
decisionengine

Release 1.1.1

Fermi Research Alliance, LLC.

Aug 27, 2020

CONTENTS

1	Release Notes	3
2	Developer Documentation	9
3	Source code	13
4	Indices and tables	33
	Python Module Index	35
	Index	37

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

RELEASE NOTES

1.1 Release 1.3.0

In this release:

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

1.1.1 Full list of commits since version 1.2

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

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

afd8837 : Configuration-manager fixes (#128)

571e2be : Remove pip installed system python packages

407d9ed : Update Dockerfile

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

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

2840813 : Switch to Jsonnet configuration system (#125)

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

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

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

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

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

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

c3d1be3 : Python 3 upgrades. (#116)

e7399af : Header fix (#114)

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

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

97c21b1 : osg version 3.5 (#112)

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

1.2 Release 1.2.0

In this release:

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

1.2.1 decisionengine

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

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

1.2.2 decisionengine_modules

30d928b : clone version 1.2.0 of decisionengine
ae7c5a6 : Jenkins pipeline improvements (#236)
310befd : T198 (#235)
a65886d : Fix import as reported in : <https://github.com/HEPcloud/decisionengin...> (#232)
93711cc : Run coveralls even if tests fail (#229)
03d763a : Jenkins pipeline improvements (#230)
f48d30f : Fix/223 (#228)
c8aa262 : github ticket 199 (#222)
0323bda : Address : https://github.com/HEPcloud/decisionengine_modules/issues/224 (#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 argument -DPYVER=3.6 to build python3 library <https://fermicloud140.fnal.gov/reviews/r/31/>
843f30c : Minor cleanups per travis-lint
a538cac : Remove unused C++ files.
4c9d125 : Update repo where action is taken from
87fb2d9 : Update rpms installed in docker image. Update entrypoint.sh to use cmake3.
199ee87 : Find python3 libraries using cmake3 from epel rpm Also need to install python3-devel
4c79d2c : Remove unnused GNUmakefiles.
94342ee : Add unit test as a Github Action
1a0e102 : more advanced travis.yml
0be413f : Add helper file for pip
7794327 : Make recursive import happy
7005c78 : Add simple target
de8b0fa : python3 compliance: replace string.join() where appropriate, handle UserDict
2662e6c : note required packages
3b87119 : Add missing header includes.
3e79b84 : Remove defunct code and its tests

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

DEVELOPER DOCUMENTATION

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

2.1 Decisionengine framework

2.1.1 Prerequisites:

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

2.1.2 Build & test

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

export PYTHONPATH=`pwd`

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

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

(continues on next page)

(continued from previous page)

```

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

```

```

===== 26 passed in 23.86s_
↳=====

```

2.2 Decisionengine_modules

2.2.1 Prerequisites:

In Addition to above installed packages

```

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

```

2.2.2 Test

```

cd

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

```

Current status:

```

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

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

```

(continues on next page)

(continued from previous page)

```

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

```

(continues on next page)

(continued from previous page)

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

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

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


SOURCE CODE

3.1 Welcome to decisionengine's documentation!

3.1.1 decisionengine package

Subpackages

decisionengine.framework package

Subpackages

decisionengine.framework.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(load)
```

```
decisionengine.framework.configmanager.tests.test_configmanager.test_channel_empty_dict(load)
```

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

```
decisionengine.framework.configmanager.tests.test_configmanager.test_channel_no_config_file(load)
```

```
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_leading(load)
```

```
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,  
                                                                                       cap-  
                                                                                       sys)
```

```
decisionengine.framework.configmanager.tests.test_configmanager.test_minimal_python_default
```

```
decisionengine.framework.configmanager.tests.test_configmanager.test_program_options_default
```

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

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

Module contents

Submodules

decisionengine.framework.dataspace.datasources.postgresql module

