)
Start thread with an optional delay to start the thread in X seconds

stop()

class decisionengine.framework.dataspace.dataspace.**Singleton**
Bases: type

Singleton pattern using Metaclass http://stackoverflow.com/questions/6760685/creating-a-singleton-in-python

_instances = {}

class decisionengine.framework.dataspace.dataspace.**State**(*value*)
Bases: enum.Enum

An enumeration.

ERROR = 7

IDLE = 1

RUNNING = 3

SLEEPING = 4

STARTING = 2

STOPPED = 6

STOPPING = 5

Module contents

decisionengine.framework.engine package

Submodules

decisionengine.framework.engine.DecisionEngine module

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

class decisionengine.framework.engine.DecisionEngine.**DecisionEngine**(*cfg*, *server_address*)

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

_disable_channels_with_terminated_processes()

_dispatch(*method*, *params*)

Dispatches the XML-RPC method.

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

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

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

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

get_logger()

handle_sighup(*signum*, *frame*)

reaper_start(*delay*)

reaper_status()

reaper_stop()

reload_config()

rpc_get_channel_log_level(*channel*)

rpc_get_log_level()

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

rpc_print_products()

rpc_reaper_start(*delay=0*)

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

rpc_reaper_status()

rpc_reaper_stop()

rpc_reload_config()

rpc_set_channel_log_level(*channel*, *log_level*)

Assumes log_level is a string corresponding to the supported logging-module levels.

rpc_show_config(*channel*)

rpc_show_de_config()

rpc_start_channel(*channel*)

rpc_start_channels()

rpc_status()

rpc_stop()

rpc_stop_channel(*channel*)

rpc_stop_channels()

service_actions()
Called by the serve_forever() loop.

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

start_channel(*channel*)

start_channels()

stop_channel(*channel*)

stop_channels()

class decisionengine.framework.engine.DecisionEngine.**RequestHandler**(*request*, *client_address*, *server*)
Bases: xmlrpc.server.SimpleXMLRPCRequestHandler

rpc_paths = ('/RPC2',)

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

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

decisionengine.framework.engine.DecisionEngine.**_get_de_conf_manager**(*args=None*)
If args is a list, it will be used instead of sys.argv If args is not an instance of 'list', sys.argv will be parsed

decisionengine.framework.engine.DecisionEngine.**_start_de_server**(*conf_manager*)
start the DE server with the passed config manager

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

decisionengine.framework.engine.DecisionEngine.**parse_program_options**(*args*)

decisionengine.framework.engine.de_client module

decisionengine.framework.engine.de_client.**build_xmlrpc_connection**(*host*, *port*)

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

decisionengine.framework.engine.de_client.**execute_command_from_args**(*argsparsed*, *xmlrpc-socket*)

argsparsed should be from create_parser in this file

decisionengine.framework.engine.de_client.**main**(*args_to_parse=None*)

If you pass a list of args, they will be used instead of sys.argv

Module contents

decisionengine.framework.modules package

Submodules

decisionengine.framework.modules.LogicEngine module

class decisionengine.framework.modules.LogicEngine.**LogicEngine**(*set_of_parameters*)

Bases: *decisionengine.framework.modules.Module.Module*

evaluate(*data_block*)

decisionengine.framework.modules.Module module

class decisionengine.framework.modules.Module.**Module**(*set_of_parameters*)

Bases: object

get_data_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*)

decisionengine.framework.modules.Source module

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

acquire()

produces(*name_schema_id_list*)

decisionengine.framework.modules.SourceProxy module

Fill in data from another channel data block

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

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

"AWSJobLimits" : { "module" : "modules.source_proxy", "name" : "SourceProxy", "parameters": {"channel_name": "channel_aws_config_data",

"Dataproducts":[("aws_instance_limits", "Job_Limits")], "retries": 3, "retry_timeout": 20,

},

"schedule": 360,

},

_get_data(*data_block*, *key*)

acquire()
Overrides Source class method

consumes()

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

must_have = ('channel_name', 'Dataproducts')

produces()

Assumes that self.datakeys has the following structure or

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

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

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

decisionengine.framework.modules.Transform module

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

consumes(*name_list*)

produces(*name_schema_id_list*)

transform()

decisionengine.framework.modules.de_logger module

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_file_name='/tmp/decision_engine_logs/decision_*
max_file_size=200000000,
max_backup_count=6,
log_level='DEBUG')

