
decisionengine

Release 0.3.12.dev670+gaeb6b974

Fermilab

Apr 07, 2022

CONTENTS

1	Release Notes	3
2	Install Decision Engine	25
3	Developer Documentation	31
4	Jenkins CI pipeline	33
5	Source code	39
6	Indices and tables	107
	Python Module Index	109
	Index	113

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 Notes

HEPCloud's Decision Engine release notes.

The latest release is the designated production release. Decision Engine will support also N-1. New feature development will happen in the development branch and go in the next (N+1) release.

1.1.1 Release 1.7.1

Patch level (bug fix) release.

Issues fixed in this release

Bugs fixed

- [DE 522](#): Decision engine log files get split between several different processes with several different versions open: ([fd1e99ce](#))

Enhancements:

- Added dummy sources ([de1536fa](#))

Full list of commits since version 1.7.0

[606e1e9f](#): Merge pull request #585 from vitodb/fix/1.7/vito_port_PR527

[538cf940](#): Merge pull request #586 from HEPCloud/goodenou-patch-remove-gc

[67febfd0](#): remove unnecessary 'import gc'

[9da797d3](#): Improve parameter and variable names.

[55a5b547](#): porting #PR515 into 1.7 (simplifications to logging in Modules) cherry-picked from commit d515741

[fd1e99ce](#): porting #PR563 into 1.7 (adding queue logging into de_logger)

[c75deef4](#): Also run tests on PRs for backports to 1.7

[de1536fa](#): add dummy source and test

1.1.2 Release 1.7.0

This release features:

- New produces-consumes structure using decorators. This will improve the code quality, improving static checks and reducing the lines of code by removing repetitive boilerplates, especially in the modules.
- Added structured logging. Improved python logging and adoption of structured logs format that will increase the semantic content of the messages and ease the export of information for dashboards and Elastic Search.
- Added SQLAlchemy object-relational mapper to increase the testability of DB interactions and to allow different database backends. Switching between datasource backends requires dropping all objects if you wish to reuse the tablespace.
- Packaging via setuptools for both decisionengine and decisionengine_modules: Dependencies are not yet fully listed in the RPMs.
- A new, optional, configuration parameter called “channel_name” is available. “channel_name” is one of the keys in the output dictionary of the structured logging and will be used in the upcoming monitoring. If the variable is not defined in the configuration file, then it is taken from the name of the file, e.g. the job_classification.jsonnet config file gives a default “channel_name” value of “job_classification”.

Note: Added requirement on SQLAlchemy (for new datasource backend). Non-SQLAlchemy users should ensure the indexes from [13c2f283](#) are in their database.

Note: Added requirement on prometheus-client. Prometheus will be used as optional monitoring component.

Note: The “channel_name” key in the Source Proxy config dictionaries needs to be changed to “source_channel”. “channel_name” is now being used to describe the name of the channel itself, not the name of the channel the Source Proxy is getting information from.

Issues fixed in this release

- [481](#): Channel name should be available to all worker types in TaskManager
- [456](#): Logic engine messages show in the main DE log (1.6.99 post4) prj_testing
- [458](#): Exception in new SQLAlchemy data source 1.6.99post4
- [455](#): New postgresql exception in 1.6.99post4 (aka Fixed database inconsistency silently ignored in v1.6)
- [456](#): Logic engine messages show in the main DE log (1.6.99 post4)
- [451](#): Transforms executed in wrong order in 1.6.99.post3
- [367](#): Test race conditions bug
- [406](#): Taskmanager doesn't use/honor global log level
- [379](#): Add postgresql.sql to distributed decisionengine rpm
- [329](#): Docker container is missing pylint
- [293](#): Drop requirements.txt setup mode
- [285](#): Unify ProcessingState with Reaper state management code

- 253: Decision engine can sometimes start up at boot time before network name resolution is working ([ae04db5](#))

Full list of commits since version 1.6.0

f42558df: Updated documentation for 1.7.0 release
029d118a: Updated release notes for 1.7.0 RC4 (1.6.99.post8)
0e19c754: fix SP
810994af: Update release_notes_1.7.rst
fbee95e7: Update release_notes_1.7.rst
68b955b0: Make sure product is a string
ef7a8b96: Automatically adjust PYTHONPATH for tests
e292d388: Updated release notes for 1.7.0 RC3 (1.6.99.post7)
d60b6e4e: new changes for logging with common logger name “channel”
8cdeb67e: Simplify return expression
8fb128d3: Ensure file is “flushed” so name is fully established
7806aa00: Add github CodeQL analysis
9f09bca9: removed modules/LogicEngine.py and corresponding test
b9d28fbf: Cleaner check for *Any*
cc91aa24: Switch to fstring formatting
7bb5b64f: Just return created value rather than store then return
f4847fbe: Combine nested *with* blocks
4ba38bcd: Drop redundant brackets
bdcfe8c9: By convention, pandas is usually imported as *pd*
1dd904ff: Use more traditional expression order
cccd31bc: Unused loop vars should start with *_*
c055a5cd: Drop *_keys* in favor of DB backed *keys*
e8c689b4: Moved prometheus-client requirement to proper place in list
5391500d: Added metrics API module
c2d7835c: Drop unnecessary timeout
c167fc50: Add tests for de-query-tool entry point
efabfeb3: Updated release notes for 1.7.0 RC2 (1.6.99.post6)
b2739c14: moved logging of LogicEngine from decisionengine logger to channel loggers
0c0532f3: Add locks to help ensure data changes are “atomic”
ae63c6ee: Use DB generated known keys so it always matches DB state
b2259e9e: Use public *.keys()* rather than internal implementation
85b6c3ba: Real world data shows the defaults are fine
95fb3fdf: Further constrain tablespace

3e8e8619: Finish implementation of get_datablock
edbb3568: Add entry point for de-query-tool
fed95c62: adding logging of importlib imports of modules
53e62f03: Sometimes pypy times out on the cleanup.
a44d4bc4: Don't test sqlite on pypy it isn't necessary
b13aa8a9: Some corrections
94c14110: Fix missing defines
5f102095: More detailed testing of datablock
b6c99021: Make sure our sqlite tests have ForeignKeyConditional support
6b76ba7c: Fix typo
6694369d: Ensure dbutils uses transactions
1df400ae: Fix spaces
5278fd99: Raise timeout for numpy on pypy
6d0a1a74: Release notes ready for v1.7.0
084f74e1: Initial SQLAlchemy Datasource
3353aa00: Make sure our jsonnet is json syntax valid
402b1c26: Fix transform-ordering problem.
49297573: Fix incorrect packaging of tests at top level
fbfae499: The test_channel loads data once per second.
33f9ade1: Rename taskmanager test nodb
308343e9: Initial modifications for addition of structured logging
6f337b75: Add missing error message
23a4b770: Call fixtures in a cleaner manner for xdist
1f2fe8c4: Add self.config so I can introspect the fixtures later
689c0020: Add missing *config* attrib test
d2732816: Best practices are for fixtures to *yield* vs *return*
acce50a: Seed SQLAlchemy fixtures for later activation
31002bc5: Help define the fixture interlocking
0f5fb129: The pandas 1.3.0 doesn't build against PyPy any longer
a7d18a41: Correctly test datablock construction paths
9af4c144: the *mock* package was a backport for python2.
5dda8f8f: Add another constructor test
9ae9ad13: Make sure if the client says to stop we don't override it
a581cd2b: run pyupgrade against codebase for python3.6
09e4e79c: Handle reaper duplicate shutdowns more cleanly
64d29dc5: Drop pointless cache restore

1c6b2588: Update PyPy to 3.7 for testing
2bae173e: Increase wait for overloaded test workers, update log messages
b67c185c: When aborting CI builds cleanup all processes
6c5d6306: Trim pytest fast functions, add required plugin
8c63ca6b: note why we're ignoring this line
2bd4ecbc: Add a syntax check for the toml files
e2dca404: Sometimes these get stuck
6d012fab: Add in Jenkinsfile pipeline configuration a timeout at stage level
baf07973: Add timeout option to block-while/until
970faf92: Make pre-commit happy
0cea2285: Fix alignment issue
5620c65b: List why we aren't checking
88611d90: Ensure fixtures are cleaned up between invocations
0ba135d2: Setup blank DB for SQLAlchemy tests and prep fixtures
3793e674: Setup pre-commit
9e6d1317: Migrate test_Reaper to pytest fixtures
51df43bf: Cleanup a bunch of pointless whitespace
96e5d069: Fix typo
9f96f418: Setup datablock to use our paramaterized fixture
36ebc66c: Add config for LGTM
c6032e5f: Use topologically sorted transforms to remove some multi-threading.
e063f82a: Drop pointless comma
bfd6689e: Begin prepwork for PEP517
72c5725f: Stub out null source rather than more complex mocking
3b65e5e2: Push Singleton into its own space
fb5b177e: Put fixtures in central location
5ab3cbaa: Add more details to channel startup logs
afe7f7d7: Add log about what DB we are hitting
38034b2c: Let the datasource handle the connections internally
5e03b6fe: Since we are opening an IPv4 socket, just use 127.0.0.1 to check
cac2bef3: Fix missing version requirements
3be8f84f: Add line lenght for autoformater
90e2baad: Protect against inappropriate wait under error condition.
943a17a7: Fix de-client typo and adjust tests accordingly.
3b104eba: Set the logs to DEBUG for testing
4c5564d4: Add another sync method to try and make tests less spotty

66bd81f2: Make sure to encourage updates to tools
d16f04cc: Put postgresql datasource schema into RPM
62b97e79: Fix __str__ so it includes all the data
611ef1f8: Drop pointless lines
5b9e2fb6: Drop unreachable excepts
6991f65f: Restore product-name translation required for some source-proxy cases.
f6258c09: Fixed formatting and updated content
104a0446: Update index.rst
2ed61289: Update index.rst
cb687150: Create release_notes.rst
3b57d4a2: Note new requirement
871af08b: Added 1.7.0 release notes
ce42b802: improved 1.6 release note
583c10fb: fixed rst error
96d4dc1e: Added 1.6.2 release notes, from branch 1.6
13c2f283: Add some helpful indexes to our default schema
29c32571: Log as workers are started
619021c2: One of these tests seems to be spotty, break them out to find which one
29a2c72d: Run the test in a way that gives us colors
4e36bfd2: Drop unused table create logic
5511f69e: Stronger notify state for when we've a lot of watchers.
b6cc7a46: Test the dataspace abstractions
e3b1f594: Better messages about our state
2d2feab9: Drop duplicate tests, leave specifics
8e737329: Add parameter based datasource api tests
5c023aa5: Don't do debug logs for flake8, they aren't helpful
f5d1a12f: Setup list of public exports for dataspace.py
7158b422: Merge pull request #365 from jcpunk/bad-update-is-error
cd98cc4a: Update should error out if you try to do it wrongly
eb7907fe: Add option to set taskmanager datestamp and sample usage
e124532c: Make sure the fixture uses the production flow
a8241b6e: Make sure RPM also owns the .egg-info so we don't confuse the namespaces
da87376e: Ensure the DE server is fully started before running query
622bfac: Simplify use of our PG fixtures
df98ecdf: Fixed flake8 issue
061ff6cf: decisionengine/framework: stop_channel runs Publisher shutdown methods

3727b80b: Fixup comment to avoid assuming this test uses the DB
d45aaf6b: Fix script path typo
a25a4a30: Fix ABC to match our actual usage
1510b2d1: Address minor linting issues
945e4b16: Fix missing attribute insert
5eace9d5: Add note for how to get modules in place
50a8e268: Add list of packages in the CI env to output
b9cb197d: Sanity check the home directory
cd17223c: Have client provide a hint when you ask for no behavior
95b02365: Fix de-query-tool to support produce/consume model
e660ca72: Update required versions for bugfixes
6863cb81: Fix path error
bb52e8b1: Merge pull request #340 from jcpunk/service-stop
6d7aba95: Drop obsolete files
168ae7aa: Name the tests better
0f60c4e3: Support new produces/consumes/configuration-description infrastructure.
81912469: Add de-query-tool
2a26c944: ExecStopPre is not supported on all systemd instances
67a54d5c: Merge pull request #338 from jcpunk/fix-pytest-postgres
70ab133f: Fixup use of pytest_postgresql for version 3.0.0
f8f4255e: Merge pull request #337 from jcpunk/thread-names
5f49a4f6: Set names for the various parallel code
64da77c6: Merge pull request #327 from jcpunk/datablock-expire
de33a60a: Merge pull request #336 from knoepfel/use-toposort
31a8a905: Merge pull request #328 from knoepfel/de-class-inference
410e383d: Merge pull request #331 from jcpunk/reaper-interval-tests
719ff0c8: Test datablock expire funtions
e14c49d8: The 'name' parameter is optional.
7846c9f3: Enable DE class inference based on configuration.
32ab7e44: Use third-party topological sort.
01aa8ae6: Merge pull request #325 from jcpunk/channel-tests
52b48479: Merge pull request #326 from jcpunk/valid-config-tests
8c4749e7: Merge pull request #330 from jcpunk/pylint-actions
a37770c9: Ensure validation testing is tested
d8ab5eb6: Add missing test to ensure the run interval is actually used
0cd9c42b: Also run pylint for extra sanity checks

c5cf1fff: Ensure our errors error out
baf01700: Merge pull request #324 from jcpunk/cleanup-trivial-tests
2a0133aa: Try to cleanup trivial missing coverage
44e0ad6f: Merge pull request #323 from jcpunk/about-coverage
d811f617: Merge pull request #322 from knoepfel/fix-fail-on-error
cb426262: Merge pull request #312 from jcpunk/finish-setuptools
8f6d407d: Merge pull request #316 from jcpunk/abc-coverage
4d0676bb: Merge pull request #317 from vitodb/pylint
d7c43b96: Use regular expression to support fail_on_error feature.
ada66925: add support to run pylint tests
efb1e57b: Finish migration to pure setuptools
bc4720cf: We aren't testing 'unversioned' releases
e4dc35e3: Merge pull request #314 from jcpunk/jsonnet_syntax
87e32c22: Merge pull request #294 from jcpunk/move-reaper
dec85d5e: Merge pull request #319 from jcpunk/task-loop
4108472a: Merge pull request #320 from jcpunk/container-swig
920af1c9: Merge pull request #321 from knoepfel/include-init-files
650dfa7: Don't forget __init__.py files.
1b412e03: The latest m2crypto seems to need swig now
a6e3ab1c: Merge pull request #313 from jcpunk/conf-test
1205636a: Simplify run loop
30e59dc9: fix test_client_with_no_server_verbose unit test for Jenkins CI (#315)
10384a8c: Move reaper into its own place and reuse state logic
940584e4: No real way to test abstract base classes
250c14b1: The *_validate* function doesn't permit missing 'PRODUCES'
5ae1ce9f: Make sure syntax error in config names the problem
b899fa23: Add SourceProxy module test. (#307)
7b3df14c: Increase coverage of utils (#304)
ddba2a31: Fix duplicate entry warning (#311)
915673fa: Test modules minimally (#298)
bc0c21a9: Some repos may error out, don't let them kill the build (#297)
924a7047: doc: add 1.6.1 release notes
b1ab4d31: doc: fix typo
85e5d714: postgresql: do not print stack trace for low level library (#309)
255c6415: Setuptools uses entry return value as an error msg (#303)
2fd8db45: Fix name to match expectations (#305)

9cddb70a: updated release notes

7fe0358e: Error in more clean methods (#300)

84aa506c: Fix a bug in setup.py parsing of requirements. (#301)

a58b61bb: fix typo in release notes

1.1.3 Release 1.6.2

Patch level (bug fix) release.

Issues fixed in this release

Bugs fixed

- DEM 200 (part of it): Invoke correctly channels shutdown: (75eaa90)
- no issue: Use regular expression to support fail_on_error feature (1386d20)

Enhancements:

- Improved CI support (e.g. added pylint tests)
- 217: Add option to de-client --print-product to only print the column names in a data block and-or to print one or more records in key/value format. (c4c7681)

Full list of commits since version 1.6.1

c4c7681: Updated de-query-tool w/ cherry pick of fixes from latest version of PR#332

f964d4b: Fixup use of pytest_postgresql for version 3.0.0

635ffd1: Also run pylint for extra sanity checks

11676ff: Fixed function w/ the same name

b8278f6: Add de-query-tool

75eaa90: Merge pull request #335 from shreyb/publisher_shutdown_from_1.6

77e3d79: Added set_to_shutdown method to TaskManager and accompanying test

1386d20: Merge branch 'knoepfel-fix-fail-on-error' into 1.6

73a18b1: Merge branch 'fix-fail-on-error' of <https://github.com/knoepfel/decisionengine> into knoepfel-fix-fail-on-error

4f49fb7: Merge branch 'jcpunk-finish-setuptools' into 1.6

a5e5d39: Merge branch 'finish-setuptools' of <https://github.com/jcpunk/decisionengine> into jcpunk-finish-setuptools

a1ed252: Merge branch 'vitodb-pylint' into 1.6

c8eddda: Merge branch 'pylint' of <https://github.com/vitodb/decisionengine> into vitodb-pylint Meerging PR#317 to release branch 1.6

d7c43b9: Use regular expression to support fail_on_error feature.

ada6692: add support to run pylint tests

efb1e57: Finish migration to pure setuptools

e4dc35e: Merge pull request #314 from jcpunk/jsonnet_syntax

87e32c2: Merge pull request #294 from jcpunk/move-reaper
dec85d5: Merge pull request #319 from jcpunk/task-loop
4108472: Merge pull request #320 from jcpunk/container-swig
920af1c: Merge pull request #321 from knoepfel/include-init-files
650dffa: Don't forget __init__.py files.
1b412e0: The latest m2crypto seems to need swig now
a6e3ab1: Merge pull request #313 from jcpunk/conf-test
1205636: Simplify run loop
de553a7: fix test_client_with_no_server_verbose unit test for Jenkins CI (#315)
30e59dc: fix test_client_with_no_server_verbose unit test for Jenkins CI (#315)
10384a8: Move reaper into its own place and reuse state logic
250c14b: The `_validate` function doesn't permit missing 'PRODUCES'
5ae1ce9: Make sure syntax error in config names the problem
b899fa2: Add SourceProxy module test. (#307)
7b3df14: Increase coverage of utils (#304)
ddba2a3: Fix duplicate entry warning (#311)
915673f: Test modules minimally (#298)
bc0c21a: Some repos may error out, don't let them kill the build (#297)
924a704: doc: add 1.6.1 release notes
b1ab4d3: doc: fix typo
85e5d71: postgresql: do not print stack trace for low level library (#309)
255c641: Setuptools uses entry return value as an error msg (#303)
2fd8db4: Fix name to match expectations (#305)
9cddb70: updated release notes
7fe0358: Error in more clean methods (#300)
84aa506: Fix a bug in setup.py parsing of requirements. (#301)
a58b61b: fix typo in release notes
33660bf: fixed a typo[locuser@fermicloud462 decisionengine]

1.1.4 Release 1.6.1

Patch level (bug fix) release.

Issues fixed in this release

- [306](#) : /etc/decisionengine/decision_engine.conf as shipped in RPM is wrong format ([de0aef3](#))
- [275](#) : Running de-client --stop-channel <channel> results in KeyError ([59fb44e](#))

Full list of commits since version 1.6.0

[d7ccd8a](#) : doc: fix typo

[ac48e50](#) : updated release notes

[de0aef3](#) : Fix name to match expectations (#305)

[59fb44e](#) : postgresql: do not print stack trace for low level library (#309) (#310)

[2162bbe](#) : Setuptools uses entry return value as an error msg (#308)

[b0fd9fb](#) : 1.6.0 package backports (#302)

1.1.5 Release 1.6.0

In this release:

- The logic engine has been rewritten in pure python. This removes the last C++ dependency the decision engine had. The build system has been updated accordingly.
- Migrated to setuptools package development library. This build system is the standard vanilla python build system provided with the python distribution. Build configurations have been updated and rpm packaging remains the primary distribution method.
- Completed logging implementation.
- Improvements in error handling and code coverage.
- Improvements in Jenkins and GitHub actions CI/CD pipelines.

Issues fixed in this release

- [44](#) : Logic Engine doesn't handle missing values gracefully ([743effc](#))
- [253](#) : Decision engine can sometimes start up at boot time before network name resolution is working ([ae04db5](#))

Full list of commits since version 1.5.0

[2551e07](#) : More coverage for de-client (#296)

[dde3945](#) : Make sure actions either complete in time or die (#295)

[381861c](#) : Update Jenkins pipeline configuration (#292)

[eb771f4](#) : Try to cleanup Dockerfile PATH issue (#291)

[780cb56](#) : fix unittest doc

[8680942](#) : update unittest documentation

[8154b24](#) : Fixup sphinx doc (#290)

[5f7e13a](#) : enhancements in logging and error handling in dataspace dir (#283)

3d92725 : Add missing runtime requirement (#286)
743effc : Allow conversion from errors to false values in logic-engine expressions. (#284)
124dcab : Inherit version from setuptools_scm if possible (#287)
3669803 : added missing "" as line continuation
761f1d9 : Drop invalid **init.py**
dc0e71b : migrate to setuptools (#264)
3b6f1bf : Make reaper reset state when starting from stopped proc (#280)
b2f9061 : added ISO-8601 format to time in logging. changed name of function for better clarity. (#279)
0a74fe1 : Improved DE client usage (#281)
ebf53e3 : Added shutdown method to Publisher class (#278)
f95ab6d : Address some flake8/black reports (#274)
1c383b7 : Automatically pull in our settings from about.py (#273)
e71f186 : logging and error handling enhancements to taskmanager directory (#277)
7de9ab9 : Increase Reaper log verbosity (#267)
019d245 : Update actions to follow new best practices (#272)
b84e847 : Avoid possible sync issues in reaper startup (#271)
891975f : Remove vestigial C++ files. (#270)
42e5e1f : enhancements in logging and exception handling in newly added logicengine files (#265)
38effe6 : Ensure the scheduler has started the thread before returning (#269)
db54fa1 : Start testing on PyPy with psycpg2cffi (#223)
cc44058 : Squashed commit of the following: (#263)
d6548e9 : Enhanced logging in the logicengine directory files (#261)
c341bf7 : Better match our workflow with codecov (#260)
1fbe44d : Use 'new' syntax for forward compat (#259)
2294b0b : Do a limited pin on version requirements (#256)
bcd470 : Python implementation of logic engine (#246)
c6721b4 : address comment on RB
ae04db5 : Add Wants and After (network-online.target) dependency
1a96b14 : Fix action repodata
a70cee8 : Move to CodeCov.io
7b16b4e : Add Wants and Requires dependencies (#258)
76c3670 : Move to CodeCov.io (#254)
e7ba013 : Fix action repodata (#255)
d7e72f2 : revert 3.9 test
b04154b : added 1.5.0 release notes
a03da29 : remove 3.9 to see if documentatoin gets generated

1.1.6 Release 1.5.0

In this release:

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

Issues fixed in this release

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

Full list of commits since version 1.4.1

[d66c54b](#) : Add PEP-0396 metadata (#243)

[bfc91a6](#) : More compat between psycpg2/psycpg2cffi (#248)

[f5d31a6](#) : Cleanup Fixture FIXME (#249)

[0dfaf3c](#) : Adding docker documentation (#251)

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

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

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

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

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

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

[51370fb](#) : Resolve fixture 'quickstart' issue (#238)

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

[fe9faa9](#) : implement data product browsing interface (#239)

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

[d91722f](#) : Compat with psycpg2cffi (#233)

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

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

[6e6f4c9](#) : Updated Jenkins configuration documentation (#229)

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

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

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

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

b059696 : Update workflow per changes at github (#225)
7a71cac : Use newer compilers/runtimes (#224)
15ffd93 : Add header for strict includes (#222)
71b141a : Add special PyPy only requirement (#221)
9dbb932 : Move Python C extension to versioned .so file (#220)
ea7ade5 : Migrate from boost-python to pybind11 (#215)
e6b2eae : Add python 3.9 to testing matrix (#219)
04c8f9c : Add the option to print columns types on de-client (#216)
8815dc6 : Logic-engine cleanups (#211)
086d0d5 : fix missing back tick
54cc084 : modified release notes
24744cf : Synchronize access to the task managers (#214)
87a7fda : replde dash with underscore
743d0fd : try sphinx_rtd_theme
18c7909 : added 1.4.0 release notes
ff3d491 : force docker pull when building the docker container to make sure to use an updated base layer (#210)

1.1.7 Release 1.4.1

In this release:

- Bug fixes to 1.4.0 release

Issues fixed in this release

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

Full list of commits since version 1.4.0

9799b9a : update release version to 1.4.1
84ecfe2 : Synchronize access to the task managers (#214)
751b6b8 : Address data races; remove need to sleep in unit tests (#205)

1.1.8 Release 1.4.0

In this release:

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

Issues fixed in this release

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

Full list of commits since version 1.3

84eccf3 : Fix typo in reaper script. (#209)
d836abf : next RC
926944a : Fix coveralls reporting (#198)
b95c323 : Updating base Dockerfile (#199)
d302e31 : Help jsonnet, which doesn't understand PosixPath objects. (#204)
2d791a7 : Test configuration policies. (#197)
236e27a : Ensure items are returned in a stable order (#202)
e974f5f : add pylint and pycodestyle (#203)
fbe7616 : Test task manager (#196)
686ca80 : require more recent version of pytest-postgresql (#195)
99d20a5 : Fix double-logging problem. (#192)
4ce3d17 : A set of fixtures to simplify unit tests (#183)
65f8052 : Fix typo (#190)
f3a4be8 : Protect against None workers (#187)
ec310fb : remove py3 from package name
7006489 : bump version to 1.4.0rc
158d835 : decisionengine/framework/modules: Fix SourceProxy retries (#184)
1356bf1 : Add support to test any branch in Jenkins (#182)
692fa8e : Add timeout support for unit test on Jenkins (#181)
e3d6e6a : Updated Jenkins documentation to take into account unit tests timeout parametr (#180)
2586a3e : Configuration redesign (#168)
fac984d : Fix error with DBUtils import. Looks like names of modules changed (#175)
7d661ee : Move postgres-specific implementation to postgres source. (#174)
eb85c97 : Rpm (#173)
10fe843 : Adding log rotation by time (#170)
a8d239b : Various improvements. (#167)
d9b92ee : Ignore vim's *.swp files (#166)
d9f72ef : Fix call to shutdown_timeout (and add sample entry to config) (#165)
3161795 : Add drops for items using tables being dropped (#164)

77d186d : Show output of test runtimes in travis (#163)
81820a4 : Allow server to start with no channels. (#161)
49879a6 : DE server and client usability improvements (#160)
de91c4f : Add tests to default and override config (#158)
14df1f6 : Use python fallback for options (#159)
ac64a92 : Drop python 2.7 integration tests since we are python3 only (#157)
d963301 : Update Jenkins pipeline to properly test closing PR (#156)
64248cb : Merge 'runtime' tests into running channel tests (#150)
065ad77 : Adding Jenkins pipeline documentation (#155)
18a950c : fix print-product to report non-existing product as such (#154)
6493735 : Fix invalid attribute name (#152)
d953c6a : Remove unnecessary set_start_method call (#149)
c8c9b65 : guarantee that process is killed so test never hang (#147)
f1542b6 : Channel test (#146)
7f349a8 : Fix faulty TaskManager state type (#145)
d50f1c4 : fix logging regression introduced in f5e299969e0611e3480e9fa2782052df... (#142)
becfa26 : Pass the correct type. (#144)
1a60daf : DecisionEngine: fix typo (#143)
9e7b867 : Updating Jenkins pipeline configuration (#140)
e3a6703 : fix regression introduced in f5e299969e0611e3480e9fa2782052df86d7c4ed (#141)
4900bc6 : Restore runtime test. (#139)
0823f3d : Consolidate DE server/client tests into one file. (#138)
4f84435 : A few more access fixes.
160cfd1 : Fix task manager state access.
c00d819 : A few more cleanups.
ec087e2 : Various cleanups
a309ffe : Improvements to DE client CLI.

1.1.9 Release 1.3.0

In this release:

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

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)

1.1.10 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

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 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)

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...](https://github.com/HEPCloud/decisionengine_modules/issues/232) (#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)

1.1.11 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 omission.

24e0795 : Apply clang-format

17c17f9 : Remove JSON dependency.

faa0b22 : Massive cleanup.

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

fef6c11 : Fix errors when running pylint.sh multiple times

da6f077 : Autopep8 -i fixes

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

17396da : logicengine: get rid of compiler warnings

01dc3d1 : Only track what we need

b609d73 : Configure coveralls (and some minor cleanup)

bd9ed5e : Many C++ cleanups

2a61876 : Add Badges

c864f27 : Do not call pytest fixtures directly.

307db5f : white space fix

882b58f : fix unit tests

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

5a6e6b1 : Run tests on push

8404245 : Add missing Boost regex library dependency.

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

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

8a8f560 : Cache venv directory instead

ad017ce : Build private boost for testing

928c64a : Test pip cache

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

9f0ddb3 : Add pylint github action.

5e6ce4a : Remove more unused C++ files.

63717fe : Setup travis to use new cmake var

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

843f30c : Minor cleanups per travis-lint

a538cac : Remove unused C++ files.

4c9d125 : Update repo where action is taken from

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

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

4c79d2c : Remove unused GNUmakefiles.

94342ee : Add unit test as a Github Action

1a0e102 : more advanced travis.yml

0be413f : Add helper file for pip

7794327 : Make recursive import happy

7005c78 : Add simple target

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

2662e6c : note required packages

3b87119 : Add missing header includes.