```
class decisionengine.framework.dataspace.datasources.postgresql.Postgresql (config_dict)
    Bases: decisionengine.framework.dataspace.datasource.DataSource
    Implementation of postgresql data source

    __query (query_string, values=None, cursor_factory=None)
    _abc_cache = <_weakrefset.WeakSet object>
    _abc_negative_cache = <_weakrefset.WeakSet object>
    _abc_negative_cache_version = 185
    _abc_registry = <_weakrefset.WeakSet object>
    _delete (sql_query, values=None)
    _insert (table_name_or_sql_query, record=None)
    _insert_returning_result (table_name_or_sql_query, record=None)
    _remove (sql_query, values=None)
    _select (query_string, values=None, cursor_factory=None)
    _select_dictresult (sql_query, values=None)
    _select_getresult (sql_query, values=None)
    _select_tuple (sql_query, values)
    _update (query_string, values=None)
    _update_returning_result (query_string, values=None)

    close ()
        Close all connections to the database

    connect ()
        Create a pool of database connections

    create_tables ()
        Create database tables

    delete_data_older_than (days)
        Delete data older than days interval :type days: int :arg days: remove data older than days interval

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

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

    get_connection ()
```

get_datablock (*taskmanager_id*, *generation_id*)

Return the entire datablock from the dataproduct table for the given taskmanager_id, generation_id

Parameters

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

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

Return the data from the dataproduct table for the given taskmanager_id, generation_id, key

Parameters

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

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

Return the header from the header table for the given taskmanager_id, generation_id, key

Parameters

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

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

Return last generation id for current task manager or taskmanager w/ task_manager_id.

Parameters

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

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

Return the metadata from the metadata table for the given taskmanager_id, generation_id, key

Parameters

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

get_schema (*table=None*)

Given the table name return it's schema

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

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

Retrieve TaskManager :type taskmanager_name: *string* :arg taskmanager_name: name of taskmanager to retrieve :type taskmanager_id: *string* :arg taskmanager_id: id of taskmanager to retrieve

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

Insert data into respective tables for the given taskmanager_id, generation_id, key

Parameters

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

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

store_taskmanager (name, taskmanager_id)

Store TaskManager :type taskmanager_name: string :arg taskmanager_name: name of taskmanager to

retrieve :type taskmanager_id: string :arg taskmanager_id: id of taskmanager to retrieve

tables = {'dataprodukt': ['taskmanager_id TEXT', 'generation_id INT', 'key TEXT', 'va

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

Update the data in respective tables for the given taskmanager_id, generation_id, key

Parameters

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

decisionengine.framework.dataspace.datasources.postgresql.**generate_insert_query** (table_name,
keys)

Generate insert query given table name and list of fields

Parameters

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

Keys list

Return type str - insert query

Module contents

Submodules

decisionengine.framework.dataspace.datablock module

```
class decisionengine.framework.dataspace.datablock.DataBlock (dataspace,  
                                                             name, taskman-  
                                                             ager_id=None,  
                                                             genera-  
                                                             tion_id=None, se-  
                                                             quence_id=None)
```

Bases: object

_insert (key, value, header, metadata)

Insert a new product into database with header and metadata

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

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

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

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

Return type *DataBlock*

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

Return type *dict*

get_header (*key*)
 Return the Header associated with the key in the database

Return type *Header*

get_metadata (*key*)
 Return the metadata associated with the key in the database

Return type *Metadata*

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

The dictionary returned looks like : {'datestamp': datetime.datetime(2017, 12, 20, 17, 37, 17, 503210,

tzinfo=psycpg2.tz.FixedOffsetTimezone(offset=-360, name=None)),

'sequence_id': 135L, 'name': 'AWS_Calculations', 'taskmanager_id': '77B16EB5-C79E-45B0-B1B1-37E846692E1D'}

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

keys ()

mark_expired (*expiration_time*)
 Set the expiration_time for the current generation of the dataproduct and mark it as expired if expiration_time <= current time

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

store_taskmanager (*taskmanager_name, taskmanager_id*)
 Persist TaskManager, returns sequence number :type taskmanager_name: *string* :type taskmanager_id: :obj: *string* :rtype: *int*