Parameters

- **log_file_name** (str) – log file name
- **max_file_size** (int) – maximal size of log file. If reached save and start new log.
- **max_backup_count** (int) – start rotaion after this number is reached

Return type logging.Logger - rotating file logger

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

Parameters **logger_name** (str) – logger name

Return type logging.Logger

Module contents

decisionengine.framework.taskmanager package

Submodules

decisionengine.framework.taskmanager.TaskManager module

Task Manager

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

Decision Channel. Instantiates workers according to channel configuration

class decisionengine.framework.taskmanager.TaskManager.**State**(*value*)
Bases: enum.Enum

An enumeration.

BOOT = 0

OFFLINE = 2

SHUTDOWN = 4

SHUTTINGDOWN = 3

STEADY = 1

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

Bases: object

Task Manager

_take_offline(*current_data_block*)
offline and stop task manager

data_block_put(*data*, *header*, *data_block*)
Put data into data block

Parameters

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

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

do_backup()
Duplicate current data block and return its copy

Return type DataBlock

get_loglevel()

get_state()

get_state_name()

run()
Task Manager main loop

run_logic_engine(*data_block=None*)
Run Logic Engine.

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

run_publishers(*actions*, *facts*, *data_block=None*)
Run Publishers in main process.

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

run_source(*src*)
Get the data from source and put it into the data block

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

run_transform(*transform*, *data_block*)
Run a transform

Parameters

- **transform** (*Worker*) – source Worker
- **data_block** (DataBlock) – data block

run_transforms(*data_block=None*)
Run transforms. So far in main process.

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

set_loglevel(*log_level*)
Assumes log_level is a string corresponding to the supported logging-module levels.

set_state(*state*)

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

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

wait_for_all(*events_done*)
Wait for all sources or transforms to finish

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

wait_for_any(*events_done*)
Wait for any sources to finish

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

class decisionengine.framework.taskmanager.TaskManager.**Worker**(*conf_dict*)
Bases: object

Provides interface to loadable modules an events to sycronise execution

decisionengine.framework.taskmanager.TaskManager.**_make_workers_for**(*configs*)

Module contents

decisionengine.framework.tests package

Submodules

decisionengine.framework.tests.PublisherNOP module

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

consumes(*name_list=None*)

publish(*data_block=None*)

decisionengine.framework.tests.SourceNOP module

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

acquire()

produces()

decisionengine.framework.tests.TransformNOP module

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

consumes(*name_list=None*)

produces(*name_schema_id_list=None*)

transform(*data_block*)

decisionengine.framework.tests.test_channel module

class decisionengine.framework.tests.test_channel.**TestChannel**(*methodName='runTest'*)
Bases: unittest.case.TestCase

de_client_request(**args*)

pytestmark = [Mark(name='usefixtures', args=('fixtures',), kwargs={})]

setUp()
Hook method for setting up the test fixture before exercising it.

tearDown()
Hook method for deconstructing the test fixture after testing it.

test_client_can_get_de_server_reaper_start_delay()

test_client_can_get_de_server_reaper_status()

test_client_can_get_de_server_reaper_stop()

test_client_can_get_de_server_reload_config()

test_client_can_get_de_server_show_channel_logger_level()

test_client_can_get_de_server_show_config()

test_client_can_get_de_server_show_logger_level()

test_client_can_get_de_server_status()

test_global_channel_log_level_in_config()

class decisionengine.framework.tests.test_channel.**Worker**(*db_parameters*, *port*)
Bases: multiprocessing.context.Process

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

decisionengine.framework.tests.test_channel.**datasource**(*request*, *postgresql*)

decisionengine.framework.tests.test_channel.**fixtures**(*request*, *datasource*)