3e79b84 : Remove defunct code and its tests

b1dbe1a : Ensure attribs are defined at **init**

c4ad78a : Correct logger arguments do avoid duplicate string parse

a8dcc67 : Remove unused imports (per pylint)

d3502b5 : Remove obsolete CVS directories.

d744111 : add six module to the list of required modules

0a9b1e8 : Fix class declaration

b83157e : Handle metaclasses

549f33b : Add config for Travis CI

ee71044 : Drop trailing white space

3f82af6 : Python3 forward compatible syntax

28bf291 : Add safe (for python 2.7) python3 compatible syntax

1d1d76f : prepare for python3

INSTALL DECISION ENGINE

Here are instructions for operators and developers to install the Decision Engine using the distributed RPM packages.

2.1 Installing and running HEPCloud's Decision Engine

Decision engine uses a PostgreSQL database back-end and Redis as message broker and cache.

You need to install first PostgreSQL, Redis, and then the Decision engine framework (decisionengine) and install and add the standard channels (decisionengine_modules).

The following instructions assume a system installation, performed as root. decisionengine will run as the decisionengine user.

2.1.1 Install PostgreSQL

The default postgresql installed on RH7 is 9.2 which is outdated. Suggest to remove it and install 12 instead :

1. Remove old postgresql

```
yum erase -y postgresql*
```

2. Install postgresql 12

```
yum install -y https://download.postgresql.org/pub/repos/yum/repopms/EL-7-x86_64/  
↪pgdg-redhat-repo-latest.noarch.rpm  
yum install -y postgresql12 postgresql12-server  
# optional, also: postgresql11-devel
```

3. Enable postgresql

```
systemctl enable postgresql-12
```

4. Init the database

```
/usr/pgsql-12/bin/postgresql-12-setup initdb
```

5. edit /var/lib/pgsql/12/data/pg_hba.conf like the following:

```
[root@fermicloud371 ~]# diff /var/lib/pgsql/12/data/pg_hba.conf~ /var/lib/pgsql/12/  
↪data/pg_hba.conf  
80c80
```

(continues on next page)

(continued from previous page)

```

< local  all          all          peer
---
> local  all          all          trust
82c82
< host   all          all          127.0.0.1/32  ident
---
> host   all          all          127.0.0.1/32  trust
84c84
< host   all          all          ::1/128      ident
---
> host   all          all          ::1/128      trust

```

This is setting the authentication method to *trust*

6. start the database

```
systemctl start postgresql-12
```

7. create decisionengine

```
createdb -U postgres decisionengine
```

The schema and the connection will be created and configured during the Decision engine framework installation.

To use the database you have to add it to the environment:

```
export PG_VERSION=12
export PATH="/usr/pgsql-${PG_VERSION}/bin:~/.local/bin:$PATH"
```

2.1.2 Install Redis

Install and start the message broker (Redis) as explained in the redis document

2.1.3 Install Decision Engine and the standard modules

1. Prerequisites setup. Make sure that the required yum repositories and some required packages (python3, gcc, ...) are installed and up to date.

```

yum install -y http://ftp.scientificlinux.org/linux/scientific/7x/repos/x86_64/yum-
↳ conf-softwarecollections-2.0-1.el7.noarch.rpm
yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.
↳ rpm
# gcc, swig and make are needed for dependencies (jsonnet)
yum -y install python3 python3-pip python3-setuptools python3-wheel \
    gcc gcc-c++ make \
    python3-devel swig openssl-devel git rpm-build
python3 -m pip install --upgrade --user pip
python3 -m pip install --upgrade --user setuptools wheel setuptools-scm[toml]

# To install the modules you will also need GlideinWMS Frontend, which is in the
↳ OSG repository.
# Assuming the use of OSG 3.5 that supports both GSI and tokens, here is a brief
↳ summary of the setup:

```

(continues on next page)

(continued from previous page)

```

yum install -y yum-priorities
yum install -y https://repo.opensciencegrid.org/osg/3.5/osg-3.5-el7-release-latest.
↳rpm
# HTCondor 8.9.x or 9.x, required by GlideinWMS, is in the osg-upcoming repository.↳
↳It should be enabled to find the dependency
# GlideinWMS 3.9.x is in osg-contrib. The repository should be enabled to find the↳
↳dependency
# In both the following files set: enabled=1
vi /etc/yum.repos.d/osg-upcoming.repo
vi /etc/yum.repos.d/osg-contrib.repo
# Change the Epel repository priority to make sure that comes after the OSG↳
↳repositories, which are 98. Make sure that epel has:
priority=99
vi /etc/yum.repos.d/epel.repo

```

The complete version of the GlideinWMS installation instructions is available [here](https://opensciencegrid.org/docs/other/install-gwms-frontend/)<<https://opensciencegrid.org/docs/other/install-gwms-frontend/>>

2. Setup the decision engine yum repositories

```

wget -O /etc/yum.repos.d/ssi-hepcloud.repo http://ssi-rpm.fnal.gov/hep/ssi-hepcloud.
↳repo
wget -O /etc/yum.repos.d/ssi-hepcloud-dev.repo http://ssi-rpm.fnal.gov/hep/ssi-
↳hepcloud-dev.repo

```

3. Install the decision engine (add --enablerepo=ssi-hepcloud-dev for the latest development version)

```

yum install decisionengine
yum install decisionengine_modules
# The modules RPM was renamed. For versions <1.7 use instead
yum install decisionengine-standard-library

```

4. Not all packages are available as RPM. It is necessary to install directly some Python dependencies. To avoid to pollute the system Python we will install them for the decisionengine user, the user the service is running as. Install the required Python packages (these are taken from setup.py)

```

su decisionengine -s /bin/bash
python3 -m pip install --upgrade pip setuptools wheel --user
python3 /path/to/decisionengine/setup.py develop --user
python3 /path/to/decisionengine/setup.py develop --user --uninstall
python3 /path/to/decisionengine_modules/setup.py develop --user
python3 /path/to/decisionengine_modules/setup.py develop --user --uninstall
exit

```

The commands above should be sufficient. Anyway, here is an explicit list you can use in alternative:

```

su decisionengine -s /bin/bash
# from decisionengine setup.py
python3 -m pip install --user jsonnet tabulate toposort structlog
python3 -m pip install --user wheel DBUtils sqlalchemy
python3 -m pip install --user pandas==1.1.5 numpy==1.19.5
python3 -m pip install --user "psycogp2-binary >= 2.8.6; platform_python_
↳implementation == 'CPython'"

```

(continues on next page)

(continued from previous page)

```
python3 -m pip install --user "psycpg2cffi >= 2.9.0; platform_python_
↳implementation == 'PyPy'"
# from decisionengine_modules setup.py
python3 -m pip install --user boto3 google_auth google-api-python-client
python3 -m pip install --user gcs-oauth2-boto-plugin
# Condor should be already there from the RPM, if not add: python3 -m pip install_
↳htcondor
# bill-calculator-hep is only for versions >= 1.7
python3 -m pip install --user bill-calculator-hep

# The following are additional requirements for v1.6 and earlier
python3 -m pip install --user boto packaging
# This is not in pypi
python3 -m pip install --user https://test-files.pythonhosted.org/packages/f4/a5/
↳17a14b4ef85bc412a0ddb771771de3f562430328b0d83da6091a4131bb26/bill_calculator_hep_
↳mapsacosta-0.0.10-py3-none-any.whl

exit
```

Now you can type `decisionengine --help` to print the help message. To do more you need first to configure Decision Engine.

2.1.4 Configure Decision Engine

The default configuration file lives in `/etc/decisionengine/decision_engine.jsonnet`.

A number of defaults are set for you.

Selecting your datasource

You need a datasource to store in the database the channel's data (datablocks). Each datasource has its own unique schema and cannot be used with a different datasource.

The SQLAlchemy Data Source

SQLAlchemy is the default Data Source after v1.7 and is setup with a configuration like:

```
"datasource": {
  "module": "decisionengine.framework.dataspace.datasources.sqlalchemy_ds",
  "name": "SQLAlchemyDS",
  "config": {
    "url": "postgresql://{db_user}:{db_password}@{db_host}:{db_port}/{db_dbname}",
  }
}
```

Any extra keywords you can pass to the `sqlalchemy.engine.Engine` constructor may be set under `config`.

SQLAlchemy will create any tablespace objects it requires automatically.

The PostgreSQL Data Source

The postgresql Data Source is the only one supported pre v1.7 and is setup with a config like:


```
"datasource": {
  "module": "decisionengine.framework.dataspace.datasources.postgresql",
  "name": "Postgresql",
  "config": {
    "user": "postgres",
    "blocking": true,
    "host": "localhost",
    "port": 5432,
    "database": "decisionengine",
    "maxconnections": 100,
    "maxcached": 10
  }
}
```

If you use this datasource you must also load the database schema by hand. To load the database schema run:

```
psql -U postgres decisionengine -f /usr/share/doc/decisionengine/datasources/postgresql.
↪ sql
```

2.1.5 Start decision engine

Start the service

```
systemctl start decisionengine
```

2.1.6 Add channels to decision engine

Decision engine decision cycles happen in channels. You can add channels by adding configuration files in `/etc/decisionengine/config.d/` and restarting the decision engine.

Here is a simple test channel configuration. This test channel is using some NOP classes currently defined in the unit tests and not distributed. First, copy these classes from the Git repository:

```
cd YOUR_decisionengine_REPO
# OR download the files from GitHub
mkdir /tmp/derepo
cd /tmp/derepo
wget https://github.com/HEPCloud/decisionengine/archive/refs/heads/master.zip
unzip master.zip
cd decisionengine-master
# Now copy the files
cp -r src/decisionengine/framework/tests /lib/python3.6/site-packages/decisionengine/
↪ framework/
```

Then, add the channel by placing this in `/etc/decisionengine/config.d/test_channel.jsonnet`:

```
{
  sources: {
    source1: {
      module: "decisionengine.framework.tests.SourceNOP",
      parameters: {},
    }
  }
}
```

(continues on next page)

(continued from previous page)

```
    schedule: 1,
  },
},
transforms: {
  transform1: {
    module: "decisionengine.framework.tests.TransformNOP",
    parameters: {},
    schedule: 1
  }
},
logicengines: {
  le1: {
    module: "decisionengine.framework.logicengine.LogicEngine",
    parameters: {
      facts: {
        pass_all: "True"
      },
      rules: {
        r1: {
          expression: 'pass_all',
          actions: ['publisher1']
        }
      }
    }
  }
},
publishers: {
  publisher1: {
    module: "decisionengine.framework.tests.PublisherNOP",
    parameters: {}
  }
}
}
```

Finally, restart decision engine to start the new channel:

```
systemctl restart decisionengine
```

de-client --status should show the active test channel

DEVELOPER DOCUMENTATION

The developer documentation is in the [GitHub Wiki](#)

Instructions to build the package, or to run unit tests and other CI tests, and to install decisionengine are in the [GitHub Wiki](#) as well.

JENKINS CI PIPELINE

4.1 Decisionengine CI with Jenkins pipeline

Jenkins dashboard with Decisionengine framework CI results is available [here](#).

A CI build is triggered any time a PR is created/closed or a commit is made to an existing PR. There are also *nightly CI builds* to test a list of predefined branches.

The Jenkins pipeline runs *pylint* and *unit_tests* test suites alongside the *rpmbuild* stage.

The Jenkins dashboard looks like this:

Jenkins > CI > decisionengine_pipeline >

[Back to Dashboard](#)
[Status](#)
[Changes](#)
[Build with Parameters](#)
[Delete Pipeline](#)
[Configure](#)
[Full Stage View](#)
[GitHub](#)
[Job Config History](#)
[Rename](#)
[GitHub PR](#)
[Pipeline Syntax](#)
[GitHub PR Polling Log](#)
[Set Next Build Number](#)

Pipeline decisionengine_pipeline

DE pipeline

Last Successful Artifacts

mail.results	2.04 KB	view
pep8.merge150.log	0 B	view
pylint.merge150.log	0 B	view
pytest.log	7.89 KB	view
results.merge150.log	5.16 KB	view
rpmbuild.tar	1.37 MB	view

Recent Changes

Stage View

Average stage times:
(Average full run time: ~27min 25s)

	Declarative: Checkout SCM	DE tests	pylint	unit_tests	rpmbuild
	1s	51ms	15min 17s	13min 24s	9min 44s
319#PR#150 Sep 02 17:00 No Changes	602ms	49ms	12min 45s	13min 49s failed	6min 20s
318#PR#150 Sep 02 16:36 No Changes	617ms	49ms	27min 5s	23min 43s	19min 15s
317#PR#150 Sep 02 16:25 No Changes	1s	54ms	21min 14s	20min 1s	15min 5s
316#PR#150 Sep 02 16:23 No Changes	1s	54ms	21min 4s	18min 41s	14min 36s
315#PR#150 Sep 02 16:16 No Changes	2s	57ms	22min 32s	19min 24s	14min 31s
314#PR#150 Sep 02 16:10 No Changes	921ms	55ms	16min 43s	14min 16s	7min 23s
313#PR#150 Sep 02 16:08 4 commits	789ms	42ms	16min 56s	14min 11s	8min 17s

Build History [trend](#)

find

- 319#PR#150 Sep 2, 2020 5:00 PM [#150](#)
- 318#PR#150 Sep 2, 2020 4:36 PM [#150](#)
- 317#PR#150 Sep 2, 2020 4:25 PM [#150](#)
- 316#PR#150 Sep 2, 2020 4:23 PM [#150](#)
- 315#PR#150 Sep 2, 2020 4:16 PM [#150](#)
- 314#PR#150 Sep 2, 2020 4:10 PM [#150](#)
- 313#PR#150 Sep 2, 2020 4:08 PM [#150](#)
- 312#PR#149 Sep 2, 2020 3:38 PM [#149](#)
- 311#PR#149 Sep 2, 2020 3:31 PM [#149](#)
- 310#PR#149 Sep 2, 2020 3:23 PM [#149](#)
- 309#PR#147 Sep 2, 2020 12:42 PM [#147](#)

On the bottom left side there is the list of recent CI builds that are named after the PR or the branch tested.

On the bottom right side the dashboard shows for each CI build detailed status for each test suite.

Hovering the mouse over the *status box* for each CI build stage, a tool-tip with a button to access log details shows up.

Next to the build number the symbol  gives access to a menu with the list of artifacts stored for that build. Those artifacts include logs and the tarball with RPMs.

From the panel on the left side it is possible to access the PR on GitHub by clicking on the PR icon that looks like this  [#142](#).

On occasion it could be useful to trigger a manual CI build to test a branch on the official DE GitHub repository or on the user fork. For this purpose, on the top left panel the user can click on the  **Build with Parameters** button, and this panel shows up

Pipeline decisionengine_pipeline

This build requires parameters:

DOCKER_IMAGE	<input type="text" value="vitodb/decision-engine-ci:jenkins"/>
	Docker image name to use. Default is: vitodb/decision-engine-ci:jenkins
DE_REPO	<input type="text" value="https://github.com/HEPCloud/decisionengine/"/>
	Decisionengine repo. Default is: https://github.com/HEPCloud/decisionengine/
BRANCH	<input type="text" value="master"/>
	Branch to test. Default is: master
PYTEST_TIMEOUT	<input type="text" value="300"/>
	Timeout in seconds for unit_tests (it applies to individual unit test) Default is: 300

Build

the user can modify these parameters to customize what code to test with the CI build.

The *DE_REPO* parameter can point to the user fork or to the main repository.

The *BRANCH* parameter can point to the desired branch to test.

The *PYTEST_TIMEOUT* parameter is the timeout in seconds for *unit_tests*.

When ready, by clicking on the *Build* button, the CI build will start.

The [pipeline configuration](#) is part of the decisionengine repo.

4.1.1 Nightly CI build configuration

The nightly CI build for Decisionengine framework uses this [Jenkins project](#) that triggers a CI build using the Jenkins pipeline described above to test a list of predefined branches.

Jenkins ▸ CI ▸ decisionengine_ci ▸

[Back to Dashboard](#)

Status

[Changes](#)

[Workspace](#)

[Build Now](#)

[Configure](#)

[Delete Multi-configuration project](#)

[Rebuild Last](#)

[Job Config History](#)

[Rename](#)

[Set Next Build Number](#)

Build History [trend](#) ^

find

#295	Nov 19, 2020 2:23 AM
#294	Nov 18, 2020 2:23 AM
#293	Nov 17, 2020 2:23 AM

Project decisionengine_ci

Decision Engine CI running inside dedicated docker container

Configurations

[BRANCH=master](#) [BRANCH=1.4](#)

Subprojects

Static

- [decisionengine_modules_pipeline\(non-blocking\)](#)
- [decisionengine_pipeline\(non-blocking\)](#)

Permalinks

- [Last build \(#295\), 7 hr 6 min ago](#)
- [Last stable build \(#295\), 7 hr 6 min ago](#)
- [Last successful build \(#295\), 7 hr 6 min ago](#)
- [Last completed build \(#295\), 7 hr 6 min ago](#)

Branches to test are defined using the project matrix as shown in the picture below. Each branch in the list (here *master* and *1.4*) spawns an independent CI build.

The screenshot shows the Jenkins Configuration Matrix configuration page. The 'Configuration Matrix' tab is selected. Under the 'User-defined Axis' section, the 'Name' field is set to 'BRANCH' and the 'Values' field is set to 'master 1.4'. There is a red 'X' icon in the top right corner of the axis configuration area and a blue question mark icon next to the values field.

In the *Build* section of the configuration it is set the list of Jenkins subprojects to be triggered, in this case we have *decisionengine_pipeline* and *decisionengine_modules_pipeline*.

The *Parameters* text box is used to override parameters of each Jenkins subproject with a custom value.

In total this Jenkins project triggers 4 CI builds, i.e. 2 branches X 2 Jenkins subprojects.

The screenshot shows the Jenkins Build configuration page. The 'Build' tab is selected. Under the 'Trigger/call builds on other projects' section, the 'Projects to build' field is set to 'decisionengine_pipeline,decisionengine_modules_pipeline'. The 'Block until the triggered projects finish their builds' checkbox is unchecked. Under the 'Predefined parameters' section, the 'Parameters' field is set to 'BRANCH=\${BRANCH}'. There are red 'X' icons in the top right corner of the 'Trigger/call builds on other projects' and 'Predefined parameters' sections, and blue question mark icons next to the 'Projects to build' and 'Parameters' fields.

Finally the *Build Triggers* section is used to setup the schedule for the periodic build, in this case it is scheduled to run at about 2 AM.

Jenkins will choose the actual time depending on the actual load on the system.

General

Advanced Project Options

Source Code Management

Build Triggers

Configuration Matrix

Build Environment

Build

Post-build Actions

Build Triggers

☐ Trigger builds remotely (e.g., from scripts)

☐ Build after other projects are built

☒ Build periodically

Schedule

H 2 * * *

Would last have run at Wednesday, November 4, 2020 2:23:53 AM CST;
would next run at Thursday, November 5, 2020 2:23:53 AM CST.

?

?

?

?

SOURCE CODE

5.1 Welcome to decisionengine's documentation!

5.1.1 decisionengine package

Subpackages

`decisionengine.framework` package

Subpackages

`decisionengine.framework.config` package

Subpackages

`decisionengine.framework.config.tests` package

Submodules

`decisionengine.framework.config.tests.test_config` module

`decisionengine.framework.config.tests.test_config._channel_config_dir`(*relative_dir*)

`decisionengine.framework.config.tests.test_config._global_config_file`(*relative_filename*)

`decisionengine.framework.config.tests.test_config.load`()

`decisionengine.framework.config.tests.test_config.test_channel_empty_config`(*load*, *caplog*)

`decisionengine.framework.config.tests.test_config.test_channel_empty_dictionary`(*load*,
caplog)

`decisionengine.framework.config.tests.test_config.test_channel_invalid_modules_list`(*load*,
caplog)

`decisionengine.framework.config.tests.test_config.test_channel_invalid_modules_no_keys`(*load*,
caplog)

```
decisionengine.framework.config.tests.test_config.test_channel_invalid_modules_string(load,
                                                                                       caplog)
decisionengine.framework.config.tests.test_config.test_channel_loading(caplog)
decisionengine.framework.config.tests.test_config.test_channel_module_missing_all(load,
                                                                                   caplog)
decisionengine.framework.config.tests.test_config.test_channel_module_missing_module(load,
                                                                                   caplog)
decisionengine.framework.config.tests.test_config.test_channel_module_missing_parameters(load,
                                                                                       caplog)
decisionengine.framework.config.tests.test_config.test_channel_names(load)
decisionengine.framework.config.tests.test_config.test_channel_no_config_files(load)
decisionengine.framework.config.tests.test_config.test_channel_no_modules(load)
decisionengine.framework.config.tests.test_config.test_empty_config(load)
decisionengine.framework.config.tests.test_config.test_minimal_jsonnet_right_extension(load,
                                                                                       cap-
                                                                                       sys)
decisionengine.framework.config.tests.test_config.test_minimal_jsonnet_wrong_extension(load,
                                                                                       cap-
                                                                                       sys)
decisionengine.framework.config.tests.test_config.test_syntax_error_in_config_names_bad_file(load)
decisionengine.framework.config.tests.test_config.test_valid_but_empty_config(load)
```

decisionengine.framework.config.tests.test_de_std module

```
decisionengine.framework.config.tests.test_de_std.config(basename, jpathdirs=None)
decisionengine.framework.config.tests.test_de_std.test_allow_duplicate_keys_same_values()
decisionengine.framework.config.tests.test_de_std.test_allow_duplicate_source_proxy_keys()
decisionengine.framework.config.tests.test_de_std.test_combine_one_level()
decisionengine.framework.config.tests.test_de_std.test_combine_one_level_skip_proxies()
decisionengine.framework.config.tests.test_de_std.test_error_on_duplicate_keys()
decisionengine.framework.config.tests.test_de_std.test_jpath()
```

decisionengine.framework.config.tests.test_policies module

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

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

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

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

Module contents

Submodules

decisionengine.framework.config.ChannelConfigHandler module

Manager of channel configurations.

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

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

Bases: object

_load_channel(*channel_name*, *path*)

get_channels()

load_all_channels()

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

Any cached configurations will be dropped prior to reloading.

load_channel(*channel_name*)

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

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

<cached channel config. dir>/{channel_name}.jsonnet

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

print_channel_config(*channel*)

decisionengine.framework.config.ChannelConfigHandler.**_check_keys**(*channel_conf_dict*)

check that channel config has mandatory keys :type data: dict

decisionengine.framework.config.ChannelConfigHandler.**_make_de_logger**(*global_config*)

decisionengine.framework.config.ValidConfig module

ValidConfig represents a valid JSON document.

The decision engine requires each of its configuration files to be valid JSON. This is achieved by either supplying a valid Jsonnet or JSON document upfront.

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

class decisionengine.framework.config.ValidConfig.ValidConfig(*filename, jpathdirs=None*)

Bases: collections.UserDict

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

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

_abc_impl = <_abc._abc_data object>

dump()

Print dictionary data to a valid JSON string.

decisionengine.framework.config.ValidConfig._config_from_file(*config_file, jpaths=None*)

decisionengine.framework.config.policies module

Decision-engine default configuration policies.

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

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

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

decisionengine.framework.config.policies.channel_config_dir(*parent_dir=None*)

Retrieve the channel configuration directory as a pathlib.Path object.

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

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

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

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

Retrieve global configuration dir as `pathlib.Path` object.

This is the directory that houses the ‘decision_engine.jsonnet’ global configuration file.

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

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

Return the `pathlib.Path` object corresponding to the global configuration.

If supplied, the ‘parent_dir’ is assumed to be the full path corresponding to a directory containing the ‘decision_engine.jsonnet’ file. If not provided, the global configuration directory is determined based on the behavior of the ‘global_config_dir()’ function.

An exception is raised if no ‘decision_engine.jsonnet’ file is found.

`decisionengine.framework.config.policies.valid_dir(path, scope)`

Throws if the supplied path object is not a directory, otherwise returns the path object.

Module contents

decisionengine.framework.dataspace package

Subpackages

decisionengine.framework.dataspace.datasources package

Subpackages

decisionengine.framework.dataspace.datasources.sqlalchemy_ds package

Submodules

decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource_api module

The datasource layer for our abstraction

class `decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource_api.SQLAlchemyDS(config_dict)`

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

A DecisionEngine data source via the SQL Alchemy ORM

```
{
    "dataspace": {
        "datasource": { "module": "decisionengine.framework.dataspace.datasources.sqlalchemy_ds",
                        "name": "SQLAlchemyDS", "params": {
                            "pool_size": 5, "max_overflow": 10, "timeout": 30,
                            # url is mandatory, but any engine keyword is accepted here.
                            "url": "dialect[+driver]://user:password@host/dbname"
                        }
                    }
    }
```

```
    }  
  }  
}
```

Exceptions should be caught and logged by the caller.

_abc_impl = <_abc._abc_data object>

close()

Close all connections to the database

Returns None

connect()

Create a pool of database connections

Returns None

create_tables()

Create database tables

Returns None

delete_data_older_than(days)

Delete data older than interval

Parameters **days** (*int*) – remove data older than this many days

Returns None

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** (*str/uuid*) – id of taskmanager to retrieve
- **generation_id** (*int*) – generation id to clone
- **new_generation_id** (*int*) – generation id to create

Returns None

get_datablock(taskmanager_id, generation_id)

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

Parameters

- **taskmanager_id** (*str/uuid*) – id of taskmanager to retrieve
- **generation_id** (*int*) – generation id to locate

Returns with all set keys and their associated values

Return type dict

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** (*str/uuid*) – id of taskmanager to retrieve
- **generation_id** (*int*) – generation id to locate

- **key** (*str*) – key for the value

Returns The possibly binary value stored earlier

Return type *obj*

get_dataproducts(*taskmanager_id*, *key=None*)

Return list of all data products associated with with *taskmanager_id*

Parameters

- **taskmanager_id** (*str/uuid*) – id of taskmanager to retrieve
- **key** (*str*) – key for the value

Returns each element is the matching row as a dict()

Return type *tuple*

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** (*str/uuid*) – id of taskmanager to retrieve
- **generation_id** (*int*) – generation id to locate
- **key** (*str*) – key for the value

Returns

fields in order are: *taskmanager.taskmanager_id*, *header.taskmanager_id*,
header.generation_id, *header.key*, *header.create_time*, *header.expiration_time*,
header.scheduled_create_time, *header.creator*, *header.schema_id*

Return type *tuple*

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

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

Parameters

- **taskmanager_name** (*str*) – name of taskmanager to retrieve
- **taskmanager_id** (*str/uuid*) – id of taskmanager to retrieve

Returns the largest generation stored within the database

Return type *int*

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** (*str/uuid*) – id of taskmanager to retrieve
- **generation_id** (*int*) – generation id to locate
- **key** (*str*) – key for the value

Returns

fields in order are: *taskmanager.taskmanager_id*, *metadata.taskmanager_id*, *meta-*
data.generation_id, *metadata.key*, *metadata.state*, *metadata.generation_time*, *meta-*
data.missed_update_count

Return type tuple

get_schema(*table=None*)

Given the table name return it's schema

get_taskmanager(*taskmanager_name, taskmanager_id=None*)

Find the task manager by name/uuid in the database get back the primary key.

If multiples match, find highest primary key.

Parameters

- **taskmanager_name** (*str*) – name of taskmanager to retrieve
- **taskmanager_id** (*str/uuid*) – id of taskmanager to retrieve

Returns the matching row, column names as keys

Return type dict

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

Find taskmanagers that meet our search

Parameters

- **taskmanager_name** (*str*) – name of taskmanager to retrieve
- **start_time** (*datetime*) – Datetime to confine against
- **end_time** (*datetime*) – Datetime to confine against

Returns each element is a dict() matching row, column names as keys

Return type list

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** (*str/uuid*) – id of taskmanager to retrieve
- **generation_id** (*int*) – generation id to create
- **key** (*str*) – key for the value
- **value** (*obj*) – Value can be an object or dict or a binary
- **header** (*datablock.Header*) – Header for the value
- **metadata** (*datablock.Metadata*) – Metadata for the value

Returns None

reset_connections()

Reset the connection to the database. So long as self.engine isn't undef, the engine can still make new connections if new db actions happen. It just wont have any open at this time.

Returns None

store_taskmanager(*name, taskmanager_id, datestamp=None*)

Store TaskManager in database

Parameters

- **name** (*str*) – name of taskmanager to retrieve
- **taskmanager_id** (*str/uuid*) – id of taskmanager to retrieve

- **timestamp** (*datetime*) – datetime of created object, defaults to ‘now’

Returns the primary key of the row in the database

Return type int

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** (*str/uuid*) – id of taskmanager to retrieve
- **generation_id** (*int*) – generation id to update
- **key** (*str*) – key for the value
- **value** (*obj*) – Value can be an object or dict or a binary
- **header** (*datablock.Header*) – Header for the value
- **metadata** (*datablock.Metadata*) – Metadata for the value

Returns None

decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema module

The table layout and utilities for our SQLAlchemy ORM

class decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.**Base**(***kwargs*)

Bases: object

The base class of the class hierarchy.

When called, it accepts no arguments and returns a new featureless instance that has no instance attributes and cannot be given any.

_sa_registry = <sqlalchemy.orm.decl_api.registry object>

metadata = MetaData()

registry = <sqlalchemy.orm.decl_api.registry object>

class decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.**Dataproduct**(***kwargs*)

Bases: *sqlalchemy.orm.decl_api.Base*

The PRIMARY KEY on this table isn’t used...

Existing code appears to depend on column order.

```
_sa_class_manager = {'generation_id':
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'id':
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'key':
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'taskmanager':
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'taskmanager_id':
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'value':
<sqlalchemy.orm.attributes.InstrumentedAttribute object>}
```

generation_id

id

key

taskmanager

taskmanager_id

value

```
class decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Header(**kwargs)
```

Bases: *sqlalchemy.orm.decl_api.Base*

The PRIMARY KEY on this table isn't used...