```
class decisionengine.framework.dataspace.datablock.Header(taskmanager_id, create_time=None, expiration_time=None, scheduled_create_time=None, creator='module', schema_id=None)

    Bases: collections.UserDict

    _abc_cache = <_weakrefset.WeakSet object>
    _abc_negative_cache = <_weakrefset.WeakSet object>
    _abc_negative_cache_version = 185
    _abc_registry = <_weakrefset.WeakSet object>
    default_data_lifetime = 1800

    is_valid()
        Check if the Header has minimum required information

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

exception decisionengine.framework.dataspace.datablock.InvalidMetadataError
    Bases: Exception

    Errors due to invalid Metadata

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

    dataproducit_table = 'dataproducit'
        Name of the dataproducit 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 dataproducit table for the given taskmanager_id, generation_id

        Parameters
        • taskmanager_id (string) – taskmanager_id for generation to be retrieved
        • generation_id (int) – generation_id of the data

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

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

    abstract get_header (taskmanager_id, generation_id, key)
        Return the header from the header table for the given taskmanager_id, generation_id, key
```

Parameters

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

abstract get_last_generation_id (name, taskmanager_id=None)

Return last generation id for current task manager or taskmanager w/ task_manager_id.

Parameters

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

abstract get_metadata (taskmanager_id, generation_id, key)

Return the metadata from the metadata table for the given taskmanager_id, generation_id, key

Parameters

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

abstract get_schema (table=None)

Given the table name return it's schema

Parameters table (string) – Name of the table

abstract get_taskmanager (taskmanager_name, taskmanager_id)

Retrieve TaskManager :type taskmanager_name: string :arg taskmanager_name: name of taskmanager to retrieve :type taskmanager_id: string :arg taskmanager_id: id of taskmanager to retrieve

header_table = 'header'

Name of the header table

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

Insert data into respective tables for the given taskmanager_id, generation_id, key

Parameters

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

metadata_table = 'metadata'

Name of the metadata table

abstract store_taskmanager (taskmanager_name, taskmanager_id)

Store TaskManager :type taskmanager_name: string :arg taskmanager_name: name of taskmanager to retrieve :type taskmanager_id: string :arg taskmanager_id: id of taskmanager to retrieve

taskmanager_table = 'taskmanager'

Name of the taskmanager table

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

Update the data in respective tables for the given taskmanager_id, generation_id, key

Parameters

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

decisionengine.framework.dataspace.dataspace module

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

Bases: object

_ds = None

static create_datasource (*module_name, class_name, config*)

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

Bases: object

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

_tables_created = False

Description of tables and their columns

close ()

delete (*taskmanager_id, all_generations=False*)

duplicate_datablock (*taskmanager_id, generation_id, new_generation_id*)

get_dataproduct (*taskmanager_id, generation_id, key*)

get_header (*taskmanager_id, generation_id, key*)

get_last_generation_id (*taskmanager_name, taskmanager_id=None*)

get_metadata (*taskmanager_id, generation_id, key*)

get_taskmanager (*taskmanager_name, taskmanager_id=None*)

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

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

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

store_taskmanager (*name, id*)

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

exception decisionengine.framework.dataspace.dataspace.**DataSpaceConfigurationError**

Bases: Exception

Errors related to database access

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

    Errors related to database access

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

    Errors related to database access

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

    Errors related to database access

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

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

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

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

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

    _instances = {}

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

    An enumeration.

    ERROR = 7
    IDLE = 1
    RUNNING = 3
    SLEEPING = 4
    STARTING = 2
    STOPPED = 6
    STOPPING = 5
```

