

---

# **decisionengine**

***Release 1.2.1.dev633+g22488d7***

**Fermilab**

**Apr 11, 2024**



## CONTENTS

<b>1</b>	<b>Release Notes</b>	<b>3</b>
<b>2</b>	<b>Install Decision Engine</b>	<b>33</b>
<b>3</b>	<b>Developer Documentation</b>	<b>55</b>
<b>4</b>	<b>Jenkins CI pipeline</b>	<b>57</b>
<b>5</b>	<b>Source code</b>	<b>63</b>
<b>6</b>	<b>Indices and tables</b>	<b>135</b>
	<b>Python Module Index</b>	<b>137</b>
	<b>Index</b>	<b>141</b>



The Decision Engine is a critical component of the HEP Cloud Facility. It provides the functionality of resource scheduling for disparate resource providers, including those which may have a cost or restricted allocation of cycles. This package, decisionengine, provides the framework and base classes, the [decisionengine\\_modules](#) package contains provider specific implementations of the base classes.



## 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 2.0.2

This is mainly a bug fix and documentation release. Instructions to run on EL8 have been added. Also a UP/DOWN status metric was added via Prometheus.

##### Issues fixed in this release

- 428 : Decision engine 1.7.3 bug too many open file descriptors in glide\_frontend\_element.py
- 427 : Set CONTINUE\_IF\_NO\_PROXY to False to allow hybrid configuration

##### Full list of commits since version 2.0.1

7ec132e9: [pre-commit.ci] auto fixes from pre-commit.com hooks

b942241a: Add installation instructions for CentOS 8

4f6fc134: [pre-commit.ci] auto fixes from pre-commit.com hooks

e8d1922e: Fix docstrings errors and warnings

fc6aefd5: Docker container and test setup for EL8

51d5293f: [pre-commit.ci] auto fixes from pre-commit.com hooks

0c15d3bd: Added UP/DOWN status metric of the decision engine

fc76a1f0: Fixup coverage for new version

04b18750: Set upper limit version for flake8. This is needed to have pytest-flake8 and flake8 versions working together.

98797411: Add 'Setup pressure-based pilot submission' section to install document

0165183c: make RPM requires more flexible

28e2a0d4: Updated release notes for 2.0.1 and porting of 1.7.3

## 1.1.2 Release 2.0.1

Patch level (bug fix) release.

### Issues fixed in this release

Bugs fixed

- [DE 639](#): de-client –status stalls whenever channels are not yet in STEADY state
- [DE 638](#): Sources should go offline if the client channel offline
- [DE 634](#): de-client –stop-channel / –start-channel doesn't work in 2.0rc2
- [DE 626](#): New DE 2.0rc2 regularly takes 2-3 minutes to shut down
- [DE 599](#): Clarify timeout variable in block\_while()
- [DE 522](#): Decision engine log files get split between several different processes with several different versions open
- [DE 236](#): New race condition in de-client

Enhancements:

- [DE 650](#): Added separate log files for Sources

### Full list of commits since version 2.0.0

[b5e56ab8](#): Remove signal handler.

[0fb6814b](#): Prevent blocking (if possible) during service actions.

[bb68fc31](#): Add logging handler to client-message receiver.

[53fefbc5](#): Update kombu version.

[009cdd95](#): Use kombu queues for server/client communication.

[29a1ee25](#): add distinct logging for sources

[e44e9210](#): Update GitHub actions; pylint workaround.

[d192f8fb](#): Lock typing\_extensions for Python 3.6 compat

[2b946043](#): Fix pre-commit node version to 17.9.0, the last to support SL7.

[76f3ddfb](#): lock pyupgrade to python3.6 support

[c9c7cb3e](#): Include psutil as part of runtime requirements.

[df8a3941](#): Make sure to kill worker process.

[69924d0c](#): Do not block de-client calls during startup.

[ddb18d7c](#): Minor cleanups.

[f4dc7da7](#): Do not take source offline more than once during detach.

[cbffa992](#): Update Docker entrypoint script for DE 2.0 branch

[e10fe5af](#): Fixed cross-package link in the documentation

[9da1eac8](#): Added cross-package link in the documentation

[d278726b](#): Updated 2.0 release notes and indexes, ready for 2.0.0



### 1.1.3 Release 2.0.0

This release series follows 1.7. A lot started to happen in 1.7.0 and has happened since, so we felt it was proper to change the major version number. We are proud to introduce Decision Engine 2.0.0 to outside users: it provides a friendlier installation procedure and configuration samples to test it on all resources supported by the GlideinWMS Factory, like OSG, some HPC resources and commercial cloud providers.

- New architecture with redesigned source system using Kombu message passing with a Redis backend.
- Token support via DE modules: support for SciToken, WlcgToken (for CE authentication) and HTCondor Idtokens (for Glideins and Factory communication)
- Separation from the GlideinWMS Frontend. Decision Engine still shares some libraries with GlideinWMS but you don't need any more to install and configure the Frontend.
- 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.
- Monitoring via Prometheus.
- SQLAlchemy object-relational mapper to increase the testability of DB interactions and to allow different database backends.
- Packaging via setuptools for both decisionengine and decisionengine\_modules: Dependencies are not yet fully listed in the RPMs.
- Added support of CentOS8 (RHEL7 is still out main platform)
- Configuration example using HTC resources via GlideinWMS Factory
- Decision Engine is distributed under the Apache 2.0 license
- We increased our CI tests including also code auto-formatting and license compliance. We introduced integration tests and we are proud of our over 95% unit test coverage.

---

**Note:** SQLAlchemy is required and is now the only datasource backend supported. Upgrading from a different datasource backend (1.6 or earlier were using direct PostgreSQL, 1.7 was supporting both) is a one-way change with a migration tool. We suggest dropping all objects if you wish to reuse the tablespace. You can preserve a copy of the old database to query historical information.

---

---

**Note:** Added requirement on the Kombu library and a Redis server. We suggest to [install Redis using a container](#).

---

---

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

---

#### Issues fixed in this release

- [528](#): Update license and add copyright notices
- [207](#): Under certain circumstances the fetch of the “consumes” information fails but the channel does not go offline operations
- [547](#): Update DE client libs to postgresql-12
- [459](#): Setuptools issues in decisionengine rpm
- [546](#): Request CentOS8 Stream support for Decision Engine

- 453: Struct Logging Self test errors with pytest-xdist
- 418: Add auto-formatting of the code
- 134: Yum update on decisionengine rpm doesn't restart the service
- 480: Request: Make postgresql migration script to migrate from old postgresql schema to new sqlalchemy schema

### **Full list of commits since version 1.7.0**

685a3a8e: Added changelog file for developers curated list of changes

044f4463: Updated 1.7 and 2.0 release notes, ready for 2.0.0 RC4

19994fb5: Convert timeout program options to floats.

e2055f92: Address Marco's review comments.

abdf35ad: Restore multiple queues but purge source queue after each publish.

52936cb5: Improve error-handling.

aad20744: Change to multiprocessing.Lock for protecting channel/source workers.

24bbe41: Adjust launching of source workers in attempt to avoid deadlock.

6d13a392: Remove unnecessary (and perhaps harmful) external updating of channel states.

5456f32f: Improve test coverage.

1afabb70: Use service\_actions to disable sources whenever client channels fail.

7f67a172: Various naming and logging adjustments

e6e49184: Adjust de-client --status and add --product-dependencies program option.

a7c1f351: Apply block-while timeout to all channels, not each channel.

3d739ec7: Update ci workflow to include workflow\_dispatch mechanism and to customize artifact file name

c5a05650: Archive unit test logs in case of unit test failure and make them available as artifacts

e94c2abb: Update Python 3.6-compatible pre-commit hooks.

aeb6b974: Update Countdown docstrings.

525eb3a8: Add Countdown class to address global timeout problem.

4c458e0c: Updated release notes for 2.0.0 RC3 (1.7.99.post3)

137b574a: Add a minimal container image more suited to production usage

9d7f6875: Provide de-client --queue-status program option.

a7dcc30d: Ensure that channels and sources shared the same queues.

49a316e0: Restore pyupgrade to v2.30, which works on Python 3.6

2ce5ccb6: [pre-commit.ci] pre-commit autoupdate

7bd41851: Print number of pickled bytes of source-produced data.

97aed846: Protect tests from Redis DB/routing-key collisions.

4d3abab7: Flush the Redis DB once the DE server stops.

e36c2150: Remove unnecessary @pytest.mark.usefixtures(...) decorations.

30d68610: More unit testing

7850995d: We should have one path where we test without -v.

a81a52cc: A simple test to ensure the metrics can run

7547720c: Logger tests are a bit unstable at high parallelization.

56516df7: Add missing test to ensure we can change the channel level twice

abde7d0f: Add missing tests for inherited functions

6522ed37: Note lines we are not testing

de7829a4: Remove the unit test log directory if it got created

28fbd599: pin jsonnet 0.17.0

9c5c827e: Metrics seems to want the channels setup to complete

b8829997: Pin pytest version

b348d6f7: Fix deadlock starting cherrypy metrics server

7697e6c1: Log invocation of random port

9e7e4813: Clarify note on xdist, run more workers

0b495fbf: Leave note to remember to cleanup temp files

ca5ddf6f: Ensure we are calling the cherrypy shutdown methods

e60efe78: Move metrics fixtures to the fixtures file

9c717cc5: Log finished with DB init

55965f9e: Prep the server fixture to permit the metrics webserver

732ff99b: Add a 'ping' method

6117cc95: [pre-commit.ci] pre-commit autoupdate

b5af73ca: More logging about cherrypy state

dfe4278f: Added unlinked release notes for DE 2.0.0

7d6484ad: Test source shared between two channels.

ae29d9d1: Test same source types, separate channels

6095d33f: Test LatestMessages utility.

dfbf3e06: Separate sources from channels.

2c10391e: Remove source proxy

afcc7cff: Add some more logging to try and trace startup state

dbd49a66: Explicitly pass .coveragerc to pytest.

e6b03216: Set max retry timeout for sqlite in unit tests.

51bed3d6: Updated documentation for 1.7.1 release

3829151f: Allow duplicate keys if their values are the same.

1ea288e0: [pre-commit.ci] pre-commit autoupdate

6b6611e5: Use pre-commit.ci rather than local actions

dedbe4bd: Use local time for structlog timestamp

461c506e: Make sure de\_std.libsonnet is provided when packaged.  
f93b5963: Update pre-commit hook versions and accommodate python-debian issue.  
bba51609: Reduce number of fixtures.  
a4510cb1: Segment the update for setuptools so it gets cached correctly  
40098f35: Merge pull request #584 from jcpunk/user-pip  
4e1b79a1: Merge pull request #583 from jcpunk/drop-dbutils  
72c8db4a: Recommend using the site user pip dir instead.  
ff604495: Drop unneeded module  
b203e2c4: remove extraneous 'import gc'  
ee2278e7: replace needed import  
4b7dedf2: add licensing info  
e5a56816: add licensing info  
a114abba: add licensing info  
c2d511cd: adding queue logging to de\_logger  
77dd8d5a: Also run checks on backports to 1.7  
7c029578: Updated developers instructions w/ license maintenance via REUSE information  
e66b985d: Fix faulty tests.  
d1a86c57: Set Apache 2.0 license and added REUSE compliance  
e488030e: Ensure that redis is running.  
6c982c11: Report PID for source process.  
3f844ca4: Further flesh out the documentation  
1d750001: Simplifications and rearrangements.  
b85dca45: Set state to error for exceptions caught before the thread start.  
c4727acb: Changed summaries to histograms in DecisionEngine and TaskManager modules  
6fa0bf4d: Added install document and updated the index and development instructions accordingly  
e4de391e: Do the build of the wheel as not-root per our requirements  
24ba5272: Add a redis server to the CI testing containers  
c939a6ed: Address Pat's comments.  
1925a7b0: First implementation using Kombu/redis to communicate data from sources to cycles.  
82faa271: Don't try to package obsolete sql file  
9cbffe94: Drop redundant tests.  
ab0de9a5: Drop obsolete raw postgresql interface  
164b36d3: Removed unnecessary comment  
91f7a76f: Fixed rebase errors  
e475fbd5: Added import statement to fix MultiProcessCollector  
a409f126: Add no-webserver setting to all DE Test Workers

39cca32e: Moved multiprocessing import to metrics to clean up imports.

73762e90: Added `--no-webserver` to invocations of DEServer

303ee4be: Added `__all__` global to control what is exported.

5170224b: Allow for metrics disabling from systemd unit file

2cacef4f: Added check for proper metrics environment and associated unit tests

a637a088: Make webserver operation configurable

b3d6445a: Changed `set_to_function` calls to `set()` calls for metrics

5dccc7fa: Changed metric names to match prometheus convention

7371c2e8: Added cherrypy requirement

2c511cea: Added metrics to record time to run Modules and DecisionEngine rpc calls

c24d33bc: Renamed `prometheus.py` to `metrics.py`

2335134d: Moved TaskManager metrics to `util/prometheus.py` to avoid duplicates

3c1b790c: Added metrics endpoint to RPC server, changed prometheus to multiprocessing mode, and added CherryPy webserver for prometheus metrics

d8972de0: Added unit tests for metrics API

7b0f641b: Add instructions for running the Redis container.

8d0c4919: Block `pytest-postgresql` 4

d988f1a0: Lower timeout for actions.

4f920dcc: Simplifications in preparation for Kombu.

eb9f4292: Make TaskManager not executable

00c8f6e6: Remove unused files.

dd990d2c: Adding `de-logparser`, a tool to help parsing Decision Engine semi-structured logs

1da0d61e: Added a comment to help developers with incomplete installation

1cbc7334: Drop testing/support for PyPy

3ba3e8e6: Ignoring E203, whitespace after `'.'`, since black is adding the whitespace

814669d5: Disable PyPy test that fails for `PG_DE_DB_WITH_SCHEMA` fixture value.

8d68c287: Fix debug message

33db6425: Test composite workflows using source proxies and configuration combination.

30951a5b: EL7 doesn't ship with a new enough go lang for `jsonnetfmt`

e72eb3fd: Forbid inheritance from `SourceProxy`.

e95071fd: Automatically format jsonnet files with `jsonnetfmt`

355ccd45: Correct tests for python 3.10

64119161: Start testing python 3.10

c1cb8258: add dummy source and test

31b0f30b: Check for duplicate keys after source proxies have been removed.

140a4c47: Fix configuration-combination function signature.

d4a05299: Remove now unnecessary blocking.  
1e78a889: Don't run setup.py as root  
a71d5b0a: Add error for running server as root  
cd345701: Increase coverage in LogicEngine  
6c132924: Fix out of sync devel requirements  
8bfab003: Start running tests with xdist  
aeb7d49: Remove unnecessary conversion to Pandas dataframe.  
28919b16: Allow channels to boot in parallel.  
c4fc5997: Improve parameter and variable names.  
e021419b: Encourage use of automatic nag hook  
cc4e469a: Update hooks to latest via *pre-commit autoupgrade*  
cef30b69: Further simplify some cases  
590bea3f: Add channel-combination facilities.  
a4a7938c: Various simplifications recommended by flake8-simple  
d5157416: Possible simplification to logging.  
81e3d1ee: Fix pylint error on *create\_runner* and *ProcessingState*  
2a328c25: Added missing init file to make managers a package  
d7f44015: Rework tests for #454  
9521d3ce: Add debug statement when default logic-engine configuration is used.  
c6dc778c: Unconditionally execute publishers with default configured logic engine.  
a48dd7d8: Remove now-unnecessary Python-to-Jsonnet conversion.  
a6a81ce7: Run autoformatters  
49dac1ec: Setup pre-commit hooks for autoformatters  
3800cc2a: Run the code style/standards checks early.  
1d42eb0d: TaskManager now inherits from ComponentManager. Also added SourceManager, ChannelManager, and SourceSubscriptionManager files for future integration.  
85a16f3b: Python optimised byte code removes assert under some conditions  
bed2f5d9: Support latest setuptools\_scm release

## **1.1.4 Release 1.7.5**

Fixed source logging. Pinned some dependencies to maintain Python 3.6 compatibility.