The existing code has a hard expectation on the time columns being BIGINT rather than datetime objects burried within the classes.

Looks like there was an initial goal of a relationship with the Schema table, but it may not be in use

Existing code appears to depend on column order.

```
_sa_class_manager = {'create_time':  
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'creator':  
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'expiration_time':  
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'generation_id':  
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'id':  
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'key':  
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'scheduled_create_time':  
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'schema_id':  
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'taskmanager':  
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'taskmanager_id':  
<sqlalchemy.orm.attributes.InstrumentedAttribute object>}
```

create_time

creator

expiration_time

generation_id

id

key

scheduled_create_time

schema_id

taskmanager

taskmanager_id

```
class decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Metadata(**kwargs)
```

Bases: *sqlalchemy.orm.decl_api.Base*

The PRIMARY KEY on this table isn't used...

The existing code has a hard expectation on the state field as a 'text' element.

The existing code has a hard expectation on the time columns being BIGINT rather than datetime objects burried within the classes.

Existing code appears to depend on column order.

```

_sa_class_manager = {'generation_id':
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'generation_time':
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'id':
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'key':
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'missed_update_count':
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'state':
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'taskmanager':
<sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'taskmanager_id':
<sqlalchemy.orm.attributes.InstrumentedAttribute object>}

```

generation_id

generation_time

id

key

missed_update_count

state

taskmanager

taskmanager_id

```
class decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Schema(**kwargs)
```

Bases: [sqlalchemy.orm.decl_api.Base](#)

This table may not be in use

Has a one-to-many relationship with: Header - may not be in use

```

_sa_class_manager = {'schema': <sqlalchemy.orm.attributes.InstrumentedAttribute
object>, 'schema_id': <sqlalchemy.orm.attributes.InstrumentedAttribute object>}

```

schema

schema_id

```
class decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Taskmanager(**kwargs)
```

Bases: [sqlalchemy.orm.decl_api.Base](#)

Has a one-to-many relationship with: Header Metadata Dataproduct

changes cascade on: Header Metadata Dataproduct

Existing code appears to depend on column order.

```

_sa_class_manager = {'datestamp': <sqlalchemy.orm.attributes.InstrumentedAttribute
object>, 'name': <sqlalchemy.orm.attributes.InstrumentedAttribute object>,
'sequence_id': <sqlalchemy.orm.attributes.InstrumentedAttribute object>,
'task_dataproduct': <sqlalchemy.orm.attributes.InstrumentedAttribute object>,
'task_header': <sqlalchemy.orm.attributes.InstrumentedAttribute object>,
'task_metadata': <sqlalchemy.orm.attributes.InstrumentedAttribute object>,
'taskmanager_id': <sqlalchemy.orm.attributes.InstrumentedAttribute object>}

```

datestamp

```
name
sequence_id
task_dataproduct
task_header
task_metadata
taskmanager_id
```

decisionengine.framework.dataspace.datasources.sqlalchemy_ds.utils module

Code not written by us

```
decisionengine.framework.dataspace.datasources.sqlalchemy_ds.utils.add_engine_pidguard(engine)
```

Based on <https://stackoverflow.com/questions/62920507/using-sqlalchemy-connection-pooling-queues-with-python-multiprocessing>

```
decisionengine.framework.dataspace.datasources.sqlalchemy_ds.utils.clone_model(model,
                                                                                **kwargs)
```

Based on <https://stackoverflow.com/a/55991358>

```
decisionengine.framework.dataspace.datasources.sqlalchemy_ds.utils.orm_as_dict(obj)
```

Based on : <https://stackoverflow.com/a/37350445>

Module contents

Top level import so we can rationally segment items of the ORM

```
class decisionengine.framework.dataspace.datasources.sqlalchemy_ds.SQLAlchemyDS(config_dict)
```

Bases: [*decisionengine.framework.dataspace.datasource.DataSource*](#)

A DecisionEngine data source via the SQL Alchemy ORM

```
{
    "dataspace": {
        "datasource": { "module": "decisionengine.framework.dataspace.datasources.sqlalchemy_ds",
                        "name": "SQLAlchemyDS", "params": {
                            "pool_size": 5, "max_overflow": 10, "timeout": 30,
                            # url is mandatory, but any engine keyword is accepted here.
                            "url": "dialect[+driver]://user:password@host/dbname"
                        }
                    }
    }
}
```

Exceptions should be caught and logged by the caller.

```
_abc_impl = <_abc._abc_data object>
```

close()

Close all connections to the database

Returns None

connect()

Create a pool of database connections

Returns None

create_tables()

Create database tables

Returns None

delete_data_older_than(days)

Delete data older than interval

Parameters **days** (*int*) – remove data older than this many days

Returns None

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** (*str/uuid*) – id of taskmanager to retrieve
- **generation_id** (*int*) – generation id to clone
- **new_generation_id** (*int*) – generation id to create

Returns None

get_datablock(taskmanager_id, generation_id)

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

Parameters

- **taskmanager_id** (*str/uuid*) – id of taskmanager to retrieve
- **generation_id** (*int*) – generation id to locate

Returns with all set keys and their associated values

Return type dict

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** (*str/uuid*) – id of taskmanager to retrieve
- **generation_id** (*int*) – generation id to locate
- **key** (*str*) – key for the value

Returns The possibly binary value stored earlier

Return type obj

get_dataproducts(*taskmanager_id*, *key=None*)

Return list of all data products associated with with *taskmanager_id*

Parameters

- **taskmanager_id** (*str/uuid*) – id of taskmanager to retrieve
- **key** (*str*) – key for the value

Returns each element is the matching row as a dict()

Return type tuple

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** (*str/uuid*) – id of taskmanager to retrieve
- **generation_id** (*int*) – generation id to locate
- **key** (*str*) – key for the value

Returns

fields in order are: taskmanager.taskmanager_id, header.taskmanager_id,
header.generation_id, header.key, header.create_time, header.expiration_time,
header.scheduled_create_time, header.creator, header.schema_id

Return type tuple

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

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

Parameters

- **taskmanager_name** (*str*) – name of taskmanager to retrieve
- **taskmanager_id** (*str/uuid*) – id of taskmanager to retrieve

Returns the largest generation stored within the database

Return type int

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** (*str/uuid*) – id of taskmanager to retrieve
- **generation_id** (*int*) – generation id to locate
- **key** (*str*) – key for the value

Returns

fields in order are: taskmanager.taskmanager_id, metadata.taskmanager_id, meta-
data.generation_id, metadata.key, metadata.state, metadata.generation_time, meta-
data.missed_update_count

Return type tuple

get_schema(*table=None*)

Given the table name return it's schema

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

Find the task manager by name/uuid in the database get back the primary key.

If multiples match, find highest primary key.

Parameters

- **taskmanager_name** (*str*) – name of taskmanager to retrieve
- **taskmanager_id** (*str/uuid*) – id of taskmanager to retrieve

Returns the matching row, column names as keys

Return type dict

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

Find taskmanagers that meet our search

Parameters

- **taskmanager_name** (*str*) – name of taskmanager to retrieve
- **start_time** (*datetime*) – Datetime to confine against
- **end_time** (*datetime*) – Datetime to confine against

Returns each element is a dict() matching row, column names as keys

Return type list

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** (*str/uuid*) – id of taskmanager to retrieve
- **generation_id** (*int*) – generation id to create
- **key** (*str*) – key for the value
- **value** (*obj*) – Value can be an object or dict or a binary
- **header** (*datablock.Header*) – Header for the value
- **metadata** (*datablock.Metadata*) – Metadata for the value

Returns None

reset_connections()

Reset the connection to the database. So long as self.engine isn't undef, the engine can still make new connections if new db actions happen. It just wont have any open at this time.

Returns None

store_taskmanager(*name*, *taskmanager_id*, *timestamp=None*)

Store TaskManager in database

Parameters

- **name** (*str*) – name of taskmanager to retrieve
- **taskmanager_id** (*str/uuid*) – id of taskmanager to retrieve
- **timestamp** (*datetime*) – datetime of created object, defaults to 'now'

Returns the primary key of the row in the database

Return type int

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** (*str/uuid*) – id of taskmanager to retrieve
- **generation_id** (*int*) – generation id to update
- **key** (*str*) – key for the value
- **value** (*obj*) – Value can be an object or dict or a binary
- **header** (*datablock.Header*) – Header for the value
- **metadata** (*datablock.Metadata*) – Metadata for the value

Returns None

decisionengine.framework.dataspace.datasources.tests package

Submodules

decisionengine.framework.dataspace.datasources.tests.fixtures module

pytest fixtures/constants

decisionengine.framework.dataspace.datasources.tests.fixtures.PG_DE_DB_WITHOUT_SCHEMA(*request*:
_pytest.fixtures.FixtureRequest
→
psycopg2.extensions.cursor)

Fixture factory for PostgreSQL.

Parameters **request** – fixture request object

Returns postgresql client

decisionengine.framework.dataspace.datasources.tests.fixtures.PG_PROG(*request*:
_pytest.fixtures.FixtureRequest,
tmp_path_factory:
_pytest.tmpdir.TempPathFactory)
→
Iterator[pytest_postgresql.executor.PostgreSQLExecutor])

Process fixture for PostgreSQL.

Parameters

- **request** – fixture request object
- **tmp_path_factory** – temporary path object (fixture)

Returns tcp executor

decisionengine.framework.dataspace.datasources.tests.fixtures.SQLALCHEMY_PG_WITH_SCHEMA(*PG_DE_DB_WITHOUT_SCHEMA*,
tmp_path_factory)
Get a blank database from pytest_postgresql. Then setup the SQLAlchemy style URL with that DB. The SQLAlchemyDS will create the schema as needed.

`decisionengine.framework.dataspace.datasources.tests.fixtures.SQLALCHEMY_TEMPFILE_SQLITE(tmp_path)`
 Setup an SQLite database with the pytest tmp_path fixture. Then setup the SQLAlchemy style URL with that DB. The SQLAlchemyDS will create the schema as needed.

`decisionengine.framework.dataspace.datasources.tests.fixtures.datasource(request)`

This parameterized fixture will setup up various datasources.

Add datasource objects to DATASOURCES_TO_TEST once they've got our basic schema loaded. And adjust our *if* statements here until we are SQLAlchemy only.

Pytest should take it from there and automatically run it through all the tests using this fixture.

`decisionengine.framework.dataspace.datasources.tests.fixtures.mock_data_block()`

This fixture replaces the standard datablock implementation.

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

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

decisionengine.framework.dataspace.datasources.tests.test_datasource_api module

This test plan covers a generic dataspace object via pytest parameters.

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_create_tables(datasource)`
 create_tables() should be safe to call multiple times

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_delete_data_older_than_age(datasource)`
 Can we delete old entries

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_duplicate_datablock(datasource)`
 Can we duplicate taskmanager1 and all its entries

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_get_datablock(datasource)`

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_get_dataproduct(datasource)`
 Can we get the dataproduct by uuid with key

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_get_dataproduct_not_exist(datasource)`
 Does it error out if we ask for bogus information?

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_get_dataproducts(datasource)`
 Can we get the dataproducts by uuid and uuid with key

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_get_dataproducts_not_exist(datasource)`
 Does it error out if we ask for bogus information?

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_get_header(datasource)`
 Can we fetch a header?

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_get_header_not_exist(datasource)`
 Does it error out if we ask for a bogus header?

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_get_last_generation_id(datasource)`
 Can we get the last generation id by name or name and uuid

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_get_last_generation_id_no`

Does it error out if we ask for a bogus taskmanager?

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_get_metadata(datasource)`

Can we fetch a metadata element?

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_get_metadata_not_exist(da`

Does it error out if we ask for a bogus metadata element?

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_get_taskmanager_exists(da`

Can I get a taskmanager by name or name and uuid

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_get_taskmanager_not_exists`

This should error out

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_get_taskmanagers(datasource`

Can I get multiple task managers

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_get_taskmanagers_not_exis`

Do I error out when asking for garbage

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_has_config(datasource)`

This should have a *config* dict we can pass to jsonnet

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_insert(datasource)`

Can we insert new elements

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_reset_connections(datasource`

`reset_connections()` should be safe to call any time

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_store_taskmanager(datasource`

Can we make new entries

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_update(datasource)`

Do updates work as expected

`decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_update_bad(datasource)`

Do updates fail to work on bogus taskmanager as expected

Module contents

Submodules

decisionengine.framework.dataspace.datasources.null module

class `decisionengine.framework.dataspace.datasources.null.NullDataSource(config_dict)`

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

Implementation of data source ABC that does nothing

`_abc_impl = <_abc._abc_data object>`

`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 interval :type days: long :arg days: remove data older than 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_datablock(taskmanager_id, generation_id)

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

Parameters

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

get_dataproduct(taskmanager_id, generation_id, key)

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

Parameters

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

get_dataproducts(taskmanager_id, key=None)

Return list of all data products associated with taskmanager_id

Parameters **key** (string) – data product key

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

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

get_metadata(*taskmanager_id, generation_id, key*)

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

Parameters

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

get_schema(*table=None*)

Given the table name return it's schema

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

get_taskmanager(*taskmanager_name, taskmanager_id=None*)

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

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

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

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

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

Parameters

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

reset_connections()

Drop any cached connections and reconnect to the database

store_taskmanager(*name, taskmanager_id, datestamp=None*)

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 :type datestamp: datetime :arg datestamp: datetime of created object, defaults to 'now'

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

Module contents

decisionengine.framework.dataspace.tests package

Submodules

decisionengine.framework.dataspace.tests.fixtures module

decisionengine.framework.dataspace.tests.fixtures.**PG_DE_DB_WITHOUT_SCHEMA**(request: *_pytest.fixtures.FixtureRequest*) → *psycopg2.extensions.connection*

Fixture factory for PostgreSQL.

Parameters **request** – fixture request object

Returns postgresql client

decisionengine.framework.dataspace.tests.fixtures.**PG_PROG**(request: *_pytest.fixtures.FixtureRequest*, tmp_path_factory: *_pytest.tmpdir.TempPathFactory*) → *Iterator[pytest_postgresql.executor.PostgreSQLExecutor]*

Process fixture for PostgreSQL.

Parameters

- **request** – fixture request object
- **tmp_path_factory** – temporary path object (fixture)

Returns tcp executor

decisionengine.framework.dataspace.tests.fixtures.**SQLALCHEMY_PG_WITH_SCHEMA**(PG_DE_DB_WITHOUT_SCHEMA) Get a blank database from pytest_postgresql. Then setup the SQLAlchemy style URL with that DB. The SQLAlchemyDS will create the schema as needed.

decisionengine.framework.dataspace.tests.fixtures.**SQLALCHEMY_TEMPFILE_SQLITE**(tmp_path) Setup an SQLite database with the pytest tmp_path fixture. Then setup the SQLAlchemy style URL with that DB. The SQLAlchemyDS will create the schema as needed.

decisionengine.framework.dataspace.tests.fixtures.**datasource**(request)

This parameterized fixture will setup up various datasources.

Add datasource objects to DATASOURCES_TO_TEST once they've got our basic schema loaded. And adjust our *if* statements here until we are SQLAlchemy only.

Pytest should take it from there and automatically run it through all the tests using this fixture.

decisionengine.framework.dataspace.tests.fixtures.**dataspace**(request)

This parameterized fixture will setup up various datasources. Add datasource objects to DATASOURCES_TO_TEST once they've got our basic schema loaded. And adjust our *if* statements here until we are SQLAlchemy only.

Pytest should take it from there and automatically run it through all the tests using this fixture.

decisionengine.framework.dataspace.tests.fixtures.**load_sample_data_into_datasource**(schema_only_db)

load our sample test data into a dataspace This is a function not a fixture so you can run it on any datasource providing the right API.

decisionengine.framework.dataspace.tests.test_Reaper module

```
decisionengine.framework.dataspace.tests.test_Reaper.config()
decisionengine.framework.dataspace.tests.test_Reaper.reaper(request)
decisionengine.framework.dataspace.tests.test_Reaper.test_fail_bad_config(reaper, config)
decisionengine.framework.dataspace.tests.test_Reaper.test_fail_missing_config(reaper,
                                                                                config)
decisionengine.framework.dataspace.tests.test_Reaper.test_fail_missing_config_key(reaper,
                                                                                    config)
decisionengine.framework.dataspace.tests.test_Reaper.test_fail_small_retain(reaper)
decisionengine.framework.dataspace.tests.test_Reaper.test_fail_small_run_interval(reaper)
decisionengine.framework.dataspace.tests.test_Reaper.test_fail_start_two_reapers(reaper)
decisionengine.framework.dataspace.tests.test_Reaper.test_fail_wrong_config_key(reaper,
                                                                                config)
decisionengine.framework.dataspace.tests.test_Reaper.test_just_stop_no_error(reaper)
decisionengine.framework.dataspace.tests.test_Reaper.test_loop_of_start_stop_in_clumps(reaper)
decisionengine.framework.dataspace.tests.test_Reaper.test_reap_default_state(reaper)
decisionengine.framework.dataspace.tests.test_Reaper.test_reaper_can_reap(reaper)
decisionengine.framework.dataspace.tests.test_Reaper.test_source_fail_can_be_fixed(reaper)
decisionengine.framework.dataspace.tests.test_Reaper.test_start_delay(reaper)
decisionengine.framework.dataspace.tests.test_Reaper.test_start_stop(reaper)
decisionengine.framework.dataspace.tests.test_Reaper.test_start_stop_stop(reaper)
decisionengine.framework.dataspace.tests.test_Reaper.test_state_can_be_active(reaper)
decisionengine.framework.dataspace.tests.test_Reaper.test_state_sets_timer_and_uses_it(reaper)
```

decisionengine.framework.dataspace.tests.test_datablock module

```
decisionengine.framework.dataspace.tests.test_datablock.test_DataBlock_constructor(dataspace)
decisionengine.framework.dataspace.tests.test_datablock.test_DataBlock_duplicate(dataspace)
decisionengine.framework.dataspace.tests.test_datablock.test_DataBlock_get_dataproducts(dataspace)
decisionengine.framework.dataspace.tests.test_datablock.test_DataBlock_get_header(dataspace)
decisionengine.framework.dataspace.tests.test_datablock.test_DataBlock_get_metadata(dataspace)
decisionengine.framework.dataspace.tests.test_datablock.test_DataBlock_get_taskmanager(dataspace)
```


`decisionengine.framework.dataspace.tests.test_datablock.test_DataBlock_is_expired(dataspace)`

This test just validates the method/function exists. The stub within our default code should be replaced by a class inheriting from it. That class should have more rational return types.

`decisionengine.framework.dataspace.tests.test_datablock.test_DataBlock_is_expired_with_key(dataspace)`

This test just validates the method/function exists. The stub within our default code should be replaced by a class inheriting from it. That class should have more rational return types.

`decisionengine.framework.dataspace.tests.test_datablock.test_DataBlock_key_management(dataspace)`

`decisionengine.framework.dataspace.tests.test_datablock.test_DataBlock_key_management_change_name(dataspace)`

`decisionengine.framework.dataspace.tests.test_datablock.test_DataBlock_mark_expired(dataspace)`

`decisionengine.framework.dataspace.tests.test_datablock.test_DataBlock_no_key_by_name(dataspace)`

`decisionengine.framework.dataspace.tests.test_datablock.test_DataBlock_to_str(dataspace)`

`decisionengine.framework.dataspace.tests.test_datablock.test_Header_constructor(dataspace)`

`decisionengine.framework.dataspace.tests.test_datablock.test_Header_is_valid(dataspace)`

`decisionengine.framework.dataspace.tests.test_datablock.test_Metadata_constructor(dataspace)`

`decisionengine.framework.dataspace.tests.test_datablock.test_Metadata_set_state(dataspace)`

decisionengine.framework.dataspace.tests.test_datablock_zlib module

`decisionengine.framework.dataspace.tests.test_datablock_zlib.test_compress()`

`decisionengine.framework.dataspace.tests.test_datablock_zlib.test_zdumps()`

`decisionengine.framework.dataspace.tests.test_datablock_zlib.test_zloads()`

decisionengine.framework.dataspace.tests.test_datasource module

`decisionengine.framework.dataspace.tests.test_datasource.test_has_methods_we_expect()`

decisionengine.framework.dataspace.tests.test_dataspace module

`decisionengine.framework.dataspace.tests.test_dataspace.test_dataspace_config_finds_bad()`

`decisionengine.framework.dataspace.tests.test_dataspace.test_delete(dataspace)`

`decisionengine.framework.dataspace.tests.test_dataspace.test_duplicate_datablock(dataspace)`

Can we duplicate taskmanager1 and all its entries

`decisionengine.framework.dataspace.tests.test_dataspace.test_get_datablock(dataspace)`

Can we get the datablock content

`decisionengine.framework.dataspace.tests.test_dataspace.test_get_dataproduct(dataspace)`

Can we get the dataproduct by uuid with key

`decisionengine.framework.dataspace.tests.test_dataspace.test_get_dataproduct_not_exist(dataspace)`
Does it error out if we ask for bogus information?

`decisionengine.framework.dataspace.tests.test_dataspace.test_get_dataproducts(dataspace)`
Can we get the dataproducts by uuid and uuid with key

`decisionengine.framework.dataspace.tests.test_dataspace.test_get_dataproducts_not_exist(dataspace)`
Does it error out if we ask for bogus information?

`decisionengine.framework.dataspace.tests.test_dataspace.test_get_header(dataspace)`
Can we fetch a header?

`decisionengine.framework.dataspace.tests.test_dataspace.test_get_header_not_exist(dataspace)`
Does it error out if we ask for a bogus header?

`decisionengine.framework.dataspace.tests.test_dataspace.test_get_last_generation_id(dataspace)`
Can we get the last generation id by name or name and uuid

`decisionengine.framework.dataspace.tests.test_dataspace.test_get_last_generation_id_not_exist(dataspace)`
Does it error out if we ask for a bogus taskmanager?

`decisionengine.framework.dataspace.tests.test_dataspace.test_get_metadata(dataspace)`
Can we fetch a metadata element?

`decisionengine.framework.dataspace.tests.test_dataspace.test_get_metadata_not_exist(dataspace)`
Does it error out if we ask for a bogus metadata element?

`decisionengine.framework.dataspace.tests.test_dataspace.test_get_taskmanager_exists(dataspace)`
Can I get a taskmanager by name or name and uuid

`decisionengine.framework.dataspace.tests.test_dataspace.test_get_taskmanager_not_exists(dataspace)`
This should error out

`decisionengine.framework.dataspace.tests.test_dataspace.test_get_taskmanagers(dataspace)`
Can I get multiple task managers

`decisionengine.framework.dataspace.tests.test_dataspace.test_get_taskmanagers_not_exist(dataspace)`
Do I error out when asking for garbage

`decisionengine.framework.dataspace.tests.test_dataspace.test_has_config(dataspace)`
verify our config entry exists

`decisionengine.framework.dataspace.tests.test_dataspace.test_insert(dataspace)`
Can we insert new elements

`decisionengine.framework.dataspace.tests.test_dataspace.test_mark_expired(dataspace)`

`decisionengine.framework.dataspace.tests.test_dataspace.test_store_taskmanager(dataspace)`
Can we make new entries

`decisionengine.framework.dataspace.tests.test_dataspace.test_update(dataspace)`
Do updates work as expected

`decisionengine.framework.dataspace.tests.test_dataspace.test_update_bad(dataspace)`
Do updates fail to work on bogus taskmanager as expected

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

```
__update(key, value, header, metadata)
```

Update an existing product in the database with header and metadata

```
_setitem(key, value, header, metadata=None)
```

put a product in the database with header and metadata

```
duplicate()
```

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

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

Return type *DataBlock*

```
get(key, default=None)
```

Return the value associated with the key in the database

Return type dict

```
get_dataproducts(key=None)
```

```
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=psycogp2.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 expira-
    tion_time <= current time

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

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

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

    Bases: collections.UserDict

    _abc_impl = <_abc._abc_data object>

    default_data_lifetime = 1800

    is_valid()
        Check if the Header has minimum required information

    required_keys = {'create_time', 'creator', 'expiration_time',
                    'scheduled_create_time', 'schema_id', 'taskmanager_id'}

exception decisionengine.framework.dataspace.datablock.InvalidMetadataError
    Bases: Exception
    Errors due to invalid Metadata

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

    Bases: collections.UserDict

    _abc_impl = <_abc._abc_data object>

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

    set_state(state)
        Set the state for the Metadata
```

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

```
class decisionengine.framework.dataspace.datablock.ProductRetriever(product_name,  
                                                                    product_type,  
                                                                    product_source)
```

Bases: object

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

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

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

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

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

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

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

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

Parameters **zbytes** – compressed bytes

Returns returns python object

decisionengine.framework.dataspace.datasource module

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

Bases: object

```
_abc_impl = <_abc._abc_data object>
```

```
abstract close()
```

Close all connections to the database

```
abstract connect()
```

Create a pool of database connections

```
abstract create_tables()
```

Create database tables

```
dataprodut_table = 'dataprodut'
```

Name of the dataprodut table

```
abstract delete_data_older_than(days)
```

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

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

For the given taskmanager_id, make a copy of the datablock with given generation_id, set the generation_id for the datablock copy

Parameters

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

abstract get_datablock(*taskmanager_id, generation_id*)

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

Parameters

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

abstract get_dataproduct(*taskmanager_id, generation_id, key*)

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

Parameters

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

abstract get_dataproducts(*taskmanager_id, key*)

Return list of all data products associated with with taskmanager_id

Parameters **key** (string) – data product key

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(*taskmanager_name, taskmanager_id=None*)

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

Parameters

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

abstract get_metadata(*taskmanager_id, generation_id, key*)

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

Parameters

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

abstract get_schema(*table=None*)

Given the table name return it's schema

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

abstract get_taskmanager(*taskmanager_name, taskmanager_id*)

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

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

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

header_table = 'header'

Name of the header table

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

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

Parameters

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

metadata_table = 'metadata'

Name of the metadata table

abstract reset_connections()

Drop any cached connections and reconnect to the database

abstract store_taskmanager(*taskmanager_name, taskmanager_id, datestamp=None*)

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 :type datestamp: datetime :arg datestamp: datetime of created object, defaults to 'now'

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.**DataSpace**(*config*)

Bases: object

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

close()**delete**(*taskmanager_id*, *all_generations=False*)**duplicate_datablock**(*taskmanager_id*, *generation_id*, *new_generation_id*)**get_datablock**(*taskmanager_id*, *generation_id*)**get_dataproduct**(*taskmanager_id*, *generation_id*, *key*)**get_dataproducts**(*taskmanager_id*, *key=None*)**get_header**(*taskmanager_id*, *generation_id*, *key*)**get_last_generation_id**(*taskmanager_name*, *taskmanager_id=None*)**get_metadata**(*taskmanager_id*, *generation_id*, *key*)**get_taskmanager**(*taskmanager_name*, *taskmanager_id=None*)**get_taskmanagers**(*taskmanager_name=None*, *start_time=None*, *end_time=None*)**insert**(*taskmanager_id*, *generation_id*, *key*, *value*, *header*, *metadata*)**mark_demented**(*taskmanager_id*, *keys*, *generation_id=None*)**mark_expired**(*taskmanager_id*, *generation_id*, *key*, *expiry_time*)**store_taskmanager**(*name*, *taskmanager_id*, *timestamp=None*)**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

decisionengine.framework.dataspace.maintain module

class decisionengine.framework.dataspace.maintain.Reaper(*config*)

Bases: object

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

The class attributes indicate a rational set of defaults that shouldn't be altered by user configuration.

MIN_RETENTION_INTERVAL_DAYS = 7

MIN_SECONDS_BETWEEN_RUNS = 7080

_reaper_loop(*delay*)

The thread actually runs this.

reap()

Actually spawn the query to delete the old records. Lock the state as this task doesn't have a cancel option.

property retention_interval

We have data constraints, so use a property to track

property seconds_between_runs

We have data constraints, so use a property to track

start(*delay=0*)

Start thread with an optional delay to start the thread in X seconds

stop()

Try to stop the reaper, will block if the reaper cannot be interrupted.

Module contents

decisionengine.framework.engine package

Subpackages

decisionengine.framework.engine.tests package

Submodules

decisionengine.framework.engine.tests.fixtures module

pytest defaults

decisionengine.framework.engine.tests.fixtures.DEServer(*conf_path=None, conf_override=None, channel_conf_path=None, channel_conf_override=None, host='127.0.0.1', port=None, make_conf_dirs_if_missing=False*)

A DE Server using a private database

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

Fixture factory for PostgreSQL.

Parameters **request** – fixture request object

Returns postgresql client

```
decisionengine.framework.engine.tests.fixtures.PG_PROG(request: _pytest.fixtures.FixtureRequest,
    tmp_path_factory:
    _pytest.tmpdir.TempPathFactory) →
    Iterator[pytest_postgresql.executor.PostgreSQLExecutor]
```

Process fixture for PostgreSQL.