Module contents

decisionengine.framework.engine package

Submodules

decisionengine.framework.engine.DecisionEngine module

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

```
class decisionengine.framework.engine.DecisionEngine.DecisionEngine (cfg,
                                                                    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.

    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()
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.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_block()
    get_parameters()
    set_data_block(data_block)
```

decisionengine.framework.modules.Publisher module

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

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

decisionengine.framework.modules.Source module

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

    acquire()
    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 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,
genera-
tion_id,
chan-
nel_dict,
global_config)

Bases: object

Task Manager

_take_offline (*current_data_block*)
offline and stop task manager

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

Parameters

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

decision_cycle()

Decision cycle to be run periodically (by trigger)

do_backup()

Duplicate current data block and return its copy

Return type DataBlock**get_loglevel()****get_state()****get_state_name()****run()**

Task Manager main loop

run_logic_engine (*data_block=None*)

Run Logic Engine.

Parameters **data_block** (DataBlock) – data block**run_publishers** (*actions, facts, data_block=None*)

Run Publishers in main process.

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

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

Parameters **src** (*Worker*) – source Worker**run_transform** (*transform, data_block*)

Run a transform

Parameters

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

run_transforms (*data_block=None*)

Run transforms. So far in main process.

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

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

set_state (*state*)**start_sources** (*data_block=None*)

Start sources, each in a separate thread

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

Wait for all sources or transforms to finish

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

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

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

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

Provides interface to loadable modules and events to synchronise execution

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

Module contents

decisionengine.framework.util package

Submodules

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`

INDICES AND TABLES

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

PYTHON MODULE INDEX

d

`decisionengine`, [31](#)
`decisionengine.framework`, [31](#)
`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`,
 [26](#)
`decisionengine.framework.engine.DecisionEngine`,
 [25](#)
`decisionengine.framework.modules`, [29](#)
`decisionengine.framework.modules.de_logger`,
 [28](#)
`decisionengine.framework.modules.LogicEngine`,
 [27](#)
`decisionengine.framework.modules.Module`,
 [27](#)
`decisionengine.framework.modules.Publisher`,
 [27](#)
`decisionengine.framework.modules.Source`,
 [27](#)
`decisionengine.framework.modules.SourceProxy`,
 [27](#)
`decisionengine.framework.modules.Transform`,
 [28](#)
`decisionengine.framework.taskmanager`,
 [31](#)
`decisionengine.framework.taskmanager.TaskManager`,
 [29](#)
`decisionengine.framework.util`, [31](#)
`decisionengine.framework.util.tsort`, [31](#)