## 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](#))

## Full list of commits since version 1.7.1

[0c90cdfb6](#): fix source logging by defining logger in Sources after PR670 plus missing adjustments

[670a618f2](#): Updated release notes for 1.7.5

[ea6ef79d](#): pin ubuntu version to 20.04 to get python versions we use to run DE 1.7 tests and set upper limit for python modules to be used by tests

[a1af36f5](#): For branch 1.7 pin pytest version to 6.2.5

[352eab54](#): For branch 1.7 pin jsonnet version to 0.17.0

[bfcfef2f](#): Updated release notes for 1.7.4

[4ff9db91](#): Updated release notes for 1.7.3

[53aba118](#): Updated release notes, ready for 1.7.2

[a461a8f9](#): Updated documentation for 1.7.1 release

## 1.1.5 Release 1.7.4

Same as 1.7.1 release. Done to maintain the same version number as decisionengine\_modules.

## 1.1.6 Release 1.7.3

Same as 1.7.1 release. Done to maintain the same version number as decisionengine\_modules.

## 1.1.7 Release 1.7.2

Same as 1.7.1 release. Done to maintain the same version number as decisionengine\_modules.

## 1.1.8 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

## 1.1.9 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  
3ebe8619: 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.

5ddaff8f: 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  
622bfacf: 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

## 1.1.10 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

650dfa: 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]

### 1.1.11 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)

### 1.1.12 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)  
bcda470 : 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

### 1.1.13 Release 1.5.0

In this release:

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

#### 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)

## 1.1.14 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)

## 1.1.15 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 pyinit 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.

## 1.1.16 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)

### **1.1.17 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)  
cafff2 : 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...> (#232)  
93711cc : Run coveralls even if tests fail (#229)  
03d763a : Jenkins pipeline improvements (#230)  
f48d30f : Fix/223 (#228)  
c8aa262 : github ticket 199 (#222)  
0323bda : Address : [https://github.com/HEPCloud/decisionengine\\_modules/issues/224](https://github.com/HEPCloud/decisionengine_modules/issues/224) (#226)  
62e4df6 : Add support to run CI on Jenkins (#221)  
5ab1541 : bump master version to 1.2.0 (for now) (#219)  
bc19c65 : decisionengine\_modules/NERSC: Added retry loop for NERSC API Calls (#220)  
41a50de : Sync up pep8speaks and run\_pylint.sh with decisionengine settings (#218)  
db4634f : silence pylint error (#217)  
1b95141 : Fix whitespace around operator error  
746ea38 : ignore W503  
8a8b5f4 : remove unused variable  
a6668bf : fix PEP8 warnings  
13773ee : address pep8 warnings  
6bea4ca : silence pylint error

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

a1d0507 : fixing pep8 warning

a10bd17 : debugging one import error

ec501ad : make coveralls.io links work

deab1a7 : T201 (#204)

69f2645 : Add coveragerc

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

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

## 1.1.18 Release 1.1.0

In this release:

- Fixed. [https://github.com/HEPCloud/decisionengine\\_modules/issues/108](https://github.com/HEPCloud/decisionengine_modules/issues/108) “Supply Postgres script to delete fields in main database before a certain date”
- significant code cleanup and pep8 compliance
- unit test work
- CI (GitHub actions and Travis) is introduced

commits

f894b1d : Skip unittest (#77)

632e64b : Add ipython

f681a79 : Make python 2.7 tests run on 1.1 branch

d6a32c0 : implementation of data reaper (#75)

2ad8614 : Use sparse checkout for first checkout to get .github/actions (#65)

812f032 : Cat output of pytest log Exit pylint entrypoint with the line count of pep8 and pylint logs Deal with (detach from ...) Only tar up (S)RPMS dirs for rpm build.

6b05ec7 : Fix errors reported by run\_pylint (#62)

d9f5b66 : Setup pep8speaks

c3b8ac2 : Run github actions as non-root uid. Install packages in virtualenv and remove system rpms.

ae01f9e : Support Python 3 for Boost Python

579761c : Support Python 3 for Boost Python

044b979 : Remove unnecessary using declarations.

00f6d00 : Add extra header dependency due to Boost Python ommission.

24e0795 : Apply clang-format

17c17f9 : Remove JSON dependency.

faa0b22 : Massive cleanup.

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

fef6c11 : Fix errors when running pylint.sh multiple times

da6f077 : Autopep8 -i fixes

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

17396da : logicengine: get rid of compuler warnings

01dc3d1 : Only track what we need

b609d73 : Configure coveralls (and some minor cleanup)

bd9ed5e : Many C++ cleanups

2a61876 : Add Badges

c864f27 : Do not call pytest fixtures directly.

307db5f : white space fix

882b58f : fix unit tests

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

5a6e6b1 : Run tests on push

8404245 : Add missing Boost regex library dependency.

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

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

8a8f560 : Cache venv directory instead

ad017ce : Build private boost for testing

928c64a : Test pip cache

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

9f0ddb3 : Add pylint github action.

5e6ce4a : Remove more unused C++ files.

63717fe : Setup travis to use new cmake var

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

843f30c : Minor cleanups per travis-lint

a538cac : Remove unused C++ files.

4c9d125 : Update repo where action is taken from

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

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

4c79d2c : Remove unnused GNUmakefiles.

94342ee : Add unit test as a Github Action

1a0e102 : more advanced travis.yml

0be413f : Add helper file for pip

7794327 : Make recursive import happy

7005c78 : Add simple target

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

2662e6c : note required packages

3b87119 : Add missing header includes.  
3e79b84 : Remove defunct code and its tests  
b1dbe1a : Ensure attribs are defined at **init**  
c4ad78a : Correct logger arguments do avoid duplicate string parse  
a8dcc67 : Remove unused imports (per pylint)  
d3502b5 : Remove obsolete CVS directories.  
d744111 : add six module to the list of required modules  
0a9b1e8 : Fix class declaration  
b83157e : Handle metaclasses  
549f33b : Add config for Travis CI  
ee71044 : Drop trailing white space  
3f82af6 : Python3 forward compatible syntax  
28bf291 : Add safe (for python 2.7) python3 compatible syntax  
1d1d76f : prepare for python3

## 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 on EL9

Currently the only version supporting EL9 is the development version, DE 2.0.x, which corresponds to the master branch in Git.

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 Alma Linux 9. You may need to adapt them slightly for other EL9 flavors.

These also assume a system installation, performed as root. decisionengine will run as the decisionengine user.

#### 2.1.1 Install PostgreSQL

Install the default postgresql distributed on RHEL9, Postgress 13:

1. Install postgresql

```
dnf install -y postgresql postgresql-server
# optional, also: postgresql-devel
```

2. Enable postgresql

```
systemctl enable postgresql
```

3. Init the database

```
postgresql-setup --initdb -k
```

4. edit /var/lib/pgsql/data/pg\_hba.conf like the following:

```
[root@fermicloud371 ~]# diff /var/lib/pgsql/data/pg_hba.conf~ /var/lib/pgsql/data/
↪pg_hba.conf
80c80
< local    all                all                                peer
---
> local    all                all                                trust
```

(continues on next page)

(continued from previous page)

```

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*

5. start the database

```
systemctl start postgresql
```

6. 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=13
export PATH=~/.local/bin:$PATH
# you may also add these lines to ~/.bashrc

```

## 2.1.2 Install Redis

Install and start the message broker (Redis) container on your system. You can find more details on the redis document

1. Install Podman

```
dnf install -y podman
```

2. Run the Redis container

```

podman run --name decisionengine-redis -p 127.0.0.1:6379:6379 -d redis:6 --loglevel_
↪warning
# When prompted to select an image, pick "docker.io/library/redis:6".

```

## 2.1.3 Install Decision Engine and the standard modules

### 2.1.4 Install needed RPMs prerequisites

1. Make sure the correct repositories and priorities are set.

```

# CRB ("Code Ready Builder" - PowerTools ) is used for swig and other dependencies
dnf install -y yum-utils
dnf config-manager --set-enabled crb
# EPEL is used for OSG dependencies
dnf install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.

```

(continues on next page)

(continued from previous page)

```

↪rpm
# OSG is used for GlideinWMS, HTCSS and others
dnf install https://repo.opensciencegrid.org/osg/3.6/osg-3.6-el9-release-latest.rpm
dnf repolist
# Make sure all the above repos are enabled
# And change the Epel repository priority to make sure that comes after the OSG
↪repositories, which are 98 by default.
# Make sure that epel has:
# priority=99
vi /etc/yum.repos.d/epel.repo

```

2. Install the following prerequisites. Make sure that the required packages are installed and up to date.

```

# RPMs
# gcc, swig and make are needed for dependencies (jsonnet)
dnf install -y python3 python3-devel python3-cryptography python3-pip
dnf install -y gettext git make openssl-devel gcc gcc-c++ swig
# Install also these for RPM building:
dnf install -y python3-setuptools python3-wheel rpm-build
# Update Python pip
python3 -m pip install --upgrade --user pip
python3 -m pip install --upgrade --user setuptools wheel setuptools-scm[toml]

```

You can install using the provided RPMs (recommended for production) or via PIP install (recommended for development when you want to clone the Git repository and change the code). This section is for the RPM installation, the next one for the PIP installation. Use one or the other.

1. The yum repositories are available only within Fermilab. From the outside you will have to download the RPMs from *GitHub* <<https://github.com/HEPCloud/decisionengine/releases>> or use the PIP installarion (below).

Setup the decision engine yum repositories

```

# You need the development version 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
# This is the same as the EL9 development repo: http://ssi-rpm.fnal.
↪gov/hep/hepcloud-el9/ssi-hepcloud-dev.repo (http://ssi-rpm.fnal.gov/
↪hep/hepcloud-el9/development/)

```

2. Install the decision engine and add `--enablerepo=ssi-hepcloud-dev` for the latest development version

```
dnf install decisionengine dnf install decisionengine_modules
```

3. 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

```

(continues on next page)

(continued from previous page)

```
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==0.17.0 tabulate toposort_
↪structlog
python3 -m pip install --user wheel DBUtils sqlalchemy
python3 -m pip install --user pandas==2.0.0 numpy==1.24.2
python3 -m pip install --user "psycpg2-binary >= 2.9.6; platform_
↪python_implementation == 'CPython'"
python3 -m pip install --user "psycpg2cffi >= 2.9.0; platform_python_
↪implementation == 'PyPy'"
python3 -m pip install --user "cherrypy>=18.8.0" "kombu[redis]>=5.3.0b3
↪" "prometheus-client>=0.16.0"
python3 -m pip install --user "psutil>=5.8.0" "typing_extensions==4.1.1
↪"
# from decisionengine_modules setup.py
python3 -m pip install --user boto3 google-api-python-client
python3 -m pip install --user "google_auth<2dev,>=1.16.0" "urllib3>=1.
↪26.2"
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
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. Skip the PIP installation and go to the configuration section.



## 2.1.5 Install via PIP

Skip this if you did the RPM installation. This PIP installation is recommended for development when you want to clone the Git repository and change the code. There are a few extra steps (dependencies installation and setups) that are automated in the RPM installation.

1. GlideinWMS (3.10.x) and HTCondor (aka HTCSS) are needed for Decision Engine. The glideinwms packages will pull all the other dependencies.

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/>>. For a minimal installation, you can use the following command:

```
dnf install glideinwms-vofrontend-libs glideinwms-vofrontend-glidein_
↪ glideinwms-userschedd glideinwms-usercollector
dnf install glideinwms-vofrontend-core glideinwms-vofrontend-httpd
```

2. Setup the decision engine user and git repositories

```
useradd decisionengine
sudo -u decisionengine git clone https://github.com/HEPCloud/decisionengine.git ~
↪ decisionengine/decisionengine
sudo -u decisionengine git clone https://github.com/HEPCloud/decisionengine_modules.
↪ git ~decisionengine/decisionengine_modules
```

3. Install the decision engine from the git repositories

```
# Install the decisionengine framework and modules using setuptools
su - decisionengine -s /bin/bash
# Now you should be the decisionengine user in its home directory
pushd decisionengine
python3 setup.py develop --user
popd
pushd decisionengine_modules
python3 setup.py develop --user
popd
exit

# Create the required system files and directories (as root)
mkdir /etc/decisionengine
mkdir /var/log/decisionengine/
cp ~decisionengine/decisionengine/config/decision_engine.jsonnet /etc/decisionengine
cp -r ~decisionengine/decisionengine/src/decisionengine/framework/tests/etc/
↪ decisionengine/config.d /etc/decisionengine
chown -R decisionengine:decisionengine /etc/decisionengine
chown -R decisionengine:decisionengine /var/log/decisionengine
```

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

Remember that all the times that you start a new shell as `decisionengine` you need to add the PIP binary directory to the PATH:

```
export PATH=~/.local/bin:$PATH
```

## 2.1.6 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 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, used until v1.7, is no more supported.

## 2.1.7 Start decision engine

Start the service

```
# For the RPM install, as root:
systemctl start decisionengine
# For the PIP install, as decisionengine user (Python packages are installed in ~
↳ decisionengine/.local/bin/):
export PATH=~/.local/bin:$PATH
decisionengine --no-webserver &
```

## 2.1.8 Stop decision engine

To stop the service and remove the Redis container once you are done run the following:

```
# If you are in a RPM installation, as root:
systemctl stop decisionengine
# If you installed via PIP, as decisionengine:
export PATH=~/.local/bin:$PATH
de-client --stop
# Run the following as root (root started the container)
podman stop decisionengine-redis | xargs podman rm
```

## 2.1.9 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.

The following configuration has been added as an example to `/etc/decisionengine/config.d/test_channel.jsonnet` during the installation process:

```
{
  sources: {
    source1: {
      module: "decisionengine.framework.tests.SourceNOP",
      parameters: {},
      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, start or restart decision engine to start the new channel:

```
# For the RPM install:
systemctl restart decisionengine
```

(continues on next page)

(continued from previous page)

```
# For the PIP install, as decisionengine user
decisionengine --no-webserver &
```

Once the decisionengine is running, `de-client --status` should show the active test channel.

## 2.1.10 Setup pressure-based pilot submission

At this point Decision Engine, GlideinWMS and HTCondor are supposed to be installed and able to run.

We assume that the Frontend proxy and the VO proxy are already available.

**- Configure the pressure-based submission** | Write the configuration for the Decision Engine glideinwms module | To ease the process you can use the templates available in the [config\\_template contrib repo](#). | Copy the files from the EL9 folder into `/etc/decisionengine`, and the files in `EL9/config.d/` into `/etc/decisionengine/config.d`. | If you made changes to `decision_engine.jsonnet` please merge it with the version from the repository. | The important part from the is the glideinwms import `decision_engine.jsonnet` template is the line: `glideinwms: import 'glideinwms.libsonnet',.` | Those configuration files have a placeholder field `@TEMPLATE...@` | that needs to be replaced with the proper parameters according to your specific system setup. The README file has some suggestions.

Once those configuration files have been updated, we are ready to finalize the Decision Engine configuration.

### - Setup Redis

Start the message broker (Redis) as pod container:

```
podman run --name decisionengine-redis -p 127.0.0.1:6379:6379 -d redis:6 --loglevel_
↳ warning
```

**- Create GWMS frontend configuration** For this step you need first to restart the Decision Engine and then to run a configuration script. To do so, run:

```
# as root (fix the ownership of the frontend library files)
chown -R decisionengine: /var/lib/gwms-frontend
# for RPM installation as root
systemctl stop decisionengine
systemctl start decisionengine
ksu decisionengine -e /usr/bin/python3 /usr/lib/python3.9/site-packages/decisionengine_
↳ modules/glideinwms/configure_gwms_frontend.py
# for PIP installation as decisionengine
de-client --stop
decisionengine --no-webserver &
python3 ~decisionengine/decisionengine_modules/src/decisionengine_modules/glideinwms/
↳ configure_gwms_frontend.py
```

This command will create the file `/var/lib/gwms-frontend/vofrontend/de_frontend_config`

To allow a fresh start stop and reset everything:

1. stop the decisionengine (service):

```
# If you are in a RPM installation, as root: systemctl stop decisionengine # If you installed via PIP, as
decisionengine: de-client --stop
```

- remove the Redis container:

```
# Run the following as root (root started the container) podman stop decisionengine-redis | xargs podman
rm
```

- and reset the decisionengine DB in PostgreSQL:

```
dropdb -U postgres decisionengine
createdb -U postgres decisionengine
```

**- Run Decision Engine** Now all should be ready to run Decision Engine with a fresh start. Start the Redis container and the decisionengine service.

- Run Redis container:

```
podman run --name decisionengine-redis -p 127.0.0.1:6379:6379 -d redis:6 --loglevel_
↪warning
```

- Start decisionengine service and check its status:

```
# For RPM installations as root:
systemctl start decisionengine
sleep 5
systemctl status decisionengine
# For PIP installations as decisionengine:
decisionengine --no-webserver &
sleep 5
de-client --status
```

**- Submit a test job** Finally you can submit a test job to trigger Glidein requests and test the system.

- Switch to decisionengine user and make sure channel and sources are STEADY:

```
ksu decisionengine -e /bin/bash de-client --status
```

- prepare a Condor submission file mytest.submit with the following content:

```
# A test Condor submission file - mytest.submit
executable = /bin/hostname
universe = vanilla
+DESIRED_Sites = "@CHANGEME@"
log = test.log
output = test.out.%(Cluster).%(Process)
error = test.err.%(Cluster).%(Process)
queue 1
```

- submit the test job:

```
condor_submit mytest.submit
```

- check jobs in the queue:

```
condor_q
```

- check for available glideins:

```
condor_status
```

after test jobs are submitted it will take few minutes (usually no more than 10 minutes) to get some glideins and then get the job running.

Now the decisionengine user session can be closed to get back to the root session.

### - Stop Decision Engine service

Finally stop Decision Engine service and remove the Redis container:

```
# If you installed via RPMs run
systemctl stop decisionengine.service
# Run de-client --stop as decisionengine if you installed w/ PIP
podman stop decisionengine-redis | xargs podman rm
```

## 2.1.11 Troubleshooting

There is a known podman bug. podman is leaking volumes each time it starts a container, in the long run this is exhausting system resources. To check current volumes used by podman user can run `podman volume list`. To clean up volumes user can run `podman volume prune -f` after all podman container have been stopped and removed.

## 2.2 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.

### 2.2.1 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/reporpms/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
< 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"
```

## 2.2.2 Install Redis

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

## 2.2.3 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]
```

(continues on next page)

(continued from previous page)

```
# 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:
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](#)

## 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
```

## 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==0.17.0 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 "psycpg2-binary >= 2.8.6; platform_python_<
```

(continues on next page)



(continued from previous page)

```

↪implementation == 'CPython'"
python3 -m pip install --user "psycpg2cffi >= 2.9.0; platform_python_
↪implementation == 'PyPy'"
python3 -m pip install --user "cherrypy>=18.6.0" "kombu[redis]>=5.2.0rc1"
↪"prometheus-client>=0.10.0"
python3 -m pip install --user "psutil>=5.8.0" "typing_extensions==4.1.1"
# from decisionengine_modules setup.py
python3 -m pip install --user boto3 google-api-python-client
python3 -m pip install --user "google_auth<2dev,>=1.16.0" "urllib3>=1.26.2"
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
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.

## 2.2.4 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
```

## 2.2.5 Start decision engine

Start the service

```
systemctl start decisionengine
```

## 2.2.6 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",
```

(continues on next page)

(continued from previous page)

```

    parameters: {},
    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

## 2.2.7 Setup pressure-based pilot submission

At this point Decision Engine, GlideinWMS and HTCondor are supposed to be installed and able to run. We assume that the Frontend proxy and the VO proxy are already available.

Decision Engine configuration templates referred in this section are available in the [contrib repo](#).

Files from decisionengine folder need to be copied inside /etc/decisionengine. Those configuration files have the placeholder field @CHANGEME@ that needs to be replaced with a proper parameter according to the specific system setup.

Once those configuration file have been updated, we are ready to finalize the Decision Engine configuration.

### - Setup Redis

Start the message broker (Redis) as pod container:

```
podman run --name decisionengine-redis -p 127.0.0.1:6379:6379 -d redis:6 --loglevel_
↪warning
```

**- Create GWMS frontend configuration** For this step it is needed to run:

```
chown -R decisionengine: /var/lib/gwms-frontend
systemctl start decisionengine
ksu decisionengine -e /usr/bin/python3 /usr/lib/python3.6/site-packages/decisionengine_
↪modules/glideinwms/configure_gwms_frontend.py
```

This command will create the file `/var/lib/gwms-frontend/vofrontend/de_frontend_config`

At this point it is needed to stop decisionengine service and remove the Redis container:

```
systemctl stop decisionengine
podman stop decisionengine-redis | xargs podman rm
```

Now all should be ready to run Decision Engine.

### - Run Decision Engine

The procedure to run Decision Engine is as follow:

- Reset decisionengine DB:

```
dropdb -U postgres decisionengine
createdb -U postgres decisionengine
```

- Run Redis container:

```
podman run --name decisionengine-redis -p 127.0.0.1:6379:6379 -d redis:6 --loglevel_
↪warning
```

- Start decisionengine service and check its status:

```
systemctl start decisionengine
sleep 5
systemctl status decisionengine
```

### - Submit a test job

- Switch to decisionengine user and make sure channel and sources are STEADY:

```
ksu decisionengine -e /bin/bash de-client --status
```

- prepare a Condor submission file `mytest.submit` with the following content:

```
# A test Condor submission file - mytest.submit
executable = /bin/hostname
universe = vanilla
+DESIRED_Sites = "@CHANGEME@"
log = test.log
output = test.out.%(Cluster).%(Process)
```

(continues on next page)

(continued from previous page)

```
error = test.err.$(Cluster).$(Process)
queue 1
```

- submit the test job:

```
condor_submit mytest.submit
```

- check jobs in the queue:

```
condor_q
```

- check for available glideins:

```
condor_status
```

after test jobs are submitted it will take few minutes (usually no more than 10 minutes) to get some glideins and then get the job running.

Now the decisionengine user session can be closed to get back to the root session.

### - Stop Decision Engine service

Finally stop Decision Engine service and remove the Redis container:

```
systemctl stop decisionengine.service
podman stop decisionengine-redis | xargs podman rm
```

## 2.3 Installing and running HEPCloud's Decision Engine on EL8

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.

### 2.3.1 Install PostgreSQL

The default postgresql installed on RH8 is 9.2 which is outdated. Suggest to remove it and install 12 instead :

1. Disable the built-in PostgreSQL module

```
sudo dnf -qy module disable postgresql
```

2. Install postgresql 12

```
dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-8-x86_64/
↪pgdg-redhat-repo-latest.noarch.rpm
dnf install -y postgresql12 postgresql12-server
# optional, also: postgresql12-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
< 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"
# you may also add these lines to ~/.bashrc
```

## 2.3.2 Install Redis

Install and start the message broker (Redis) container on your system. You can find more details on the redis document

1. Install Podman

```
dnf install -y podman
```

2. Run the Redis container

```
podman run --name decisionengine-redis -p 127.0.0.1:6379:6379 -d redis:6 --loglevel_
↪warning
# When prompted to select an image, pick "docker.io/library/redis:6".
```

### 2.3.3 Install Decision Engine and the standard modules

1. Prerequisites setup. Make sure that the required packages (python39, gcc, ...) are installed and up to date.

```
# gcc, swig and make are needed for dependencies (jsonnet)
dnf install python39 python39-pip python39-setuptools python39-wheel \
    gcc gcc-c++ make \
    python39-devel swig openssl-devel git rpm-build
python3.9 -m pip install --upgrade --user pip
python3.9 -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 3.6, here is a brief summary of the setup:
dnf install -y https://repo.opensciencegrid.org/osg/3.6/osg-3.6-el8-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/>>. For a minimal installation, you can use the following command:

```
dnf install glideinwms-vofrontend-libs glideinwms-vofrontend-glidein glideinwms-userschedd
glideinwms-usercollector
```

2. Setup the decision engine user and git repositories

```
useradd decisionengine
sudo -u decisionengine git clone https://github.com/HEPCloud/decisionengine.git ~
↪decisionengine/decisionengine
sudo -u decisionengine git clone https://github.com/HEPCloud/decisionengine_modules.
↪git ~decisionengine/decisionengine_modules
```

3. Install the decision engine from the git repositories

```
# Install the decisionengine framework and modules using setuptools
su - decisionengine
pushd decisionengine
python3.9 setup.py develop --user
popd
pushd decisionengine_modules
python3.9 setup.py develop --user
popd
exit
```

(continues on next page)

(continued from previous page)

```
# Create the required system files and directories (as root)
mkdir /etc/decisionengine
mkdir /var/log/decisionengine/
cp ~decisionengine/decisionengine/config/decision_engine.jsonnet /etc/decisionengine
cp -r ~decisionengine/decisionengine/src/decisionengine/framework/tests/etc/
  ↳ decisionengine/config.d /etc/decisionengine
chown -R decisionengine:decisionengine /etc/decisionengine
chown -R decisionengine:decisionengine /var/log/decisionengine
```

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

## 2.3.4 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
  }
}
```

(continues on next page)



(continued from previous page)

```
}
}
```

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
```

### 2.3.5 Start decision engine

Start the service

```
# As decisionengine user
decisionengine --no-webserver &
```

### 2.3.6 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.

The following configuration has been added as an example to `/etc/decisionengine/config.d/test_channel.jsonnet` during the installation process:

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

```

(continues on next page)

(continued from previous page)

```
        actions: ['publisher1']
      }
    }
  },
  publishers: {
    publisher1: {
      module: "decisionengine.framework.tests.PublisherNOP",
      parameters: {}
    }
  }
}
```

Once the decisionengine is running, `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	<a href="#">view</a>
pep8.merge150.log	0 B	<a href="#">view</a>
pylint.merge150.log	0 B	<a href="#">view</a>
pytest.log	7.89 KB	<a href="#">view</a>
results.merge150.log	5.16 KB	<a href="#">view</a>
rpmbuild.tar	1.37 MB	<a href="#">view</a>

**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

## 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

**Build History** [trend](#)

find

<a href="#">#295</a>	<a href="#">Nov 19, 2020 2:23 AM</a>
<a href="#">#294</a>	<a href="#">Nov 18, 2020 2:23 AM</a>
<a href="#">#293</a>	<a href="#">Nov 17, 2020 2:23 AM</a>

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 'User-defined Axis' section is expanded, showing a 'Name' field with the value 'BRANCH' and a 'Values' field with the value '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 'Trigger/call builds on other projects' section is expanded, showing 'Projects to build' as 'decisionengine\_pipeline,decisionengine\_modules\_pipeline' and a checkbox for 'Block until the triggered projects finish their builds'. Below this, the 'Predefined parameters' section is expanded, showing 'Parameters' as '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: 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: `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,
```

(continues on next page)

(continued from previous page)

```
        # 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, metadata.generation\_id, metadata.key, metadata.state, metadata.generation\_time, metadata.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.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: *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: *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 buried 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: [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: [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: [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: *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, metadata.generation\_id, metadata.key, metadata.state, metadata.generation\_time, metadata.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 won't 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
- **datestamp** (*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: FixtureRequest*)

→  
Iterator[Connection]

Fixture factory for PostgreSQL.

**Parameters**

**request** – fixture request object

**Returns**

postgresql client

```
decisionengine.framework.dataspace.datasources.tests.fixtures.PG_PROG(request: FixtureRequest,
                               tmp_path_factory:
                               TempPathFactory) →
                               Iterator[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_WITH_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.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
```

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_not_exist(datasource)
    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(datasource)
    Does it error out if we ask for a bogus metadata element?
decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_get_taskmanager_exists(datasource)
    Can I get a taskmanager by name or name and uuid
decisionengine.framework.dataspace.datasources.tests.test_datasource_api.test_get_taskmanager_not_exist(datasource)
    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_exist(datasource)
    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: *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 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*, *timestamp=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 timestamp: **datetime** :arg timestamp: 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*:  
*FixtureRequest*) →  
Iterator[Connection]

Fixture factory for PostgreSQL.

**Parameters**

**request** – fixture request object

**Returns**

postgresql client

decisionengine.framework.dataspace.tests.fixtures.**PG\_PROG**(*request*: *FixtureRequest*,  
*tmp\_path\_factory*: *TempPathFactory*) →  
Iterator[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.datasources(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

**Parameters**

- **taskmanager\_name** (*str*) – Name of the TaskManager
- **taskmanager\_id** (*str, optional*) – ID of the TaskManager to retrieve. Defaults to None.

**Returns**

TaskManager information

**Return type**

dict

The dictionary returned looks like :

```
{
  'datestamp': datetime.datetime(2017, 12, 20, 17, 37, 17, 503210,
    tzinfo=psycpg2.tz.FixedOffsetTimezone(offset=-360, name=None)),
  'sequence_id': 135L,
  'name': 'AWS_Calculations',
  'taskmanager_id': '77B16EB5-C79E-45B0-B1B1-37E846692E1D'
}
```

**is\_expired(key=None)**

Check if the dataproduct for a given key or any key is expired

**keys()**

**mark\_expired**(*expiration\_time*)

Set the expiration\_time for the current generation of the dataproduct and mark it as expired if expiration\_time <= current time

**put**(*key, value, header, metadata=None*)

Put data into the DataBlock

**store\_taskmanager**(*taskmanager\_name, taskmanager\_id*)

Persist TaskManager, returns sequence number :type taskmanager\_name: string :type taskmanager\_id: :obj: string :rtype: int

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

Bases: 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: 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 dataprodut 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,
                                                         block_until_startup_complete=True)
```

A DE Server using a private database

```
decisionengine.framework.engine.tests.fixtures.PG_DE_DB_WITHOUT_SCHEMA(request:
                                                                           FixtureRequest) →
                                                                           Iterator[Connection]
```

Fixture factory for PostgreSQL.

**Parameters**

**request** – fixture request object

**Returns**

postgresql client

```
decisionengine.framework.engine.tests.fixtures.PG_PROG(request: FixtureRequest, tmp_path_factory:
                                                         TempPathFactory) →
                                                         Iterator[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\_SourceWorkers module**

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

`decisionengine.framework.engine.tests.test_SourceWorkers.source_config(channel_name,  
source_name)`

`decisionengine.framework.engine.tests.test_SourceWorkers.source_worker_for(src_config,  
global_config)`

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

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

`decisionengine.framework.engine.tests.test_SourceWorkers.test_worker_logger_wrong_rotation_method(global_config)`

`decisionengine.framework.engine.tests.test_SourceWorkers.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: 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 unguarded access:

```
ws = workers.get_unguarded() # 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

```
access()
```

```
accessed_by_another_thread()
```

```
get_unguarded()
```

### decisionengine.framework.engine.ClientMessageReceiver module

```
class decisionengine.framework.engine.ClientMessageReceiver.ClientMessageReceiver(exchange_name,  
                                                                                   ex-  
                                                                                   change_type,  
                                                                                   bro-  
                                                                                   ker_url,  
                                                                                   rout-  
                                                                                   ing_key_suffix,  
                                                                                   log-  
                                                                                   ger_name)
```

Bases: object

```
_receive(body, message)
```

```
execute(func, *args)
```

### 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: ThreadingMixIn, 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*)

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

**get\_logger**()

**metrics**()

**reaper\_start**(*delay*)

**reaper\_status**()

**reaper\_stop**()

**rm\_channel**(*channel, maybe\_timeout*)

**rpc\_block\_while**(*client\_queue, state\_str, timeout=None*)

**rpc\_get\_channel\_log\_level**(*client\_queue, channel*)

**rpc\_get\_log\_level**(*client\_queue*)

**rpc\_get\_source\_log\_level**(*client\_queue, source*)

**rpc\_kill\_channel**(*client\_queue, channel, timeout=None*)

**rpc\_metrics**(*client\_queue*)

Display collected metrics

**rpc\_ping**(*client\_queue*)

**rpc\_print\_product**(*client\_queue, product, columns=None, query=None, types=False, format=None*)

**rpc\_print\_products**(*client\_queue*)

**rpc\_product\_dependencies**(*client\_queue*)

**rpc\_query\_tool**(*client\_queue, product, format=None, start\_time=None*)

**rpc\_queue\_status**(*client\_queue*)

**rpc\_reaper\_start**(*client\_queue, delay=0*)

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

**rpc\_reaper\_status**(*client\_queue*)

**rpc\_reaper\_stop**(*client\_queue*)



**rpc\_rm\_channel**(*client\_queue, channel, maybe\_timeout*)

**rpc\_set\_channel\_log\_level**(*client\_queue, channel, log\_level*)

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

**rpc\_set\_source\_log\_level**(*client\_queue, source, log\_level*)

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

**rpc\_show\_config**(*client\_queue, channel*)

Show the configuration for a channel.

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

**rpc\_start\_channel**(*client\_queue, channel\_name*)

**rpc\_start\_channels**(*client\_queue*)

**rpc\_status**(*client\_queue*)

**rpc\_stop**(*client\_queue=None*)

**rpc\_stop\_channel**(*client\_queue, channel*)

**rpc\_stop\_channels**(*client\_queue*)

**service\_actions**()

Called by the serve\_forever() loop.

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

**start\_channel**(*channel\_name, src\_workers*)

**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: SimpleXMLRPCRequestHandler

**rpc\_paths** = ('/RPC2',)

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

Bases: 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._initial_start_channels(server)`

`decisionengine.framework.engine.DecisionEngine._queue_name_and_type(redis_obj, redis_member)`

`decisionengine.framework.engine.DecisionEngine._requests_in_flight(redis_obj, exchange_name)`

`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, logger_config,  
channel_name, exchange,  
broker_url)`

Bases: `Process`

Provides interface to loadable modules and events to synchronize execution

**get\_loglevel()**

**run()**

Get the data from source

**set\_loglevel\_value(log\_level)**

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

**setup\_logger()**

**take\_offline()**