Module contents

decisionengine.framework.util package

Submodules

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

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, 33
decisionengine.framework, 33
decisionengine.framework.configmanager, 15
decisionengine.framework.configmanager.ConfigManager, 14
decisionengine.framework.configmanager.tests, 14
decisionengine.framework.configmanager.tests.test_configmanager, 13
decisionengine.framework.dataspace, 25
decisionengine.framework.dataspace.datablock, 18
decisionengine.framework.dataspace.datasource, 21
decisionengine.framework.dataspace.datasources, 18
decisionengine.framework.dataspace.datasources.postgresql, 16
decisionengine.framework.dataspace.datasources.tests, 16
decisionengine.framework.dataspace.datasources.tests.test_postgresql, 15
decisionengine.framework.dataspace.dataspace, 23
decisionengine.framework.engine, 27
decisionengine.framework.engine.de_client, 27
decisionengine.framework.engine.DecisionEngine, 25
decisionengine.framework.modules, 29
decisionengine.framework.modules.de_logger, 29
decisionengine.framework.modules.LogicEngine, 27
decisionengine.framework.modules.Module, 27
decisionengine.framework.modules.Publisher, 27
decisionengine.framework.modules.Source, 28
decisionengine.framework.modules.SourceProxy, 28
decisionengine.framework.modules.Transform, 29
decisionengine.framework.taskmanager, 31
decisionengine.framework.taskmanager.TaskManager, 29
decisionengine.framework.tests, 33
decisionengine.framework.tests.PublisherNOP, 31
decisionengine.framework.tests.SourceNOP, 32
decisionengine.framework.tests.test_channel, 32
decisionengine.framework.tests.TransformNOP, 32
decisionengine.framework.util, 33
decisionengine.framework.util.sockets, 33
decisionengine.framework.util.tsort, 33

INDEX

Symbols

__query() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 16
_abc_cache (*decisionengine.framework.dataspace.datablock.Header attribute*), 20
_abc_cache (*decisionengine.framework.dataspace.datablock.Metadata attribute*), 20
_abc_cache (*decisionengine.framework.dataspace.datasource.DataSource attribute*), 21
_abc_cache (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql attribute*), 16
_abc_negative_cache (*decisionengine.framework.dataspace.datablock.Header attribute*), 20
_abc_negative_cache (*decisionengine.framework.dataspace.datablock.Metadata attribute*), 20
_abc_negative_cache (*decisionengine.framework.dataspace.datasource.DataSource attribute*), 21
_abc_negative_cache (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql attribute*), 16
_abc_negative_cache_version (*decisionengine.framework.dataspace.datablock.Header attribute*), 20
_abc_negative_cache_version (*decisionengine.framework.dataspace.datablock.Metadata attribute*), 20
_abc_negative_cache_version (*decisionengine.framework.dataspace.datasource.DataSource attribute*), 21
_abc_negative_cache_version (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql attribute*), 16
_abc_registry (*decisionengine.framework.dataspace.datablock.Header attribute*), 20
_abc_registry (*decisionengine.framework.dataspace.datablock.Metadata attribute*), 20
_abc_registry (*decisionengine.framework.dataspace.datasource.DataSource attribute*), 21
_abc_registry (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql attribute*), 16
_check_keys() (*in module decisionengine.framework.configmanager.ConfigManager*), 14
_config_from_file() (*in module decisionengine.framework.configmanager.ConfigManager*), 14
_convert_to_json() (*in module decisionengine.framework.configmanager.ConfigManager*), 14
_delete() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 16
_disable_channels_with_terminated_processes() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 25
_dispatch() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 25
_ds (*decisionengine.framework.dataspace.dataspace.DataSourceLoader attribute*), 23
_get_data() (*decisionengine.framework.modules.SourceProxy.SourceProxy method*), 28
_get_de_conf_manager() (*in module decisionengine.framework.engine.DecisionEngine*), 26
_insert() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 18
_insert() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 16
_insert_returning_result() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 16
_instances (*decisio-*

nengine.framework.dataspace.dataspace.Singleton attribute), 24
_load_channels() (*decisionengine.framework.configmanager.ConfigManager.ConfigManager method*), 14
_make_logger() (*in module decisionengine.framework.configmanager.ConfigManager*), 14
_make_workers_for() (*in module decisionengine.framework.taskmanager.TaskManager*), 31
_reaper_loop() (*decisionengine.framework.dataspace.dataspace.Reaper method*), 24
_remove() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 16
_select() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 16
_select_dictresult() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 16
_select_getresult() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 16
_select_tuple() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 16
_set_state() (*decisionengine.framework.dataspace.dataspace.Reaper method*), 24
_setitem() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 18
_start_de_server() (*in module decisionengine.framework.engine.DecisionEngine*), 26
_tables_created (*decisionengine.framework.dataspace.dataspace.DataSpace attribute*), 23
_take_offline() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 30
_update() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 19
_update() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 16
_update_returning_result() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 16
_validate_channel() (*in module decisionengine.framework.configmanager.ConfigManager*), 14