INDEX

Symbols

<code>__query()</code> (<i>decisionengine.framework.dataspace.datasources.postgresql.Postgresql</i> method), 16	<code>_abc_registry</code> (<i>decisionengine.framework.dataspace.datablock.Metadata</i> attribute), 20
<code>_abc_cache</code> (<i>decisionengine.framework.dataspace.datablock.Header</i> attribute), 20	<code>_abc_registry</code> (<i>decisionengine.framework.dataspace.datasource.DataSource</i> attribute), 21
<code>_abc_cache</code> (<i>decisionengine.framework.dataspace.datablock.Metadata</i> attribute), 20	<code>_abc_registry</code> (<i>decisionengine.framework.dataspace.datasources.postgresql.Postgresql</i> attribute), 16
<code>_abc_cache</code> (<i>decisionengine.framework.dataspace.datasource.DataSource</i> attribute), 21	<code>_check_keys()</code> (in module <i>decisionengine.framework.configmanager.ConfigManager</i>), 14
<code>_abc_cache</code> (<i>decisionengine.framework.dataspace.datasources.postgresql.Postgresql</i> attribute), 16	<code>_config_from_file()</code> (in module <i>decisionengine.framework.configmanager.ConfigManager</i>), 14
<code>_abc_negative_cache</code> (<i>decisionengine.framework.dataspace.datablock.Header</i> attribute), 20	<code>_convert_to_json()</code> (in module <i>decisionengine.framework.configmanager.ConfigManager</i>), 14
<code>_abc_negative_cache</code> (<i>decisionengine.framework.dataspace.datablock.Metadata</i> attribute), 20	<code>_delete()</code> (<i>decisionengine.framework.dataspace.datasources.postgresql.Postgresql</i> method), 16
<code>_abc_negative_cache</code> (<i>decisionengine.framework.dataspace.datasource.DataSource</i> attribute), 21	<code>_dispatch()</code> (<i>decisionengine.framework.engine.DecisionEngine.DecisionEngine</i> method), 25
<code>_abc_negative_cache</code> (<i>decisionengine.framework.dataspace.datasources.postgresql.Postgresql</i> attribute), 16	<code>_ds</code> (<i>decisionengine.framework.dataspace.dataspace.DataSourceLoader</i> attribute), 23
<code>_abc_negative_cache_version</code> (<i>decisionengine.framework.dataspace.datablock.Header</i> attribute), 20	<code>get_data()</code> (<i>decisionengine.framework.modules.SourceProxy.SourceProxy</i> method), 28
<code>_abc_negative_cache_version</code> (<i>decisionengine.framework.dataspace.datablock.Metadata</i> attribute), 20	<code>_insert()</code> (<i>decisionengine.framework.dataspace.datablock.DataBlock</i> method), 18
<code>_abc_negative_cache_version</code> (<i>decisionengine.framework.dataspace.datasource.DataSource</i> attribute), 21	<code>_insert()</code> (<i>decisionengine.framework.dataspace.datasources.postgresql.Postgresql</i> method), 16
<code>_abc_negative_cache_version</code> (<i>decisionengine.framework.dataspace.datasource.DataSource</i> attribute), 21	<code>_insert_returning_result()</code> (<i>decisionengine.framework.dataspace.datasources.postgresql.Postgresql</i> method), 16
<code>_abc_negative_cache_version</code> (<i>decisionengine.framework.dataspace.datasources.postgresql.Postgresql</i> attribute), 16	<code>_instances</code> (<i>decisionengine.framework.dataspace.dataspace.Singleton</i> attribute), 24
<code>_abc_registry</code> (<i>decisionengine.framework.dataspace.datablock.Header</i> attribute), 20	<code>_load_channels()</code> (<i>decisionengine.framework.configmanager.ConfigManager.ConfigManager</i> method), 14
	<code>_make_logger()</code> (in module <i>decisionengine.framework.configmanager.ConfigManager</i>), 14

engine.framework.configmanager.ConfigManager), 14
 build_xmlrpc_connection() (in module decisionengine.framework.engine.de_client), 26
 _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
 _tables_created (decisionengine.framework.dataspace.dataspace.DataSpace attribute), 23
 _take_offline() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 29
 _update() (decisionengine.framework.dataspace.datablock.DataBlock static method), 14
 _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
 acquire() (decisionengine.framework.modules.Source.Source method), 27
 acquire() (decisionengine.framework.modules.SourceProxy.SourceProxy method), 28
 BOOT (decisionengine.framework.taskmanager.TaskManager.State attribute), 29
 Channel (class in decisionengine.framework.taskmanager.TaskManager), 29
 close() (decisionengine.framework.dataspace.datasource.DataSource method), 21
 close() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 18
 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.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), 28
 create() (decisionengine.framework.configmanager.ConfigManager.Constructor static method), 14
 create_datasource() (decisionengine.framework.dataspace.dataspace.DataSourceLoader static method), 23
 create_parser() (in module decisionengine.framework.engine.de_client), 26
 create_tables() (decisionengine.framework.dataspace.datasource.DataSource method), 21
 create_tables() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 16
 data() (in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 15
 data_block_put() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 29