```
class decisionengine.framework.engine.SourceWorkers.SourceWorkers(exchange, broker_url, log-
                                                                    ger=<BoundLoggerLazyProxy(logger=None,
                                                                    wrapper_class=None,
                                                                    processors=None,
                                                                    context_class=None,
                                                                    initial_values={}, log-
                                                                    ger_factory_args=('decisionengine',
                                                                    ))>)
```

Bases: object

This class manages and provides access to the Source 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_source'] = SourceWorker(...)
```

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

```
ws = workers.get_unguarded() # Access to ws is unprotected ws['new_source'].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

```
access()
```

```
detach(channel_name, source_names)
```

```
get_unguarded()
```

```
prune(channel_name, source_names)
```

```
remove_all(timeout)
```

```
update(channel_name, source_configs, logger_config)
```

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

```
decisionengine.framework.engine.de_client.command_for_args(argsparsed, de_socket)
```

argsparsed should be from create\_parser in this file

```
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.main(args_to_parse=None, logger_name='de_client')
```

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.**command\_for\_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.**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.**main**(*args\_to\_parse=None*,  
*logger\_name='de\_query\_tool'*)

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: [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: 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: Module
    _consumes = {}
    publish(data_block=None)
    shutdown()
decisionengine.framework.modules.Publisher.consumes(**kwargs)
decisionengine.framework.modules.Publisher.describe(cls, program_options=<class 'decision-
engine.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: 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: 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 'decisionengine.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 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

An enumeration.

```
ACTIVE = 2
```

```
BOOT = 0
```

```
ERROR = 7
```

```
IDLE = 1
```

```
OFFLINE = 6
```

```
SHUTDOWN = 5
```

```
SHUTTINGDOWN = 4
```

```
STEADY = 3
```

## decisionengine.framework.taskmanager.PublisherStatus module

PublisherStatus

The status of each decision-engine publisher is captured by a PublisherStatus object. The status can be queried to determine if a given publisher is enabled, and when it was last enabled or disabled. To access this information from a datablock, one must specify the a consumes statement:

The API for each relevant class is given below.

```
class decisionengine.framework.taskmanager.PublisherStatus.PublisherState(enabled, duration,
                                                                              since)
```

Bases: NamedTuple

```
_asdict()
```

Return a new dict which maps field names to their values.

```
_field_defaults = {}
```

```
_fields = ('enabled', 'duration', 'since')
```

```
classmethod _make(iterable)
```

Make a new PublisherState object from a sequence or iterable

```
_replace(**kws)
```

Return a new PublisherState object replacing specified fields with new values

**duration:** `timedelta`

datetime.timedelta object representing duration between now and when publisher was last enabled/disabled

**enabled:** `bool`

Boolean value indicating if publisher is enabled.

**since:** `datetime`

`datetime.datetime` object representing when publisher was last enabled/disabled

**class** `decisionengine.framework.taskmanager.PublisherStatus.PublisherStatus`(*status\_snapshot*)

Bases: `object`

Proxy object that provides publisher-status information.

**is\_enabled**(*publisher\_name*)

**Parameters**

**publisher\_name** (*str*) – The name of the configured publisher

**Returns**

If publisher is enabled or disabled

**Return type**

`bool`

**state**(*publisher\_name*)

**Parameters**

**publisher\_name** (*str*) – The name of the configured publisher

**Returns**

Full state of publisher

**Return type**

*`PublisherState`*

**class** `decisionengine.framework.taskmanager.PublisherStatus.PublisherStatusBoard`(*publisher\_names*)

Bases: `object`

Publisher status board owned by each decision channel

The status board is not a user-facing entity; it is owned by each decision channel, which updates the status of each publisher after they have been run.

**snapshot**()

**Returns**

An publisher-status object corresponding to now

**Return type**

*`PublisherStatus`*

**update**(*publisher\_name, result\_of\_publish*)

**Parameters**

- **publisher\_name** (*str*) – The name of the configured publisher
- **result\_of\_publish** (*bool*) – Whether the last execution of the publisher was successful

**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*,  
*routing\_keys*)

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 and 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, logger=<BoundLoggerLazyProxy(logger=None, wrapper\_class=None, processors=None, context\_class=None, initial\_values={}, logger\_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: *Transform*

`_consumes` = {'B': None}

`_produces` = {'A': None}

#### decisionengine.framework.tests.BATransform module

`class decisionengine.framework.tests.BATransform.BATransform`(*module\_parameters, \*args, \*\*kwargs*)

Bases: *Transform*

`_consumes` = {'A': None}

`_produces` = {'B': None}

#### decisionengine.framework.tests.DynamicPublisher module

`class decisionengine.framework.tests.DynamicPublisher.DynamicPublisher`(*config*)

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

**decisionengine.framework.tests.DynamicTransform module**

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

    transform(data_block)
```

**decisionengine.framework.tests.ErringPublisher module**

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

**decisionengine.framework.tests.ErrorOnAcquire module**

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

**decisionengine.framework.tests.FailingPublisher module**

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

**decisionengine.framework.tests.IntSource module**

```

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

### decisionengine.framework.tests.SourceAlias module

### decisionengine.framework.tests.SourceNOP module

```
class decisionengine.framework.tests.SourceNOP.SourceNOP(config)
```

Bases: *Source*

```
_produces = {'foo': <class 'pandas.core.frame.DataFrame'>}
```

```
acquire()
```

### decisionengine.framework.tests.SourceWithMissingProduces module

```
class decisionengine.framework.tests.SourceWithMissingProduces.SourceWithMissingProduces(set_of_parameters)
```

Bases: *Source*

### decisionengine.framework.tests.SourceWithSampleConfigNOP module

```
class decisionengine.framework.tests.SourceWithSampleConfigNOP.SourceWithSampleConfigNOP(config)
```

Bases: *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: 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: 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: 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: 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, block\_until\_startup\_complete=True*)

A DE Server using a private database

`decisionengine.framework.tests.fixtures.PG_DE_DB_WITHOUT_SCHEMA`(*request: FixtureRequest*) → *Iterator[Connection]*

Fixture factory for PostgreSQL.

### Parameters

**request** – fixture request object

### Returns

postgresql client

`decisionengine.framework.tests.fixtures.PG_PROG`(*request: FixtureRequest, tmp\_path\_factory: TempPathFactory*) → *Iterator[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_get_loglevel(deserver)
```

```
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_logger(deserver,  
                                                                                    caplog)
```

Make sure the actual client console call goes to a logging destination

### **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_error_on_acquire(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\_publisher\_status module**

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

### **decisionengine.framework.tests.test\_publisher\_status\_board module**

```
decisionengine.framework.tests.test_publisher_status_board.test_publisher_status_board()
```

### **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*,  
*caplog*)

Verify client can get channel products even when none are run

## decisionengine.framework.tests.test\_status\_during\_startup module

decisionengine.framework.tests.test\_status\_during\_startup.test\_status\_during\_startup(*deserver\_no\_wait*)

## 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 marched

**Parameters**

- **constraint** (*dict* / *None*) – combined constraints
- **linelist** (*list*) – List of line fields
- **linedict** (*dict*) – Dictionary with structured elements

**Returns**

True is 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:
    ~prometheus_client.registry.CollectorRegistry |
    None =
    <prometheus_client.registry.CollectorRegistry
    object>, _labelvalues: ~typing.Sequence[str] |
    None = None)
```

Bases: Counter

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

```
class decisionengine.framework.util.metrics.Gauge(*args, **kwargs)
```

Bases: Gauge

Override prometheus client Gauge so that multiprocessing\_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:
    ~prometheus_client.registry.CollectorRegistry
    | None =
    <prometheus_client.registry.CollectorRegistry
    object>, _labelvalues: ~typing.Sequence[str]
    | None = None, buckets:
    ~typing.Sequence[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: 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:
    ~prometheus_client.registry.CollectorRegistry |
    None =
    <prometheus_client.registry.CollectorRegistry
    object>, _labelvalues: ~typing.Sequence[str] |
    None = None)
```

Bases: 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: [Singleton](#)

Singleton pattern using Metaclass with weak refs

**\_instances** = <WeakValueDictionary>

**class** decisionengine.framework.util.singleton.ScopedSingletonABC(*name, bases, namespace, \*\*kwargs*)

Bases: ABCMeta, [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: ABCMeta, [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, 134  
decisionengine.framework, 134  
decisionengine.framework.about, 133  
decisionengine.framework.config, 67  
decisionengine.framework.config.ChannelConfigHandler, 65  
decisionengine.framework.config.policies, 66  
decisionengine.framework.config.tests, 65  
decisionengine.framework.config.tests.test\_config, 63  
decisionengine.framework.config.tests.test\_de\_std, 64  
decisionengine.framework.config.tests.test\_policies, 65  
decisionengine.framework.config.ValidConfig, 66  
decisionengine.framework.dataspace, 95  
decisionengine.framework.dataspace.datablock, 88  
decisionengine.framework.dataspace.datasource, 91  
decisionengine.framework.dataspace.datasources, 84  
decisionengine.framework.dataspace.datasources.null, 82  
decisionengine.framework.dataspace.datasources.sqlalchemy\_ds, 75  
decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.datasource\_api, 67  
decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema, 72  
decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.utils, 75  
decisionengine.framework.dataspace.datasources.tests, 82  
decisionengine.framework.dataspace.datasources.tests.fixtures, 79  
decisionengine.framework.dataspace.datasources.tests.test\_datasource\_api, 80  
decisionengine.framework.dataspace.dataspace, 93  
decisionengine.framework.dataspace.maintain, 94  
decisionengine.framework.dataspace.tests, 88  
decisionengine.framework.dataspace.tests.fixtures, 84  
decisionengine.framework.dataspace.tests.test\_datablock, 86  
decisionengine.framework.dataspace.tests.test\_datablock\_zl, 87  
decisionengine.framework.dataspace.tests.test\_datasource, 87  
decisionengine.framework.dataspace.tests.test\_dataspace, 87  
decisionengine.framework.dataspace.tests.test\_Reaper, 85  
decisionengine.framework.engine, 104  
decisionengine.framework.engine.ChannelWorkers, 98  
decisionengine.framework.engine.ClientMessageReceiver, 99  
decisionengine.framework.engine.de\_client, 103  
decisionengine.framework.engine.de\_query\_tool, 104  
decisionengine.framework.engine.DecisionEngine, 99  
decisionengine.framework.engine.SourceWorkers, 102  
decisionengine.framework.engine.tests, 98  
decisionengine.framework.engine.tests.fixtures, 95  
decisionengine.framework.engine.tests.test\_ChannelWorkers, 96  
decisionengine.framework.engine.tests.test\_client\_only, 97  
decisionengine.framework.engine.tests.test\_query\_tool\_only, 97  
decisionengine.framework.engine.tests.test\_SourceWorkers, 96  
decisionengine.framework.engine.tests.test\_startup, 97  
decisionengine.framework.engine.tests.test\_verify\_redis\_se

98	decisionengine.framework.modules.tests.test_Module,
decisionengine.framework.logicengine, 110	110
decisionengine.framework.logicengine.BooleanExpression,	decisionengine.framework.modules.tests.test_module_decorat
107	111
decisionengine.framework.logicengine.FactLookup,	decisionengine.framework.modules.tests.test_Publisher,
107	110
decisionengine.framework.logicengine.LogicEngine,	decisionengine.framework.modules.tests.test_QueueLogger,
108	110
decisionengine.framework.logicengine.Rule,	decisionengine.framework.modules.tests.test_Source,
109	111
decisionengine.framework.logicengine.RuleEngine,	decisionengine.framework.modules.tests.test_Transform,
109	111
decisionengine.framework.logicengine.tests,	decisionengine.framework.modules.tests.test_translate_pro
107	111
decisionengine.framework.logicengine.tests.test_booleanengine,	decisionengine.framework.modules.Transform,
104	113
decisionengine.framework.logicengine.tests.test_casadeengine,	decisionengine.framework.modules.translate_product_name,
105	115
decisionengine.framework.logicengine.tests.test_decisionengine,	decisionengine.framework.taskmanager, 121
105	decisionengine.framework.taskmanager.LatestMessages,
decisionengine.framework.logicengine.tests.test_duplicate_fact_names,	
105	decisionengine.framework.taskmanager.module_graph,
decisionengine.framework.logicengine.tests.test_facts,	120
105	decisionengine.framework.taskmanager.ProcessingState,
decisionengine.framework.logicengine.tests.test_fail_dne,	116
106	decisionengine.framework.taskmanager.PublisherStatus,
decisionengine.framework.logicengine.tests.test_pandas_fact,	
106	decisionengine.framework.taskmanager.SourceProductCache,
decisionengine.framework.logicengine.tests.test_rule_with_negated_fact,	
106	decisionengine.framework.taskmanager.TaskManager,
decisionengine.framework.logicengine.tests.test_simple_configuration,	
106	decisionengine.framework.tests, 129
decisionengine.framework.modules, 115	decisionengine.framework.tests.ABTransform,
decisionengine.framework.modules.de_logger,	121
114	decisionengine.framework.tests.BATransform,
decisionengine.framework.modules.describe,	121
114	decisionengine.framework.tests.DynamicPublisher,
decisionengine.framework.modules.EmptySource,	121
112	decisionengine.framework.tests.DynamicSource,
decisionengine.framework.modules.logging_configDict,	122
115	decisionengine.framework.tests.DynamicTransform,
decisionengine.framework.modules.Module, 112	122
decisionengine.framework.modules.print_descriptions,	decisionengine.framework.tests.ErringPublisher,
115	122
decisionengine.framework.modules.Publisher,	decisionengine.framework.tests.ErrorOnAcquire,
112	122
decisionengine.framework.modules.QueueLogger,	decisionengine.framework.tests.FailingPublisher,
113	122
decisionengine.framework.modules.Source, 113	decisionengine.framework.tests.fixtures, 126
decisionengine.framework.modules.tests, 112	decisionengine.framework.tests.IntSource, 123
decisionengine.framework.modules.tests.test_decisionengine,	decisionengine.framework.tests.ModuleProgramOptions,
111	123
decisionengine.framework.modules.tests.test_EmptySource,	decisionengine.framework.tests.PublisherNOP,
110	124