A

acquire() (*decisionengine.framework.modules.Source.Source method*), 28
acquire() (*decisionengine.framework.modules.SourceProxy.SourceProxy method*), 28
acquire() (*decisionengine.framework.tests.SourceNOP.SourceNOP method*), 32

B

BOOT (*decisionengine.framework.taskmanager.TaskManager.State attribute*), 29
build_xmlrpc_connection() (*in module decisionengine.framework.engine.de_client*), 27

C

Channel (*class in decisionengine.framework.taskmanager.TaskManager*), 29
close() (*decisionengine.framework.dataspace.datasource.DataSource method*), 21
close() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 16
close() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 23
compress() (*in module decisionengine.framework.dataspace.datablock*), 20
ConfigManager (*class in decisionengine.framework.configmanager.ConfigManager*), 14
connect() (*decisionengine.framework.dataspace.datasource.DataSource method*), 21
connect() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 16
consumes() (*decisionengine.framework.modules.Publisher.Publisher method*), 27
consumes() (*decisionengine.framework.modules.SourceProxy.SourceProxy method*), 28
consumes() (*decisionengine.framework.modules.Transform.Transform method*), 29
consumes() (*decisionengine.framework.tests.PublisherNOP.PublisherNOP method*), 31
consumes() (*decisionengine.framework.tests.TransformNOP.TransformNOP method*), 32
create() (*decisionengine.framework.configmanager.ConfigManager.ConfigManager static method*), 14
create_datasource() (*decisionengine.framework.dataspace.dataspace.DataSourceLoader static method*), 23

create_parser() (*in module decisionengine.framework.engine.de_client*), 27
create_tables() (*decisionengine.framework.dataspace.datasource.DataSource method*), 21
create_tables() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 16

D

data() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 15
data_block_put() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 30
DataBlock (*class in decisionengine.framework.dataspace.datablock*), 18
dataproduct() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 15
dataproduct_table (*decisionengine.framework.dataspace.datasource.DataSource attribute*), 21
DataSource (*class in decisionengine.framework.dataspace.datasource*), 21
datasource() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 15
datasource() (*in module decisionengine.framework.tests.test_channel*), 32
DataSourceLoader (*class in decisionengine.framework.dataspace.dataspace*), 23
DataSpace (*class in decisionengine.framework.dataspace.dataspace*), 23
DataSpaceConfigurationError, 23
DataSpaceConnectionError, 23
DataSpaceError, 24
DataSpaceExistsError, 24
de_client_request() (*decisionengine.framework.tests.test_channel.TestChannel method*), 32
decision_cycle() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 30
decisionengine
    module, 33
DecisionEngine (*class in decisionengine.framework.engine.DecisionEngine*), 25
decisionengine.framework
    module, 33
decisionengine.framework.configmanager
    module, 15
decisionengine.framework.configmanager.ConfigManager
    module, 14
decisionengine.framework.configmanager.tests
    module, 14
decisionengine.framework.configmanager.tests.test_c
    module, 13
decisionengine.framework.dataspace
    module, 25
decisionengine.framework.dataspace.datablock
    module, 18
decisionengine.framework.dataspace.datasource
    module, 21
decisionengine.framework.dataspace.datasources
    module, 18
decisionengine.framework.dataspace.datasources.post
    module, 16
decisionengine.framework.dataspace.datasources.test
    module, 16
decisionengine.framework.dataspace.datasources.test
    module, 15
decisionengine.framework.dataspace.dataspace
    module, 23
decisionengine.framework.engine
    module, 27
decisionengine.framework.engine.de_client
    module, 27
decisionengine.framework.engine.DecisionEngine
    module, 25
decisionengine.framework.modules
    module, 29
decisionengine.framework.modules.de_logger
    module, 29
decisionengine.framework.modules.LogicEngine
    module, 27
decisionengine.framework.modules.Module
    module, 27
decisionengine.framework.modules.Publisher
    module, 27
decisionengine.framework.modules.Source
    module, 28
decisionengine.framework.modules.SourceProxy
    module, 28
decisionengine.framework.modules.Transform
    module, 29
decisionengine.framework.taskmanager
    module, 31
decisionengine.framework.taskmanager.TaskManager
    module, 29
decisionengine.framework.tests
    module, 33