Parameters

- **request** – fixture request object
- **tmp_path_factory** – temporary path object (fixture)

Returns tcp executor

```
decisionengine.framework.engine.tests.fixtures.SQLALCHEMY_PG_WITH_SCHEMA(PG_DE_DB_WITHOUT_SCHEMA)
```

Get a blank database from `pytest_postgresql`. Then setup the SQLAlchemy style URL with that DB. The SQLAlchemyDS will create the schema as needed.

```
decisionengine.framework.engine.tests.fixtures.SQLALCHEMY_TEMPFILE_SQLITE(tmp_path)
```

Setup an SQLite database with the `pytest tmp_path` fixture. Then setup the SQLAlchemy style URL with that DB. The SQLAlchemyDS will create the schema as needed.

decisionengine.framework.engine.tests.test_ChannelWorkers module

```
class decisionengine.framework.engine.tests.test_ChannelWorkers.TaskManager
```

Bases: object

name = 'test_channel'

set_loglevel_value(value)

```
decisionengine.framework.engine.tests.test_ChannelWorkers.global_config(dataspace)
```

```
decisionengine.framework.engine.tests.test_ChannelWorkers.test_worker_logger_sized_rotation(global_config)
```

```
decisionengine.framework.engine.tests.test_ChannelWorkers.test_worker_logger_timed_rotation(global_config)
```

```
decisionengine.framework.engine.tests.test_ChannelWorkers.test_worker_name(global_config)
```

decisionengine.framework.engine.tests.test_client_only module

`decisionengine.framework.engine.tests.test_client_only.test_client_err_returned_as_rc()`
no de server is running, so `--status` should error

`decisionengine.framework.engine.tests.test_client_only.test_client_err_returned_verbose_as_rc()`
no de server is running, so `--status` should error

`decisionengine.framework.engine.tests.test_client_only.test_client_help(capfd)`

`decisionengine.framework.engine.tests.test_client_only.test_client_with_no_command_says_use_help()`

`decisionengine.framework.engine.tests.test_client_only.test_client_with_no_server()`

`decisionengine.framework.engine.tests.test_client_only.test_client_with_no_server_verbose()`

`decisionengine.framework.engine.tests.test_client_only.test_exclusive_options()`

decisionengine.framework.engine.tests.test_query_tool_only module

`decisionengine.framework.engine.tests.test_query_tool_only.test_client_err_returned_as_rc()`
no de server is running, so `--status` should error

`decisionengine.framework.engine.tests.test_query_tool_only.test_client_err_returned_verbose_as_rc()`
no de server is running, so `--status` should error

`decisionengine.framework.engine.tests.test_query_tool_only.test_query_tool_help()`

`decisionengine.framework.engine.tests.test_query_tool_only.test_query_tool_with_no_server()`

`decisionengine.framework.engine.tests.test_query_tool_only.test_query_tool_with_no_server_verbose()`

decisionengine.framework.engine.tests.test_startup module

`decisionengine.framework.engine.tests.test_startup._check_override(arguments)`

`decisionengine.framework.engine.tests.test_startup.metrics_env_setup(tmp_path, monkeypatch)`
Make sure we have a directory set for `PROMETHEUS_MULTIPROC_DIR` so that metric instantiation gives us multiprocess metrics

`decisionengine.framework.engine.tests.test_startup.test_change_port()`

`decisionengine.framework.engine.tests.test_startup.test_check_metrics_env_no_webserver()`

`decisionengine.framework.engine.tests.test_startup.test_check_metrics_env_var_set()`

`decisionengine.framework.engine.tests.test_startup.test_check_metrics_env_var_unset()`

`decisionengine.framework.engine.tests.test_startup.test_default_config()`

decisionengine.framework.engine.tests.test_verify_redis_server module

```
decisionengine.framework.engine.tests.test_verify_redis_server.test_verify_bad_broker()
decisionengine.framework.engine.tests.test_verify_redis_server.test_verify_bad_redis_server()
decisionengine.framework.engine.tests.test_verify_redis_server.test_verify_bad_url()
decisionengine.framework.engine.tests.test_verify_redis_server.test_verify_redis_server()
decisionengine.framework.engine.tests.test_verify_redis_server.test_verify_redis_url()
```

Module contents

Submodules

decisionengine.framework.engine.ChannelWorkers module

```
class decisionengine.framework.engine.ChannelWorkers.ChannelWorker(task_manager,
                                                                    logger_config)
```

Bases: multiprocessing.context.Process

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

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

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

get_consumes()

get_produces()

get_state_name()

run()

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

setup_logger()

wait_while(*state, timeout=None*)

```
class decisionengine.framework.engine.ChannelWorkers.ChannelWorkers
```

Bases: object

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

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

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

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

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

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

```
class Access(workers, lock)
    Bases: object
    _update_channel_states()
    access()
    unguarded_access()
```

decisionengine.framework.engine.DecisionEngine module

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

```
class decisionengine.framework.engine.DecisionEngine.DecisionEngine(global_config,
                                                                    channel_config_loader,
                                                                    server_address)
```

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

```
_dataframe_to_column_names(df)
_dataframe_to_csv(df)
_dataframe_to_json(df)
_dataframe_to_table(df)
_dataframe_to_vertical_tables(df)
_dispatch(method, params)
```

Dispatches the XML-RPC method.

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

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

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

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

```
block_while(state, timeout=None)
get_logger()
handle_sighup(signum, frame)
metrics()
reaper_start(delay)
```

```
reaper_status()
reaper_stop()
rm_channel(channel, maybe_timeout)
rpc_block_while(state_str, timeout=None)
rpc_get_channel_log_level(channel)
rpc_get_log_level()
rpc_kill_channel(channel, timeout=None)
rpc_metrics()
    Display collected metrics
rpc_ping()
rpc_print_product(product, columns=None, query=None, types=False, format=None)
rpc_print_products()
rpc_query_tool(product, format=None, start_time=None)
rpc_queue_status()
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_rm_channel(channel, maybe_timeout)
rpc_set_channel_log_level(channel, log_level)
    Assumes log_level is a string corresponding to the supported logging-module levels.
rpc_show_config(channel)
    Show the configuration for a channel.

rpc_show_de_config()
rpc_start_channel(channel_name)
rpc_start_channels()
rpc_status()
rpc_stop()
rpc_stop_channel(channel)
rpc_stop_channels()
start_channel(channel_name, channel_config)
start_channels()
```

start_webserver()

Start CherryPy webserver using configured port. If port is not configured use default webserver port.

stop_channels()

stop_worker(*worker, timeout*)

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

Bases: xmlrpc.server.SimpleXMLRPCRequestHandler

rpc_paths = ('/RPC2',)

class decisionengine.framework.engine.DecisionEngine.**StopState**(*value*)

Bases: enum.Enum

An enumeration.

Clean = 2

NotFound = 1

Terminated = 3

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

decisionengine.framework.engine.DecisionEngine.**_check_metrics_env**(*options*)

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

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

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

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

decisionengine.framework.engine.DecisionEngine.**_start_de_server**(*server*)

Start the DE server and listen forever

decisionengine.framework.engine.DecisionEngine.**_verify_redis_server**(*broker_url*)

decisionengine.framework.engine.DecisionEngine.**_verify_redis_url**(*broker_url*)

decisionengine.framework.engine.DecisionEngine.**main**(*args=None*)

If args is None, sys.argv will be used instead If args is a list, it will be used instead of sys.argv (for unit testing)

decisionengine.framework.engine.DecisionEngine.**parse_program_options**(*args=None*)

If args is a list, it will be used instead of sys.argv

decisionengine.framework.engine.SourceWorkers module

class decisionengine.framework.engine.SourceWorkers.**SourceWorker**(*key, config, channel_name, exchange, broker_url*)

Bases: multiprocessing.context.Process

Provides interface to loadable modules and events to synchronise execution

decrement_use_count()

increment_use_count()

run()

Get the data from source

should_stop()

take_offline()

class decisionengine.framework.engine.SourceWorkers.**SourceWorkers**(*exchange, broker_url, logger=<BoundLoggerLazyProxy(logger=None, wrapper_class=None, processors=None, context_class=None, initial_values={}, logger_factory_args=('decisionengine',))>*)

Bases: object

prune(*source_names*)

remove_all(*timeout*)

update(*channel_name, source_configs*)

decisionengine.framework.engine.de_client module

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

This is the entry point for the setuptools auto generated scripts. Setuptools thinks a return from this function is an error message.

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

decisionengine.framework.engine.de_client.**execute_command_from_args**(*argsparsed, de_socket*)

argsparsed should be from create_parser in this file

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

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

decisionengine.framework.engine.de_query_tool module

`decisionengine.framework.engine.de_query_tool.console_scripts_main(args_to_parse=None)`

This is the entry point for the setuptools auto generated scripts. Setuptools thinks a return from this function is an error message.

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

`decisionengine.framework.engine.de_query_tool.execute_command_from_args(argsparsed, de_socket)`

Calls the proper function for the arguments passed to `de_query_tool`.

Parameters

- **argsparsed** (*Namespace*) – Should be from `create_parser` in this file.
- **de_socket** (*ServerProxy*) – RPC Server Proxy.

Returns Output of the command.

Return type str

`decisionengine.framework.engine.de_query_tool.main(args_to_parse=None)`

Main function for `de_query_tool`

Parameters **args_to_parse** (*list, optional*) – If you pass a list of args, they will be used instead of `sys.argv`. Defaults to `None`.

Returns Query result

Return type str

Module contents

decisionengine.framework.logicengine package

Subpackages

decisionengine.framework.logicengine.tests package

Submodules

decisionengine.framework.logicengine.tests.test_bool_function_name module

`decisionengine.framework.logicengine.tests.test_bool_function_name.test_error_conditions()`

decisionengine.framework.logicengine.tests.test_cascaded_rules module

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

`decisionengine.framework.logicengine.tests.test_cascaded_rules.test_rule_that_does_not_fire(myengine)`

`decisionengine.framework.logicengine.tests.test_cascaded_rules.test_rule_that_fires(myengine)`

decisionengine.framework.logicengine.tests.test_construction module

`decisionengine.framework.logicengine.tests.test_construction.test_configuration_with_fact_using_function()`

`decisionengine.framework.logicengine.tests.test_construction.test_configuration_with_numpy_facts()`

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

LogicEngine is not default constructible.

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

Logic engine constructed with trivial rules and facts.

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

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

decisionengine.framework.logicengine.tests.test_duplicate_fact_names module

`decisionengine.framework.logicengine.tests.test_duplicate_fact_names.test_duplicate_fact_names()`

decisionengine.framework.logicengine.tests.test_facts module

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

`decisionengine.framework.logicengine.tests.test_facts.test_compound_fact_with_spaces()`

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

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

`decisionengine.framework.logicengine.tests.test_facts.test_fact_with_fail_on_error()`

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

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

`decisionengine.framework.logicengine.tests.test_facts.test_syntax_error(caplog)`

decisionengine.framework.logicengine.tests.test_fail_on_error module

`decisionengine.framework.logicengine.tests.test_fail_on_error.logic_engine_with_fact(fact)`
`decisionengine.framework.logicengine.tests.test_fail_on_error.test_conditional_fact()`
`decisionengine.framework.logicengine.tests.test_fail_on_error.test_fact_with_misspecified_attribute()`
`decisionengine.framework.logicengine.tests.test_fail_on_error.test_fail_on_error(caplog)`
`decisionengine.framework.logicengine.tests.test_fail_on_error.test_false_fact_with_spaces()`
`decisionengine.framework.logicengine.tests.test_fail_on_error.test_false_literal_fact()`
`decisionengine.framework.logicengine.tests.test_fail_on_error.test_index_error()`
`decisionengine.framework.logicengine.tests.test_fail_on_error.test_misspecified_fact()`
`decisionengine.framework.logicengine.tests.test_fail_on_error.test_true_fact()`
`decisionengine.framework.logicengine.tests.test_fail_on_error.test_true_literal_fact()`

decisionengine.framework.logicengine.tests.test_pandas_fact module

`decisionengine.framework.logicengine.tests.test_pandas_fact.mydata(y)`
Return a 'datablock' surrogate carrying a Pandas DataFrame, and a parameter named 'y' with value y.
`decisionengine.framework.logicengine.tests.test_pandas_fact.myengine()`
`decisionengine.framework.logicengine.tests.test_pandas_fact.test_rule_that_does_not_fire(myengine)`
Rules that do not fire do not create entries in the returned actions and newfacts.
`decisionengine.framework.logicengine.tests.test_pandas_fact.test_rule_that_fires(myengine)`

decisionengine.framework.logicengine.tests.test_rule_with_negated_fact module

`decisionengine.framework.logicengine.tests.test_rule_with_negated_fact.myengine()`
`decisionengine.framework.logicengine.tests.test_rule_with_negated_fact.test_rule_that_does_not_fire(myengine)`
Rules that do not fire do not create entries in the returned actions and newfacts.
`decisionengine.framework.logicengine.tests.test_rule_with_negated_fact.test_rule_that_fires(myengine)`

decisionengine.framework.logicengine.tests.test_simple_configuration module

`decisionengine.framework.logicengine.tests.test_simple_configuration.myengine()`
`decisionengine.framework.logicengine.tests.test_simple_configuration.test_error_on_bad_names(myengine)`
`decisionengine.framework.logicengine.tests.test_simple_configuration.test_rule_that_does_not_fire(myengine)`
Rules that do not fire do not create entries in the returned actions and newfacts.
`decisionengine.framework.logicengine.tests.test_simple_configuration.test_rule_that_fires(myengine)`

Module contents

Submodules

decisionengine.framework.logicengine.BooleanExpression module

class decisionengine.framework.logicengine.BooleanExpression.**BooleanExpression**(*expr*)

Bases: object

evaluate(*d*)

Return the evaluated Boolean value of this expression in the context of the given data 'd'.

exception decisionengine.framework.logicengine.BooleanExpression.**LogicError**

Bases: TypeError

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

decisionengine.framework.logicengine.BooleanExpression.**maybe_fail_on_error**(*expr*)

decisionengine.framework.logicengine.FactLookup module

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

Bases: object

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

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

As an example, consider the following configuration:

```
facts: { should_publish: "(True)",
}, rules: {
  publish_1: { expression: "should_publish", facts: ["should_publish"]
}, publish_2: {
  expression: "should_publish", actions: ["go_to_press"] facts: ["should_publish"]
} retract: {
  expression: "not should_publish", facts: ["should_retract"]
}
```

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

rule_for(*fact_name*)

Selects rule required to evaluate fact with the supplied name.

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

Return type str

Returns Rule name

sorted_rules(*rules_cfg*)

Rules sorted according to rule dependencies.

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

Return type list

Returns Rules to be evaluated by the rule engine.

decisionengine.framework.logicengine.LogicEngine module

class decisionengine.framework.logicengine.LogicEngine.**LogicEngine**(*cfg*)

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

_create_facts_dataframe(*newfacts*)

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

facts dict format: 'newfacts': {

```
    'publish_glidein_requests': { 'allow_hpc_new': True, 'allow_foo': True
    }, 'dummy_rule': {
        'dummy_new_fact': True
    }
}
```

consumes()

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

evaluate(*db*)

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

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

evaluate_facts(*db*)

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

Return type dict

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

produces()

decisionengine.framework.logicengine.LogicEngine.**passthrough_configuration**(*publisher_names*)

Assembles logic-engine configuration to unconditionally execute all publishers.

decisionengine.framework.logicengine.Rule module

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

Bases: object

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

evaluate(*evaluated_facts*)

Evaluates a compiled expression given the supplied facts.

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

Return type bool

decisionengine.framework.logicengine.RuleEngine module

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

Bases: object

Engine responsible for evaluating logic-engine rules.

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

execute(*evaluated_facts*)

Evaluates all rules given the supplied facts.

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

Return type tuple

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

Module contents

decisionengine.framework.modules package

Subpackages

decisionengine.framework.modules.tests package

Submodules

decisionengine.framework.modules.tests.test_EmptySource module

decisionengine.framework.modules.tests.test_EmptySource.**test_empty_source_structure**()

decisionengine.framework.modules.tests.test_EmptySource.**test_missing_data_product_name_not_supported**()

decisionengine.framework.modules.tests.test_Module module

`decisionengine.framework.modules.tests.test_Module.test_module_structure()`

The module.Module itself is a bit of a skeleton...

decisionengine.framework.modules.tests.test_Publisher module

`decisionengine.framework.modules.tests.test_Publisher.test_publisher_structure()`

The module.publisher itself is a bit of a skeleton...

decisionengine.framework.modules.tests.test_QueueLogger module

`decisionengine.framework.modules.tests.test_QueueLogger.handler_setup()`

`decisionengine.framework.modules.tests.test_QueueLogger.log_setup()`

`decisionengine.framework.modules.tests.test_QueueLogger.queue_logger_setup()`

`decisionengine.framework.modules.tests.test_QueueLogger.setup_queue_logging(queue_logger_setup,
log_setup,
handler_setup)`

`decisionengine.framework.modules.tests.test_QueueLogger.test_setup_queue_logging(queue_logger_setup,
log_setup,
han-
dler_setup)`

`decisionengine.framework.modules.tests.test_QueueLogger.test_start_queue_logger(queue_logger_setup,
log_setup,
han-
dler_setup)`

`decisionengine.framework.modules.tests.test_QueueLogger.test_stop_queue_logger(queue_logger_setup,
log_setup,
han-
dler_setup)`

decisionengine.framework.modules.tests.test_Source module

`decisionengine.framework.modules.tests.test_Source.test_source_structure()`

The module.Source itself is a bit of a skeleton...

decisionengine.framework.modules.tests.test_Transform module

decisionengine.framework.modules.tests.test_Transform.test_transform_structure()

The module.Transform itself is a bit of a skeleton...

decisionengine.framework.modules.tests.test_de_logger module

decisionengine.framework.modules.tests.test_de_logger.log_setup()

decisionengine.framework.modules.tests.test_de_logger.test_by_nonsense_is_err(log_setup)

decisionengine.framework.modules.tests.test_de_logger.test_by_size(log_setup)

decisionengine.framework.modules.tests.test_de_logger.test_by_time(log_setup)

decisionengine.framework.modules.tests.test_module_decorators module

decisionengine.framework.modules.tests.test_module_decorators.test_multiple_consumes_declarations()

decisionengine.framework.modules.tests.test_module_decorators.test_multiple_produces_declarations()

decisionengine.framework.modules.tests.test_module_decorators.test_supports_config()

decisionengine.framework.modules.tests.test_module_decorators.test_wrong_product_names()

decisionengine.framework.modules.tests.test_module_decorators.test_wrong_product_types()

decisionengine.framework.modules.tests.test_translate_product_name module

decisionengine.framework.modules.tests.test_translate_product_name.test_translate_all()

decisionengine.framework.modules.tests.test_translate_product_name.test_translate_illegal_characters()

decisionengine.framework.modules.tests.test_translate_product_name.test_translate_none()

decisionengine.framework.modules.tests.test_translate_product_name.test_translate_simple()

decisionengine.framework.modules.tests.test_translate_product_name.test_translate_with_underscores()

Module contents

Submodules

decisionengine.framework.modules.EmptySource module

This dummy source takes the name of a source datablock from config file as parameter “data_product_name” and produces an empty pandas DataFrame as a datablock with that name

class decisionengine.framework.modules.EmptySource.EmptySource(*config*)

Bases: *decisionengine.framework.modules.Source.Source*


```
_supported_config = {'data_product_name': (<class 'str'>, '', None)}  
acquire()
```

decisionengine.framework.modules.Module module

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

Bases: object

A skeleton of a module

```
get_data_block()
```

```
get_parameters()
```

```
set_data_block(data_block)
```

```
decisionengine.framework.modules.Module.consumes(**kwargs)
```

```
decisionengine.framework.modules.Module.produces(**kwargs)
```

```
decisionengine.framework.modules.Module.verify_products(producer, data)
```

decisionengine.framework.modules.Publisher module

```
class decisionengine.framework.modules.Publisher.Parameter(name, type=None, default=None,  
                                                           comment=None)
```

Bases: object

```
class decisionengine.framework.modules.Publisher.Publisher(set_of_parameters)
```

Bases: [decisionengine.framework.modules.Module.Module](#)

```
_consumes = {}
```

```
publish(data_block=None)
```

```
shutdown()
```

```
decisionengine.framework.modules.Publisher.consumes(**kwargs)
```

```
decisionengine.framework.modules.Publisher.describe(cls, program_options=<class 'decisionengine.framework.modules.describe.ModuleProgramOptions'>)
```

```
decisionengine.framework.modules.Publisher.supports_config(*args)
```

decisionengine.framework.modules.QueueLogger module

```
class decisionengine.framework.modules.QueueLogger.QueueLogger
```

Bases: object

```
configure_listener(handlers)
```

```
format_logger(logger)
```

```
initialize_q()
setup_queue_logging(logger, handlers)
start()
stop()
```

decisionengine.framework.modules.Source module

```
class decisionengine.framework.modules.Source.Parameter(name, type=None, default=None,
                                                         comment=None)
```

Bases: object

```
class decisionengine.framework.modules.Source.Source(set_of_parameters)
```

Bases: [decisionengine.framework.modules.Module.Module](#)

```
_produces = {}
```

```
acquire()
```

```
decisionengine.framework.modules.Source.describe(cls, sample_config=None)
```

```
decisionengine.framework.modules.Source.produces(**kwargs)
```

```
decisionengine.framework.modules.Source.supports_config(*args)
```

decisionengine.framework.modules.Transform module

```
class decisionengine.framework.modules.Transform.Parameter(name, type=None, default=None,
                                                            comment=None)
```

Bases: object

```
class decisionengine.framework.modules.Transform.Transform(set_of_parameters)
```

Bases: [decisionengine.framework.modules.Module.Module](#)

```
_consumes = {}
```

```
_produces = {}
```

```
transform()
```

```
decisionengine.framework.modules.Transform.consumes(**kwargs)
```

```
decisionengine.framework.modules.Transform.describe(cls, program_options=<class 'decisionengine.framework.modules.describe.ModuleProgramOptions'>)
```

```
decisionengine.framework.modules.Transform.produces(**kwargs)
```

```
decisionengine.framework.modules.Transform.supports_config(*args)
```

decisionengine.framework.modules.de_logger module

Logger to use in all modules

```
decisionengine.framework.modules.de_logger.configure_logging(log_level='DEBUG',
                                                            file_rotate_by='size',
                                                            rotation_time_unit='D',
                                                            rotation_interval=1,
                                                            max_backup_count=6,
                                                            max_file_size=200000000,
                                                            log_file_name='/tmp/decision_engine_logs/decisioneng
                                                            start_q_logger=True')
```

Parameters

- **log_level** (str) – log level
- **file_rotate_by** – files rotation by size or by time
- **rotation_time_unit** (str) – unit of time for file rotation
- **rotation_interval** (int) – time in rotation_time_units between file rotations
- **log_file_name** (str) – log file name
- **max_file_size** (int) – maximal size of log file. If reached save and start new log.
- **max_backup_count** (int) – start rotaion after this number is reached

Return type None

```
decisionengine.framework.modules.de_logger.get_logger()
```

get default logger - "decisionengine" :rtype: logging.Logger - rotating file logger

```
decisionengine.framework.modules.de_logger.get_queue_logger()
```

get QueueLogger which owns the logging queues and listeners :rtype: decisionengine.framework.modules.QueueLogger`

```
decisionengine.framework.modules.de_logger.stop_queue_logger()
```

decisionengine.framework.modules.describe module

```
class decisionengine.framework.modules.describe.ModuleProgramOptions(module_spec, cls)
```

Bases: object

process_args()

```
class decisionengine.framework.modules.describe.Parameter(name, type=None, default=None,
                                                         comment=None)
```

Bases: object

```
decisionengine.framework.modules.describe._par_default(par_type, default_value)
```

```
decisionengine.framework.modules.describe._par_type(par_type, default_value)
```

```
decisionengine.framework.modules.describe.main_wrapper(cls, program_options=<class 'decision-
engine.framework.modules.describe.ModuleProgramOptions'
```

```
decisionengine.framework.modules.describe.supports_config(*args)
```

decisionengine.framework.modules.logging_configDict module

Global Logger config dictionary used by all loggers (in their own subkeys)

decisionengine.framework.modules.print_description module

```
decisionengine.framework.modules.print_description._print_comment(comment)
decisionengine.framework.modules.print_description._print_type(type_or_value)
decisionengine.framework.modules.print_description._print_value(v)
decisionengine.framework.modules.print_description._spec_from_file_name(filename)
decisionengine.framework.modules.print_description.print_consumes(cls)
decisionengine.framework.modules.print_description.print_produces(cls)
decisionengine.framework.modules.print_description.print_supported_config(module_spec, cls)
decisionengine.framework.modules.print_description.spec_if_main(cls)
```

decisionengine.framework.modules.translate_product_name module

```
decisionengine.framework.modules.translate_product_name.translate(spec)
    Break apart the string 'old -> new' into a tuple ('old', 'new')
decisionengine.framework.modules.translate_product_name.translate_all(specs)
```

Module contents

decisionengine.framework.taskmanager package

Submodules

decisionengine.framework.taskmanager.LatestMessages module

The LatestMessages class listens for messages from a set of queues and retains only the last unconsumed message from each queue.

The latest messages are consumed by calling the consume() instance method, which returns a dictionary whose key is the message routing key and whose value is the full message. If no messages are available, consume() returns an empty dictionary.

The LatestMessages class is intended to be used as a context manager (e.g.):

```
with LatestMessages(queues, broker_url) as messages:
    while some_predicate():
        msgs = messages.consume()
        if not msgs:
            continue
    for routing_key, msg in msgs.items(): ...
```

Upon exiting the context, a LatestMessage object will no longer listen for any messages.

```
class decisionengine.framework.taskmanager.LatestMessages.LatestMessages(queues, broker_url)
    Bases: object
    _listen()
    consume()
        Return dictionary of latest messages, keyed by message routing key.
```

decisionengine.framework.taskmanager.ProcessingState module

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

BOOT IDLE ACTIVE STEADY OFFLINE SHUTTINGDOWN SHUTDOWN ERROR

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

The ‘RUNNING_CONDITIONS’ list is a list of states that a thread may have if it is started/starting. The ‘STOPPING_CONDITIONS’ list is a list of states that a thread may have if it is stopped/stopping. The ‘INACTIVE_CONDITIONS’ list is a list of states that a thread may have when it is not active

```
class decisionengine.framework.taskmanager.ProcessingState.ProcessingState(state=State.BOOT)
    Bases: object
    This object tracks the state of a process.
    A number of convience wrappers are provided.
    Additionally you may use the .lock attribute for with block to lock the state during specific operations.
    get()
        This function is a minimally locking check to fetch the state.
    get_state_value()
    has_value(state)
    inactive()
    property lock
    probably_running()
    set(state)
        This function will lock (and possibly block) to ensure a consistent change to the state value.
        This function can be blocked using the .lock to force state sync between threads if need be.
    should_stop()
    wait_until(state, timeout=None)
    wait_while(state, timeout=None)

class decisionengine.framework.taskmanager.ProcessingState.State(value)
    Bases: enum.Enum
    An enumeration.
    ACTIVE = 2
```

```
BOOT = 0
ERROR = 7
IDLE = 1
OFFLINE = 6
SHUTDOWN = 5
SHUTTINGDOWN = 4
STEADY = 3
```

decisionengine.framework.taskmanager.SourceProductCache module

```
class decisionengine.framework.taskmanager.SourceProductCache.SourceProductCache(expected_products,  
                                                                                   logger)
```

Bases: object

```
update(new_data)
```

decisionengine.framework.taskmanager.TaskManager module

Task manager

```
class decisionengine.framework.taskmanager.TaskManager.TaskManager(name, workers, dataspace,  
                                                                    expected_products,  
                                                                    exchange, broker_url,  
                                                                    queue_info)
```

Bases: object

Task manager

```
data_block_put(data, header, data_block)
```

Put data into data block

Parameters

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

```
decision_cycle()
```

Decision cycle to be run periodically (by trigger)

```
get_consumes()
```

```
get_loglevel()
```

```
get_produces()
```

```
get_state()
```

```
get_state_name()
```

get_state_value()

run_cycle(*messages*)

run_cycles()

Task manager main loop

run_logic_engine(*data_block*)

Run Logic Engine.

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

run_publishers(*actions, data_block*)

Run Publishers in main process.