[decisionengine.framework.tests.PublisherWithMissingConsumes](#), [124](#)  
[decisionengine.framework.util.logparser](#), [130](#)  
[decisionengine.framework.util.metrics](#), [131](#)  
[decisionengine.framework.tests.SourceAlias](#), [124](#)  
[decisionengine.framework.util.reaper](#), [132](#)  
[decisionengine.framework.util.redis\\_stats](#),  
[decisionengine.framework.tests.SourceNOP](#), [124](#) [133](#)  
[decisionengine.framework.tests.SourceWithMissingProduces](#), [124](#)  
[decisionengine.framework.util.singleton](#), [133](#)  
[decisionengine.framework.util.sockets](#), [133](#)  
[decisionengine.framework.tests.SourceWithSampleConfigNOP](#), [124](#)  
[decisionengine.framework.util.subclasses](#), [133](#)  
[decisionengine.framework.version](#), [134](#)  
[decisionengine.framework.tests.SupportsConfigPublisher](#), [125](#)  
[decisionengine.tests.test\\_framework\\_package](#),  
[decisionengine.framework.tests.test\\_client\\_errors](#), [126](#) [134](#)  
[decisionengine.framework.tests.test\\_client\\_server](#),  
[127](#)  
[decisionengine.framework.tests.test\\_combined\\_channels](#),  
[127](#)  
[decisionengine.framework.tests.test\\_defaults](#),  
[127](#)  
[decisionengine.framework.tests.test\\_dynamic\\_test\\_modules](#),  
[127](#)  
[decisionengine.framework.tests.test\\_empty\\_config](#),  
[127](#)  
[decisionengine.framework.tests.test\\_error\\_on\\_acquire](#),  
[127](#)  
[decisionengine.framework.tests.test\\_module\\_program\\_options](#),  
[128](#)  
[decisionengine.framework.tests.test\\_publisher\\_status](#),  
[128](#)  
[decisionengine.framework.tests.test\\_publisher\\_status\\_board](#),  
[128](#)  
[decisionengine.framework.tests.test\\_query\\_tool\\_server](#),  
[128](#)  
[decisionengine.framework.tests.test\\_reaper](#),  
[128](#)  
[decisionengine.framework.tests.test\\_same\\_source\\_types](#),  
[128](#)  
[decisionengine.framework.tests.test\\_sample\\_config](#),  
[128](#)  
[decisionengine.framework.tests.test\\_shared\\_sources](#),  
[129](#)  
[decisionengine.framework.tests.test\\_start\\_with\\_bad\\_channels](#),  
[129](#)  
[decisionengine.framework.tests.test\\_status\\_during\\_startup](#),  
[129](#)  
[decisionengine.framework.tests.TransformNOP](#),  
[125](#)  
[decisionengine.framework.tests.TransformWithMissingProducesConsumes](#),  
[125](#)  
[decisionengine.framework.tests.WriteToDisk](#),  
[125](#)  
[decisionengine.framework.util](#), [133](#)  
[decisionengine.framework.util.countdown](#), [129](#)  
[decisionengine.framework.util.fs](#), [130](#)





# INDEX

## Symbols

\_DEFAULT\_MULTIPROC\_MODE (decisionengine.framework.util.metrics.Gauge attribute), 131  
 \_\_determine\_multiprocess\_mode\_existence() (decisionengine.framework.util.metrics.Gauge method), 131  
 \_\_insert() (decisionengine.framework.dataspace.datablock.DataBlock method), 88  
 \_\_update() (decisionengine.framework.dataspace.datablock.DataBlock method), 88  
 \_abc\_impl (decisionengine.framework.config.ValidConfig.ValidConfig attribute), 66  
 \_abc\_impl (decisionengine.framework.dataspace.datablock.Header attribute), 90  
 \_abc\_impl (decisionengine.framework.dataspace.datablock.Metadata attribute), 90  
 \_abc\_impl (decisionengine.framework.dataspace.datasource.DataSource attribute), 91  
 \_abc\_impl (decisionengine.framework.dataspace.datasources.null.NullDataSource attribute), 82  
 \_abc\_impl (decisionengine.framework.dataspace.datasources.sqlalchemy\_us.SQLAlchemyDS attribute), 75  
 \_abc\_impl (decisionengine.framework.dataspace.datasources.sqlalchemy\_us.datasource\_api.SQLAlchemyDS attribute), 68  
 \_abc\_impl (decisionengine.framework.util.metrics.Counter attribute), 131  
 \_abc\_impl (decisionengine.framework.util.metrics.Gauge attribute), 131  
 \_abc\_impl (decisionengine.framework.util.metrics.Histogram attribute), 132  
 \_abc\_impl (decisionengine.framework.util.metrics.Summary attribute), 132  
 \_asdict() (decisionengine.framework.taskmanager.PublisherStatus.PublisherStatus method), 117  
 \_channel\_config\_dir() (in module decisionengine.framework.config.tests.test\_config), 63  
 \_channel\_preamble() (in module decisionengine.framework.engine.DecisionEngine), 101  
 \_check\_keys() (in module decisionengine.framework.config.ChannelConfigHandler), 65  
 \_check\_metrics\_env() (in module decisionengine.framework.engine.DecisionEngine), 102  
 \_check\_override() (in module decisionengine.framework.engine.tests.test\_startup), 97  
 \_config\_from\_file() (in module decisionengine.framework.config.ValidConfig), 66  
 \_consumed\_products() (in module decisionengine.framework.taskmanager.module\_graph), 120  
 \_consumes (decisionengine.framework.modules.Publisher.Publisher attribute), 112  
 \_consumes (decisionengine.framework.modules.Transform.Transform attribute), 113  
 \_consumes (decisionengine.framework.tests.ABTransform.ABTransform attribute), 121  
 \_consumes (decisionengine.framework.tests.BATransform.BATransform attribute), 121  
 \_consumes (decisionengine.framework.tests.ERTransform.ERTransform attribute), 121  
 \_consumes (decisionengine.framework.tests.ErringPublisher.ErringPublisher attribute), 122  
 \_consumes (decisionengine.framework.tests.FailingPublisher.FailingPublisher attribute), 122  
 \_consumes (decisionengine.framework.tests.PublisherNOP.PublisherNOP attribute), 124  
 \_consumes (decisionengine.framework.tests.TransformNOP.TransformNOP attribute), 125  
 \_consumes\_not\_subset() (in module decisionengine.framework.tests.test\_start\_with\_bad\_channels), 129  
 \_create\_data\_server() (in module decisionengine.framework.engine.DecisionEngine), 102  
 \_create\_facts\_dataframe() (decisionengine.framework.logicengine.LogicEngine.LogicEngine method), 108  
 \_create\_module\_instance() (in module decisionengine.framework.taskmanager.module\_graph), 120

`_dataframe_to_column_names()` (decisionengine.framework.engine.DecisionEngine.DecisionEngine63 method), 99  
`_dataframe_to_csv()` (decisionengine.framework.engine.DecisionEngine.DecisionEngine102 method), 99  
`_dataframe_to_json()` (decisionengine.framework.engine.DecisionEngine.DecisionEngine102 method), 99  
`_dataframe_to_table()` (decisionengine.framework.engine.DecisionEngine.DecisionEngine116 method), 99  
`_dataframe_to_vertical_tables()` (decisionengine.framework.engine.DecisionEngine.DecisionEngine65 method), 99  
`_derived_class()` (in module decisionengine.framework.util.subclasses), 133  
`_dispatch()` (decisionengine.framework.engine.DecisionEngine.DecisionEngine65 method), 99  
`_expected_acquire_result()` (in module decisionengine.framework.tests.ModuleProgramOptions), 123  
`_expected_circularity()` (in module decisionengine.framework.tests.test\_start\_with\_bad\_channels), 129  
`_expected_config_template()` (in module decisionengine.framework.tests.ModuleProgramOptions), 123  
`_expected_config_template_with_comments()` (in module decisionengine.framework.tests.ModuleProgramOptions), 123  
`_expected_help()` (in module decisionengine.framework.tests.ModuleProgramOptions), 123  
`_expected_source_help()` (in module decisionengine.framework.tests.ModuleProgramOptions), 123  
`_field_defaults` (decisionengine.framework.taskmanager.PublisherStatus.PublisherStatus117 attribute), 117  
`_fields` (decisionengine.framework.taskmanager.PublisherStatus.PublisherStatus117 attribute), 117  
`_find_only_one_subclass()` (in module decisionengine.framework.taskmanager.module\_graph), 120  
`_get_de_conf_manager()` (in module decisionengine.framework.engine.DecisionEngine), 102  
`_get_global_config()` (in module decisionengine.framework.engine.DecisionEngine), 102  
`_global_config_file()` (in module decisionengine.framework.config.tests.test\_config), 102  
`_initial_start_channels()` (in module decisionengine.framework.engine.DecisionEngine), 102  
`_instances` (decisionengine.framework.util.singleton.ScopedSingleton attribute), 133  
`_instances` (decisionengine.framework.util.singleton.Singleton attribute), 133  
`_listen()` (decisionengine.framework.taskmanager.LatestMessages.LatestMessages116 method), 116  
`_load_channel()` (decisionengine.framework.config.ChannelConfigHandler.ChannelConfigHandler116 method), 116  
`_make()` (decisionengine.framework.taskmanager.PublisherStatus.PublisherStatus117 class method), 117  
`_make_de_logger()` (in module decisionengine.framework.config.ChannelConfigHandler), 117  
`_make_workers_for()` (in module decisionengine.framework.taskmanager.module\_graph), 120  
`_missing_consumes()` (in module decisionengine.framework.tests.test\_start\_with\_bad\_channels), 129  
`_missing_produces()` (in module decisionengine.framework.tests.test\_start\_with\_bad\_channels), 129  
`_normalize()` (in module decisionengine.framework.tests.ModuleProgramOptions), 123  
`_par_default()` (in module decisionengine.framework.modules.describe), 114  
`_par_type()` (in module decisionengine.framework.modules.describe), 114  
`_print_comment()` (in module decisionengine.framework.modules.print\_description), 115  
`_print_type()` (in module decisionengine.framework.modules.print\_description), 115  
`_print_value()` (in module decisionengine.framework.modules.print\_description), 115  
`_produced_products()` (in module decisionengine.framework.taskmanager.module\_graph), 120  
`_produces` (decisionengine.framework.modules.Source.Source113 attribute), 113  
`_produces` (decisionengine.framework.modules.Transform.Transform113 attribute), 113  
`_produces` (decisionengine.framework.tests.ABTransform.ABTransform121 attribute), 121  
`_produces` (decisionengine.framework.tests.BATransform.BATransform121 attribute), 121

attribute), 121  
 \_produces (decisionengine.framework.tests.ErrorOnAcquire.ErrorOnAcquire), 112  
 attribute), 122  
 \_produces (decisionengine.framework.tests.IntSource.IntSource), 121  
 attribute), 123  
 \_produces (decisionengine.framework.tests.SourceNOP.SourceNOP), 121  
 attribute), 124  
 \_produces (decisionengine.framework.tests.SourceWithSampleConfigNOP.SourceWithSampleConfigNOP), 121  
 attribute), 124  
 \_produces (decisionengine.framework.tests.TransformNOP.TransformNOP), 121  
 attribute), 125  
 \_queue\_name\_and\_type() (in module decisionengine.framework.engine.DecisionEngine), 102  
 \_reaper\_loop() (decisionengine.framework.dataspace.maintain.Reaper method), 94  
 \_receive() (decisionengine.framework.engine.ClientMessageReceiver), 99  
 \_replace() (decisionengine.framework.taskmanager.PublisherStatusPublisher.State method), 117  
 \_requests\_in\_flight() (in module decisionengine.framework.engine.DecisionEngine), 102  
 \_run\_as\_main() (in module decisionengine.framework.tests.ModuleProgramOptions), 124  
 \_sa\_class\_manager (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db.schema.Dataproduct attribute), 72  
 \_sa\_class\_manager (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db.schema.Dataproduct attribute), 72  
 \_sa\_class\_manager (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db.schema.Metadata attribute), 73  
 \_sa\_class\_manager (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db.schema.Metadata attribute), 74  
 \_sa\_class\_manager (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db.schema.Metadata attribute), 74  
 \_sa\_registry (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db.schema.Metadata attribute), 72  
 \_setitem() (decisionengine.framework.dataspace.datablock.DataBlock method), 88  
 \_spec\_from\_file\_name() (in module decisionengine.framework.modules.print\_description), 115  
 \_start\_de\_server() (in module decisionengine.framework.engine.DecisionEngine), 102  
 \_supported\_config (decisionengine.framework.modules.EmptySource.EmptySource attribute), 112  
 \_supported\_config (decisionengine.framework.tests.DynamicPublisher.DynamicPublisher attribute), 121  
 \_supported\_config (decisionengine.framework.tests.DynamicSource.DynamicSource attribute), 121  
 \_supported\_config (decisionengine.framework.tests.DynamicTransform.DynamicTransform attribute), 122  
 \_supported\_config (decisionengine.framework.tests.IntSource.IntSource attribute), 123  
 \_supported\_config (decisionengine.framework.tests.SourceWithSampleConfigNOP.SourceWithSampleConfigNOP attribute), 124  
 \_supported\_config (decisionengine.framework.tests.SupportsConfigPublisher.SupportsConfigPublisher attribute), 124  
 \_supported\_config (decisionengine.framework.tests.WriteToDisk.WriteToDisk attribute), 125  
 \_verify\_redis\_server() (in module decisionengine.framework.engine.DecisionEngine), 102  
 \_verify\_redis\_url() (in module decisionengine.framework.engine.DecisionEngine), 102  
 A  
 ABTransform (class in decisionengine.framework.tests.ABTransform), 121  
 access() (decisionengine.framework.engine.ChannelWorkers.ChannelWorkers method), 103  
 access() (decisionengine.framework.engine.SourceWorkers.SourceWorkers method), 103  
 accessed\_by\_main\_thread() (decisionengine.framework.engine.ChannelWorkers.ChannelWorkers method), 99  
 acquire() (decisionengine.framework.modules.EmptySource.EmptySource method), 112  
 acquire() (decisionengine.framework.modules.Source.Source method), 112  
 acquire() (decisionengine.framework.tests.DynamicSource.DynamicSource method), 122  
 acquire() (decisionengine.framework.tests.ErrorOnAcquire.ErrorOnAcquire method), 122  
 acquire() (decisionengine.framework.tests.IntSource.IntSource method), 123  
 acquire() (decisionengine.framework.tests.SourceNOP.SourceNOP method), 124  
 acquire() (decisionengine.framework.tests.SourceWithSampleConfigNOP.SourceWithSampleConfigNOP method), 124

**AcquireWithConfig** (class in *decisionengine.framework.tests.ModuleProgramOptions*), 123  
**AcquireWithSampleConfig** (class in *decisionengine.framework.tests.ModuleProgramOptions*), 123  
**ACTIVE** (*decisionengine.framework.taskmanager.ProcessingState.State* attribute), 117  
**add\_engine\_pidguard()** (in module *decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.uns*), 75  
**all\_subclasses()** (in module *decisionengine.framework.util.subclasses*), 133  
**B**  
**Base** (class in *decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.uns*), 72  
**BATransform** (class in *decisionengine.framework.tests.BATransform*), 121  
**block\_while()** (*decisionengine.framework.engine.DecisionEngine.DecisionEngine* method), 100  
**BooleanExpression** (class in *decisionengine.framework.logicengine.BooleanExpression*), 107  
**BOOT** (*decisionengine.framework.taskmanager.ProcessingState.State* attribute), 117  
**C**  
**channel\_config\_dir()** (in module *decisionengine.framework.config.policies*), 66  
**channel\_workers()** (in module *decisionengine.framework.taskmanager.module\_graph*), 120  
**ChannelConfigHandler** (class in *decisionengine.framework.config.ChannelConfigHandler*), 65  
**ChannelWorker** (class in *decisionengine.framework.engine.ChannelWorkers*), 98  
**ChannelWorkers** (class in *decisionengine.framework.engine.ChannelWorkers*), 98  
**ChannelWorkers.Access** (class in *decisionengine.framework.engine.ChannelWorkers*), 99  
**Clean** (*decisionengine.framework.engine.DecisionEngine.StopState* attribute), 101  
**ClientMessageReceiver** (class in *decisionengine.framework.engine.ClientMessageReceiver*), 99  
**clone\_model()** (in module *decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.uns*), 75  
**close()** (*decisionengine.framework.dataspace.datasource.DataSource* method), 91  
**close()** (*decisionengine.framework.dataspace.datasources.null.NullDataSource* method), 82  
**close()** (*decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.uns* method), 68  
**close()** (*decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.uns* method), 75  
**close()** (*decisionengine.framework.dataspace.dataspace.DataSpace* method), 93  
**command\_for\_args()** (in module *decisionengine.framework.engine.de\_client*), 103  
**command\_for\_args()** (in module *decisionengine.framework.engine.de\_query\_tool*), 104  
**compress()** (*decisionengine.framework.dataspace.datablock*), 90  
**config()** (in module *decisionengine.framework.config.tests.test\_de\_std*), 64  
**config()** (in module *decisionengine.framework.dataspace.tests.test\_Reaper*), 85  
**ConfigTemplate** (class in *decisionengine.framework.tests.ModuleProgramOptions*), 123  
**configure\_listener()** (*decisionengine.framework.modules.QueueLogger.QueueLogger* method), 113  
**configure\_logging()** (in module *decisionengine.framework.modules.de\_logger*), 114  
**connect()** (*decisionengine.framework.dataspace.datasource.DataSource* method), 91  
**connect()** (*decisionengine.framework.dataspace.datasources.null.NullDataSource* method), 82  
**connect()** (*decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.uns* method), 68  
**connect()** (*decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.uns* method), 75  
**console\_scripts\_main()** (in module *decisionengine.framework.engine.de\_client*), 103  
**console\_scripts\_main()** (in module *decisionengine.framework.engine.de\_query\_tool*), 104  
**console\_scripts\_main()** (in module *decisionengine.framework.util.logparser*), 130  
**consume()** (*decisionengine.framework.taskmanager.LatestMessages.LatestMessages* method), 116  
**consumes()** (*decisionengine.framework.logicengine.LogicEngine.LogicEngine* method), 108  
**consumes()** (in module *decisionengine.framework.logicengine.LogicEngine.LogicEngine*), 108