decisionengine.framework.tests.PublisherNOP module, 31
decisionengine.framework.tests.SourceNOP module, 32
decisionengine.framework.tests.test_channel module, 32
decisionengine.framework.tests.TransformNOP module, 32
decisionengine.framework.util module, 33
decisionengine.framework.util.sockets module, 33
decisionengine.framework.util.tsort module, 33
decompress() (*in module decisionengine.framework.dataspace.datablock*), 20
default_data_lifetime (*decisionengine.framework.dataspace.datablock.Header attribute*), 20
delete() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 23
delete_data_older_than() (*decisionengine.framework.dataspace.datasource.DataSource method*), 21
delete_data_older_than() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 16
do_backup() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 30
duplicate() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 19
duplicate_datablock() (*decisionengine.framework.dataspace.datasource.DataSource method*), 21
duplicate_datablock() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 16
duplicate_datablock() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 23

E

ERROR (*decisionengine.framework.dataspace.dataspace.State attribute*), 24
evaluate() (*decisionengine.framework.modules.LogicEngine.LogicEngine method*), 27
execute_command_from_args() (*in module decisionengine.framework.engine.de_client*), 27

F

fixtures() (*in module decisionengine.framework.tests.test_channel*), 32

G

generate_insert_query() (*in module decisionengine.framework.dataspace.datasources.postgresql*), 18
get() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 19
get_channels() (*decisionengine.framework.configmanager.ConfigManager.ConfigManage method*), 14
get_connection() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 16
get_data_bock() (*decisionengine.framework.modules.Module.Module method*), 27
get_datablock() (*decisionengine.framework.dataspace.datasource.DataSource method*), 21
get_datablock() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 16
get_dataproduct() (*decisionengine.framework.dataspace.datasource.DataSource method*), 21
get_dataproduct() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 17
get_dataproduct() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 23
get_global_config() (*decisionengine.framework.configmanager.ConfigManager.ConfigManage method*), 14
get_header() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 19
get_header() (*decisionengine.framework.dataspace.datasource.DataSource method*), 21
get_header() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 17
get_header() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 23
get_last_generation_id() (*decisionengine.framework.dataspace.datasource.DataSource method*), 22
get_last_generation_id() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql*

method), 17
get_last_generation_id() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 23
get_logger() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 25
get_logger() (*in module decisionengine.framework.modules.de_logger*), 29
get_loglevel() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 30
get_metadata() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 19
get_metadata() (*decisionengine.framework.dataspace.datasource.DataSource method*), 22
get_metadata() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 17
get_metadata() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 23
get_paramaters() (*decisionengine.framework.modules.Module.Module method*), 27
get_produces() (*decisionengine.framework.configmanager.ConfigManager.ConfigManager method*), 14
get_random_port() (*in module decisionengine.framework.util.sockets*), 33
get_retention_interval() (*decisionengine.framework.dataspace.dataspace.Reaper method*), 24
get_schema() (*decisionengine.framework.dataspace.datasource.DataSource method*), 22
get_schema() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 17
get_state() (*decisionengine.framework.dataspace.dataspace.Reaper method*), 24
get_state() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 30
get_state_name() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 30
get_taskmanager() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 19
get_taskmanager() (*decisionengine.framework.dataspace.datasource.DataSource method*), 22
get_taskmanager() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 17
get_taskmanager() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 23

H

handle_sighup() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 25
Header (*class in decisionengine.framework.dataspace.datablock*), 19
header() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 15
headers_table (*decisionengine.framework.dataspace.datasource.DataSource attribute*), 22

I

IDLE (*decisionengine.framework.dataspace.dataspace.State attribute*), 24
insert() (*decisionengine.framework.dataspace.datasource.DataSource method*), 22
insert() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 17
insert() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 23
InvalidMetadataError, 20
is_expired() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 19
is_updated() (*decisionengine.framework.configmanager.ConfigManager.ConfigManager method*), 14
is_valid() (*decisionengine.framework.dataspace.datablock.Header method*), 20