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

run_transform(*worker, data_block*)

Run a transform

Parameters

- **worker** (Worker) – Transform 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_value(*log_level*)

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

take_offline()

Adjust status to stop the decision cycles and bring the task manager offline

decisionengine.framework.taskmanager.module_graph module

Ensure no circularities in produces and consumes.

class decisionengine.framework.taskmanager.module_graph.**Worker**(*key, conf_dict, base_class, channel_name*)

Bases: object

Provides interface to loadable modules an events to synchronise execution

decisionengine.framework.taskmanager.module_graph.**_consumed_products**(**worker_lists*)

decisionengine.framework.taskmanager.module_graph.**_create_module_instance**(*config_dict, base_class, channel_name*)

Create instance of dynamically loaded module

decisionengine.framework.taskmanager.module_graph.**_find_only_one_subclass**(*module, base_class*)

Search through module looking for only one subclass of the supplied base_class

```
decisionengine.framework.taskmanager.module_graph._make_workers_for(configs, base_class,  
                                                                    channel_name)
```

```
decisionengine.framework.taskmanager.module_graph._produced_products(*worker_lists)
```

```
decisionengine.framework.taskmanager.module_graph.channel_workers(channel_name,  
                                                                    channel_config, logger)
```

```
decisionengine.framework.taskmanager.module_graph.ensure_no_circularities(sources, transforms,  
                                                                    publishers)
```

Ensures no circularities among data products.

```
decisionengine.framework.taskmanager.module_graph.source_products(source_workers)
```

```
decisionengine.framework.taskmanager.module_graph.validated_workflow(channel_name, sources,  
                                                                    channel_config, log-  
                                                                    ger=<BoundLoggerLazyProxy(logger=None  
                                                                    wrapper_class=None,  
                                                                    processors=None,  
                                                                    context_class=None,  
                                                                    initial_values={}, log-  
                                                                    ger_factory_args=())>)
```

Module contents

decisionengine.framework.tests package

Submodules

decisionengine.framework.tests.ABTransform module

```
class decisionengine.framework.tests.ABTransform.ABTransform(module_parameters, *args,  
                                                            **kwargs)
```

Bases: *decisionengine.framework.modules.Transform.Transform*

```
_consumes = {'B': None}
```

```
_produces = {'A': None}
```

decisionengine.framework.tests.BATransform module

```
class decisionengine.framework.tests.BATransform.BATransform(module_parameters, *args,  
                                                            **kwargs)
```

Bases: *decisionengine.framework.modules.Transform.Transform*

```
_consumes = {'A': None}
```

```
_produces = {'B': None}
```


decisionengine.framework.tests.DynamicPublisher module

```
class decisionengine.framework.tests.DynamicPublisher.DynamicPublisher(config)
    Bases: decisionengine.framework.modules.Publisher.Publisher
    _supported_config = {'consumes': (<class 'list'>, None, None), 'expects': (<class
    'int'>, None, None)}
    publisher(data_block)
```

decisionengine.framework.tests.DynamicSource module

```
class decisionengine.framework.tests.DynamicSource.DynamicSource(config)
    Bases: decisionengine.framework.modules.Source.Source
    _supported_config = {'data_product_name': (<class 'str'>, None, None)}
    acquire()
```

decisionengine.framework.tests.DynamicTransform module

```
class decisionengine.framework.tests.DynamicTransform.DynamicTransform(config)
    Bases: decisionengine.framework.modules.Transform.Transform
    _supported_config = {'consumes': (<class 'list'>, None, None), 'data_product_name':
    (<class 'str'>, None, None)}
    transform(data_block)
```

decisionengine.framework.tests.ErrorOnAcquire module

```
class decisionengine.framework.tests.ErrorOnAcquire.ErrorOnAcquire(config)
    Bases: decisionengine.framework.modules.Source.Source
    _produces = {'_placeholder': None}
    acquire()
```

decisionengine.framework.tests.FailingPublisher module

```
class decisionengine.framework.tests.FailingPublisher.FailingPublisher(module_parameters,
                                                                    *args, **kwargs)
    Bases: decisionengine.framework.modules.Publisher.Publisher
    _consumes = {'bar': None}
    publish(data_block)
```

decisionengine.framework.tests.FailingSourceNOP module

```
class decisionengine.framework.tests.FailingSourceNOP.SourceWithMissingProduces(set_of_parameters)
    Bases: decisionengine.framework.modules.Source.Source
```

decisionengine.framework.tests.IntSource module

```
class decisionengine.framework.tests.IntSource.IntSource(config)
    Bases: decisionengine.framework.modules.Source.Source
    _produces = {'int_value': <class 'int'>}
    _supported_config = {'int_value': (<class 'int'>, None, None)}
    acquire()
```

decisionengine.framework.tests.ModuleProgramOptions module

```
class decisionengine.framework.tests.ModuleProgramOptions.AcquireWithConfig(name)
    Bases: object
    test(byte_str, expected_stderr="")

class decisionengine.framework.tests.ModuleProgramOptions.AcquireWithSampleConfig(name)
    Bases: object
    test()

class decisionengine.framework.tests.ModuleProgramOptions.ConfigTemplate(name)
    Bases: object
    test(has_comments=False)

class decisionengine.framework.tests.ModuleProgramOptions.Describe(name)
    Bases: object
    test(consumes=None, produces=None)

class decisionengine.framework.tests.ModuleProgramOptions.DescribeAlias(alias, original)
    Bases: object
    test()

class decisionengine.framework.tests.ModuleProgramOptions.Help(name)
    Bases: object
    test(has_sample_config=False)

decisionengine.framework.tests.ModuleProgramOptions._expected_acquire_result(name, con-
fig_file=None,
multiplier=1,
chan-
nel_name='test1')
```

```
decisionengine.framework.tests.ModuleProgramOptions._expected_config_template(name)
decisionengine.framework.tests.ModuleProgramOptions._expected_config_template_with_comments(name)
decisionengine.framework.tests.ModuleProgramOptions._expected_help(name)
decisionengine.framework.tests.ModuleProgramOptions._expected_source_help(name,
                                                                    has_sample_config=False)
decisionengine.framework.tests.ModuleProgramOptions._normalize(string)
decisionengine.framework.tests.ModuleProgramOptions._run_as_main(name, *program_options)
```

decisionengine.framework.tests.PublisherNOP module

```
class decisionengine.framework.tests.PublisherNOP.PublisherNOP(module_parameters, *args,
                                                                **kwargs)
    Bases: decisionengine.framework.modules.Publisher.Publisher
    _consumes = {'bar': <class 'pandas.core.frame.DataFrame'>}
    publish(data_block)
```

decisionengine.framework.tests.PublisherWithMissingConsumes module

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

decisionengine.framework.tests.SourceAlias module

decisionengine.framework.tests.SourceNOP module

```
class decisionengine.framework.tests.SourceNOP.SourceNOP(config)
    Bases: decisionengine.framework.modules.Source.Source
    _produces = {'foo': <class 'pandas.core.frame.DataFrame'>}
    acquire()
```

decisionengine.framework.tests.SourceWithSampleConfigNOP module

```
class decisionengine.framework.tests.SourceWithSampleConfigNOP.SourceWithSampleConfigNOP(config)
    Bases: decisionengine.framework.modules.Source.Source
    _produces = {'foo': <class 'pandas.core.frame.DataFrame'>}
    _supported_config = {'channel_name': (<class 'str'>, None, None), 'multiplier':
                          (<class 'int'>, None, None)}
    acquire()
```

decisionengine.framework.tests.SupportsConfigPublisher module

```
class decisionengine.framework.tests.SupportsConfigPublisher.SupportsConfig(set_of_parameters)
    Bases: decisionengine.framework.modules.Publisher.Publisher
    _supported_config = {'comment': (<class 'str'>, None, 'Single-line comment'),
        'comment_with_nl': (<class 'str'>, None, 'Comment with newline\n'), 'convert_to':
        (<class 'int'>, 3, None), 'default_only': (<class 'float'>, 2.5, None), 'no_type':
        (None, None, None), 'only_type': (<class 'int'>, None, None)}
```

decisionengine.framework.tests.TransformNOP module

```
class decisionengine.framework.tests.TransformNOP.TransformNOP(module_parameters, *args,
                                                                **kwargs)
    Bases: decisionengine.framework.modules.Transform.Transform
    _consumes = {'foo': <class 'pandas.core.frame.DataFrame'>}
    _produces = {'bar': <class 'pandas.core.frame.DataFrame'>}
    transform(data_block)
```

decisionengine.framework.tests.TransformWithMissingProducesConsumes module

```
class decisionengine.framework.tests.TransformWithMissingProducesConsumes.TransformWithMissingProducesConsumesC
    Bases: decisionengine.framework.modules.Transform.Transform
    transform(data_block)
```

decisionengine.framework.tests.WriteToDisk module

Special publisher used to register publish calls with an external file.

It is difficult to interact with individual publishers while testing a workflow. The WriteToDisk publisher therefore writes to an external file that can be read by a test. Ideally, we would implement a system so that the test and any instance of the WriteToDisk class are passed the same file-name string. Unfortunately, this is non-trivial to achieve without adjusting the behavior of the decision-engine server itself. We therefore choose the following abstruse logic:

- WriteToDisk creates a temporary file and broadcasts its name to STDOUT. Note that this temporary file must be uniquely named as multiple tests can use WriteToDisk in parallel.
- Capture STDOUT in the DETestWorker class.
- Pass STDOUT to the 'wait_for_n_writes' which will wait until the number 'n' appears in the file.

```
class decisionengine.framework.tests.WriteToDisk.WriteToDisk(config)
    Bases: decisionengine.framework.modules.Publisher.Publisher
    _supported_config = {'consumes': (<class 'list'>, None, None), 'filename': (<class
    'str'>, None, None)}
    publish(data_block)
```

```
decisionengine.framework.tests.WriteToDisk.wait_for_n_writes(stdout, n)
```

decisionengine.framework.tests.fixtures module

defaults for pytest

```
decisionengine.framework.tests.fixtures.DEServer(conf_path=None, conf_override=None,
                                                  channel_conf_path=None,
                                                  channel_conf_override=None, host='127.0.0.1',
                                                  port=None, make_conf_dirs_if_missing=False)
```

A DE Server using a private database

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

Fixture factory for PostgreSQL.

Parameters **request** – fixture request object

Returns postgresql client

```
decisionengine.framework.tests.fixtures.PG_PROG(request: _pytest.fixtures.FixtureRequest,
                                                  tmp_path_factory: _pytest.tmpdir.TempPathFactory)
→
Iterator[pytest_postgresql.executor.PostgreSQLExecutor]
```

Process fixture for PostgreSQL.

Parameters

- **request** – fixture request object
- **tmp_path_factory** – temporary path object (fixture)

Returns tcp executor

```
decisionengine.framework.tests.fixtures.SQLALCHEMY_PG_WITH_SCHEMA(PG_DE_DB_WITHOUT_SCHEMA)
```

Get a blank database from pytest_postgresql. Then setup the SQLAlchemy style URL with that DB. The SQLAlchemyDS will create the schema as needed.

```
decisionengine.framework.tests.fixtures.SQLALCHEMY_TEMPFILE_SQLITE(tmp_path)
```

Setup an SQLite database with the pytest tmp_path fixture. Then setup the SQLAlchemy style URL with that DB. The SQLAlchemyDS will create the schema as needed.

decisionengine.framework.tests.test_client_errors module

Fixture based DE Server tests of the sample config

```
decisionengine.framework.tests.test_client_errors.test_client_cannot_wait_on_bad_state(deserver)
```

Verify wait is for a valid state

decisionengine.framework.tests.test_client_server module

Fixture based DE Server for the de-client tests

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

`decisionengine.framework.tests.test_client_server.test_client_set_loglevel(deserver)`

`decisionengine.framework.tests.test_client_server.test_client_status_msg_to_stdout(deserver)`

Make sure the actual client console call goes to stdout

decisionengine.framework.tests.test_combined_channels module

`decisionengine.framework.tests.test_combined_channels.test_combined_channels(deserver)`

`decisionengine.framework.tests.test_combined_channels.test_combined_channels_3g(deserver_combined)`

decisionengine.framework.tests.test_defaults module

Fixture based DE Server tests of the sample config

`decisionengine.framework.tests.test_defaults.test_defaults(deserver)`

decisionengine.framework.tests.test_dynamic_test_modules module

`decisionengine.framework.tests.test_dynamic_test_modules.test_dynamic_publisher()`

`decisionengine.framework.tests.test_dynamic_test_modules.test_dynamic_source()`

`decisionengine.framework.tests.test_dynamic_test_modules.test_dynamic_transform()`

decisionengine.framework.tests.test_empty_config module

Fixture based DE Server tests of adding a channel later on

`decisionengine.framework.tests.test_empty_config.test_client_can_start_one_channel_added_after_startup()`

Verify client can start a single channel

decisionengine.framework.tests.test_error_on_acquire module

`decisionengine.framework.tests.test_error_on_acquire.test_source_only_channel(deserver)`

decisionengine.framework.tests.test_module_program_options module

`decisionengine.framework.tests.test_module_program_options.test_acquire_for_sources()`

`decisionengine.framework.tests.test_module_program_options.test_config_templates()`

`decisionengine.framework.tests.test_module_program_options.test_descriptions()`

`decisionengine.framework.tests.test_module_program_options.test_help()`

`decisionengine.framework.tests.test_module_program_options.test_module_alias()`

decisionengine.framework.tests.test_query_tool_server module

Fixture based DE Server for the de-query-tool tests

`decisionengine.framework.tests.test_query_tool_server.test_query_tool(deserver)`

decisionengine.framework.tests.test_reaper module

Fixture based DE Server for the reaper tests

`decisionengine.framework.tests.test_reaper.test_client_can_get_de_server_reaper_status(deserver)`

Verify reaper status

decisionengine.framework.tests.test_same_source_types module

`decisionengine.framework.tests.test_same_source_types.test_same_source_types_separate_channels(deserver)`

decisionengine.framework.tests.test_sample_config module

Fixture based DE Server tests of the defaults

`decisionengine.framework.tests.test_sample_config.stopped_channel_opts(timeout=1)`

`decisionengine.framework.tests.test_sample_config.test_client_can_get_de_server_status(deserver)`

`decisionengine.framework.tests.test_sample_config.test_client_can_kill_one_channel(deserver)`

`decisionengine.framework.tests.test_sample_config.test_client_can_restart_all_channels(deserver)`

Verify client can get channel products even when none are run

`decisionengine.framework.tests.test_sample_config.test_client_can_restart_one_channel(deserver)`

Verify client can restart a single channel

`decisionengine.framework.tests.test_sample_config.test_client_logger_level(deserver)`

`decisionengine.framework.tests.test_sample_config.test_client_non_real_channel(deserver)`

decisionengine.framework.tests.test_shared_sources module

decisionengine.framework.tests.test_shared_sources.**record_that_matches**(*substring, records*)

decisionengine.framework.tests.test_shared_sources.**test_conflicting_source_configurations**(*deserver_conflict, caplog*)

decisionengine.framework.tests.test_shared_sources.**test_shared_source**(*deserver_shared, caplog*)

decisionengine.framework.tests.test_start_with_bad_channels module

Fixture based DE Server tests of invalid channel configs

decisionengine.framework.tests.test_start_with_bad_channels.**_consumes_not_subset**(*test_str*)

decisionengine.framework.tests.test_start_with_bad_channels.**_expected_circularity**(*test_str*)

decisionengine.framework.tests.test_start_with_bad_channels.**_missing_consumes**(*name*)

decisionengine.framework.tests.test_start_with_bad_channels.**_missing_produces**(*name*)

decisionengine.framework.tests.test_start_with_bad_channels.**test_client_can_get_products_no_channels**(*deserver_conflict, caplog*)

Verify client can get channel products even when none are run

Module contents

decisionengine.framework.util package

Submodules

decisionengine.framework.util.countdown module

class decisionengine.framework.util.countdown.**Countdown**(*wait_up_to*)

Bases: object

Countdown is a context manager that keeps track of elapsed time.

It is designed to be used for cases where a sequence of operations should not take longer than a specified period of time. This is done by occasionally querying the 'time_left' attribute (e.g.):

countdown = Countdown(wait_up_to=10) for p in processes:

with countdown: rc = p.join(countdown.time_left) if rc is None:

p.terminate()

In the above example, the time it takes to shutdown all processes should not exceed 10 seconds. Upon entering the countdown context, a timer starts. Once that context is exited, the timer stops and the elapsed time is subtracted from the initial 'wait_up_to' value. If it takes 10 seconds for the first process to join, then the 'time_left' value will be 0 when joining all subsequent processes. In this way, the entire sequence of operations is constrained to occur in roughly 10 seconds.

decisionengine.framework.util.fs module

`decisionengine.framework.util.fs.files_with_extensions(dir_path, *extensions)`

Return all files in `dir_path` that match the provided extensions.

If no extensions are given, then all files in `dir_path` are returned.

Results are sorted by channel name to ensure stable output.

decisionengine.framework.util.logparser module

`decisionengine.framework.util.logparser.console_scripts_main(args_to_parse=None)`

This is the entry point for the setuptools auto generated scripts. Setuptools thinks a return from this function is an error message.

`decisionengine.framework.util.logparser.create_parser()`

`decisionengine.framework.util.logparser.execute_command_from_args(argsparsed, logfile=None, constraint=None)`

Parse the log file as requested.

Parameters

- **argsparsed** (*Namespace*) – Parsed arguments from `create_parser` in this file.
- **logfile** (*path*) – Log file path.
- **constraint** (*dict*) – Combined constraints dictionary

Returns Output of the command.

Return type str

`decisionengine.framework.util.logparser.main(args_to_parse=None)`

Main function for logparser

Parameters

- **args_to_parse** (*list, optional*) – If you pass a list of args, they will be used instead of `sys.argv`.
- **None.** (*Defaults to*) –

Returns Parsing result

Return type str

`decisionengine.framework.util.logparser.matches_constraint(constraint, linelist, linedict)`

Return True if all constraints are matched

Parameters

- **constraint** (*dict / None*) – combined constraints
- **linelist** (*list*) – List of line fields
- **linedict** (*dict*) – Dictionary with structured elements

Returns True if all constraints are matched, False otherwise

Return type bool

`decisionengine.framework.util.logparser.parse_constraints(constraints, loglevel=None)`

Parse and combine the constraints

Parameters

- **constraints** (*list/None*) – List of constraints
- **loglevel** (*str*) – Logging level, e.g. DEBUG, INFO, ...

Returns combined constraint dictionary

Return type dict

decisionengine.framework.util.metrics module

```
class decisionengine.framework.util.metrics.Counter(name: str, documentation: str, labelnames:
                                                    typing.Iterable[str] = (), namespace: str
                                                    = "", subsystem: str = "", unit: str = "", registry: typ-
ing.Optional[prometheus_client.registry.CollectorRegistry]
                                                    = <prometheus_client.registry.CollectorRegistry
object>, _labelvalues:
                                                    typing.Optional[typing.Sequence[str]] = None)
```

Bases: `prometheus_client.metrics.Counter`

`_abc_impl = <_abc._abc_data object>`

```
class decisionengine.framework.util.metrics.Gauge(*args, **kwargs)
```

Bases: `prometheus_client.metrics.Gauge`

Override prometheus client Gauge so that `multiprocess_mode` 'liveall' is the default as opposed to 'all'

`_DEFAULT_MULTIPROC_MODE = 'liveall'`

`__determine_multiprocess_mode_existence(*args, **kwargs)`

`_abc_impl = <_abc._abc_data object>`

```
class decisionengine.framework.util.metrics.Histogram(name: str, documentation: str, labelnames:
                                                       typing.Iterable[str] = (), namespace: str = "",
                                                       subsystem: str = "", unit: str = "", registry: typ-
ing.Optional[prometheus_client.registry.CollectorRegistry]
                                                       =
                                                       <prometheus_client.registry.CollectorRegistry
object>, _labelvalues:
                                                       typing.Optional[typing.Sequence[str]] =
                                                       None, buckets:
                                                       typing.Sequence[typing.Union[float, str]] =
                                                       (0.005, 0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5,
                                                       0.75, 1.0, 2.5, 5.0, 7.5, 10.0, inf))
```

Bases: `prometheus_client.metrics.Histogram`

`_abc_impl = <_abc._abc_data object>`

```

class decisionengine.framework.util.metrics.Summary(name: str, documentation: str, labelnames:
    typing.Iterable[str] = (), namespace: str
    = "", subsystem: str = "", unit: str = "", registry: typing.Optional[prometheus_client.registry.CollectorRegistry]
    = <prometheus_client.registry.CollectorRegistry
    object>, _labelvalues:
    typing.Optional[typing.Sequence[str]] = None)

    Bases: prometheus_client.metrics.Summary
    _abc_impl = <_abc._abc_data object>

decisionengine.framework.util.metrics.display_metrics()

```

decisionengine.framework.util.reaper module

A stand-alone script purges data in database older than specified in configuration. Configuration file has to have this bit added:

```

{
    "dataspace" [{ "retention_interval_in_days"[365,]
        "datasource": { ... }
    }
}

```

Can be used in a cron job.

```
decisionengine.framework.util.reaper.main()
```

decisionengine.framework.util.redis_stats module

```
decisionengine.framework.util.redis_stats.redis_stats(broker_url, exchange)
```

decisionengine.framework.util.singleton module

```

class decisionengine.framework.util.singleton.ScopedSingleton
    Bases: decisionengine.framework.util.singleton.Singleton
    Singleton pattern using Metaclass with weak refs
    _instances = <WeakValueDictionary>

class decisionengine.framework.util.singleton.ScopedSingletonABC(name, bases, namespace,
    **kwargs)
    Bases: abc.ABCMeta, decisionengine.framework.util.singleton.ScopedSingleton

class decisionengine.framework.util.singleton.Singleton
    Bases: type
    Singleton pattern using Metaclass with strong refs
    _instances = {}

```

```
class decisionengine.framework.util.singleton.SingletonABC(name, bases, namespace, **kwargs)
    Bases: abc.ABCMeta, decisionengine.framework.util.singleton.Singleton
```

decisionengine.framework.util.sockets module

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

decisionengine.framework.util.subclasses module

```
decisionengine.framework.util.subclasses._derived_class(cls, base_class)
```

Only matches subclasses that are not equal to the base class.

```
decisionengine.framework.util.subclasses.all_subclasses(module, base_class)
```

Return all of a module's subclasses of the given base class.

Module contents

Submodules

decisionengine.framework.about module

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

decisionengine.framework.version module

Module contents

decisionengine.tests package

Submodules

decisionengine.tests.test_framework_package module

Make sure decisionengine.framework is a valid python package

```
decisionengine.tests.test_framework_package.test_can_import()
```

Module contents

Module contents

5.2 Indices and tables

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

- search

INDICES AND TABLES

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

PYTHON MODULE INDEX

d

[decisionengine](#), 104
[decisionengine.framework](#), 104
[decisionengine.framework.about](#), 104
[decisionengine.framework.config](#), 43
[decisionengine.framework.config.ChannelConfigHandler](#), 41
[decisionengine.framework.config.policies](#), 42
[decisionengine.framework.config.tests](#), 41
[decisionengine.framework.config.tests.test_config](#), 39
[decisionengine.framework.config.tests.test_de_std](#), 40
[decisionengine.framework.config.tests.test_policies](#), 41
[decisionengine.framework.config.ValidConfig](#), 42
[decisionengine.framework.dataspace](#), 69
[decisionengine.framework.dataspace.datablock](#), 63
[decisionengine.framework.dataspace.datasource](#), 65
[decisionengine.framework.dataspace.datasources](#), 59
[decisionengine.framework.dataspace.datasources.null](#), 56
[decisionengine.framework.dataspace.datasources.sqlalchemy_us](#), 50
[decisionengine.framework.dataspace.datasources.sqlalchemy_us.datasource_api](#), 43
[decisionengine.framework.dataspace.datasources.sqlalchemy_us.db_schema](#), 47
[decisionengine.framework.dataspace.datasources.sqlalchemy_us.utils](#), 50
[decisionengine.framework.dataspace.datasources.tests](#), 56
[decisionengine.framework.dataspace.datasources.tests.fixtures](#), 54
[decisionengine.framework.dataspace.datasources.tests.test_datasource_api](#), 55
[decisionengine.framework.dataspace.dataspace](#), 68
[decisionengine.framework.dataspace.maintain](#), 69
[decisionengine.framework.dataspace.tests](#), 63
[decisionengine.framework.dataspace.tests.fixtures](#), 59
[decisionengine.framework.dataspace.tests.test_datablock](#), 60
[decisionengine.framework.dataspace.tests.test_datablock_zl](#), 61
[decisionengine.framework.dataspace.tests.test_datasource](#), 61
[decisionengine.framework.dataspace.tests.test_dataspace](#), 61
[decisionengine.framework.dataspace.tests.test_Reaper](#), 60
[decisionengine.framework.engine](#), 77
[decisionengine.framework.engine.ChannelWorkers](#), 72
[decisionengine.framework.engine.de_client](#), 76
[decisionengine.framework.engine.de_query_tool](#), 77
[decisionengine.framework.engine.DecisionEngine](#), 73
[decisionengine.framework.engine.SourceWorkers](#), 76
[decisionengine.framework.engine.tests](#), 72
[decisionengine.framework.engine.tests.fixtures](#), 69
[decisionengine.framework.engine.tests.test_ChannelWorkers](#), 70
[decisionengine.framework.engine.tests.test_client_only](#), 71
[decisionengine.framework.engine.tests.test_query_tool_only](#), 71
[decisionengine.framework.engine.tests.test_startup](#), 71
[decisionengine.framework.engine.tests.test_verify_redis_se](#), 72
[decisionengine.framework.logicengine](#), 82
[decisionengine.framework.logicengine.BooleanExpression](#), 80
[decisionengine.framework.logicengine.FactLookup](#),

80	83
decisionengine.framework.logicengine.LogicEngine,	decisionengine.framework.modules.tests.test_Source,
81	83
decisionengine.framework.logicengine.Rule,	decisionengine.framework.modules.tests.test_Transform,
decisionengine.framework.logicengine.RuleEngine,	84
82	decisionengine.framework.modules.tests.test_translate_pro,
decisionengine.framework.logicengine.tests,	84
80	decisionengine.framework.modules.Transform,
decisionengine.framework.logicengine.tests.test_bool_function_name,	86
77	decisionengine.framework.modules.translate_product_name,
decisionengine.framework.logicengine.tests.test_cascaded_rules,	88
78	decisionengine.framework.taskmanager, 92
decisionengine.framework.logicengine.tests.test_decisionengine,	decisionengine.framework.taskmanager.LatestMessages,
78	88
decisionengine.framework.logicengine.tests.test_dispatcher_framework,	decisionengine.framework.taskmanager.module_graph,
78	91
decisionengine.framework.logicengine.tests.test_fact,	decisionengine.framework.taskmanager.ProcessingState,
78	89
decisionengine.framework.logicengine.tests.test_failure,	decisionengine.framework.taskmanager.SourceProductCache,
79	90
decisionengine.framework.logicengine.tests.test_pipeline,	decisionengine.framework.taskmanager.TaskManager,
79	90
decisionengine.framework.logicengine.tests.test_pipeline_factory,	decisionengine.framework.tests, 100
79	decisionengine.framework.tests.ABTransform,
decisionengine.framework.logicengine.tests.test_simpleconfiguration,	92
79	decisionengine.framework.tests.BATransform,
decisionengine.framework.modules, 88	92
decisionengine.framework.modules.de_logger,	decisionengine.framework.tests.DynamicPublisher,
87	93
decisionengine.framework.modules.describe,	decisionengine.framework.tests.DynamicSource,
87	93
decisionengine.framework.modules.EmptySource,	93
84	decisionengine.framework.tests.DynamicTransform,
decisionengine.framework.modules.logging_configDict,	93
88	decisionengine.framework.tests.ErrorOnAcquire,
decisionengine.framework.modules.Module,	93
85	decisionengine.framework.tests.FailingPublisher,
decisionengine.framework.modules.print_descriptions,	93
88	decisionengine.framework.tests.FailingSourceNOP,
decisionengine.framework.modules.Publisher,	94
85	decisionengine.framework.tests.fixtures, 97
decisionengine.framework.modules.QueueLogger,	decisionengine.framework.tests.IntSource, 94
85	decisionengine.framework.tests.ModuleProgramOptions,
decisionengine.framework.modules.Source,	94
86	decisionengine.framework.tests.PublisherNOP,
decisionengine.framework.modules.tests, 84	95
decisionengine.framework.modules.tests.test_decisionengine,	decisionengine.framework.tests.PublisherWithMissingConsumer,
84	95
decisionengine.framework.modules.tests.test_EmptySource,	95
82	decisionengine.framework.tests.SourceAlias,
decisionengine.framework.modules.tests.test_Module,	95
83	decisionengine.framework.tests.SourceNOP, 95
decisionengine.framework.modules.tests.test_module,	decisionengine.framework.tests.SourceWithSampleConfigNOP,
84	95
decisionengine.framework.modules.tests.test_Publisher,	95
83	decisionengine.framework.tests.SupportsConfigPublisher,
decisionengine.framework.modules.tests.test_QueueLogger,	96

- decisionengine.framework.tests.test_client_errors,
 - 97
- decisionengine.framework.tests.test_client_server,
 - 98
- decisionengine.framework.tests.test_combined_channels,
 - 98
- decisionengine.framework.tests.test_defaults,
 - 98
- decisionengine.framework.tests.test_dynamic_test_modules,
 - 98
- decisionengine.framework.tests.test_empty_config,
 - 98
- decisionengine.framework.tests.test_error_on_acquire,
 - 98
- decisionengine.framework.tests.test_module_program_options,
 - 99
- decisionengine.framework.tests.test_query_tool_server,
 - 99
- decisionengine.framework.tests.test_reaper,
 - 99
- decisionengine.framework.tests.test_same_source_types,
 - 99
- decisionengine.framework.tests.test_sample_config,
 - 99
- decisionengine.framework.tests.test_shared_sources,
 - 100
- decisionengine.framework.tests.test_start_with_bad_channels,
 - 100
- decisionengine.framework.tests.TransformNOP,
 - 96
- decisionengine.framework.tests.TransformWithMissingProducesConsumes,
 - 96
- decisionengine.framework.tests.WriteToDisk,
 - 96
- decisionengine.framework.util, 104
- decisionengine.framework.util.countdown, 100
- decisionengine.framework.util.fs, 101
- decisionengine.framework.util.logparser, 101
- decisionengine.framework.util.metrics, 102
- decisionengine.framework.util.reaper, 103
- decisionengine.framework.util.redis_stats,
 - 103
- decisionengine.framework.util.singleton, 103
- decisionengine.framework.util.sockets, 104
- decisionengine.framework.util.subclasses, 104
- decisionengine.framework.version, 104
- decisionengine.tests, 104
- decisionengine.tests.test_framework_package,
 - 104