DataBlock	(class in decisionengine.framework.dataspace.datablock), 18	decisionengine.framework.dataspace.datasources.testmodule, 15
dataprodut ()	(in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 15	decisionengine.framework.dataspace.dataspace
dataprodut_table	(decisionengine.framework.dataspace.datasource.DataSource attribute), 21	decisionengine.framework.engine
DataSource	(class in decisionengine.framework.dataspace.datasource), 21	decisionengine.framework.engine.de_client
datasource ()	(in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 15	decisionengine.framework.engine.DecisionEngine
DataSourceLoader	(class in decisionengine.framework.dataspace.dataspace), 23	decisionengine.framework.modules
DataSpace	(class in decisionengine.framework.dataspace.dataspace), 23	decisionengine.framework.modules.de_logger
DataSpaceConfigurationError,	23	decisionengine.framework.modules.LogicEngine
DataSpaceConnectionError,	23	decisionengine.framework.modules.Module
DataSpaceError,	24	decisionengine.framework.modules.Publisher
DataSpaceExistsError,	24	decisionengine.framework.modules.Source
decision_cycle ()	(decisionengine.framework.taskmanager.TaskManager.TaskManager method), 30	decisionengine.framework.modules.SourceProxy
decisionengine	module, 31	decisionengine.framework.modules.Transform
DecisionEngine	(class in decisionengine.framework.engine.DecisionEngine), 25	decisionengine.framework.taskmanager
decisionengine.framework	module, 31	decisionengine.framework.taskmanager.TaskManager
decisionengine.framework.configmanager	module, 15	decisionengine.framework.util
decisionengine.framework.configmanager.ConfigManager	module, 14	decisionengine.framework.util.tsort
decisionengine.framework.configmanager.tests	module, 14	decompress () (in module decisionengine.framework.dataspace.datablock), 30
decisionengine.framework.configmanager.tests.test_decisionengine_framework	module, 13	default_data_lifetime (decisionengine.framework.dataspace.datablock.Header attribute), 20
decisionengine.framework.dataspace	module, 25	delete_data_older_than () (decisionengine.framework.dataspace.dataspace.DataSource method), 23
decisionengine.framework.dataspace.datablock	module, 18	delete_data_older_than () (decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 16
decisionengine.framework.dataspace.datasource	module, 21	duplicate () (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 30
decisionengine.framework.dataspace.datasource.backup ()	module, 18	duplicate () (decisionengine.framework.dataspace.datablock.DataBlock
decisionengine.framework.dataspace.datasources.postgresql	module, 16	

method), 19
duplicate_datablock() (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.datasources.postgresql.Postgresql method), 17
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), 26
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.ConfigManager method), 14
get_connection() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 16
get_data_block() (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_dataproduct() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 14
get_global_config() (decisionengine.framework.configmanager.ConfigManager.ConfigManager 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), 28
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_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), 14
[get_state\(\)](#) (decisionengine.framework.dataspace.datablock.Header method), 20
[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 load() (in module decisionengine.framework.configmanager.tests.test_configmanager), 13
[get_taskmanager\(\)](#) (decisionengine.framework.dataspace.datasource.DataSource LogicEngine (class in decisionengine.framework.modules.LogicEngine), 27
[get_taskmanager\(\)](#) (decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 17
[get_taskmanager\(\)](#) (decisionengine.framework.dataspace.dataspace.DataSpace main() (in module decisionengine.framework.engine.de_client), 26
[get_taskmanager\(\)](#) (decisionengine.framework.dataspace.dataspace.DataSpace main() (in module decisionengine.framework.engine.DecisionEngine), 26
[handle_sighup\(\)](#) (decisionengine.framework.engine.DecisionEngine.DecisionEngine main() (in module decisionengine.framework.modules.SourceProxy), 28
[Header](#) (class in decisionengine.framework.dataspace.datablock), 19
[header\(\)](#) (in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 15
[header_table](#) (decisionengine.framework.dataspace.datasource.DataSource mark_expired() (in module decisionengine.framework.dataspace.dataspace.DataSpace metadata (class in decisionengine.framework.dataspace.datablock), 20
[IDLE](#) (decisionengine.framework.dataspace.dataspace.State metadata() (in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 15
[insert\(\)](#) (decisionengine.framework.dataspace.datasource.DataSource metadata_table (decisionengine.framework.dataspace.datasource.DataSource attribute), 22
[insert\(\)](#) (decisionengine.framework.dataspace.dataspace.DataSpace decisionengine, 31
[InvalidMetadataError](#), 20
[is_expired\(\)](#) (decisionengine.framework.dataspace.datablock.DataBlock decisionengine.framework.configmanager, 15
[is_updated\(\)](#) (decisionengine.framework.configmanager.ConfigManager.ConfigManager decisionengine.framework.configmanager.ConfigManager, 14