K

keys() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 19

L

load() (*decisionengine.framework.configmanager.ConfigManager.ConfigManager method*), 14
load() (*in module decisionengine.framework.configmanager.tests.test_configmanager*), 13

LogicEngine (*class in decisionengine.framework.modules.LogicEngine*), 27

M

main() (*in module decisionengine.framework.engine.de_client*), 27
main() (*in module decisionengine.framework.engine.DecisionEngine*), 26
main() (*in module decisionengine.framework.modules.SourceProxy*), 28
mark_demented() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 23
mark_expired() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 19
mark_expired() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 23
Metadata (*class in decisionengine.framework.dataspace.datablock*), 20
metadata() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 15
metadata_table (*decisionengine.framework.dataspace.datasource.DataSource attribute*), 22
module
    decisionengine, 33
    decisionengine.framework, 33
    decisionengine.framework.configmanager, 15
    decisionengine.framework.configmanager.ConfigManager, 14
    decisionengine.framework.configmanager.tests, 14
    decisionengine.framework.configmanager.tests.test_configmanager, 13
    decisionengine.framework.dataspace, 25
    decisionengine.framework.dataspace.datablock, 18
    decisionengine.framework.dataspace.datasource, 21
    decisionengine.framework.dataspace.datasources, 18
    decisionengine.framework.dataspace.datasources.postgresql, 16
    decisionengine.framework.dataspace.datasources.tests, 16
    decisionengine.framework.dataspace.datasources.tests.test_postgresql, 15
    decisionengine.framework.dataspace.dataspace, 23
    decisionengine.framework.engine, 27
    decisionengine.framework.engine.de_client, 27
    decisionengine.framework.engine.DecisionEngine, 25
    decisionengine.framework.modules, 29
    decisionengine.framework.modules.de_logger, 29
    decisionengine.framework.modules.LogicEngine, 27
    decisionengine.framework.modules.Module, 27
    decisionengine.framework.modules.Publisher, 27
    decisionengine.framework.modules.Source, 28
    decisionengine.framework.modules.SourceProxy, 28
    decisionengine.framework.modules.Transform, 29
    decisionengine.framework.taskmanager, 31
    decisionengine.framework.taskmanager.TaskManager, 29
    decisionengine.framework.tests, 33
    decisionengine.framework.tests.PublisherNOP, 31
    decisionengine.framework.tests.SourceNOP, 32
    decisionengine.framework.tests.test_channel, 32
    decisionengine.framework.tests.TransformNOP, 32
    decisionengine.framework.util, 33
    decisionengine.framework.util.sockets, 33
    decisionengine.framework.util.tsort, 33
Module (*class in decisionengine.framework.modules.Module*), 27
module_config_info() (*in module decisionengine.framework.modules.SourceProxy*), 28
module_config_template() (*in module decisionengine.framework.modules.SourceProxy*), 28
must_have (*decisionengine.framework.modules.SourceProxy.SourceProxy attribute*), 28

O

OFFLINE (*decisionengine.framework.taskmanager.TaskManager.State*

attribute), 30

P

parse_program_options() (*in module decisionengine.framework.engine.DecisionEngine*), 26
Postgresql (*class in decisionengine.framework.dataspace.datasources.postgresql*), 16
produces() (*decisionengine.framework.modules.Source.Source method*), 28
produces() (*decisionengine.framework.modules.SourceProxy.SourceProxy method*), 28
produces() (*decisionengine.framework.modules.Transform.Transform method*), 29
produces() (*decisionengine.framework.tests.SourceNOP.SourceNOP method*), 32
produces() (*decisionengine.framework.tests.TransformNOP.TransformNOP method*), 32
publish() (*decisionengine.framework.modules.Publisher.Publisher method*), 27
publish() (*decisionengine.framework.tests.PublisherNOP.PublisherNOP method*), 31
Publisher (*class in decisionengine.framework.modules.Publisher*), 27
PublisherNOP (*class in decisionengine.framework.tests.PublisherNOP*), 31
put() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 19
pytestmark (*decisionengine.framework.tests.test_channel.TestChannel attribute*), 32