`nengine.framework.modules.Module)`, 112  
`consumes()` (in module `decisionengine.framework.modules.Publisher`), 112  
`consumes()` (in module `decisionengine.framework.modules.Transform`), 113  
`Countdown` (class in `decisionengine.framework.util.countdown`), 129  
`Counter` (class in `decisionengine.framework.util.metrics`), 131  
`create_channel()` (`decisionengine.framework.engine.DecisionEngine.DecisionEngine` method), 100  
`create_parser()` (in module `decisionengine.framework.engine.de_client`), 103  
`create_parser()` (in module `decisionengine.framework.engine.de_query_tool`), 104  
`create_parser()` (in module `decisionengine.framework.util.logparser`), 130  
`create_tables()` (`decisionengine.framework.dataspace.datasource.DataSource` method), 91  
`create_tables()` (`decisionengine.framework.dataspace.datasources.null.NullDataSource` method), 82  
`create_tables()` (`decisionengine.framework.dataspace.datasources.sqlalchemy.DataSource` method), 68  
`create_tables()` (`decisionengine.framework.dataspace.datasources.sqlalchemy.DataSource` method), 75  
`create_time` (`decisionengine.framework.dataspace.datasources.sqlalchemy.DataSource` attribute), 73  
`creator` (`decisionengine.framework.dataspace.datasources.sqlalchemy.DataSource` attribute), 73  
**D**  
`data_block_put()` (`decisionengine.framework.taskmanager.TaskManager.TaskManager` method), 119  
`DataBlock` (class in `decisionengine.framework.dataspace.datablock`), 88  
`Dataproduct` (class in `decisionengine.framework.dataspace.datasources.sqlalchemy.DataSource`), 72  
`dataproduct_table` (`decisionengine.framework.dataspace.datasource.DataSource` attribute), 91  
`DataSource` (class in `decisionengine.framework.dataspace.datasource`), 91  
`datasource()` (in module `decisionengine.framework.dataspace.datasources.tests.fixtures`), 80  
`datasource()` (in module `decisionengine.framework.dataspace.tests.fixtures`), 85  
`DataSpace` (class in `decisionengine.framework.dataspace.dataspace`), 93  
`dataspace()` (in module `decisionengine.framework.dataspace.tests.fixtures`), 85  
`DataSourceConfigurationError`, 94  
`DataSourceConnectionError`, 94  
`DataSourceError`, 94  
`DataSourceExistsError`, 94  
`datestamp` (`decisionengine.framework.dataspace.datasources.sqlalchemy.DataSource` attribute), 74  
`decision_cycle()` (`decisionengine.framework.taskmanager.TaskManager.TaskManager` method), 119  
`decisionengine` module, 134  
`DecisionEngine` (class in `decisionengine.framework.engine.DecisionEngine`), 99  
`decisionengine.framework` module, 133  
`decisionengine.framework.about` module, 133  
`decisionengine.framework.config` module, 67  
`decisionengine.framework.config.ChannelConfigHandler` module, 67  
`decisionengine.framework.config.schema.Header` module, 67  
`decisionengine.framework.config.policies` module, 65  
`decisionengine.framework.config.tests` module, 65  
`decisionengine.framework.config.tests.test_config` module, 63  
`decisionengine.framework.config.tests.test_de_std` module, 64  
`decisionengine.framework.config.tests.test_policies` module, 65  
`decisionengine.framework.config.ValidConfig` module, 66  
`decisionengine.framework.dataspace` module, 95  
`decisionengine.framework.dataspace.datablock` module, 88  
`decisionengine.framework.dataspace.datasource` module, 91  
`decisionengine.framework.dataspace.datasources`



module, 84	module, 96
decisionengine.framework.dataspace.datasourcesdecisionengine	decisionengine.framework.engine.tests.test_client_only
module, 82	module, 97
decisionengine.framework.dataspace.datasourcesdecisionengine	decisionengine.framework.engine.tests.test_query_tool_only
module, 75	module, 97
decisionengine.framework.dataspace.datasourcesdecisionengine	decisionengine.framework.engine.tests.test_SourceWorkers
module, 67	module, 96
decisionengine.framework.dataspace.datasourcesdecisionengine	decisionengine.framework.engine.tests.test_startup
module, 72	module, 97
decisionengine.framework.dataspace.datasourcesdecisionengine	decisionengine.framework.engine.tests.test_verify_redis_se
module, 75	module, 98
decisionengine.framework.dataspace.datasourcesdecisionengine	decisionengine.framework.logicengine
module, 82	module, 110
decisionengine.framework.dataspace.datasourcesdecisionengine	decisionengine.framework.logicengine.BooleanExpression
module, 79	module, 107
decisionengine.framework.dataspace.datasourcesdecisionengine	decisionengine.framework.logicengine.FactLookup
module, 80	module, 107
decisionengine.framework.dataspace.dataspace	decisionengine.framework.logicengine.LogicEngine
module, 93	module, 108
decisionengine.framework.dataspace.maintain	decisionengine.framework.logicengine.Rule
module, 94	module, 109
decisionengine.framework.dataspace.tests	decisionengine.framework.logicengine.RuleEngine
module, 88	module, 109
decisionengine.framework.dataspace.tests.fixtures	decisionengine.framework.logicengine.tests
module, 84	module, 107
decisionengine.framework.dataspace.tests.test_data_block	decisionengine.framework.logicengine.tests.test_bool_func
module, 86	module, 104
decisionengine.framework.dataspace.tests.test_data_block_line	decisionengine.framework.logicengine.tests.test_cascaded_r
module, 87	module, 105
decisionengine.framework.dataspace.tests.test_data_block	decisionengine.framework.logicengine.tests.test_constructi
module, 87	module, 105
decisionengine.framework.dataspace.tests.test_data_block	decisionengine.framework.logicengine.tests.test_duplicate
module, 87	module, 105
decisionengine.framework.dataspace.tests.test_data_block	decisionengine.framework.logicengine.tests.test_facts
module, 85	module, 105
decisionengine.framework.engine	decisionengine.framework.logicengine.tests.test_fail_on_er
module, 104	module, 106
decisionengine.framework.engine.ChannelWorkers	decisionengine.framework.logicengine.tests.test_pandas_fac
module, 98	module, 106
decisionengine.framework.engine.ClientMessageReceiver	decisionengine.framework.logicengine.tests.test_rule_with
module, 99	module, 106
decisionengine.framework.engine.de_client	decisionengine.framework.logicengine.tests.test_simple_cor
module, 103	module, 106
decisionengine.framework.engine.de_query_tool	decisionengine.framework.modules
module, 104	module, 115
decisionengine.framework.engine.DecisionEngine	decisionengine.framework.modules.de_logger
module, 99	module, 114
decisionengine.framework.engine.SourceWorkers	decisionengine.framework.modules.describe
module, 102	module, 114
decisionengine.framework.engine.tests	decisionengine.framework.modules.EmptySource
module, 98	module, 112
decisionengine.framework.engine.tests.fixtures	decisionengine.framework.modules.logging_configDict
module, 95	module, 115
decisionengine.framework.engine.tests.test_ChannelWorkers	decisionengine.framework.modules.Module

module, 112	module, 121
decisionengine.framework.modules.print_description	decisionengine.framework.tests.DynamicSource
module, 115	module, 122
decisionengine.framework.modules.Publisher	decisionengine.framework.tests.DynamicTransform
module, 112	module, 122
decisionengine.framework.modules.QueueLogger	decisionengine.framework.tests.ErringPublisher
module, 113	module, 122
decisionengine.framework.modules.Source	decisionengine.framework.tests.ErrorOnAcquire
module, 113	module, 122
decisionengine.framework.modules.tests	decisionengine.framework.tests.FailingPublisher
module, 112	module, 122
decisionengine.framework.modules.tests.test_decisionengine	decisionengine.framework.tests.fixtures
module, 111	module, 126
decisionengine.framework.modules.tests.test_EmptySource	decisionengine.framework.tests.IntSource
module, 110	module, 123
decisionengine.framework.modules.tests.test_Module	decisionengine.framework.tests.ModuleProgramOptions
module, 110	module, 123
decisionengine.framework.modules.tests.test_module_decisionengine	decisionengine.framework.tests.PublisherNOP
module, 111	module, 124
decisionengine.framework.modules.tests.test_Publisher	decisionengine.framework.tests.PublisherWithMissingConsumer
module, 110	module, 124
decisionengine.framework.modules.tests.test_QueueLogger	decisionengine.framework.tests.SourceAlias
module, 110	module, 124
decisionengine.framework.modules.tests.test_Source	decisionengine.framework.tests.SourceNOP
module, 111	module, 124
decisionengine.framework.modules.tests.test_Transform	decisionengine.framework.tests.SourceWithMissingProduces
module, 111	module, 124
decisionengine.framework.modules.tests.test_transform_decisionengine	decisionengine.framework.tests.SourceWithSampleConfigNOP
module, 111	module, 124
decisionengine.framework.modules.Transform	decisionengine.framework.tests.SupportsConfigPublisher
module, 113	module, 125
decisionengine.framework.modules.translate_producer_name	decisionengine.framework.tests.test_client_errors
module, 115	module, 126
decisionengine.framework.taskmanager	decisionengine.framework.tests.test_client_server
module, 121	module, 127
decisionengine.framework.taskmanager.LatestMessages	decisionengine.framework.tests.test_combined_channels
module, 115	module, 127
decisionengine.framework.taskmanager.module_graph	decisionengine.framework.tests.test_defaults
module, 120	module, 127
decisionengine.framework.taskmanager.ProcessingStatus	decisionengine.framework.tests.test_dynamic_test_modules
module, 116	module, 127
decisionengine.framework.taskmanager.PublisherStatus	decisionengine.framework.tests.test_empty_config
module, 117	module, 127
decisionengine.framework.taskmanager.SourceProducer	decisionengine.framework.tests.test_error_on_acquire
module, 119	module, 127
decisionengine.framework.taskmanager.TaskManager	decisionengine.framework.tests.test_module_program_options
module, 119	module, 128
decisionengine.framework.tests	decisionengine.framework.tests.test_publisher_status
module, 129	module, 128
decisionengine.framework.tests.ABTransform	decisionengine.framework.tests.test_publisher_status_board
module, 121	module, 128
decisionengine.framework.tests.BATransform	decisionengine.framework.tests.test_query_tool_server
module, 121	module, 128
decisionengine.framework.tests.DynamicPublisher	decisionengine.framework.tests.test_reaper

module, 128	delete_data_older_than()	(decisionengine.framework.dataspace.datasources.null.NullDataSource method), 82
decisionengine.framework.tests.test_same_source_types	delete_data_older_than()	(decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource method), 68
module, 128	delete_data_older_than()	(decisionengine.framework.dataspace.datasources.sqlalchemy_ds.SQLAlchemyDataSource method), 76
decisionengine.framework.tests.test_shared_sources	DescribeAlias	(class in decisionengine.framework.tests.ModuleProgramOptions), 123
module, 129	describe()	(in module decisionengine.framework.modules.Publisher), 112
decisionengine.framework.tests.test_start_with_bad_channels	describe()	(in module decisionengine.framework.modules.Source), 113
module, 129	describe()	(in module decisionengine.framework.modules.Transform), 113
decisionengine.framework.tests.test_status_duration	DEServer()	(in module decisionengine.framework.engine.tests.fixtures), 95
decisionengine.framework.tests.TransformNOP	DEServer()	(in module decisionengine.framework.tests.fixtures), 126
module, 125	detach()	(decisionengine.framework.engine.SourceWorkers.SourceWorker method), 103
decisionengine.framework.tests.TransformWithMissingProducersConsumers	display_metrics()	(in module decisionengine.framework.util.metrics), 132
module, 125	dump()	(decisionengine.framework.config.ValidConfig.ValidConfig method), 66
decisionengine.framework.tests.WriteToDisk	duplicate()	(decisionengine.framework.dataspace.datablock.DataBlock method), 88
module, 125	duplicate_datablock()	(decisionengine.framework.dataspace.datasource.DataSource method), 91
decisionengine.framework.util	duplicate_datablock()	(decisionengine.framework.dataspace.datasources.null.NullDataSource method), 82
module, 133	duplicate_datablock()	(decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource method), 68
decisionengine.framework.util.countdown	duplicate_datablock()	(decisionengine.framework.dataspace.datasources.sqlalchemy_ds.SQLAlchemyDataSource method), 76
module, 129	duration	(decisionengine.framework.taskmanager.PublisherStatus.PublisherStatus attribute), 117
decisionengine.framework.util.fs		
module, 130		
decisionengine.framework.util.logparser		
module, 130		
decisionengine.framework.util.metrics		
module, 131		
decisionengine.framework.util.reaper		
module, 132		
decisionengine.framework.util.redis_stats		
module, 133		
decisionengine.framework.util.singleton		
module, 133		
decisionengine.framework.util.sockets		
module, 133		
decisionengine.framework.util.subclasses		
module, 133		
decisionengine.framework.version		
module, 134		
decisionengine.tests		
module, 134		
decisionengine.tests.test_framework_package		
module, 134		
decompress() (in module decisionengine.framework.dataspace.datablock), 91		
default_data_lifetime (decisionengine.framework.dataspace.datablock.Header attribute), 90		
delete() (decisionengine.framework.dataspace.dataspace.DataSource method), 93		
delete_data_older_than() (decisionengine.framework.dataspace.datasource.DataSource method), 91		



DynamicPublisher (class in decisionengine.framework.tests.DynamicPublisher), 121

DynamicSource (class in decisionengine.framework.tests.DynamicSource), 122

DynamicTransform (class in decisionengine.framework.tests.DynamicTransform), 122

## E

EmptySource (class in decisionengine.framework.modules.EmptySource), 112

enabled (decisionengine.framework.taskmanager.PublisherStatus.PublisherStatus attribute), 117

ensure\_no\_circularities() (in module decisionengine.framework.taskmanager.module\_graph), 120

ErringPublisher (class in decisionengine.framework.tests.ErringPublisher), 122

ERROR (decisionengine.framework.taskmanager.ProcessingState.State attribute), 117

ErrorOnAcquire (class in decisionengine.framework.tests.ErrorOnAcquire), 122

evaluate() (decisionengine.framework.logicengine.BooleanExpression.BooleanExpression method), 107

evaluate() (decisionengine.framework.logicengine.LogicEngine.LogicEngine method), 108

evaluate() (decisionengine.framework.logicengine.Rule.Rule method), 109

evaluate\_facts() (decisionengine.framework.logicengine.LogicEngine.LogicEngine method), 108

execute() (decisionengine.framework.engine.ClientMessageReceiver.ClientMessageReceiver method), 99

execute() (decisionengine.framework.logicengine.RuleEngine.RuleEngine method), 109

execute\_command\_from\_args() (in module decisionengine.framework.util.logparser), 130

expiration\_time (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema attribute), 73

## F

FactLookup (class in decisionengine.framework.logicengine.FactLookup), 107

FailingPublisher (class in decisionengine.framework.tests.FailingPublisher), 122

files\_with\_extensions() (in module decisionengine.framework.util.fs), 130

format\_logger() (decisionengine.framework.modules.QueueLogger.QueueLogger method), 113

function\_name\_from\_call() (in module decisionengine.framework.logicengine.BooleanExpression), 107