INDEX

Symbols

- `_DEFAULT_MULTIPROC_MODE` (decisionengine.framework.util.metrics.Gauge attribute), 102
- `__determine_multiprocess_mode_existence()` (decisionengine.framework.util.metrics.Gauge method), 102
- `__insert()` (decisionengine.framework.dataspace.datablock.DataBlock method), 63
- `__update()` (decisionengine.framework.dataspace.datablock.DataBlock method), 63
- `_abc_impl` (decisionengine.framework.config.ValidConfig.ValidConfig attribute), 42
- `_abc_impl` (decisionengine.framework.dataspace.datablock.Header attribute), 64
- `_abc_impl` (decisionengine.framework.dataspace.datablock.Metadata attribute), 64
- `_abc_impl` (decisionengine.framework.dataspace.datasource.DataSource attribute), 65
- `_abc_impl` (decisionengine.framework.dataspace.datasources.null.NullDataSource attribute), 56
- `_abc_impl` (decisionengine.framework.dataspace.datasources.sqlalchemy_us.SQLAlchemyDS attribute), 50
- `_abc_impl` (decisionengine.framework.dataspace.datasources.sqlalchemy_us.datasource_api.SQLAlchemyDS attribute), 44
- `_abc_impl` (decisionengine.framework.util.metrics.Counter attribute), 102
- `_abc_impl` (decisionengine.framework.util.metrics.Gauge attribute), 102
- `_abc_impl` (decisionengine.framework.util.metrics.Histogram attribute), 102
- `_abc_impl` (decisionengine.framework.util.metrics.Summary attribute), 103
- `_channel_config_dir()` (in module decisionengine.framework.config.tests.test_config), 39
- `_channel_preamble()` (in module decisionengine.framework.engine.DecisionEngine), 75
- `_check_keys()` (in module decisionengine.framework.config.ChannelConfigHandler), 41
- `_check_metrics_env()` (in module decisionengine.framework.engine.DecisionEngine), 75
- `_check_override()` (in module decisionengine.framework.engine.tests.test_startup), 71
- `_config_from_file()` (in module decisionengine.framework.config.ValidConfig), 42
- `consumed_products()` (in module decisionengine.framework.taskmanager.module_graph), 91
- `_consumes` (decisionengine.framework.modules.Publisher.Publisher attribute), 85
- `_consumes` (decisionengine.framework.modules.Transform.Transform attribute), 86
- `_consumes` (decisionengine.framework.tests.ABTransform.ABTransform attribute), 92
- `_consumes` (decisionengine.framework.tests.BATransform.BATransform attribute), 92
- `_consumes` (decisionengine.framework.tests.FailingPublisher.FailingPublisher attribute), 93
- `_consumes` (decisionengine.framework.tests.PublisherNOP.PublisherNOP attribute), 95
- `_consumes` (decisionengine.framework.tests.TransformNOP.TransformNOP attribute), 96
- `_consumes_not_subset()` (in module decisionengine.framework.tests.test_start_with_bad_channels), 100
- `create_de_server()` (in module decisionengine.framework.engine.DecisionEngine), 75
- `_create_facts_dataframe()` (decisionengine.framework.logicengine.LogicEngine.LogicEngine method), 81
- `_create_module_instance()` (in module decisionengine.framework.taskmanager.module_graph), 91
- `_dataframe_to_column_names()` (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 73
- `_dataframe_to_csv()` (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 73

`nengine.framework.engine.DecisionEngine.DecisionEngine` `nengine.framework.config.ChannelConfigHandler.ChannelConfig`
`method`), 73 `method`), 41
`_dataframe_to_json()` (`decisionengine.framework.engine.DecisionEngine.DecisionEngine` `nengine.framework.config.ChannelConfigHandler`),
`method`), 73 41
`_dataframe_to_table()` (`decisionengine.framework.engine.DecisionEngine.DecisionEngine` `nengine.framework.taskmanager.module_graph`),
`method`), 73 91
`_dataframe_to_vertical_tables()` (`decisionengine.framework.engine.DecisionEngine.DecisionEngine` `nengine.framework.tests.test_start_with_bad_channels`),
`method`), 73 100
`_derived_class()` (`in module decisionengine.framework.util.subclasses`), 104 `_missing_produces()` (`in module decisionengine.framework.tests.test_start_with_bad_channels`),
`_dispatch()` (`decisionengine.framework.engine.DecisionEngine.DecisionEngine` `finalize()` (`in module decisionengine.framework.tests.ModuleProgramOptions`),
`method`), 73 `method`), 95
`_expected_acquire_result()` (`in module decisionengine.framework.tests.ModuleProgramOptions`), 94 `_par_default()` (`in module decisionengine.framework.modules.describe`), 87
`_expected_circularity()` (`in module decisionengine.framework.tests.test_start_with_bad_channels`), 100 `_par_type()` (`in module decisionengine.framework.modules.describe`), 87
`_expected_config_template()` (`in module decisionengine.framework.tests.ModuleProgramOptions`), 94 `_print_comment()` (`in module decisionengine.framework.modules.print_description`),
`_expected_config_template_with_comments()` (`in module decisionengine.framework.tests.ModuleProgramOptions`), 94 `_print_type()` (`in module decisionengine.framework.modules.print_description`),
`_expected_source_help()` (`in module decisionengine.framework.tests.ModuleProgramOptions`), 95 `_print_value()` (`in module decisionengine.framework.modules.print_description`),
`_expected_help()` (`in module decisionengine.framework.tests.ModuleProgramOptions`), 95 `_produced_products()` (`in module decisionengine.framework.taskmanager.module_graph`),
`_expected_source_help()` (`in module decisionengine.framework.tests.ModuleProgramOptions`), 95 `_produces` (`decisionengine.framework.modules.Source.Source`
`attribute`), 86
`_find_only_one_subclass()` (`in module decisionengine.framework.taskmanager.module_graph`), 91 `_produces` (`decisionengine.framework.modules.Transform.Transform`
`attribute`), 86
`_get_de_conf_manager()` (`in module decisionengine.framework.engine.DecisionEngine`), 75 `_produces` (`decisionengine.framework.tests.ABTransform.ABTransform`
`attribute`), 92
`_get_global_config()` (`in module decisionengine.framework.engine.DecisionEngine`), 75 `_produces` (`decisionengine.framework.tests.BATransform.BATransform`
`attribute`), 92
`_global_config_file()` (`in module decisionengine.framework.config.tests.test_config`), 39 `_produces` (`decisionengine.framework.tests.ErrorOnAcquire.ErrorOnAcquire`
`attribute`), 93
`_instances` (`decisionengine.framework.util.singleton.ScopeSingleton` `attribute`), 103 `_produces` (`decisionengine.framework.tests.IntSource.IntSource`
`attribute`), 94
`_instances` (`decisionengine.framework.util.singleton.Singleton` `attribute`), 103 `_produces` (`decisionengine.framework.tests.SourceNOP.SourceNOP`
`attribute`), 95
`_listen()` (`decisionengine.framework.taskmanager.LatestMessagesReaper.LoopMessages` `method`), 89 `_produces` (`decisionengine.framework.tests.TransformNOP.TransformNOP`
`attribute`), 96
`_load_channel()` (`decisionengine.framework.dataspace.maintain.Reaper.Reaper` `method`), 69

_run_as_main() (in module decisionengine.framework.tests.ModuleProgramOptions), 95
 _sa_class_manager (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Dataproduct attribute), 47
 _sa_class_manager (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Header attribute), 48
 _sa_class_manager (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Metadata attribute), 48
 _sa_class_manager (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Schema attribute), 49
 _sa_class_manager (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Taskmanager attribute), 49
 _sa_registry (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Base attribute), 47
 _setitem() (decisionengine.framework.dataspace.datablock.DataBlock method), 63
 _spec_from_file_name() (in module decisionengine.framework.modules.print_description), 88
 _start_de_server() (in module decisionengine.framework.engine.DecisionEngine), 75
 _supported_config (decisionengine.framework.modules.EmptySource.EmptySource attribute), 84
 _supported_config (decisionengine.framework.tests.DynamicPublisher.DynamicPublisher attribute), 93
 _supported_config (decisionengine.framework.tests.DynamicSource.DynamicSource attribute), 93
 _supported_config (decisionengine.framework.tests.DynamicTransform.DynamicTransform attribute), 93
 _supported_config (decisionengine.framework.tests.IntSource.IntSource attribute), 94
 _supported_config (decisionengine.framework.tests.SourceWithSampleConfigNOP.SourceWithSampleConfigNOP attribute), 95
 _supported_config (decisionengine.framework.tests.SupportsConfigPublisher.SupportsConfigPublisher attribute), 96
 _supported_config (decisionengine.framework.tests.WriteToDisk.WriteToDisk attribute), 96
 _update_channel_states() (decisionengine.framework.engine.ChannelWorkers.ChannelWorkers method), 73
 _verify_redis_server() (in module decisionengine.framework.engine.DecisionEngine), 75
 _verify_redis_url() (in module decisionengine.framework.engine.DecisionEngine), 75
 ABTransform (class in decisionengine.framework.tests.ABTransform), 92
 access() (decisionengine.framework.engine.ChannelWorkers.ChannelWorkers method), 75
 acquire() (decisionengine.framework.modules.EmptySource.EmptySource method), 85
 acquire() (decisionengine.framework.modules.Source.Source method), 86
 acquire() (decisionengine.framework.tests.DynamicSource.DynamicSource method), 93
 acquire() (decisionengine.framework.tests.ErrorOnAcquire.ErrorOnAcquire method), 93
 acquire() (decisionengine.framework.tests.IntSource.IntSource method), 94
 acquire() (decisionengine.framework.tests.SourceNOP.SourceNOP method), 95
 acquire() (decisionengine.framework.tests.SourceWithSampleConfigNOP.SourceWithSampleConfigNOP method), 95
 AcquireWithConfig (class in decisionengine.framework.tests.ModuleProgramOptions), 94
 AcquireWithSampleConfig (class in decisionengine.framework.tests.ModuleProgramOptions), 94
 ACTIVE (decisionengine.framework.taskmanager.ProcessingState.State attribute), 89
 add_engine_pidguard() (in module decisionengine.framework.dataspace.datasources.sqlalchemy_ds.utils), 50
 all_subclasses() (in module decisionengine.util.subclasses), 104

B

Base (class in decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema), 49
 BATransform (class in decisionengine.framework.tests.BATransform), 92
 block_write() (decisionengine.framework.engine.DecisionEngine method), 73
 BooleanExpression (class in decisionengine.framework.logicengine.BooleanExpression), 80

BOOT (decisionengine.framework.taskmanager.ProcessingState attribute), 89

C

channel_config_dir() (in module decisionengine.framework.config.policies), 42

channel_workers() (in module decisionengine.framework.taskmanager.module_graph), 92

ChannelConfigHandler (class in decisionengine.framework.config.ChannelConfigHandler), 41

ChannelWorker (class in decisionengine.framework.engine.ChannelWorkers), 72

ChannelWorkers (class in decisionengine.framework.engine.ChannelWorkers), 72

ChannelWorkers.Access (class in decisionengine.framework.engine.ChannelWorkers), 73

Clean (decisionengine.framework.engine.DecisionEngine.StopState attribute), 75

clone_model() (in module decisionengine.framework.dataspace.datasources.sqlalchemy_datasource), 50

close() (decisionengine.framework.dataspace.datasource.DataSource method), 65

close() (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 56

close() (decisionengine.framework.dataspace.datasources.sqlalchemy_datasource.DataSource method), 44

close() (decisionengine.framework.dataspace.datasources.sqlalchemy_datasource.DataSource method), 50

close() (decisionengine.framework.dataspace.dataspace.DataSpace method), 68

compress() (in module decisionengine.framework.dataspace.datablock), 65

config() (in module decisionengine.framework.config.tests.test_de_std), 40

config() (in module decisionengine.framework.dataspace.tests.test_Reaper), 60

ConfigTemplate (class in decisionengine.framework.tests.ModuleProgramOptions), 94

configure_listener() (decisionengine.framework.modules.QueueLogger.QueueLogger method), 85

configure_logging() (in module decisionengine.framework.modules.de_logger), 87

connect() (decisionengine.framework.dataspace.datasource.DataSource method), 65

connect() (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 56

connect() (decisionengine.framework.dataspace.datasources.sqlalchemy_datasource.DataSource method), 44

connect() (decisionengine.framework.dataspace.datasources.sqlalchemy_datasource.DataSource method), 51

console_scripts_main() (in module decisionengine.framework.engine.de_client), 76

console_scripts_main() (in module decisionengine.framework.engine.de_query_tool), 77

console_scripts_main() (in module decisionengine.framework.util.logparser), 101

consume() (decisionengine.framework.taskmanager.LatestMessages.LatestMessages method), 89

consumes() (decisionengine.framework.logicengine.LogicEngine.LogicEngine method), 81

consumes() (in module decisionengine.framework.modules.Module), 85

consumes() (in module decisionengine.framework.modules.Publisher), 85

consumes() (in module decisionengine.framework.modules.Transform), 86

Countdown (class in decisionengine.framework.util.countdown), 100

Counter (class in decisionengine.framework.util.countdown), 100

create_parser() (in module decisionengine.framework.engine.de_client), 76

create_parser() (in module decisionengine.framework.engine.de_query_tool), 77

create_parser() (in module decisionengine.framework.util.logparser), 101

create_tables() (decisionengine.framework.dataspace.datasource.DataSource method), 65

create_tables() (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 57

create_tables() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource.DataSource method), 44

create_tables() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.SQLAlchemy method), 51

create_time (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema attribute), 48

creator (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource.DataSource attribute), 48

attribute), 48

D

data_block_put() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 90

DataBlock (class in decisionengine.framework.dataspace.datablock), 63

Dataproduct (class in decisionengine.framework.dataspace.datasources.sqlalchemy_utils.py), 47

dataproduct_table (decisionengine.framework.dataspace.datasource.DataSource attribute), 65

DataSource (class in decisionengine.framework.dataspace.datasource), 65

datasource() (in module decisionengine.framework.dataspace.datasources.tests.fixtures), 55

datasource() (in module decisionengine.framework.dataspace.tests.fixtures), 59

DataSpace (class in decisionengine.framework.dataspace.dataspace), 68

dataspace() (in module decisionengine.framework.dataspace.tests.fixtures), 59

DataSpaceConfigurationError, 68

DataSpaceConnectionError, 68

DataSpaceError, 68

DataSpaceExistsError, 68

datestamp (decisionengine.framework.dataspace.datasources.sqlalchemy_utils.py attribute), 49

decision_cycle() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 90

decisionengine module, 104

DecisionEngine (class in decisionengine.framework.engine.DecisionEngine), 73

decisionengine.framework module, 104

decisionengine.framework.about module, 104

decisionengine.framework.config module, 43

decisionengine.framework.config.ChannelConfigurationHandler module, 41

decisionengine.framework.config.policies module, 42

decisionengine.framework.config.tests module, 41

decisionengine.framework.config.tests.test_config module, 39

decisionengine.framework.config.tests.test_de_std module, 40

decisionengine.framework.config.tests.test_policies module, 41

decisionengine.framework.config.ValidConfig module, 42

decisionengine.framework.dataspace module, 69

decisionengine.framework.dataspace.datablock module, 63

decisionengine.framework.dataspace.datasource module, 65

decisionengine.framework.dataspace.datasources module, 59

decisionengine.framework.dataspace.datasources.null module, 56

decisionengine.framework.dataspace.datasources.sqlalchemy module, 50

decisionengine.framework.dataspace.datasources.sqlalchemy module, 43

decisionengine.framework.dataspace.datasources.sqlalchemy module, 47

decisionengine.framework.dataspace.datasources.sqlalchemy module, 50

decisionengine.framework.dataspace.datasources.tests module, 56

decisionengine.framework.dataspace.datasources.tests.fixture module, 54

decisionengine.framework.dataspace.datasources.tests.test module, 55

decisionengine.framework.dataspace.datasources.us_db_schema.TaskManager module, 68

decisionengine.framework.dataspace.maintain module, 69

decisionengine.framework.dataspace.tests module, 63

decisionengine.framework.dataspace.tests.fixtures module, 59

decisionengine.framework.dataspace.tests.test_datablock module, 60

decisionengine.framework.dataspace.tests.test_datablock_zl module, 61

decisionengine.framework.dataspace.tests.test_datasource module, 61

decisionengine.framework.dataspace.tests.test_dataspace module, 61

decisionengine.framework.dataspace.tests.test_Reaper module, 60

decisionengine.framework.engine module, 77

```

decisionengine.framework.engine.ChannelWorkersdecisionengine.framework.logicengine.tests.test_simple_cor
    module, 72                                module, 79
decisionengine.framework.engine.de_client    decisionengine.framework.modules
    module, 76                                module, 88
decisionengine.framework.engine.de_query_tool decisionengine.framework.modules.de_logger
    module, 77                                module, 87
decisionengine.framework.engine.DecisionEnginedecisionengine.framework.modules.describe
    module, 73                                module, 87
decisionengine.framework.engine.SourceWorkers decisionengine.framework.modules.EmptySource
    module, 76                                module, 84
decisionengine.framework.engine.tests        decisionengine.framework.modules.logging_configDict
    module, 72                                module, 88
decisionengine.framework.engine.tests.fixturesdecisionengine.framework.modules.Module
    module, 69                                module, 85
decisionengine.framework.engine.tests.test_ChannelWorkersdecisionengine.framework.modules.print_description
    module, 70                                module, 88
decisionengine.framework.engine.tests.test_client_sideenginedecisionengine.framework.modules.Publisher
    module, 71                                module, 85
decisionengine.framework.engine.tests.test_query_sideenginedecisionengine.framework.modules.QueueLogger
    module, 71                                module, 85
decisionengine.framework.engine.tests.test_startupdecisionengine.framework.modules.Source
    module, 71                                module, 86
decisionengine.framework.engine.tests.test_verificationenginedecisionengine.framework.modules.tests
    module, 72                                module, 84
decisionengine.framework.logicengine        decisionengine.framework.modules.tests.test_de_logger
    module, 82                                module, 84
decisionengine.framework.logicengine.BooleanExpressiondecisionengine.framework.modules.tests.test_EmptySource
    module, 80                                module, 82
decisionengine.framework.logicengine.FactLookupdecisionengine.framework.modules.tests.test_Module
    module, 80                                module, 83
decisionengine.framework.logicengine.LogicEnginedecisionengine.framework.modules.tests.test_module_decorat
    module, 81                                module, 84
decisionengine.framework.logicengine.Rule    decisionengine.framework.modules.tests.test_Publisher
    module, 82                                module, 83
decisionengine.framework.logicengine.RuleEnginedecisionengine.framework.modules.tests.test_QueueLogger
    module, 82                                module, 83
decisionengine.framework.logicengine.tests    decisionengine.framework.modules.tests.test_Source
    module, 80                                module, 83
decisionengine.framework.logicengine.tests.test_dispatcherengineframework.modules.tests.test_Transform
    module, 77                                module, 84
decisionengine.framework.logicengine.tests.test_dispatcherengineframework.modules.tests.test_translate_pro
    module, 78                                module, 84
decisionengine.framework.logicengine.tests.test_dispatcherengineframework.modules.Transform
    module, 78                                module, 86
decisionengine.framework.logicengine.tests.test_dispatcherengineframework.modules.translate_product_name
    module, 78                                module, 88
decisionengine.framework.logicengine.tests.test_dispatcherengineframework.taskmanager
    module, 78                                module, 92
decisionengine.framework.logicengine.tests.test_dispatcherengineframework.taskmanager.LatestMessages
    module, 79                                module, 88
decisionengine.framework.logicengine.tests.test_dispatcherengineframework.taskmanager.module_graph
    module, 79                                module, 91
decisionengine.framework.logicengine.tests.test_dispatcherengineframework.taskmanager.ProcessingState
    module, 79                                module, 89

```

decisionengine.framework.taskmanager.SourceProducer module, 90
 decisionengine.framework.taskmanager.TaskManager module, 90
 decisionengine.framework.tests module, 100
 decisionengine.framework.tests.ABTransform module, 92
 decisionengine.framework.tests.BATransform module, 92
 decisionengine.framework.tests.DynamicPublisher module, 93
 decisionengine.framework.tests.DynamicSource module, 93
 decisionengine.framework.tests.DynamicTransform module, 93
 decisionengine.framework.tests.ErrorOnAcquire module, 93
 decisionengine.framework.tests.FailingPublisher module, 93
 decisionengine.framework.tests.FailingSourceNOP module, 94
 decisionengine.framework.tests.fixtures module, 97
 decisionengine.framework.tests.IntSource module, 94
 decisionengine.framework.tests.ModuleProgramOptions module, 94
 decisionengine.framework.tests.PublisherNOP module, 95
 decisionengine.framework.tests.PublisherWithMissingConsumers module, 95
 decisionengine.framework.tests.SourceAlias module, 95
 decisionengine.framework.tests.SourceNOP module, 95
 decisionengine.framework.tests.SourceWithSampleConfigNOP module, 95
 decisionengine.framework.tests.SupportsConfigPublisher module, 96
 decisionengine.framework.tests.test_client_error module, 97
 decisionengine.framework.tests.test_client_server module, 98
 decisionengine.framework.tests.test_combined_channels module, 98
 decisionengine.framework.tests.test_defaults module, 98
 decisionengine.framework.tests.test_dynamic_test_modules module, 98
 decisionengine.framework.tests.test_empty_config module, 98
 decisionengine.framework.tests.test_error_on_abort module, 98
 decisionengine.framework.tests.test_module_program_options module, 99
 decisionengine.framework.tests.test_query_tool_server module, 99
 decisionengine.framework.tests.test_reaper module, 99
 decisionengine.framework.tests.test_same_source_types module, 99
 decisionengine.framework.tests.test_sample_config module, 99
 decisionengine.framework.tests.test_shared_sources module, 100
 decisionengine.framework.tests.test_start_with_bad_channel module, 100
 decisionengine.framework.tests.TransformNOP module, 96
 decisionengine.framework.tests.TransformWithMissingProducers module, 96
 decisionengine.framework.tests.WriteToDisk module, 96
 decisionengine.framework.util module, 104
 decisionengine.framework.util.countdown module, 100
 decisionengine.framework.util.fs module, 101
 decisionengine.framework.util.logparser module, 101
 decisionengine.framework.util.metrics module, 102
 decisionengine.framework.util.reaper module, 103
 decisionengine.framework.util.redis_stats module, 103
 decisionengine.framework.util.singleton module, 103
 decisionengine.framework.util.sockets module, 104
 decisionengine.framework.util.subclasses module, 104
 decisionengine.framework.version module, 104
 decisionengine.tests module, 104
 decisionengine.tests.test_framework_package module, 104
 decompress() (in module decisionengine.framework.dataspace.datablock),
 decrement_use_count() (decisionengine.framework.engine.SourceWorkers.SourceWorker method), 76
 default_data_lifetime (decisionengine.framework.dataspace.datablock.Header

attribute), 64
 delete() (decisionengine.framework.dataspace.dataspace.DataSpace method), 68
 delete_data_older_than() (decisionengine.framework.dataspace.datasource.DataSource method), 65
 delete_data_older_than() (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 57
 delete_data_older_than() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.DataSource method), 44
 delete_data_older_than() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.DataSource method), 51
 Describe (class in decisionengine.framework.tests.ModuleProgramOptions), 94
 describe() (in module decisionengine.framework.modules.Publisher), 85
 describe() (in module decisionengine.framework.modules.Source), 86
 describe() (in module decisionengine.framework.modules.Transform), 86
 DescribeAlias (class in decisionengine.framework.tests.ModuleProgramOptions), 94
 DEServer() (in module decisionengine.framework.engine.tests.fixtures), 69
 DEServer() (in module decisionengine.framework.tests.fixtures), 97
 display_metrics() (in module decisionengine.framework.util.metrics), 103
 dump() (decisionengine.framework.config.ValidConfig.ValidConfig method), 42
 duplicate() (decisionengine.framework.dataspace.datablock.DataBlock method), 63
 duplicate_datablock() (decisionengine.framework.dataspace.datasource.DataSource method), 65
 duplicate_datablock() (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 57
 duplicate_datablock() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.DataSource method), 44
 duplicate_datablock() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.DataSource method), 51
 duplicate_datablock() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.DataSource method), 51
 engine.framework.dataspace.dataspace.DataSpace
 DynamicPublisher (class in decisionengine.framework.tests.DynamicPublisher), 93
 DynamicSource (class in decisionengine.framework.tests.DynamicSource), 93
 DynamicTransform (class in decisionengine.framework.tests.DynamicTransform), 93
 engine.framework.dataspace.datasource_api.SQLAlchemyDS
E
 EmptySQLAlchemyDS (class in decisionengine.framework.modules.EmptySource), 84
 ensure_no_circularities() (in module decisionengine.framework.taskmanager.module_graph), 92
 ERROR (decisionengine.framework.taskmanager.ProcessingState.State attribute), 90
 ErrorOnAcquire (class in decisionengine.framework.tests.ErrorOnAcquire), 93
 evaluate() (decisionengine.framework.logicengine.BooleanExpression.BooleanExpression method), 80
 evaluate() (decisionengine.framework.logicengine.LogicEngine.LogicEngine method), 81
 evaluate() (decisionengine.framework.logicengine.Rule.Rule method), 82
 evaluate_facts() (decisionengine.framework.logicengine.LogicEngine.LogicEngine method), 81
 execute() (decisionengine.framework.logicengine.RuleEngine.RuleEngine method), 82
 execute_command_from_args() (in module decisionengine.framework.engine.de_client), 76
 execute_command_from_args() (in module decisionengine.framework.engine.de_query_tool), 77
 execute_command_from_args() (in module decisionengine.framework.util.logparser), 101
 expiration_time (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema attribute), 48
F
 FactLookup (class in decisionengine.framework.logicengine.FactLookup), 86
 engine.framework.dataspace.datasource_api.SQLAlchemyDS
 FailingPublisher (class in decisionengine.framework.tests.FailingPublisher), 93
 engine.framework.dataspace.datasource_api.SQLAlchemyDS
 files_with_extensions() (in module decisionengine.framework.util.fs), 101

format_logger() (decisionengine.framework.modules.QueueLogger.QueueLogger method), 66
 method), 85
 get_dataproduct() (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 57
 function_name_from_call() (in module decisionengine.framework.logicengine.BooleanExpression), method), 57
 80
 get_dataproduct() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource method), 44
 G
 Gauge (class in decisionengine.framework.util.metrics), get_dataproduct() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.SQLAlchemy method), 51
 102
 generation_id (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource attribute), 47
 get_dataproduct() (decisionengine.framework.dataspace.dataspace.DataSpace method), 68
 generation_id (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource attribute), 48
 get_dataproducts_loader (decisionengine.framework.dataspace.datablock.DataBlock method), 63
 generation_id (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource attribute), 49
 get_dataproducts_loader (decisionengine.framework.dataspace.datasource.DataSource method), 66
 generation_time (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource attribute), 49
 get_dataproducts_loader (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 57
 get() (decisionengine.framework.dataspace.datablock.DataBlock method), 63
 get_dataproducts() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource method), 45
 method), 89
 get_channels() (decisionengine.framework.config.ChannelConfigHandler.ChannelConfigHandler method), 41
 get_dataproducts() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.SQLAlchemy method), 51
 get_consumes() (decisionengine.framework.engine.ChannelWorkers.ChannelWorkers method), 72
 get_dataproducts() (decisionengine.framework.dataspace.dataspace.DataSpace method), 68
 get_consumes() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 90
 get_header() (decisionengine.framework.dataspace.datablock.DataBlock method), 63
 get_data_block() (decisionengine.framework.modules.Module.Module method), 85
 get_header() (decisionengine.framework.dataspace.datasource.DataSource method), 66
 get_datablock() (decisionengine.framework.dataspace.datasource.DataSource method), 65
 get_header() (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 57
 get_datablock() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource method), 44
 get_header() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.SQLAlchemy method), 52
 get_datablock() (decisionengine.framework.dataspace.datasource.DataSource method), 51
 get_header() (decisionengine.framework.dataspace.dataspace.DataSpace method), 68
 get_datablock() (decisionengine.framework.dataspace.dataspace.DataSpace method), 68
 get_last_generation_id() (decisionengine.framework.dataspace.datasource.DataSource method), 66
 get_dataproduct() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 90
 get_last_generation_id() (decisionengine.framework.dataspace.datasource.DataSource method), 66
 get_last_generation_id() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 90