R

reap() (*decisionengine.framework.dataspace.dataspace.Reaper method*), 24
Reaper (*class in decisionengine.framework.dataspace.dataspace*), 24
reaper_start() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 25
reaper_status() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 25
reaper_stop() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 25
reload() (*decisionengine.framework.configmanager.ConfigManager.Con method*), 14
reload_config() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 25
RequestHandler (*class in decisionengine.framework.engine.DecisionEngine*), 26
required_keys (*decisionengine.framework.dataspace.datablock.Header attribute*), 20
required_keys (*decisionengine.framework.dataspace.datablock.Metadata attribute*), 20
rpc_get_channel_log_level() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 25
rpc_get_log_level() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 25
rpc_paths (*decisionengine.framework.engine.DecisionEngine.RequestH attribute*), 26
rpc_print_product() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 25
rpc_print_products() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 25
rpc_reaper_start() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 25
rpc_reaper_status() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 25
rpc_reaper_stop() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 25
rpc_reload_config() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 25
rpc_set_channel_log_level() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 25
rpc_show_config() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 25
rpc_show_de_config() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 26
rpc_start_channel() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 26
rpc_start_channels() (*decisio-*

nengine.framework.engine.DecisionEngine.DecisionEngine method), 26
rpc_status() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 26
rpc_stop() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 26
rpc_stop_channel() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 26
rpc_stop_channels() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 26
run() (*decisionengine.framework.engine.DecisionEngine.Worker method*), 26
run() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 30
run() (*decisionengine.framework.tests.test_channel.Worker method*), 32
run_logic_engine() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 30
run_publishers() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 30
run_source() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 30
run_transform() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 30
run_transforms() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 31
RUNNING (*decisionengine.framework.dataspace.dataspace.State attribute*), 24

S

service_actions() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 26
set_data_bock() (*decisionengine.framework.modules.Module.Module method*), 27
set_logging() (*in module decisionengine.framework.modules.de_logger*), 29
set_loglevel() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 31
set_retention_interval() (*decisionengine.framework.dataspace.dataspace.Reaper method*), 24
setstate() (*decisionengine.framework.dataspace.datablock.Metadata method*), 20
setstate() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 31
setup_stream_logging() (*in module decisionengine.framework.modules.de_logger*), 29
setUp() (*decisionengine.framework.tests.test_channel.TestChannel method*), 32
SHUTDOWN (*decisionengine.framework.taskmanager.TaskManager.State attribute*), 30
SHUTTINGDOWN (*decisionengine.framework.taskmanager.TaskManager.State attribute*), 30
Singleton (*class in decisionengine.framework.dataspace.dataspace*), 24
SLEEPING (*decisionengine.framework.dataspace.dataspace.State attribute*), 24
Source (*class in decisionengine.framework.modules.Source*), 28
SourceNOP (*class in decisionengine.framework.tests.SourceNOP*), 32
SourceProxy (*class in decisionengine.framework.modules.SourceProxy*), 28
start() (*decisionengine.framework.dataspace.dataspace.Reaper method*), 24
start_channel() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 26
start_channels() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 26
start_sources() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 31
STARTING (*decisionengine.framework.dataspace.dataspace.State attribute*), 24
State (*class in decisionengine.framework.dataspace.dataspace*), 24
State (*class in decisionengine.framework.taskmanager.TaskManager*), 29
STEADY (*decisionengine.framework.taskmanager.TaskManager.State attribute*), 30
stop() (*decisionengine.framework.dataspace.dataspace.Reaper method*), 24
stop_channel() (*decisionengine.framework.engine.DecisionEngine.DecisionEngine method*), 26
stop_channels() (*decisio-*

nengine.framework.engine.DecisionEngine.DecisionEngine method), 26
STOPPED (*decisionengine.framework.dataspace.dataspace.State attribute*), 24
STOPPING (*decisionengine.framework.dataspace.dataspace.State attribute*), 24
store_taskmanager() (*decisionengine.framework.dataspace.datablock.DataBlock method*), 19
store_taskmanager() (*decisionengine.framework.dataspace.datasource.DataSource method*), 22
store_taskmanager() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 18
store_taskmanager() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 23