## G

Gauge (class in decisionengine.framework.util.metrics), 131

generation\_id (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema attribute), 72

generation\_id (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema attribute), 73

generation\_id (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema attribute), 73

generation\_time (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema attribute), 73

get() (decisionengine.framework.dataspace.datablock.DataBlock method), 89

get() (decisionengine.framework.taskmanager.ProcessingState.ProcessingState method), 117

get\_channels() (decisionengine.framework.config.ChannelConfigHandler.ChannelConfigHandler method), 65

get\_consumes() (decisionengine.framework.engine.ChannelWorkers.ChannelWorker method), 98

get\_consumes() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 99

get\_data\_block() (decisionengine.framework.modules.Module.Module method), 112

get\_datablock() (decisionengine.framework.dataspace.datasource.DataSource method), 91

get\_data\_block() (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 82

get\_datablock() (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.datasource method), 68

get\_datablock() (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.SQLAlchemy method), 76

get\_datablock() (decisionengine.framework.dataspace.dataspace.DataSpace method), 107

<i>method</i> ), 93	<i>method</i> ), 92
get_dataproduct() <i>nengine.framework.dataspace.datasource.DataSource</i> <i>method</i> ), 91	(decision- get_last_generation_id() <i>nengine.framework.dataspace.datasources.null.NullDataSource</i> <i>method</i> ), 83
get_dataproduct() <i>nengine.framework.dataspace.datasources.null.NullDataSource</i> <i>method</i> ), 82	(decision- get_last_generation_id() <i>nengine.framework.dataspace.datasources.sqlalchemy_ds.datasource</i> <i>method</i> ), 69
get_dataproduct() <i>nengine.framework.dataspace.datasources.sqlalchemy_ds.datasource</i> <i>method</i> ), 69	(decision- get_last_generation_id() <i>nengine.framework.dataspace.datasources.sqlalchemy_ds.datasource</i> <i>method</i> ), 77
get_dataproduct() <i>nengine.framework.dataspace.datasources.sqlalchemy_ds.datasource</i> <i>method</i> ), 76	(decision- get_last_generation_id() <i>nengine.framework.dataspace.dataspace.DataSpace</i> <i>method</i> ), 94
get_dataproduct() <i>nengine.framework.dataspace.dataspace.DataSpace</i> <i>method</i> ), 93	(decision- get_logger() <i>nengine.framework.engine.DecisionEngine.DecisionEngine</i> <i>method</i> ), 100
get_dataproducts() <i>nengine.framework.dataspace.datablock.DataBlock</i> <i>method</i> ), 89	(decision- get_logger() (in module decision- <i>nengine.framework.modules.de_logger</i> ), 114
get_dataproducts() <i>nengine.framework.dataspace.datasource.DataSource</i> <i>method</i> ), 92	(decision- get_loglevel() <i>nengine.framework.engine.SourceWorkers.SourceWorker</i> <i>method</i> ), 102
get_dataproducts() <i>nengine.framework.dataspace.datasources.null.NullDataSource</i> <i>method</i> ), 83	(decision- get_loglevel() <i>nengine.framework.taskmanager.TaskManager.TaskManager</i> <i>method</i> ), 119
get_dataproducts() <i>nengine.framework.dataspace.datasources.sqlalchemy_ds.datasource</i> <i>method</i> ), 69	(decision- get_metadata() <i>nengine.framework.datablock.DataBlock</i> <i>method</i> ), 89
get_dataproducts() <i>nengine.framework.dataspace.datasources.sqlalchemy_ds.datasource</i> <i>method</i> ), 76	(decision- get_metadata() <i>nengine.framework.dataspace.datasource.DataSource</i> <i>method</i> ), 92
get_dataproducts() <i>nengine.framework.dataspace.dataspace.DataSpace</i> <i>method</i> ), 94	(decision- get_metadata() <i>nengine.framework.dataspace.datasources.null.NullDataSource</i> <i>method</i> ), 83
get_header() <i>nengine.framework.dataspace.datablock.DataBlock</i> <i>method</i> ), 89	(decision- get_metadata() <i>nengine.framework.dataspace.datasources.sqlalchemy_ds.datasource</i> <i>method</i> ), 70
get_header() <i>nengine.framework.dataspace.datasource.DataSource</i> <i>method</i> ), 92	(decision- get_metadata() <i>nengine.framework.dataspace.datasources.sqlalchemy_ds.datasource</i> <i>method</i> ), 77
get_header() <i>nengine.framework.dataspace.datasources.null.NullDataSource</i> <i>method</i> ), 83	(decision- get_metadata() <i>nengine.framework.dataspace.dataspace.DataSpace</i> <i>method</i> ), 94
get_header() <i>nengine.framework.dataspace.datasources.sqlalchemy_ds.datasource</i> <i>method</i> ), 69	(decision- get_parameters() <i>nengine.framework.module.Module</i> <i>method</i> ), 112
get_header() <i>nengine.framework.dataspace.datasources.sqlalchemy_ds.datasource</i> <i>method</i> ), 77	(decision- get_produces() <i>nengine.framework.engine.ChannelWorkers.ChannelWorker</i> <i>method</i> ), 98
get_header() <i>nengine.framework.dataspace.dataspace.DataSpace</i> <i>method</i> ), 94	(decision- get_produces() <i>nengine.framework.taskmanager.TaskManager.TaskManager</i> <i>method</i> ), 119
get_last_generation_id() <i>nengine.framework.datasource.DataSource</i>	(decision- get_queue_logger() (in module decision- <i>nengine.framework.modules.de_logger</i> ),





90  
 Metadata (class in decisio-  
     nengine.framework.dataspace.datasources.sqlalchemy\_75  
     73 ds.db\_schema),  
 metadata (decisionengine.framework.dataspace.datasources.sqlalchemy\_75  
     73 ds.db\_schema.Base  
     attribute), 72  
 metadata\_table (decisio-  
     75 nengine.framework.dataspace.datasource.DataSource  
     attribute), 93  
 metrics() (decisionengine.framework.engine.DecisionEngine.  
     method), 100  
 metrics\_env\_setup() (in module decisio-  
     nengine.framework.engine.tests.test\_startup),  
     97  
 MIN\_RETENTION\_INTERVAL\_DAYS (decisio-  
     nengine.framework.dataspace.maintain.Reaper  
     attribute), 94  
 MIN\_SECONDS\_BETWEEN\_RUNS (decisio-  
     nengine.framework.dataspace.maintain.Reaper  
     attribute), 94  
 missed\_update\_count (decisio-  
     nengine.framework.dataspace.datasources.sqlalchemy\_75  
     73 ds.db\_schema.Base  
     attribute), 74  
 mock\_data\_block() (in module decisio-  
     nengine.framework.dataspace.datasources.tests.fixtures),  
     80  
 module  
     decisionengine, 134  
     decisionengine.framework, 134  
     decisionengine.framework.about, 133  
     decisionengine.framework.config, 67  
     decisionengine.framework.config.ChannelConfigHandler, 65  
     decisionengine.framework.config.policies, 66  
     decisionengine.framework.config.tests, 65  
     decisionengine.framework.config.tests.test\_config, 63  
     decisionengine.framework.config.tests.test\_de\_cli, 64  
     decisionengine.framework.config.tests.test\_policy, 65  
     decisionengine.framework.config.ValidConfig, 66  
     decisionengine.framework.dataspace, 95  
     decisionengine.framework.dataspace.datablock, 88  
     decisionengine.framework.dataspace.datasource, 91  
     decisionengine.framework.dataspace.datasources, 84  
     decisionengine.framework.dataspace.datasources.sqlalchemy\_75  
     73 ds.db\_schema, 82  
     decisionengine.framework.dataspace.datasources.sqlalchemy\_75  
     73 ds.db\_schema.Base, 72  
     decisionengine.framework.dataspace.datasources.sqlalchemy\_75  
     73 ds.db\_schema.Base, 72  
     decisionengine.framework.dataspace.datasources.tests, 79  
     decisionengine.framework.dataspace.datasources.tests.f, 82  
     decisionengine.framework.dataspace.datasources.tests.f, 79  
     decisionengine.framework.dataspace.datasources.tests.t, 80  
     decisionengine.framework.dataspace.dataspace, 93  
     decisionengine.framework.dataspace.maintain, 94  
     decisionengine.framework.dataspace.tests, 88  
     decisionengine.framework.dataspace.tests.fixtures, 84  
     decisionengine.framework.dataspace.tests.test\_databloc, 86  
     decisionengine.framework.dataspace.tests.test\_databloc, 87  
     decisionengine.framework.dataspace.tests.test\_datasour, 87  
     decisionengine.framework.dataspace.tests.test\_dataspac, 87  
     decisionengine.framework.dataspace.tests.test\_Reaper, 85  
     decisionengine.framework.engine, 104  
     decisionengine.framework.engine.ChannelWorkers, 98  
     decisionengine.framework.engine.ClientMessageReceiver, 99  
     decisionengine.framework.engine.de\_client, 103  
     decisionengine.framework.engine.de\_query\_tool, 104  
     decisionengine.framework.engine.DecisionEngine, 99  
     decisionengine.framework.engine.SourceWorkers, 102  
     decisionengine.framework.engine.tests, 98  
     decisionengine.framework.engine.tests.fixtures, 95  
     decisionengine.framework.engine.tests.test\_ChannelWork, 96  
     decisionengine.framework.engine.tests.test\_client\_only, 97  
     decisionengine.framework.engine.tests.test\_query\_tool, 97  
     decisionengine.framework.engine.tests.test\_SourceWork, 97



96	113
decisionengine.framework.engine.tests.test_state,	decisionengine.framework.modules.tests,
97	112
decisionengine.framework.engine.tests.test_verification,	decisionengine.framework.modules.tests.test_de_logger,
98	111
decisionengine.framework.logicengine, 110	decisionengine.framework.modules.tests.test_EmptySource,
decisionengine.framework.logicengine.BooleanExpression,	decisionengine.framework.modules.tests.test_Module,
107	decisionengine.framework.modules.tests.test_module_dec,
decisionengine.framework.logicengine.FactLookup, 110	decisionengine.framework.modules.tests.test_Publisher,
107	110
decisionengine.framework.logicengine.LogicEngine, 111	decisionengine.framework.modules.tests.test_QueueLogger,
108	110
decisionengine.framework.logicengine.Rule,	decisionengine.framework.modules.tests.test_Source,
109	111
decisionengine.framework.logicengine.RuleEngine, 110	decisionengine.framework.modules.tests.test_Transform,
109	decisionengine.framework.modules.tests.test_translate,
decisionengine.framework.logicengine.tests, 111	decisionengine.framework.modules.Transform,
107	decisionengine.framework.modules.translate_product_name,
decisionengine.framework.logicengine.tests.test_bool_function_name,	decisionengine.framework.taskmanager, 121
104	decisionengine.framework.taskmanager.LatestMessages,
decisionengine.framework.logicengine.tests.test_cascaded_rules,	115
105	decisionengine.framework.taskmanager.module_graph,
decisionengine.framework.logicengine.tests.test_construction,	120
105	decisionengine.framework.taskmanager.ProcessingState,
decisionengine.framework.logicengine.tests.test_duplicate_fact_names,	116
105	decisionengine.framework.taskmanager.PublisherStatus,
decisionengine.framework.logicengine.tests.test_fast,	117
105	decisionengine.framework.taskmanager.SourceProductCache,
decisionengine.framework.logicengine.tests.test_failure,	119
106	decisionengine.framework.taskmanager.TaskManager,
decisionengine.framework.logicengine.tests.test_infinite,	119
106	decisionengine.framework.tests, 129
decisionengine.framework.logicengine.tests.test_invalid_keyfork,	decisionengine.framework.tests.ABTransform,
106	121
decisionengine.framework.logicengine.tests.test_invalid_keyfork,	decisionengine.framework.tests.BATransform,
106	121
decisionengine.framework.logicengine.tests.test_invalid_keyfork,	decisionengine.framework.tests.DynamicPublisher,
106	121
decisionengine.framework.logicengine.tests.test_invalid_keyfork,	decisionengine.framework.tests.DynamicSource,
106	122
decisionengine.framework.modules, 115	decisionengine.framework.tests.DynamicTransform,
decisionengine.framework.modules.de_logger,	122
114	decisionengine.framework.tests.ErpingPublisher,
decisionengine.framework.modules.describe,	122
114	decisionengine.framework.tests.ErrorOnAcquire,
decisionengine.framework.modules.EmptySource, 112	122
112	decisionengine.framework.tests.FailingPublisher,
decisionengine.framework.modules.logging_configuration,	
115	
decisionengine.framework.modules.Module,	
112	
decisionengine.framework.modules.print_descriptions,	
115	
decisionengine.framework.modules.Publisher,	
112	
decisionengine.framework.modules.QueueLogger,	
113	
decisionengine.framework.modules.Source,	

122  
decisionengine.framework.tests.fixtures,  
126  
decisionengine.framework.tests.IntSource,  
123  
decisionengine.framework.tests.ModuleProgramOptions,  
123  
decisionengine.framework.tests.PublisherNOP,  
124  
decisionengine.framework.tests.PublisherWithMissingConsumes,  
124  
decisionengine.framework.tests.SourceAlias,  
124  
decisionengine.framework.tests.SourceNOP,  
124  
decisionengine.framework.tests.SourceWithMissingProducers,  
124  
decisionengine.framework.tests.SourceWithSampleConfigNOP,  
124  
decisionengine.framework.tests.SupportsConfigPublisher,  
125  
decisionengine.framework.tests.test\_client\_errors,  
126  
decisionengine.framework.tests.test\_client\_server,  
127  
decisionengine.framework.tests.test\_combined\_channels,  
127  
decisionengine.framework.tests.test\_defaults,  
127  
decisionengine.framework.tests.test\_dynamic\_test\_modules,  
127  
decisionengine.framework.tests.test\_empty\_config,  
127  
decisionengine.framework.tests.test\_error\_on\_acquire,  
127  
decisionengine.framework.tests.test\_module\_program\_options,  
128  
decisionengine.framework.tests.test\_publisher\_status,  
128  
decisionengine.framework.tests.test\_publisher\_status\_board,  
128  
decisionengine.framework.tests.test\_query\_resolver,  
128  
decisionengine.framework.tests.test\_reaper,  
128  
decisionengine.framework.tests.test\_same\_source\_types,  
128  
decisionengine.framework.tests.test\_sample\_config,  
128  
decisionengine.framework.tests.test\_shared\_sources,  
129  
decisionengine.framework.tests.test\_start\_with\_bad\_channels,  
129  
decisionengine.framework.tests.test\_status\_during\_startup,  
129  
decisionengine.framework.tests.TransformNOP,  
125  
decisionengine.framework.tests.TransformWithMissingProducers,  
125  
decisionengine.framework.tests.WriteToDisk,  
125  
decisionengine.framework.util, 133  
decisionengine.framework.util.countdown,  
decisionengine.framework.util.fs, 130  
decisionengine.framework.util.logparser,  
130  
decisionengine.framework.util.metrics,  
131  
decisionengine.framework.util.reaper, 132  
decisionengine.framework.util.redis\_stats,  
decisionengine.framework.util.configNOP,  
decisionengine.framework.util.singleton,  
decisionengine.framework.util.sockets,  
decisionengine.framework.util.subclasses,  
decisionengine.framework.version, 134  
decisionengine.tests, 134  
decisionengine.tests.test\_framework\_package,  
134  
Module (class in decisio-  
decisionengine.framework.modules.Module), 112  
ModuleProgramOptions (class in decisio-  
decisionengine.framework.modules.describe), 114  
mydata() (in module decisio-  
decisionengine.framework.logicengine.tests.test\_pandas\_fact),  
106  
myengine(options,(in module decisio-  
decisionengine.framework.logicengine.tests.test\_cascaded\_rules),  
105  
myengine() (in module decisio-  
decisionengine.framework.logicengine.tests.test\_pandas\_fact),  
106  
myengine() (in module decisio-  
decisionengine.framework.logicengine.tests.test\_rule\_with\_negated\_fact),  
106  
myengine() (in module decisio-  
decisionengine.framework.logicengine.tests.test\_simple\_configuration),  
106  
N  
decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_  
attribute), 74  
decisionengine.framework.engine.tests.test\_ChannelWorkers.TaskMa  
attribute), 96