`nengine.framework.dataspace.datasources.null.NullDataSource` method), 66
`method`), 57 `get_schema()` (decision-
`get_last_generation_id()` (decision- `nengine.framework.dataspace.datasources.null.NullDataSource`
`nengine.framework.dataspace.datasources.sqlalchemy_ds.DataSource` 58 `api SQLAlchemyDS`
`method`), 45 `get_schema()` (decision-
`get_last_generation_id()` (decision- `nengine.framework.dataspace.datasources.sqlalchemy_ds.DataSource`
`nengine.framework.dataspace.datasources.sqlalchemy_ds.DataSource` 46 `api SQLAlchemyDS`
`method`), 52 `get_schema()` (decision-
`get_last_generation_id()` (decision- `nengine.framework.dataspace.datasources.sqlalchemy_ds.DataSource`
`nengine.framework.dataspace.dataspace.DataSpace` method), 52
`method`), 68 `get_state()` (decision-
`get_logger()` (decision- `nengine.framework.taskmanager.TaskManager.TaskManager`
`nengine.framework.engine.DecisionEngine.DecisionEngine` method), 90
`method`), 73 `get_state_name()` (decision-
`get_logger()` (in module decision- `nengine.framework.engine.ChannelWorkers.ChannelWorker`
`nengine.framework.modules.de_logger`), method), 72
87 `get_state_name()` (decision-
`get_loglevel()` (decision- `nengine.framework.taskmanager.TaskManager.TaskManager`
`nengine.framework.taskmanager.TaskManager.TaskManager` method), 90
`method`), 90 `get_state_value()` (decision-
`get_metadata()` (decision- `nengine.framework.taskmanager.ProcessingState.ProcessingState`
`nengine.framework.dataspace.datablock.DataBlock` method), 89
`method`), 63 `get_state_value()` (decision-
`get_metadata()` (decision- `nengine.framework.taskmanager.TaskManager.TaskManager`
`nengine.framework.dataspace.datasource.DataSource` method), 90
`method`), 66 `get_taskmanager()` (decision-
`get_metadata()` (decision- `nengine.framework.dataspace.datablock.DataBlock`
`nengine.framework.dataspace.datasources.null.NullDataSource` method), 63
`method`), 57 `get_taskmanager()` (decision-
`get_metadata()` (decision- `nengine.framework.dataspace.datasource.DataSource`
`nengine.framework.dataspace.datasources.sqlalchemy_ds.DataSource` 64 `api SQLAlchemyDS`
`method`), 45 `get_taskmanager()` (decision-
`get_metadata()` (decision- `nengine.framework.dataspace.datasources.null.NullDataSource`
`nengine.framework.dataspace.datasources.sqlalchemy_ds.DataSource` 58 `api SQLAlchemyDS`
`method`), 52 `get_taskmanager()` (decision-
`get_metadata()` (decision- `nengine.framework.dataspace.datasources.sqlalchemy_ds.DataSource`
`nengine.framework.dataspace.dataspace.DataSpace` method), 46
`method`), 68 `get_taskmanager()` (decision-
`get_parameters()` (decision- `nengine.framework.dataspace.datasources.sqlalchemy_ds.DataSource`
`nengine.framework.modules.Module.Module` method), 52
`method`), 85 `get_taskmanager()` (decision-
`get_produces()` (decision- `nengine.framework.dataspace.dataspace.DataSpace`
`nengine.framework.engine.ChannelWorkers.ChannelWorker` method), 68
`method`), 72 `get_taskmanagers()` (decision-
`get_produces()` (decision- `nengine.framework.dataspace.datasource.DataSource`
`nengine.framework.taskmanager.TaskManager.TaskManager` method), 66
`method`), 90 `get_taskmanagers()` (decision-
`get_queue_logger()` (in module decision- `nengine.framework.dataspace.datasources.null.NullDataSource`
`nengine.framework.modules.de_logger`), method), 58
87 `get_taskmanagers()` (decision-
`get_random_port()` (in module decision- `nengine.framework.dataspace.datasources.sqlalchemy_ds.DataSource`
`nengine.framework.util.sockets`), 104 method), 46
`get_schema()` (decision- `get_taskmanagers()` (decision-
`nengine.framework.dataspace.datasource.DataSource` `nengine.framework.dataspace.datasources.sqlalchemy_ds.DataSource`

method), 53

get_taskmanagers() (decisionengine.framework.dataspace.dataspace.DataSource method), 67

get_taskmanagers() (decisionengine.framework.dataspace.dataspace.DataSource method), 68

global_config() (in module decisionengine.framework.engine.tests.test_ChannelWorkers), 70

global_config_dir() (in module decisionengine.framework.config.policies), 42

global_config_file() (in module decisionengine.framework.config.policies), 43

H

handle_sighup() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 73

handler_setup() (in module decisionengine.framework.modules.tests.test_QueueLogger), 83

has_value() (decisionengine.framework.taskmanager.ProcessingState.ProcessingState method), 89

Header (class in decisionengine.framework.dataspace.datablock), 64

Header (class in decisionengine.framework.dataspace.datasources.sqlalchemy_datasources), 48

header_table (decisionengine.framework.dataspace.datasource.DataSource attribute), 67

Help (class in decisionengine.framework.tests.ModuleProgramOptions), 94

Histogram (class in decisionengine.framework.util.metrics), 102

I

id(decisionengine.framework.dataspace.datasources.sqlalchemy_datasources.sqlalchemy_datasources attribute), 47

id(decisionengine.framework.dataspace.datasources.sqlalchemy_datasources.sqlalchemy_datasources attribute), 48

id(decisionengine.framework.dataspace.datasources.sqlalchemy_datasources.sqlalchemy_datasources attribute), 49

IDLE(decisionengine.framework.taskmanager.ProcessingState.ProcessingState attribute), 90

inactive() (decisionengine.framework.taskmanager.ProcessingState.ProcessingState method), 89

increment_use_count() (decisionengine.framework.engine.SourceWorkers.SourceWorker method), 76

initialize_q() (decisionengine.framework.modules.QueueLogger.QueueLogger method), 85

insert() (decisionengine.framework.dataspace.datasource.DataSource method), 67

insert() (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 58

insert() (decisionengine.framework.dataspace.datasources.sqlalchemy_datasources.sqlalchemy_datasources method), 46

insert() (decisionengine.framework.dataspace.datasources.sqlalchemy_datasources.sqlalchemy_datasources method), 53

insert() (decisionengine.framework.dataspace.dataspace.DataSource method), 68

IntSource (class in decisionengine.framework.tests.IntSource), 94

InvalidMetadataError, 64

is_expired() (decisionengine.framework.dataspace.datablock.DataBlock method), 64

is_valid() (decisionengine.framework.dataspace.datablock.Header method), 64

K

key(decisionengine.framework.dataspace.datasources.sqlalchemy_datasources.sqlalchemy_datasources attribute), 47

key(decisionengine.framework.dataspace.datasources.sqlalchemy_datasources.sqlalchemy_datasources attribute), 48

key(decisionengine.framework.dataspace.datasources.sqlalchemy_datasources.sqlalchemy_datasources attribute), 49

keys() (decisionengine.framework.dataspace.datablock.DataBlock method), 64

L

LatestMessages (class in decisionengine.framework.taskmanager.LatestMessages), 88

load() (in module decisionengine.framework.config.tests.test_config), 39

load_all_channels() (decisionengine.framework.config.ChannelConfigHandler.ChannelConfigHandler method), 41

load_channel() (decisionengine.framework.config.ChannelConfigHandler.ChannelConfigHandler method), 41

load_sample_data.MetadataDataSource() (in module decisionengine.framework.dataspace.tests.fixtures), 59

load_sample_data.MetadataDataSource() (in module decisionengine.framework.taskmanager.ProcessingState.ProcessingState property), 89

log_setup() (in module decisionengine.framework.modules.tests.test_de_logger), 84

log_setup() (in module decisionengine.framework.modules.tests.test_QueueLogger), 83

[logic_engine_with_fact\(\)](#) (in module [decisionengine.framework.logicengine.tests.test_fail_on_error](#)), [79](#)
[LogicEngine](#) (class in [decisionengine.framework.logicengine.LogicEngine](#)), [81](#)
[LogicError](#), [80](#)
M
[main\(\)](#) (in module [decisionengine.framework.engine.de_client](#)), [76](#)
[main\(\)](#) (in module [decisionengine.framework.engine.de_query_tool](#)), [77](#)
[main\(\)](#) (in module [decisionengine.framework.engine.DecisionEngine](#)), [75](#)
[main\(\)](#) (in module [decisionengine.framework.util.logparser](#)), [101](#)
[main\(\)](#) (in module [decisionengine.framework.util.reaper](#)), [103](#)
[main_wrapper\(\)](#) (in module [decisionengine.framework.modules.describe](#)), [87](#)
[make_db\(\)](#) (in module [decisionengine.framework.logicengine.tests.test_facts](#)), [78](#)
[mark_demented\(\)](#) ([decisionengine.framework.dataspace.dataspace.DataSpace](#) method), [68](#)
[mark_expired\(\)](#) ([decisionengine.framework.dataspace.datablock.DataBlock](#) method), [64](#)
[mark_expired\(\)](#) ([decisionengine.framework.dataspace.dataspace.DataSpace](#) method), [68](#)
[matches_constraint\(\)](#) (in module [decisionengine.framework.util.logparser](#)), [101](#)
[maybe_fail_on_error\(\)](#) (in module [decisionengine.framework.logicengine.BooleanExpression](#)), [80](#)
[Metadata](#) (class in [decisionengine.framework.dataspace.datablock](#)), [64](#)
[Metadata](#) (class in [decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema](#)), [48](#)
[metadata](#) ([decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Base](#) attribute), [47](#)
[metadata_table](#) ([decisionengine.framework.dataspace.datasource.DataSource](#) attribute), [67](#)
[metrics\(\)](#) ([decisionengine.framework.engine.DecisionEngine.DecisionEngine](#) method), [73](#)
[metrics_env_setup\(\)](#) (in module [decisionengine.framework.engine.tests.test_startup](#)), [71](#)
[MIN_RETENTION_INTERVAL_DAYS](#) ([decisionengine.framework.dataspace.maintain.Reaper](#) attribute), [69](#)
[MIN_SECONDS_BETWEEN_RUNS](#) ([decisionengine.framework.dataspace.maintain.Reaper](#) attribute), [69](#)
[missed_update_count](#) ([decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema](#) attribute), [49](#)
[mock_data_block\(\)](#) (in module [decisionengine.framework.dataspace.datasources.tests.fixtures](#)), [55](#)
[module](#)
[decisionengine](#), [104](#)
[decisionengine.framework](#), [104](#)
[decisionengine.framework.about](#), [104](#)
[decisionengine.framework.config](#), [43](#)
[decisionengine.framework.config.ChannelConfigHandler](#), [41](#)
[decisionengine.framework.config.policies](#), [42](#)
[decisionengine.framework.config.tests](#), [41](#)
[decisionengine.framework.config.tests.test_config](#), [39](#)
[decisionengine.framework.config.tests.test_de_std](#), [40](#)
[decisionengine.framework.config.tests.test_policies](#), [41](#)
[decisionengine.framework.config.ValidConfig](#), [42](#)
[decisionengine.framework.dataspace](#), [69](#)
[decisionengine.framework.dataspace.datablock](#), [63](#)
[decisionengine.framework.dataspace.datasource](#), [65](#)
[decisionengine.framework.dataspace.datasources](#), [59](#)
[decisionengine.framework.dataspace.datasources.null](#), [56](#)
[decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema](#), [50](#)
[decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Base](#), [43](#)
[decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Base](#), [47](#)
[decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Base](#), [50](#)
[decisionengine.framework.dataspace.datasources.tests](#), [56](#)
[decisionengine.framework.dataspace.datasources.tests.f](#), [54](#)

[decisionengine.framework.dataspace.datasources.tests.test_datasource_api,](#)
[55](#) [decisionengine.framework.logicengine.tests,](#)
[decisionengine.framework.dataspace.dataspace,](#) [80](#)
[68](#) [decisionengine.framework.logicengine.tests.test_bool_f](#)
[decisionengine.framework.dataspace.maintain,](#) [77](#)
[69](#) [decisionengine.framework.logicengine.tests.test_cascad](#)
[decisionengine.framework.dataspace.tests,](#) [78](#)
[63](#) [decisionengine.framework.logicengine.tests.test_constr](#)
[decisionengine.framework.dataspace.tests.fixtures,](#) [78](#)
[59](#) [decisionengine.framework.logicengine.tests.test_duplic](#)
[decisionengine.framework.dataspace.tests.test_data_block,](#) [78](#)
[60](#) [decisionengine.framework.logicengine.tests.test_facts,](#)
[decisionengine.framework.dataspace.tests.test_data_block_zlib,](#) [78](#)
[61](#) [decisionengine.framework.logicengine.tests.test_fail_o](#)
[decisionengine.framework.dataspace.tests.test_data_source,](#) [79](#)
[61](#) [decisionengine.framework.logicengine.tests.test_pandas](#)
[decisionengine.framework.dataspace.tests.test_data_space,](#) [79](#)
[61](#) [decisionengine.framework.logicengine.tests.test_rule_w](#)
[decisionengine.framework.dataspace.tests.test_Reaper,](#) [79](#)
[60](#) [decisionengine.framework.logicengine.tests.test_simple](#)
[decisionengine.framework.engine,](#) [77](#) [79](#)
[decisionengine.framework.engine.ChannelWorkers,](#) [decisionengine.framework.modules,](#) [88](#)
[72](#) [decisionengine.framework.modules.de_logger,](#)
[decisionengine.framework.engine.de_client,](#) [87](#)
[76](#) [decisionengine.framework.modules.describe,](#)
[decisionengine.framework.engine.de_query_tool,](#) [87](#)
[77](#) [decisionengine.framework.modules.EmptySource,](#)
[decisionengine.framework.engine.DecisionEngine,](#) [84](#)
[73](#) [decisionengine.framework.modules.logging_configDict,](#)
[decisionengine.framework.engine.SourceWorkers,](#) [88](#)
[76](#) [decisionengine.framework.modules.Module,](#)
[decisionengine.framework.engine.tests,](#) [72](#) [85](#)
[decisionengine.framework.engine.tests.fixtures,](#) [decisionengine.framework.modules.print_description,](#)
[69](#) [88](#)
[decisionengine.framework.engine.tests.test_ChannelWorkers,](#) [decisionengine.framework.modules.Publisher,](#)
[70](#) [85](#)
[decisionengine.framework.engine.tests.test_client_side,](#) [decisionengine.framework.modules.QueueLogger,](#)
[71](#) [85](#)
[decisionengine.framework.engine.tests.test_query_engine,](#) [decisionengine.framework.modules.Source,](#)
[71](#) [86](#)
[decisionengine.framework.engine.tests.test_stats,](#) [decisionengine.framework.modules.tests,](#)
[71](#) [84](#)
[decisionengine.framework.engine.tests.test_verification,](#) [decisionengine.framework.modules.tests.test_de_logger,](#)
[72](#) [84](#)
[decisionengine.framework.logicengine,](#) [82](#) [decisionengine.framework.modules.tests.test_EmptySource](#)
[decisionengine.framework.logicengine.BooleanExpression,](#) [82](#)
[80](#) [decisionengine.framework.modules.tests.test_Module,](#)
[decisionengine.framework.logicengine.FactLookup,](#) [83](#)
[80](#) [decisionengine.framework.modules.tests.test_module_dec](#)
[decisionengine.framework.logicengine.LogicEngine,](#) [84](#)
[81](#) [decisionengine.framework.modules.tests.test_Publisher,](#)
[decisionengine.framework.logicengine.Rule,](#) [83](#)
[82](#) [decisionengine.framework.modules.tests.test_QueueLogge](#)
[decisionengine.framework.logicengine.RuleEngine,](#) [83](#)

decisionengine.framework.modules.tests.test_Sources,	decisionengine.framework.tests.SupportsConfigPublisher,
83	96
decisionengine.framework.modules.tests.test_Transform,	decisionengine.framework.tests.test_client_errors,
84	97
decisionengine.framework.modules.tests.test_translate_engine_iframe,	decisionengine.framework.tests.test_client_server,
84	98
decisionengine.framework.modules.Transform,	decisionengine.framework.tests.test_combined_channels,
86	98
decisionengine.framework.modules.translate_product_name,	decisionengine.framework.tests.test_defaults,
88	98
decisionengine.framework.taskmanager, 92	decisionengine.framework.tests.test_dynamic_test_module,
decisionengine.framework.taskmanager.LatestMessages,	98
88	decisionengine.framework.tests.test_empty_config,
decisionengine.framework.taskmanager.module_graph,	98
91	decisionengine.framework.tests.test_error_on_acquire,
decisionengine.framework.taskmanager.ProcessingState,	98
89	decisionengine.framework.tests.test_module_program_options,
decisionengine.framework.taskmanager.SourceProductCache,	99
90	decisionengine.framework.tests.test_query_tool_server,
decisionengine.framework.taskmanager.TaskManager,	99
90	decisionengine.framework.tests.test_reaper,
decisionengine.framework.tests, 100	99
decisionengine.framework.tests.ABTransform,	decisionengine.framework.tests.test_same_source_types,
92	99
decisionengine.framework.tests.BATransform,	decisionengine.framework.tests.test_sample_config,
92	99
decisionengine.framework.tests.DynamicPublisher,	decisionengine.framework.tests.test_shared_sources,
93	100
decisionengine.framework.tests.DynamicSource,	decisionengine.framework.tests.test_start_with_bad_channels,
93	100
decisionengine.framework.tests.DynamicTransform,	decisionengine.framework.tests.TransformNOP,
93	96
decisionengine.framework.tests.ErrorOnAcquire,	decisionengine.framework.tests.TransformWithMissingProducts,
93	96
decisionengine.framework.tests.FailingPublisher,	decisionengine.framework.tests.WriteToDisk,
93	96
decisionengine.framework.tests.FailingSourceNOP,	decisionengine.framework.util, 104
94	decisionengine.framework.util.countdown,
decisionengine.framework.tests.fixtures,	100
97	decisionengine.framework.util.fs, 101
decisionengine.framework.tests.IntSource,	decisionengine.framework.util.logparser,
94	101
decisionengine.framework.tests.ModuleProgramOptions,	decisionengine.framework.util.metrics,
94	102
decisionengine.framework.tests.PublisherNOP,	decisionengine.framework.util.reaper, 103
95	decisionengine.framework.util.redis_stats,
decisionengine.framework.tests.PublisherWithMissingConsumes,	95
95	decisionengine.framework.util.singleton,
decisionengine.framework.tests.SourceAlias,	103
95	decisionengine.framework.util.sockets,
decisionengine.framework.tests.SourceNOP,	104
95	decisionengine.framework.util.subclasses,
decisionengine.framework.tests.SourceWithSampleConfigNOP,	95
95	decisionengine.framework.version, 104

decisionengine.tests, 104
 decisionengine.tests.test_framework_package, 104
 Module (class in decisionengine.framework.modules.Module), 85
 ModuleProgramOptions (class in decisionengine.framework.modules.describe), 87
 mydata() (in module decisionengine.framework.logicengine.tests.test_pandas_fact), 79
 myengine() (in module decisionengine.framework.logicengine.tests.test_cascaded_rules), 78
 myengine() (in module decisionengine.framework.logicengine.tests.test_pandas_fact), 79
 myengine() (in module decisionengine.framework.logicengine.tests.test_rule_with_negated_fact), 79
 myengine() (in module decisionengine.framework.logicengine.tests.test_simple_configuration), 79
 N
 name (decisionengine.framework.dataspace.datasources.sqlalchemy_db_schema.Taskmanager attribute), 49
 name (decisionengine.framework.engine.tests.test_ChannelWorkers.Taskmanager attribute), 70
 NotFound (decisionengine.framework.engine.DecisionEngine.StopState attribute), 75
 NullDataSource (class in decisionengine.framework.dataspace.datasources.null), 56
 O
 OFFLINE (decisionengine.framework.taskmanager.ProcessingState.State attribute), 90
 orm_as_dict() (in module decisionengine.framework.dataspace.datasources.sqlalchemy_ds.units), 50
 P
 Parameter (class in decisionengine.framework.modules.describe), 87
 Parameter (class in decisionengine.framework.modules.Publisher), 85
 Parameter (class in decisionengine.framework.modules.Source), 86
 Parameter (class in decisionengine.framework.modules.Transform), 86
 parse_constraints() (in module decisionengine.framework.util.logparser), 101
 parse_program_options() (in module decisionengine.framework.engine.DecisionEngine), 75
 passthrough_configuration() (in module decisionengine.framework.logicengine.LogicEngine), 81
 PG_DE_DB_WITHOUT_SCHEMA() (in module decisionengine.framework.dataspace.datasources.tests.fixtures), 54
 PG_DE_DB_WITHOUT_SCHEMA() (in module decisionengine.framework.dataspace.tests.fixtures), 59
 PG_DE_DB_WITHOUT_SCHEMA() (in module decisionengine.framework.engine.tests.fixtures), 69
 PG_DE_DB_WITHOUT_SCHEMA() (in module decisionengine.framework.tests.fixtures), 97
 PG_PROG() (in module decisionengine.framework.dataspace.datasources.tests.fixtures), 54
 PG_PROG() (in module decisionengine.framework.dataspace.tests.fixtures), 59
 PG_PROG() (in module decisionengine.framework.engine.tests.fixtures), 70
 PG_PROG() (in module decisionengine.framework.tests.fixtures), 97
 print_channel_config() (decisionengine.framework.config.ChannelConfigHandler.ChannelConfig method), 41
 print_consumes() (in module decisionengine.framework.modules.print_description), 88
 print_produces() (in module decisionengine.framework.modules.print_description), 88
 print_supported_config() (in module decisionengine.framework.modules.print_description), 88
 probably_running() (decisionengine.framework.taskmanager.ProcessingState.ProcessingState method), 89
 process_args() (decisionengine.framework.modules.describe.ModuleProgramOptions method), 87
 ProcessingState (class in decisionengine.framework.taskmanager.ProcessingState), 89
 produces() (decisionengine.framework.logicengine.LogicEngine.LogicEngine method), 81
 produces() (in module decisionengine.framework.modules.Module), 85
 produces() (in module decisionengine.framework.modules.Source), 86

[produces\(\)](#) (in module `decisionengine.framework.modules.Transform`), 86
[ProductRetriever](#) (class in `decisionengine.framework.dataspace.datablock`), 65
[prune\(\)](#) (`decisionengine.framework.engine.SourceWorkers.SourceWorkers` method), 76
[publish\(\)](#) (`decisionengine.framework.modules.Publisher.Publisher` method), 85
[publish\(\)](#) (`decisionengine.framework.tests.FailingPublisher.FailingPublisher` method), 93
[publish\(\)](#) (`decisionengine.framework.tests.PublisherNOP.PublisherNOP` method), 95
[publish\(\)](#) (`decisionengine.framework.tests.WriteToDisk.WriteToDisk` method), 96
[Publisher](#) (class in `decisionengine.framework.modules.Publisher`), 85
[publisher\(\)](#) (`decisionengine.framework.tests.DynamicPublisher.DynamicPublisher` method), 93
[PublisherNOP](#) (class in `decisionengine.framework.tests.PublisherNOP`), 95
[PublisherWithMissingConsumes](#) (class in `decisionengine.framework.tests.PublisherWithMissingConsumes`), 95
[put\(\)](#) (`decisionengine.framework.dataspace.datablock.DataBlock` method), 64
Q
[queue_logger_setup\(\)](#) (in module `decisionengine.framework.modules.tests.test_QueueLogger`), 83
[QueueLogger](#) (class in `decisionengine.framework.modules.QueueLogger`), 85
R
[reap\(\)](#) (`decisionengine.framework.dataspace.maintain.Reaper` method), 69
[Reaper](#) (class in `decisionengine.framework.dataspace.maintain`), 69
[reaper\(\)](#) (in module `decisionengine.framework.dataspace.tests.test_Reaper`), 60
[reaper_start\(\)](#) (`decisionengine.framework.engine.DecisionEngine.DecisionEngine` method), 73
[reaper_status\(\)](#) (`decisionengine.framework.engine.DecisionEngine.DecisionEngine` method), 73
[reaper_stop\(\)](#) (`decisionengine.framework.engine.DecisionEngine.DecisionEngine` method), 74
[record_that_matches\(\)](#) (in module `decisionengine.framework.tests.test_shared_sources`), 100
[RedisStats](#) (in module `decisionengine.framework.util.redis_stats`), 103
[Registry](#) (`decisionengine.framework.dataspace.datasources.sqlalchemy_ds` attribute), 47
[refining_publisher\(\)](#) (`decisionengine.framework.engine.SourceWorkers.SourceWorkers` method), 76
[RequestHandler](#) (class in `decisionengine.framework.engine.DecisionEngine`), 75
[required_keys](#) (`decisionengine.framework.dataspace.datablock.Header` attribute), 64
[required_keys](#) (`decisionengine.framework.dataspace.datablock.Metadata` attribute), 64
[reset_connections\(\)](#) (`decisionengine.framework.dataspace.datasource.DataSource` method), 67
[reset_connections\(\)](#) (`decisionengine.framework.dataspace.datasources.null.NullDataSource` method), 58
[reset_connections\(\)](#) (`decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource` method), 46
[reset_connections\(\)](#) (`decisionengine.framework.dataspace.datasources.sqlalchemy_ds.SQLAlchemyDataSource` method), 53
[retention_interval](#) (`decisionengine.framework.dataspace.maintain.Reaper` property), 69
[rm_channel\(\)](#) (`decisionengine.framework.engine.DecisionEngine.DecisionEngine` method), 74
[rpc_block_while\(\)](#) (`decisionengine.framework.engine.DecisionEngine.DecisionEngine` method), 74
[rpc_get_channel_log_level\(\)](#) (`decisionengine.framework.engine.DecisionEngine.DecisionEngine` method), 74
[rpc_get_log_level\(\)](#) (`decisionengine.framework.engine.DecisionEngine.DecisionEngine` method), 74
[rpc_kill_channel\(\)](#) (`decisionengine.framework.engine.DecisionEngine.DecisionEngine` method), 74
[rpc_metrics\(\)](#) (`decisionengine.framework.engine.DecisionEngine.DecisionEngine` method), 74

method), 74
 rpc_paths (decisionengine.framework.engine.DecisionEngineRuleRequestHandler, class in decisionengine.framework.logicengine.Rule), 82
 attribute), 75
 rpc_ping() (decisionengine.framework.engine.DecisionEngineRuleDefn, class in decisionengine.framework.logicengine.FactLookup.FactLookup), 80
 method), 74
 rpc_print_product() (decisionengine.framework.engine.DecisionEngineRuleEngine, class in decisionengine.framework.logicengine.RuleEngine), 82
 method), 74
 rpc_print_products() (decisionengine.framework.engine.ChannelWorkers.ChannelWorker, class in decisionengine.framework.logicengine.RuleEngine), 82
 method), 72
 method), 74
 run() (decisionengine.framework.engine.SourceWorkers.SourceWorker, class in decisionengine.framework.logicengine.RuleEngine), 82
 method), 76
 rpc_query_tool() (decisionengine.framework.engine.DecisionEngineRuleEngine, class in decisionengine.framework.logicengine.RuleEngine), 82
 method), 74
 rpc_queue_status() (decisionengine.framework.engine.DecisionEngineRuleEngine, class in decisionengine.framework.logicengine.RuleEngine), 82
 method), 91
 method), 91
 rpc_reaper_start() (decisionengine.framework.engine.DecisionEngineRuleEngine, class in decisionengine.framework.logicengine.RuleEngine), 82
 method), 91
 method), 91
 rpc_reaper_status() (decisionengine.framework.engine.DecisionEngineRuleEngine, class in decisionengine.framework.logicengine.RuleEngine), 82
 method), 91
 method), 91
 rpc_reaper_stop() (decisionengine.framework.engine.DecisionEngineRuleEngine, class in decisionengine.framework.logicengine.RuleEngine), 82
 method), 91
 method), 91
 rpc_rm_channel() (decisionengine.framework.engine.DecisionEngineRuleEngine, class in decisionengine.framework.logicengine.RuleEngine), 82
 method), 91
 method), 91
 rpc_set_channel_log_level() (decisionengine.framework.engine.DecisionEngineRuleEngine, class in decisionengine.framework.logicengine.RuleEngine), 82
 method), 91
 method), 91
 rpc_show_config() (decisionengine.framework.engine.DecisionEngineRuleEngine, class in decisionengine.framework.logicengine.RuleEngine), 82
 method), 48
 attribute), 48
 rpc_show_de_config() (decisionengine.framework.engine.DecisionEngineRuleEngine, class in decisionengine.framework.logicengine.RuleEngine), 82
 method), 49
 attribute), 49
 rpc_start_channel() (decisionengine.framework.engine.DecisionEngineRuleEngine, class in decisionengine.framework.logicengine.RuleEngine), 82
 method), 49
 schema_id (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema), 49
 attribute), 48
 rpc_start_channels() (decisionengine.framework.engine.DecisionEngineRuleEngine, class in decisionengine.framework.logicengine.RuleEngine), 82
 method), 48
 schema_id (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema), 49
 attribute), 49
 rpc_status() (decisionengine.framework.engine.DecisionEngineRuleEngine, class in decisionengine.framework.logicengine.RuleEngine), 82
 method), 103
 ScopedSingleton (class in decisionengine.framework.util.singleton), 103
 ScopedSingletonABC (class in decisionengine.framework.util.singleton), 103
 rpc_stop() (decisionengine.framework.engine.DecisionEngineRuleEngine, class in decisionengine.framework.logicengine.RuleEngine), 82
 method), 103
 seconds_between_runs (decisionengine.framework.dataspace.maintain.Reaper, class in decisionengine.framework.logicengine.RuleEngine), 82
 method), 69
 property), 69
 rpc_stop_channel() (decisionengine.framework.engine.DecisionEngineRuleEngine, class in decisionengine.framework.logicengine.RuleEngine), 82
 method), 69
 sequence_id (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema), 49
 attribute), 50