T

tables (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql attribute*), 18
TaskManager (*class in decisionengine.framework.taskmanager.TaskManager*), 30
taskmanager() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 15
taskmanager_table (*decisionengine.framework.dataspace.datasource.DataSource attribute*), 22
tearDown() (*decisionengine.framework.tests.test_channel.TestChannel method*), 32
test_channel_empty_config() (*in module decisionengine.framework.configmanager.tests.test_configmanager*), 13
test_channel_empty_dictionary() (*in module decisionengine.framework.configmanager.tests.test_configmanager*), 13
test_channel_names() (*in module decisionengine.framework.configmanager.tests.test_configmanager*), 13
test_channel_no_config_files() (*in module decisionengine.framework.configmanager.tests.test_configmanager*), 13
test_channel_no_modules() (*in module decisionengine.framework.configmanager.tests.test_configmanager*), 13
test_client_can_get_de_server_reaper_start_delay() (*decisionengine.framework.tests.test_channel.TestChannel method*), 32
test_client_can_get_de_server_reaper_status() (*decisionengine.framework.tests.test_channel.TestChannel method*), 32
test_client_can_get_de_server_reaper_stop() (*decisionengine.framework.tests.test_channel.TestChannel method*), 32
test_client_can_get_de_server_reload_config() (*decisionengine.framework.tests.test_channel.TestChannel method*), 32
test_client_can_get_de_server_show_channel_logger_l (*decisionengine.framework.tests.test_channel.TestChannel method*), 32
test_client_can_get_de_server_show_config() (*decisionengine.framework.tests.test_channel.TestChannel method*), 32
test_client_can_get_de_server_show_logger_level() (*decisionengine.framework.tests.test_channel.TestChannel method*), 32
test_client_can_get_de_server_status() (*decisionengine.framework.tests.test_channel.TestChannel method*), 32
test_create_tables() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 15
test_empty_config() (*in module decisionengine.framework.configmanager.tests.test_configmanager*), 13
test_empty_dict() (*in module decisionengine.framework.configmanager.tests.test_configmanager*), 13
test_empty_dict_with_leading_comment() (*in module decisionengine.framework.configmanager.tests.test_configmanager*), 13
test_generate_insert_query() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 15
test_get_last_generation_id() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 15
test_get_taskmanager() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 15
test_global_channel_log_level_in_config() (*decisionengine.framework.tests.test_channel.TestChannel method*), 32
test_insert() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 15
test_minimal_jsonnet_right_extension() (*in module decisio-*

nengine.framework.configmanager.tests.test_configmanager), 13
test_minimal_jsonnet_wrong_extension() (*in module decisionengine.framework.configmanager.tests.test_configmanager*), 14
test_minimal_python() (*in module decisionengine.framework.configmanager.tests.test_configmanager*), 14
test_minimal_python_default() (*in module decisionengine.framework.configmanager.tests.test_configmanager*), 14
test_program_options_default() (*in module decisionengine.framework.configmanager.tests.test_configmanager*), 14
test_program_options_update() (*in module decisionengine.framework.configmanager.tests.test_configmanager*), 14
test_store_taskmanager() (*in module decisionengine.framework.dataspace.datasources.tests.test_postgresql*), 15
test_wrong_type() (*in module decisionengine.framework.configmanager.tests.test_configmanager*), 14
TestChannel (*class in decisionengine.framework.tests.test_channel*), 32
Transform (*class in decisionengine.framework.modules.Transform*), 29
transform() (*decisionengine.framework.modules.Transform.Transform method*), 29
transform() (*decisionengine.framework.tests.TransformNOP.TransformNOP method*), 32
TransformNOP (*class in decisionengine.framework.tests.TransformNOP*), 32
tsort() (*in module decisionengine.framework.util.tsort*), 33

U

update() (*decisionengine.framework.dataspace.datasource.DataSource method*), 22
update() (*decisionengine.framework.dataspace.datasources.postgresql.Postgresql method*), 18
update() (*decisionengine.framework.dataspace.dataspace.DataSpace method*), 23

V

valid_states (*decisionengine.framework.dataspace.datablock.Metadata attribute*), 20

W

wait_for_all() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 31
wait_for_any() (*decisionengine.framework.taskmanager.TaskManager.TaskManager method*), 31
Worker (*class in decisionengine.framework.engine.DecisionEngine*), 26
Worker (*class in decisionengine.framework.taskmanager.TaskManager*), 31
Worker (*class in decisionengine.framework.tests.test_channel*), 32

Z

zdumps() (*in module decisionengine.framework.dataspace.datablock*), 20
zloads() (*in module decisionengine.framework.dataspace.datablock*), 20