decisionengine.framework.configmanager.must_have (decisionengine.framework.modules.SourceProxy.SourceProxy
 14 attribute), 28
 decisionengine.framework.configmanager.tests.test_configmanager,
 13
 decisionengine.framework.dataspace, OFFLINE (decisionengine.framework.taskmanager.TaskManager.State
 25 attribute), 29
 decisionengine.framework.dataspace.datablock,
 18
 decisionengine.framework.dataspace.datasources, parse_program_options() (in module decisio-
 21 nengine.framework.engine.DecisionEngine),
 decisionengine.framework.dataspace.datasources, 26,
 18
 decisionengine.framework.dataspace.datasources.postgresql (class in decisio-
 decisionengine.framework.dataspace.datasources.postgresql),
 16 16
 decisionengine.framework.dataspace.datasources.tests, (decisio-
 16 nengine.framework.modules.Source.Source
 decisionengine.framework.dataspace.datasources.tests.test_postgresql,
 15 method), 27
 decisionengine.framework.dataspace.dataspace, produces() (decisio-
 23 nengine.framework.modules.SourceProxy.SourceProxy
 method), 28
 decisionengine.framework.engine, 27 produces() (decisio-
 decisionengine.framework.engine.de_client, nengine.framework.modules.Transform.Transform
 26 method), 28
 decisionengine.framework.engine.DecisionEngine, publish() (decisionengine.framework.modules.Publisher.Publisher
 25 method), 27
 decisionengine.framework.modules, 29 Publisher (class in decisio-
 decisionengine.framework.modules.de_logger, nengine.framework.modules.Publisher),
 28 27
 decisionengine.framework.modules.LogicEngine, put() (decisionengine.framework.dataspace.datablock.DataBlock
 27 method), 19
 decisionengine.framework.modules.Module,
 27
 decisionengine.framework.modules.Publisher, reap() (decisionengine.framework.dataspace.dataspace.Reaper
 27 method), 24
 decisionengine.framework.modules.SourceReaper (class in decisio-
 27 nengine.framework.dataspace.dataspace),
 decisionengine.framework.modules.SourceProxy, 24
 27
 decisionengine.framework.modules.Transform, reaper_start() (decisio-
 28 nengine.framework.engine.DecisionEngine.DecisionEngine
 method), 25
 decisionengine.framework.taskmanager, reaper_status() (decisio-
 31 nengine.framework.engine.DecisionEngine.DecisionEngine
 method), 25
 decisionengine.framework.taskmanager.TaskManager, 29
 decisionengine.framework.util, 31 reaper_stop() (decisio-
 decisionengine.framework.util.tsort, nengine.framework.engine.DecisionEngine.DecisionEngine
 31 method), 25
 Module (class in decisio-
 nengine.framework.modules.Module), 27
 module_config_info() (in module decisio-
 nengine.framework.modules.SourceProxy),
 28
 module_config_template() (in module decisio-
 nengine.framework.modules.SourceProxy), 28