set() (decisionengine.framework.taskmanager.ProcessingState.ProcessingState), 89
 set_data_block() (decisionengine.framework.modules.Module.Module), 85
 set_loglevel_value() (decisionengine.framework.engine.tests.test_ChannelWorkers.TaskManager), 70
 set_loglevel_value() (decisionengine.framework.taskmanager.TaskManager.TaskManager), 91
 set_state() (decisionengine.framework.dataspace.datablock.Metadata), 64
 setup_logger() (decisionengine.framework.engine.ChannelWorkers.ChannelWorkers), 72
 setup_queue_logging() (decisionengine.framework.modules.QueueLogger.QueueLogger), 86
 setup_queue_logging() (in module decisionengine.framework.modules.tests.test_QueueLogger), 83
 should_stop() (decisionengine.framework.engine.SourceWorkers.SourceWorker), 76
 should_stop() (decisionengine.framework.taskmanager.ProcessingState.ProcessingState), 89
 SHUTDOWN (decisionengine.framework.taskmanager.ProcessingState.ProcessingState), 90
 shutdown() (decisionengine.framework.modules.Publisher.Publisher), 85
 SHUTTINGDOWN (decisionengine.framework.taskmanager.ProcessingState.State), 90
 Singleton (class in decisionengine.framework.util.singleton), 103
 SingletonABC (class in decisionengine.framework.util.singleton), 103
 sorted_rules() (decisionengine.framework.logicengine.FactLookup.FactLookup), 80
 Source (class in decisionengine.framework.modules.Source), 86
 source_products() (in module decisionengine.framework.taskmanager.module_graph), 92
 SourceNOP (class in decisionengine.framework.tests.SourceNOP), 95
 SourceProductCache (class in decisionengine.framework.taskmanager.SourceProductCache), 90
 SourceWithMissingProduces (class in decisionengine.framework.tests.FailingSourceNOP), 94
 SourceWithSampleConfigNOP (class in decisionengine.framework.tests.SourceWithSampleConfigNOP), 95
 SourceWorker (class in decisionengine.framework.engine.SourceWorkers), 76
 SourceWorkers (class in decisionengine.framework.engine.SourceWorkers), 76
 spec_if_main() (in module decisionengine.framework.modules.print_description), 88
 SQLAlchemy_PG_WITH_SCHEMA() (in module decisionengine.framework.dataspace.datasources.tests.fixtures), 54
 SQLAlchemy_PG_WITH_SCHEMA() (in module decisionengine.framework.dataspace.tests.fixtures), 59
 SQLAlchemy_PG_WITH_SCHEMA() (in module decisionengine.framework.engine.tests.fixtures), 70
 SQLAlchemy_PG_WITH_SCHEMA() (in module decisionengine.framework.tests.fixtures), 97
 SQLAlchemy_TEMPFILE_SQLITE() (in module decisionengine.framework.dataspace.datasources.tests.fixtures), 54
 SQLAlchemy_TEMPFILE_SQLITE() (in module decisionengine.framework.dataspace.tests.fixtures), 59
 SQLAlchemy_TEMPFILE_SQLITE() (in module decisionengine.framework.engine.tests.fixtures), 70
 SQLAlchemy_TEMPFILE_SQLITE() (in module decisionengine.framework.tests.fixtures), 97
 SQLAlchemyDS (class in decisionengine.framework.dataspace.datasources.sqlalchemy_ds), 50
 SQLAlchemyDS (class in decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource), 43
 start() (decisionengine.framework.dataspace.maintain.Reaper), 69
 start() (decisionengine.framework.modules.QueueLogger.QueueLogger), 86
 start_channel() (decisionengine.framework.engine.DecisionEngine.DecisionEngine), 74
 start_channels() (decisionengine.framework.engine.DecisionEngine.DecisionEngine), 74
 start_webserver() (decisionengine.framework.engine.DecisionEngine.DecisionEngine), 74
 State (class in decisionengine.framework.taskmanager.ProcessingState), 89

state(*decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Metadata*,
 attribute), 49 96
 STEADY(*decisionengine.framework.taskmanager.ProcessingState.State*
 attribute), 90
 stop()(*decisionengine.framework.dataspace.maintain.Reap*,*take_offline()* (decisio-
 method), 69 engine.framework.engine.SourceWorkers.SourceWorker
 stop()(*decisionengine.framework.modules.QueueLogger.QueueLogger*,*method*), 76
 method), 86 *take_offline()* (decisio-
 stop_channels() (decisio- engine.framework.taskmanager.TaskManager.TaskManager
 engine.framework.engine.DecisionEngine.DecisionEngine method), 91
 method), 75 *task_dataproduct* (decisio-
 stop_queue_logger() (in module decisio- engine.framework.dataspace.datasources.sqlalchemy_ds.db_sch
 engine.framework.modules.de_logger), attribute), 50
 87 *task_header* (decisio-
 stop_worker() (decisio- engine.framework.dataspace.datasources.sqlalchemy_ds.db_sch
 engine.framework.engine.DecisionEngine.DecisionEngine attribute), 50
 method), 75 *task_metadata* (decisio-
 stopped_channel_opts() (in module decisio- engine.framework.dataspace.datasources.sqlalchemy_ds.db_sch
 engine.framework.tests.test_sample_config), attribute), 50
 99 *Taskmanager* (class in decisio-
 StopState (class in decisio- engine.framework.dataspace.datasources.sqlalchemy_ds.db_sch
 engine.framework.engine.DecisionEngine), 49
 75 *TaskManager* (class in decisio-
 store_taskmanager() (decisio- engine.framework.engine.tests.test_ChannelWorkers),
 engine.framework.dataspace.datablock.DataBlock 70
 method), 64 *TaskManager* (class in decisio-
 store_taskmanager() (decisio- engine.framework.taskmanager.TaskManager),
 engine.framework.dataspace.datasource.DataSource 90
 method), 67 *taskmanager* (decisio-
 store_taskmanager() (decisio- engine.framework.dataspace.datasources.sqlalchemy_ds.db_sch
 engine.framework.dataspace.datasources.null.NullDataSource attribute), 48
 method), 58 *taskmanager* (decisio-
 store_taskmanager() (decisio- engine.framework.dataspace.datasources.sqlalchemy_ds.db_sch
 engine.framework.dataspace.datasources.sqlalchemy_ds.datasource_api.SQLAlchemyDS
 method), 46 *taskmanager* (decisio-
 store_taskmanager() (decisio- engine.framework.dataspace.datasources.sqlalchemy_ds.db_sch
 engine.framework.dataspace.datasources.sqlalchemy_ds.SQLAlchemyDS
 method), 53 *taskmanager_id* (decisio-
 store_taskmanager() (decisio- engine.framework.dataspace.datasources.sqlalchemy_ds.db_sch
 engine.framework.dataspace.dataspace.DataSpace attribute), 48
 method), 68 *taskmanager_id* (decisio-
 Summary (class in decisio- engine.framework.dataspace.datasources.sqlalchemy_ds.db_sch
 engine.framework.util.metrics), 102 attribute), 48
 supports_config() (in module decisio- *taskmanager_id* (decisio-
 engine.framework.modules.describe), 87 engine.framework.dataspace.datasources.sqlalchemy_ds.db_sch
 supports_config() (in module decisio- attribute), 49
 engine.framework.modules.Publisher), *taskmanager_id* (decisio-
 85 engine.framework.dataspace.datasources.sqlalchemy_ds.db_sch
 supports_config() (in module decisio- attribute), 50
 engine.framework.modules.Source), 86 *taskmanager_table* (decisio-
 supports_config() (in module decisio- engine.framework.dataspace.datasource.DataSource
 engine.framework.modules.Transform), attribute), 67
 86 *Terminated* (*decisionengine.framework.engine.DecisionEngine.StopState*
 SupportsConfig (class in decisio- attribute), 75

```

test() (decisionengine.framework.tests.ModuleProgramOptions.AcquireWithConfig module decision-
method), 94 engine.framework.config.tests.test_config),
test() (decisionengine.framework.tests.ModuleProgramOptions.AcquireWithSampleConfig
method), 94 test_channel_loading() (in module decisio-
test() (decisionengine.framework.tests.ModuleProgramOptions.ConfigTimeLimit module decision-
method), 94 40 engine.framework.config.tests.test_config),
test() (decisionengine.framework.tests.ModuleProgramOptions.TestChannel module_missing_all()
method), 94 (in module decision-
test() (decisionengine.framework.tests.ModuleProgramOptions.DescribeAll module_missing_all()
method), 94 40 engine.framework.config.tests.test_config),
test() (decisionengine.framework.tests.ModuleProgramOptions.TestChannel module_missing_module()
method), 94 (in module decision-
test_acquire_for_sources() (in module decisio- engine.framework.config.tests.test_config),
engine.framework.tests.test_module_program_options), 40
99 test_channel_module_missing_parameters()
test_allow_duplicate_keys_same_values() (in module decision-
(in module decision- engine.framework.config.tests.test_config),
engine.framework.config.tests.test_de_std), 40
40 test_channel_names() (in module decisio-
test_allow_duplicate_source_proxy_keys() engine.framework.config.tests.test_config),
(in module decision- 40
engine.framework.config.tests.test_de_std), 40 test_channel_no_config_files() (in module deci-
40 sionengine.framework.config.tests.test_config),
test_by_nonsense_is_err() (in module decisio- 40
engine.framework.modules.tests.test_de_logger),test_channel_no_modules() (in module decisio-
84 engine.framework.config.tests.test_config),
test_by_size() (in module decisio- 40
engine.framework.modules.tests.test_de_logger),test_check_metrics_env_no_webserver()
84 (in module decision-
test_by_time() (in module decisio- engine.framework.engine.tests.test_startup),
engine.framework.modules.tests.test_de_logger), 71
84 test_check_metrics_env_var_set()
test_can_import() (in module decisio- (in module decision-
engine.tests.test_framework_package), 104 engine.framework.engine.tests.test_startup),
test_change_port() (in module decisio- 71
engine.framework.engine.tests.test_startup), 71 test_check_metrics_env_var_unset()
71 (in module decision-
test_channel_config_dir() (in module decisio- engine.framework.engine.tests.test_startup),
engine.framework.config.tests.test_policies), 71
41 test_client_can_get_de_server_reaper_status()
test_channel_empty_config() (in module decisio- (in module decision-
engine.framework.config.tests.test_config), 39 engine.framework.tests.test_reaper), 99
test_channel_empty_dictionary() (in module deci- test_client_can_get_de_server_status()
sionengine.framework.config.tests.test_config), (in module decision-
39 engine.framework.tests.test_sample_config),
test_channel_invalid_modules_list() 99
(in module decision- test_client_can_get_products_no_channels()
engine.framework.config.tests.test_config), (in module decision-
39 engine.framework.tests.test_start_with_bad_channels),
test_channel_invalid_modules_no_keys() 100
(in module decision- test_client_can_kill_one_channel()
engine.framework.config.tests.test_config), (in module decision-
39 engine.framework.tests.test_sample_config),
test_channel_invalid_modules_string() 99

```


test_client_can_restart_all_channels() (in module decisionengine.framework.tests.test_sample_config), 99	test_client_with_no_server() (in module decisionengine.framework.engine.tests.test_client_only), 71
test_client_can_restart_one_channel() (in module decisionengine.framework.tests.test_sample_config), 99	test_client_with_no_server_verbose() (in module decisionengine.framework.engine.tests.test_client_only), 71
test_client_can_start_one_channel_added_after_startup() (in module decisionengine.framework.tests.test_empty_config), 98	test_combine_one_level() (in module decisionengine.framework.config.tests.test_de_std), 40
test_client_cannot_wait_on_bad_state() (in module decisionengine.framework.tests.test_client_errors), 97	test_combine_one_level_skip_proxies() (in module decisionengine.framework.config.tests.test_de_std), 40
test_client_err_returned_as_rc() (in module decisionengine.framework.engine.tests.test_client_only), 71	test_combined_channels() (in module decisionengine.framework.tests.test_combined_channels), 98
test_client_err_returned_as_rc() (in module decisionengine.framework.engine.tests.test_query_tool_only), 71	test_combined_channels_3g() (in module decisionengine.framework.tests.test_combined_channels), 98
test_client_err_returned_verbose_as_rc() (in module decisionengine.framework.engine.tests.test_client_only), 71	test_compound_fact_with_spaces() (in module decisionengine.framework.logicengine.tests.test_facts), 78
test_client_err_returned_verbose_as_rc() (in module decisionengine.framework.engine.tests.test_query_tool_only), 71	test_compress() (in module decisionengine.framework.dataspace.tests.test_datablock_zlib), 61
test_client_help() (in module decisionengine.framework.engine.tests.test_client_only), 71	test_conditional_fact() (in module decisionengine.framework.logicengine.tests.test_fail_on_error), 79
test_client_logger_level() (in module decisionengine.framework.tests.test_sample_config), 99	test_config_templates() (in module decisionengine.framework.tests.test_module_program_options), 99
test_client_non_real_channel() (in module decisionengine.framework.tests.test_sample_config), 99	test_configuration_with_fact_using_function() (in module decisionengine.framework.logicengine.tests.test_construction), 78
test_client_print_product() (in module decisionengine.framework.tests.test_client_server), 98	test_configuration_with_numpy_facts() (in module decisionengine.framework.logicengine.tests.test_construction), 78
test_client_set_loglevel() (in module decisionengine.framework.tests.test_client_server), 98	test_conflicting_source_configurations() (in module decisionengine.framework.tests.test_shared_sources), 100
test_client_status_msg_to_stdout() (in module decisionengine.framework.tests.test_client_server), 98	test_create_tables() (in module decisionengine.framework.dataspace.datasources.tests.test_datasource_55)
test_client_with_no_command_says_use_help() (in module decisionengine.framework.engine.tests.test_client_only), 71	test_DataBlock_constructor() (in module decisionengine.framework.dataspace.tests.test_datablock), 60
	test_DataBlock_duplicate() (in module decisionengine.framework.dataspace.tests.test_datablock), 60

test_DataBlock_get_dataproducts() (in module decisionengine.framework.dataspace.tests.test_datablock), 60	test_delete_data_older_than_arg() (in module decisionengine.framework.dataspace.datasources.tests.test_datasource_...), 55
test_DataBlock_get_header() (in module decisionengine.framework.dataspace.tests.test_datablock), 60	test_descriptions() (in module decisionengine.framework.tests.test_module_program_options), 99
test_DataBlock_get_metadata() (in module decisionengine.framework.dataspace.tests.test_datablock), 60	test_duplicate_datablock() (in module decisionengine.framework.dataspace.datasources.tests.test_datasource_...), 55
test_DataBlock_get_taskmanager() (in module decisionengine.framework.dataspace.tests.test_datablock), 60	test_duplicate_datablock() (in module decisionengine.framework.dataspace.tests.test_dataspace), 61
test_DataBlock_is_expired() (in module decisionengine.framework.dataspace.tests.test_datablock), 60	test_duplicate_fact_names() (in module decisionengine.framework.logicengine.tests.test_duplicate_fact_names), 78
test_DataBlock_is_expired_with_key() (in module decisionengine.framework.dataspace.tests.test_datablock), 61	test_dynamic_publisher() (in module decisionengine.framework.tests.test_dynamic_test_modules), 98
test_DataBlock_key_management() (in module decisionengine.framework.dataspace.tests.test_datablock), 61	test_dynamic_source() (in module decisionengine.framework.tests.test_dynamic_test_modules), 98
test_DataBlock_key_management_change_name() (in module decisionengine.framework.dataspace.tests.test_datablock), 61	test_dynamic_transform() (in module decisionengine.framework.tests.test_dynamic_test_modules), 98
test_DataBlock_mark_expired() (in module decisionengine.framework.dataspace.tests.test_datablock), 61	test_empty_config() (in module decisionengine.framework.config.tests.test_config), 40
test_DataBlock_no_key_by_name() (in module decisionengine.framework.dataspace.tests.test_datablock), 61	test_empty_source_structure() (in module decisionengine.framework.modules.tests.test_EmptySource), 82
test_DataBlock_to_str() (in module decisionengine.framework.dataspace.tests.test_datablock), 61	test_error_conditions() (in module decisionengine.framework.logicengine.tests.test_bool_function_name), 77
test_dataspace_config_finds_bad() (in module decisionengine.framework.dataspace.tests.test_dataspace), 61	test_error_on_bad_names() (in module decisionengine.framework.logicengine.tests.test_simple_configuration), 79
test_default_config() (in module decisionengine.framework.engine.tests.test_startup), 71	test_error_on_duplicate_keys() (in module decisionengine.framework.config.tests.test_de_std), 40
test_default_construction() (in module decisionengine.framework.logicengine.tests.test_construction), 78	test_exclusive_options() (in module decisionengine.framework.engine.tests.test_client_only), 71
test_defaults() (in module decisionengine.framework.tests.test_defaults), 98	test_fact_using_numpy_array() (in module decisionengine.framework.logicengine.tests.test_facts), 78
test_delete() (in module decisionengine.framework.dataspace.tests.test_dataspace), 61	test_fact_using_numpy_function() (in module decisionengine.framework.logicengine.tests.test_facts), 78
	test_fact_with_fail_on_error() (in module decisionengine.framework.logicengine.tests.test_facts), 78

test_fact_with_misspecified_attribute()	55
(in module decisionengine.framework.logicengine.tests.test_fail_on_error),	(in module decisionengine.framework.dataspace.tests.test_dataspace),
79	
test_fact_with_nested_names()	61
(in module decisionengine.framework.logicengine.tests.test_facts),	test_get_dataproducts()
78	(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_
test_fail_bad_config()	55
(in module decisionengine.framework.dataspace.tests.test_Reaper),	test_get_dataproducts()
60	(in module decisionengine.framework.dataspace.tests.test_dataspace),
test_fail_missing_config()	62
(in module decisionengine.framework.dataspace.tests.test_Reaper),	test_get_dataproducts_not_exist()
60	(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_
test_fail_missing_config_key()	55
(in module decisionengine.framework.dataspace.tests.test_Reaper),	test_get_dataproducts_not_exist()
60	(in module decisionengine.framework.dataspace.tests.test_dataspace),
test_fail_on_error()	62
(in module decisionengine.framework.logicengine.tests.test_fail_on_error),	test_get_header()
79	(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_
test_fail_small_retain()	55
(in module decisionengine.framework.dataspace.tests.test_Reaper),	test_get_header()
60	(in module decisionengine.framework.dataspace.tests.test_dataspace),
test_fail_small_run_interval()	62
(in module decisionengine.framework.dataspace.tests.test_Reaper),	test_get_header_not_exist()
60	(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_
test_fail_start_two_reapers()	55
(in module decisionengine.framework.dataspace.tests.test_Reaper),	test_get_header_not_exist()
60	(in module decisionengine.framework.dataspace.tests.test_dataspace),
test_fail_wrong_config_key()	62
(in module decisionengine.framework.dataspace.tests.test_Reaper),	test_get_last_generation_id()
60	(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_
test_false_fact_with_spaces()	55
(in module decisionengine.framework.logicengine.tests.test_fail_on_error),	test_get_last_generation_id()
79	(in module decisionengine.framework.dataspace.tests.test_dataspace),
test_false_literal_fact()	62
(in module decisionengine.framework.logicengine.tests.test_fail_on_error),	test_get_last_generation_id_not_exist()
79	(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_
test_get_datablock()	55
(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_	test_get_last_generation_id_not_exist()
55	(in module decisionengine.framework.dataspace.tests.test_dataspace),
test_get_datablock()	61
(in module decisionengine.framework.dataspace.tests.test_dataspace),	test_get_metadata()
	(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_
test_get_dataproduct()	55
(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_	test_get_metadata()
55	(in module decisionengine.framework.dataspace.tests.test_dataspace),
test_get_dataproduct()	61
(in module decisionengine.framework.dataspace.tests.test_dataspace),	test_get_metadata_not_exist()
	(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_
test_get_dataproduct_not_exist()	56
(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_	test_get_metadata_not_exist()
	(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_
test_get_dataproduct_not_exist()	62
(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_	

test_get_metadata_not_exist()	(in module decisionengine.framework.dataspace.tests.test_dataspace)	99
test_index_error()	(in module decisionengine.framework.logicengine.tests.test_fail_on_error),	
test_get_taskmanager_exists()	(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_api),	79
test_insert()	(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_api),	56
test_get_taskmanager_exists()	(in module decisionengine.framework.dataspace.tests.test_dataspace)	62
test_insert()	(in module decisionengine.framework.dataspace.tests.test_dataspace),	62
test_get_taskmanager_not_exists()	(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_api),	56
test_jpath()	(in module decisionengine.framework.config.tests.test_de_std),	40
test_get_taskmanager_not_exists()	(in module decisionengine.framework.dataspace.tests.test_Reaper),	60
test_loop_of_start_stop_in_clumps()	(in module decisionengine.framework.dataspace.tests.test_Reaper),	60
test_get_taskmanagers()	(in module decisionengine.framework.dataspace.tests.test_dataspace),	62
test_mark_expired()	(in module decisionengine.framework.dataspace.tests.test_dataspace),	62
test_get_taskmanagers_not_exist()	(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_api),	56
test_Metadata_constructor()	(in module decisionengine.framework.dataspace.tests.test_datablock),	61
test_get_taskmanagers_not_exist()	(in module decisionengine.framework.dataspace.tests.test_dataspace)	62
test_minimal_jsonnet_right_extension()	(in module decisionengine.framework.config.tests.test_config),	40
test_global_config_dir()	(in module decisionengine.framework.config.tests.test_policies),	41
test_minimal_jsonnet_wrong_extension()	(in module decisionengine.framework.config.tests.test_config),	40
test_global_config_file()	(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_api),	56
test_missing_data_product_name_not_supported()	(in module decisionengine.framework.modules.tests.test_EmptySource),	82
test_has_config()	(in module decisionengine.framework.dataspace.tests.test_dataspace)	62
test_misspecified_fact()	(in module decisionengine.framework.logicengine.tests.test_fail_on_error),	79
test_has_methods_we_expect()	(in module decisionengine.framework.tests.test_module_program_options),	99
test_Header_constructor()	(in module decisionengine.framework.modules.tests.test_Module),	83
test_Header_is_valid()	(in module decisionengine.framework.modules.tests.test_module_decorators),	84
test_help()	(in module decisionengine.framework.tests.test_module_program_options),	84

test_multiple_produces_declarations()	79
(in module decisio- nengine.framework.modules.tests.test_module_decorators),	(in module decisio- nengine.framework.logicengine.tests.test_simple_configuration),
84	79
test_publisher_structure()	test_same_source_types_separate_channels()
(in module decisio- nengine.framework.modules.tests.test_Publisher),	(in module decisio- nengine.framework.tests.test_same_source_types),
83	
test_query_tool()	99
(in module decisio- nengine.framework.tests.test_query_tool_server),	test_setup_queue_logging()
99	(in module decisio- nengine.framework.modules.tests.test_QueueLogger),
test_query_tool_help()	83
(in module decisio- nengine.framework.engine.tests.test_query_tool_only),	test_shared_source()
71	(in module decisio- nengine.framework.tests.test_shared_sources),
test_query_tool_with_no_server()	100
(in module decisio- nengine.framework.engine.tests.test_query_tool_only),	test_simple_fact()
71	(in module decisio- nengine.framework.logicengine.tests.test_facts),
test_query_tool_with_no_server_verbose()	78
(in module decisio- nengine.framework.engine.tests.test_query_tool_only),	test_source_fail_can_be_fixed()
71	(in module decisio- nengine.framework.dataspace.tests.test_Reaper),
test_reap_default_state()	60
(in module decisio- nengine.framework.dataspace.tests.test_Reaper),	test_source_only_channel()
60	(in module decisio- nengine.framework.tests.test_error_on_acquire),
test_reaper_can_reap()	98
(in module decisio- nengine.framework.dataspace.tests.test_Reaper),	test_source_structure()
60	(in module decisio- nengine.framework.modules.tests.test_Source),
test_reset_connections()	83
(in module decisio- nengine.framework.dataspace.datasources.tests.test_datasource_manager),	test_start_delay()
56	(in module decisio- nengine.framework.dataspace.tests.test_Reaper),
test_rule_that_does_not_fire()	60
(in module decisio- nengine.framework.logicengine.tests.test_cascaded_rules),	test_start_queue_logger()
78	(in module decisio- nengine.framework.modules.tests.test_QueueLogger),
test_rule_that_does_not_fire()	83
(in module decisio- nengine.framework.logicengine.tests.test_pandas_fact),	test_start_stop()
79	(in module decisio- nengine.framework.dataspace.tests.test_Reaper),
test_rule_that_does_not_fire()	60
(in module decisio- nengine.framework.logicengine.tests.test_rule_with_negated_hugone),	test_start_stop_stop()
79	(in module decisio- nengine.framework.dataspace.tests.test_Reaper),
test_rule_that_does_not_fire()	60
(in module decisio- nengine.framework.logicengine.tests.test_rule_with_negated_hugone),	test_state_can_be_active()
79	(in module decisio- nengine.framework.dataspace.tests.test_Reaper),
test_rule_that_does_not_fire()	60
(in module decisio- nengine.framework.logicengine.tests.test_simple_configuration),	test_state_sets_timer_and_uses_it()
79	(in module decisio- nengine.framework.dataspace.tests.test_Reaper),
test_rule_that_fires()	60
(in module decisio- nengine.framework.logicengine.tests.test_cascaded_rules),	test_stop_queue_logger()
78	(in module decisio- nengine.framework.modules.tests.test_QueueLogger),
test_rule_that_fires()	83
(in module decisio- nengine.framework.logicengine.tests.test_pandas_fact),	test_store_taskmanager()
79	(in module decisio- nengine.framework.dataspace.datasources.tests.test_datasource_manager),
test_rule_that_fires()	56
(in module decisio- nengine.framework.logicengine.tests.test_rule_with_negated_hugone),	test_store_taskmanager()
	(in module decisio- nengine.framework.dataspace.tests.test_dataspace),

62	test_valid_dir()	(in module decisionengine.framework.config.tests.test_policies),
test_supports_config()	(in module decisionengine.framework.modules.tests.test_module_decorators),	41
84	test_verify_bad_broker()	(in module decisionengine.framework.engine.tests.test_verify_redis_server),
test_syntax_error()	(in module decisionengine.framework.logicengine.tests.test_facts),	72
78	test_verify_bad_redis_server()	(in module decisionengine.framework.engine.tests.test_verify_redis_server),
test_syntax_error_in_config_names_bad_file()	(in module decisionengine.framework.config.tests.test_config),	72
40	test_verify_bad_url()	(in module decisionengine.framework.engine.tests.test_verify_redis_server),
test_transform_structure()	(in module decisionengine.framework.modules.tests.test_Transform),	72
84	test_verify_redis_server()	(in module decisionengine.framework.engine.tests.test_verify_redis_server),
test_translate_all()	(in module decisionengine.framework.modules.tests.test_translate_product_name),	72
84	test_verify_redis_url()	(in module decisionengine.framework.engine.tests.test_verify_redis_server),
test_translate_illegal_characters()	(in module decisionengine.framework.modules.tests.test_translate_product_name),	72
84	test_worker_logger_sized_rotation()	(in module decisionengine.framework.engine.tests.test_ChannelWorkers),
test_translate_none()	(in module decisionengine.framework.modules.tests.test_translate_product_name),	72
84	test_worker_logger_timed_rotation()	(in module decisionengine.framework.engine.tests.test_ChannelWorkers),
test_translate_simple()	(in module decisionengine.framework.modules.tests.test_translate_product_name),	70
84	test_worker_name()	(in module decisionengine.framework.engine.tests.test_ChannelWorkers),
test_translate_with_underscores()	(in module decisionengine.framework.modules.tests.test_translate_product_name),	72
84	test_wrong_configuration()	(in module decisionengine.framework.logicengine.tests.test_construction),
test_trivial_configuration()	(in module decisionengine.framework.logicengine.tests.test_construction),	78
78	test_wrong_product_names()	(in module decisionengine.framework.modules.tests.test_module_decorators),
test_true_fact()	(in module decisionengine.framework.logicengine.tests.test_fail_on_error),	84
79	test_wrong_product_types()	(in module decisionengine.framework.modules.tests.test_module_decorators),
test_true_literal_fact()	(in module decisionengine.framework.logicengine.tests.test_fail_on_error),	84
79	test_zdumps()	(in module decisionengine.framework.dataspace.tests.test_datablock_zlib),
test_update()	(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_api),	61
56	test_zloads()	(in module decisionengine.framework.dataspace.tests.test_datablock_zlib),
test_update()	(in module decisionengine.framework.dataspace.tests.test_dataspace),	61
62	Transform	(class in decisionengine.framework.modules.Transform),
test_update_bad()	(in module decisionengine.framework.dataspace.datasources.tests.test_datasource_api),	86
56	transform()	(decisionengine.framework.modules.Transform.Transform method),
test_update_bad()	(in module decisionengine.framework.dataspace.tests.test_dataspace),	86
62	transform()	(decisionengine.framework.tests.DynamicTransform.DynamicTransform method),
test_valid_but_empty_config()	(in module decisionengine.framework.config.tests.test_config),	40

transform() (decisionengine.framework.tests.TransformNOP.TransformNOP method), 96

transform() (decisionengine.framework.tests.TransformWithMissingProducesConsumes.TransformWithMissingProducesConsumes method), 96

TransformNOP (class in decisionengine.framework.tests.TransformNOP), 96

TransformWithMissingProducesConsumes (class in decisionengine.framework.tests.TransformWithMissingProducesConsumes), 96

translate() (in module decisionengine.framework.modules.translate_product_name), 88

translate_all() (in module decisionengine.framework.modules.translate_product_name), 88

U

unguarded_access() (decisionengine.framework.engine.ChannelWorkers.ChannelWorkers method), 73

update() (decisionengine.framework.dataspace.datasources.DataSource method), 67

update() (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 58

update() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource_api.SQLAlchemyDS method), 47

update() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.SQLAlchemyDS method), 54

update() (decisionengine.framework.dataspace.dataspace.DataSpace method), 68

update() (decisionengine.framework.engine.SourceWorkers.SourceWorkers method), 76

update() (decisionengine.framework.taskmanager.SourceProductCache.SourceProductCache method), 90

V

valid_dir() (in module decisionengine.framework.config.policies), 43

valid_states (decisionengine.framework.dataspace.datablock.Metadata attribute), 64

validated_workflow() (in module decisionengine.framework.taskmanager.module_graph), 92

ValidConfig (class in decisionengine.framework.config.ValidConfig), 42

value (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Dataproduct attribute), 48

W

wait_for_all_writes() (in module decisionengine.framework.tests.WriteToDisk), 96

wait_until() (decisionengine.framework.taskmanager.ProcessingState.ProcessingState method), 89

wait_while() (decisionengine.framework.engine.ChannelWorkers.ChannelWorker method), 72

wait_while() (decisionengine.framework.taskmanager.ProcessingState.ProcessingState method), 89

Worker (class in decisionengine.framework.taskmanager.module_graph), 91

WriteToDisk (class in decisionengine.framework.tests.WriteToDisk), 96

Z

zdump() (in module decisionengine.framework.dataspace.datablock), 65

zloads() (in module decisionengine.framework.dataspace.datablock), 65