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.DecisionEngine 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() (decisionengine.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.taskmanager.TaskManager.TaskManager method), 30

run() (decisionengine.framework.engine.DecisionEngine.Worker method), 26

run_logic_engine() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 30

RequestHandler (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 30

source() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 30

transform() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 30

transforms() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 30

RuleEngine (decisionengine.framework.dataspace.dataspace.State attribute), 24

set_data_bock() (decisionengine.framework.modules.Module.Module method), 27

set_logging() (in module decisionengine.framework.modules.de_logger), 28

set_loglevel() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 30

set_retention_interval() (decisionengine.framework.dataspace.dataspace.Reaper method), 24

set_state() (decisionengine.framework.dataspace.datablock.Metadata method), 20

set_state() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 30

set_stream_logging() (in module decisionengine.framework.modules.de_logger), 29

SHUTTINGDOWN (decisionengine.framework.taskmanager.TaskManager.State attribute), 29

SHUTTINGDOWN (decisionengine.framework.taskmanager.TaskManager.State attribute), 29

Singleton (class in decisionengine.framework.dataspace.dataspace), 23
24

T

SLEEPING (decisionengine.framework.dataspace.dataspace.State attribute), 24

Source (class in decisionengine.framework.modules.Source), 27

SourceProxy (class in decisionengine.framework.modules.SourceProxy), 27

start() (decisionengine.framework.dataspace.dataspace.Reaper method), 24

start_channel() (decisionengine.framework.engine.DecisionEngine.DecisionEngine attribute), 22
method), 26

start_channels() (decisionengine.framework.engine.DecisionEngine.DecisionEngine attribute), 22
method), 26

start_sources() (decisionengine.framework.taskmanager.TaskManager.TaskManager attribute), 30
method), 30

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), 29

stop() (decisionengine.framework.dataspace.dataspace.Reaper method), 24

stop_channel() (decisionengine.framework.engine.DecisionEngine.DecisionEngine attribute), 22
method), 26

stop_channels() (decisionengine.framework.engine.DecisionEngine.DecisionEngine attribute), 22
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 module), 18

store_taskmanager() (decisionengine.framework.dataspace.dataspace.DataSource method), 15

taskmanager (class in decisionengine.framework.taskmanager.TaskManager), 29

taskmanager() (in module decisionengine.framework.dataspace.datasources.tests.test_postgresql), 15

taskmanager_table (decisionengine.framework.dataspace.datasource.DataSource attribute), 22

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_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.dataspace.DataSource), 15


```

    engine.framework.dataspace.datasources.tests.test_postgresql), 20
15
test_insert() (in module decisio- W
    engine.framework.dataspace.datasources.tests.test_postgresql), 15
test_minimal_jsonnet_right_extension() (in module decisio-
    engine.framework.configmanager.tests.test_configmanager), 13
test_minimal_jsonnet_wrong_extension() (in module decisio-
    engine.framework.configmanager.tests.test_configmanager), 14
test_minimal_python() (in module decisio-
    engine.framework.configmanager.tests.test_configmanager), 14
test_minimal_python_default() (in module decisio-
    engine.framework.configmanager.tests.test_configmanager), 14
test_program_options_default() (in module decisio-
    engine.framework.configmanager.tests.test_configmanager), 14
test_program_options_update() (in module decisio-
    engine.framework.configmanager.tests.test_configmanager), 14
test_store_taskmanager() (in module decisio-
    engine.framework.dataspace.datasources.tests.test_postgresql), 15
test_wrong_type() (in module decisio-
    engine.framework.configmanager.tests.test_configmanager), 14
Transform (class in decisio-
    engine.framework.modules.Transform), 28
transform() (decisio-
    engine.framework.modules.Transform.Transform
    method), 28
tsort() (in module decisio-
    engine.framework.util.tsort), 31

```

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 (decisio-
    engine.framework.dataspace.datablock.Metadata

```