NotFound (*decisionengine.framework.engine.DecisionEngine* attribute), 101  
 NullDataSource (class in *decisionengine.framework.dataspace.datasources.null*), 82  
**O**  
 OFFLINE (*decisionengine.framework.taskmanager.ProcessingState* attribute), 117  
 orm\_as\_dict() (in module *decisionengine.framework.dataspace.datasources.sqlalchemy\_ds\_util.py*), 75  
**P**  
 Parameter (class in *decisionengine.framework.modules.describe*), 114  
 Parameter (class in *decisionengine.framework.modules.Publisher*), 112  
 Parameter (class in *decisionengine.framework.modules.Source*), 113  
 Parameter (class in *decisionengine.framework.modules.Transform*), 113  
 parse\_constraints() (in module *decisionengine.framework.util.logparser*), 131  
 parse\_program\_options() (in module *decisionengine.framework.engine.DecisionEngine*), 102  
 passthrough\_configuration() (in module *decisionengine.framework.logicengine.LogicEngine*), 109  
 PG\_DE\_DB\_WITHOUT\_SCHEMA() (in module *decisionengine.framework.dataspace.datasources.tests.fixtures*), 79  
 PG\_DE\_DB\_WITHOUT\_SCHEMA() (in module *decisionengine.framework.dataspace.tests.fixtures*), 84  
 PG\_DE\_DB\_WITHOUT\_SCHEMA() (in module *decisionengine.framework.engine.tests.fixtures*), 95  
 PG\_DE\_DB\_WITHOUT\_SCHEMA() (in module *decisionengine.framework.tests.fixtures*), 126  
 PG\_PROG() (in module *decisionengine.framework.dataspace.datasources.tests.fixtures*), 80  
 PG\_PROG() (in module *decisionengine.framework.dataspace.tests.fixtures*), 84  
 PG\_PROG() (in module *decisionengine.framework.engine.tests.fixtures*), 95  
 PG\_PROG() (in module *decisionengine.framework.tests.fixtures*), 126  
 print\_channel\_config() (*decisionengine.framework.config.ChannelConfigHandler.ChannelConfigHandler* method), 65  
 print\_consumes() (in module *decisionengine.framework.modules.print\_description*), 115  
 print\_produces() (in module *decisionengine.framework.modules.print\_description*), 115  
 print\_supported\_config() (in module *decisionengine.framework.modules.print\_description*), 115  
 probably\_running() (*decisionengine.framework.taskmanager.ProcessingState.ProcessingState* method), 116  
 process\_args() (*decisionengine.framework.modules.describe.ModuleProgramOptions* method), 114  
 ProcessingState (class in *decisionengine.framework.taskmanager.ProcessingState*), 116  
 produces() (*decisionengine.framework.logicengine.LogicEngine.LogicEngine* method), 109  
 produces() (in module *decisionengine.framework.modules.Module*), 112  
 produces() (in module *decisionengine.framework.modules.Source*), 113  
 produces() (in module *decisionengine.framework.modules.Transform*), 113  
 ProductRetriever (class in *decisionengine.framework.dataspace.datablock*), 90  
 prune() (*decisionengine.framework.engine.SourceWorkers.SourceWorkers* method), 103  
 publish() (*decisionengine.framework.modules.Publisher.Publisher* method), 112  
 publish() (*decisionengine.framework.tests.ErringPublisher.ErringPublisher* method), 122  
 publish() (*decisionengine.framework.tests.FailingPublisher.FailingPublisher* method), 122  
 publish() (*decisionengine.framework.tests.PublisherNOP.PublisherNOP* method), 124  
 publish() (*decisionengine.framework.tests.WriteToDisk.WriteToDisk* method), 125  
 Publisher (class in *decisionengine.framework.modules.Publisher*), 112  
 publisher() (*decisionengine.framework.tests.DynamicPublisher.DynamicPublisher* method), 121  
 PublisherNOP (class in *decisionengine.framework.tests.PublisherNOP*), 124



**PublisherState** (class in decisionengine.framework.taskmanager.PublisherStatus), 117  
**PublisherStatus** (class in decisionengine.framework.taskmanager.PublisherStatus), 118  
**PublisherStatusBoard** (class in decisionengine.framework.taskmanager.PublisherStatus), 118  
**PublisherWithMissingConsumes** (class in decisionengine.framework.tests.PublisherWithMissingConsumes), 124  
**put()** (decisionengine.framework.dataspace.datablock.DataBlock method), 90  
**Q**  
**queue\_logger\_setup()** (in module decisionengine.framework.modules.tests.test\_QueueLogger), 110  
**QueueLogger** (class in decisionengine.framework.modules.QueueLogger), 113  
**R**  
**reap()** (decisionengine.framework.dataspace.maintain.Reaper method), 94  
**Reaper** (class in decisionengine.framework.dataspace.maintain), 94  
**reaper()** (in module decisionengine.framework.dataspace.tests.test\_Reaper), 85  
**reaper\_start()** (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 100  
**reaper\_status()** (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 100  
**reaper\_stop()** (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 100  
**record\_that\_matches()** (in module decisionengine.framework.tests.test\_shared\_sources), 129  
**redis\_stats()** (in module decisionengine.framework.util.redis\_stats), 133  
**registry** (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema attribute), 72  
**remove\_all()** (decisionengine.framework.engine.SourceWorkers.SourceWorkers method), 103  
**RequestHandler** (class in decisionengine.framework.engine.DecisionEngine), 101  
**required\_keys** (decisionengine.framework.dataspace.datablock.Header attribute), 90  
**required\_keys** (decisionengine.framework.dataspace.datablock.Metadata attribute), 90  
**reset\_connections()** (decisionengine.framework.dataspace.datasource.DataSource method), 93  
**reset\_connections()** (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 84  
**reset\_connections()** (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.datasource method), 71  
**reset\_connections()** (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.SQLAlchemyDataSource method), 78  
**retention\_interval** (decisionengine.framework.dataspace.maintain.Reaper property), 95  
**rm\_channel()** (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 100  
**rpc\_block\_while()** (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 100  
**rpc\_get\_channel\_log\_level()** (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 100  
**rpc\_get\_log\_level()** (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 100  
**rpc\_get\_source\_log\_level()** (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 100  
**rpc\_kill\_channel()** (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 100  
**rpc\_metrics()** (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 100  
**rpc\_paths** (decisionengine.framework.engine.DecisionEngine.RequestHandler attribute), 101  
**rpc\_ping()** (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 100  
**rpc\_print\_product()** (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 100  
**rpc\_print\_products()** (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 100  
**rpc\_product\_dependencies()** (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 100

method), 100  
 rpc\_query\_tool() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 98  
 method), 100  
 rpc\_queue\_status() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 102  
 method), 100  
 rpc\_reaper\_start() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 119  
 method), 100  
 rpc\_reaper\_status() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 119  
 method), 100  
 rpc\_reaper\_stop() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 119  
 method), 100  
 rpc\_rm\_channel() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 119  
 method), 100  
 rpc\_set\_channel\_log\_level() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 101  
 method), 101  
 rpc\_set\_source\_log\_level() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 101  
 method), 101  
 rpc\_show\_config() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 101  
 method), 101  
 rpc\_show\_de\_config() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 101  
 method), 101  
 rpc\_start\_channel() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 101  
 method), 101  
 rpc\_start\_channels() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 101  
 method), 101  
 rpc\_status() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 101  
 method), 101  
 rpc\_stop() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 101  
 method), 101  
 rpc\_stop\_channel() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 101  
 method), 101  
 rpc\_stop\_channels() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 101  
 method), 101  
 Rule (class in decisionengine.framework.logicengine.Rule), 109  
 rule\_for() (decisionengine.framework.logicengine.FactLookup.FactLookup method), 107  
 RuleEngine (class in decisionengine.framework.logicengine.RuleEngine), 109  
 run() (decisionengine.framework.engine.ChannelWorkers.ChannelWorker method), 98  
 run() (decisionengine.framework.engine.SourceWorkers.SourceWorker method), 102  
 run\_engine() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 119  
 run\_engines() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 119  
 run\_logic\_engine() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 119  
 run\_publishers() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 119  
 run\_transform() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 120  
 run\_transforms() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 120  
 run\_transforms() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 120  
 S  
 scheduled\_create\_time (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema attribute), 73  
 Schema (class in decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema), 74  
 schema (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema attribute), 74  
 schema\_id (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema attribute), 73  
 schema\_id (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema attribute), 74  
 ScopedSingleton (class in decisionengine.framework.util.singleton), 133  
 ScopedSingletonABC (class in decisionengine.framework.util.singleton), 133  
 seconds\_between\_runs (decisionengine.framework.dataspace.maintain.Reaper property), 95  
 sequence\_id (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema attribute), 74  
 service\_actions() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 101  
 set() (decisionengine.framework.taskmanager.ProcessingState.ProcessingState method), 116  
 set\_data\_block() (decisionengine.framework.modules.Module.Module method), 116

method), 112  
 set\_loglevel\_value() (decisionengine.framework.engine.SourceWorkers.SourceWorker method), 102  
 set\_loglevel\_value() (decisionengine.framework.tests.test\_ChannelWorkers.SourceProductCache method), 96  
 set\_loglevel\_value() (decisionengine.framework.taskmanager.TaskManager.TaskManager.SourceWithMissingProduces method), 120  
 set\_state() (decisionengine.framework.dataspace.datablock.MetadataSourceWithSampleConfigNOP method), 90  
 setup\_logger() (decisionengine.framework.engine.ChannelWorkers.ChannelWorker method), 98  
 setup\_logger() (decisionengine.framework.engine.SourceWorkers.SourceWorkers method), 102  
 setup\_queue\_logging() (decisionengine.framework.modules.QueueLogger.QueueLogger method), 113  
 setup\_queue\_logging() (in module decisionengine.framework.modules.tests.test\_QueueLogger), 110  
 should\_stop() (decisionengine.framework.taskmanager.ProcessingState.ProcessingState.SQLALCHEMY\_PG\_WITH\_SCHEMA method), 116  
 SHUTDOWN (decisionengine.framework.taskmanager.ProcessingState.ProcessingState attribute), 117  
 shutdown() (decisionengine.framework.modules.Publisher.Publisher method), 112  
 SHUTTINGDOWN (decisionengine.framework.taskmanager.ProcessingState.ProcessingState attribute), 117  
 since (decisionengine.framework.taskmanager.PublisherStatus.PublisherStatus attribute), 117  
 Singleton (class in decisionengine.framework.util.singleton), 133  
 SingletonABC (class in decisionengine.framework.util.singleton), 133  
 snapshot() (decisionengine.framework.taskmanager.PublisherStatus.PublisherStatus method), 118  
 sorted\_rules() (decisionengine.framework.logicengine.FactLookup.FactLookup method), 108  
 Source (class in decisionengine.framework.modules.Source), 113  
 source\_config() (in module decisionengine.framework.engine.tests.test\_SourceWorkers), 96  
 source\_products() (in module decisionengine.framework.taskmanager.module\_graph), 121  
 source\_worker\_for() (in module decisionengine.framework.engine.tests.test\_SourceWorkers), 96  
 SourceNOP (class in decisionengine.framework.tests.SourceNOP), 124  
 SourceProductCache (class in decisionengine.framework.taskmanager.SourceProductCache), 119  
 SourceWithMissingProduces (class in decisionengine.framework.tests.SourceWithMissingProduces), 124  
 SourceWithSampleConfigNOP (class in decisionengine.framework.tests.SourceWithSampleConfigNOP), 124  
 SourceWorker (class in decisionengine.framework.engine.SourceWorkers), 102  
 SourceWorkers (class in decisionengine.framework.engine.SourceWorkers), 102  
 SourceWorkers.Access (class in decisionengine.framework.engine.SourceWorkers), 103  
 spec\_if\_main() (in module decisionengine.framework.modules.print\_description), 115  
 SQLALCHEMY\_PG\_WITH\_SCHEMA() (in module decisionengine.framework.dataspace.datasources.tests.fixtures), 80  
 SQLALCHEMY\_PG\_WITH\_SCHEMA() (in module decisionengine.framework.dataspace.tests.fixtures), 85  
 SQLALCHEMY\_PG\_WITH\_SCHEMA() (in module decisionengine.framework.engine.tests.fixtures), 96  
 SQLALCHEMY\_PG\_WITH\_SCHEMA() (in module decisionengine.framework.tests.fixtures), 126  
 SQLALCHEMY\_TEMPFILE\_SQLITE() (in module decisionengine.framework.dataspace.datasources.tests.fixtures), 80  
 SQLALCHEMY\_TEMPFILE\_SQLITE() (in module decisionengine.framework.dataspace.tests.fixtures), 85  
 SQLALCHEMY\_TEMPFILE\_SQLITE() (in module decisionengine.framework.engine.tests.fixtures), 96  
 SQLALCHEMY\_TEMPFILE\_SQLITE() (in module decisionengine.framework.tests.fixtures), 126  
 SQLAlchemyDS (class in decisionengine.framework.dataspace.datasources.sqlalchemy\_ds), 75  
 SQLAlchemyDS (class in decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.datasource), 67  
 start() (decisionengine.framework.dataspace.maintain.Reaper method), 95  
 start() (decisionengine.framework.modules.QueueLogger.QueueLogger method), 113

start\_channel() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 94  
 start\_channels() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 101  
 start\_webserver() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 101  
 State (class in decisionengine.framework.taskmanager.ProcessingState), 117  
 state (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema.Metadata attribute), 74  
 state() (decisionengine.framework.taskmanager.PublisherStatus.PublisherStatus method), 118  
 STEADY (decisionengine.framework.taskmanager.ProcessingState.State attribute), 117  
 stop() (decisionengine.framework.dataspace.maintain.Reaper.Reaper method), 95  
 stop() (decisionengine.framework.modules.QueueLogger.QueueLogger method), 113  
 stop\_channels() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 101  
 stop\_queue\_logger() (in module decisionengine.framework.modules.de\_logger), 114  
 stop\_worker() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 101  
 stopped\_channel\_opts() (in module decisionengine.framework.tests.test\_sample\_config), 128  
 StopState (class in decisionengine.framework.engine.DecisionEngine), 101  
 store\_taskmanager() (decisionengine.framework.dataspace.datablock.DataBlock method), 90  
 store\_taskmanager() (decisionengine.framework.dataspace.datasource.DataSource method), 93  
 store\_taskmanager() (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 84  
 store\_taskmanager() (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.datasource\_api.SQLiteDS method), 71  
 store\_taskmanager() (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.datasource\_api.SQLiteDS method), 79  
 store\_taskmanager() (decisionengine.framework.dataspace.dataspace.DataSpace method), 72  
 supports\_config() (in module decisionengine.framework.modules.describe), 115  
 supports\_config() (in module decisionengine.framework.modules.Publisher), 112  
 supports\_config() (in module decisionengine.framework.modules.Source), 113  
 supports\_config() (in module decisionengine.framework.modules.Transform), 117  
 SupportsConfig (class in decisionengine.framework.tests.SupportsConfigPublisher), 125  
 take\_offline() (decisionengine.framework.engine.SourceWorkers.SourceWorker method), 102  
 task\_dataproduct (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema attribute), 74  
 task\_header (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema attribute), 74  
 task\_metadata (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema attribute), 74  
 Taskmanager (class in decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema), 74  
 TaskManager (class in decisionengine.framework.engine.tests.test\_ChannelWorkers), 96  
 TaskManager (class in decisionengine.framework.taskmanager.TaskManager), 119  
 taskmanager (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema attribute), 72  
 taskmanager (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema attribute), 71  
 taskmanager (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema attribute), 79  
 taskmanager\_id (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema attribute), 72



taskmanager_id	(decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Metadata attribute), 73	test_channel_empty_config() (in module decisionengine.framework.config.tests.test_config), 63
taskmanager_id	(decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Metadata attribute), 74	test_channel_empty_dictionary() (in module decisionengine.framework.config.tests.test_config), 63
taskmanager_id	(decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Metadata attribute), 74	test_channel_invalid_modules_list() (in module decisionengine.framework.config.tests.test_config), 63
taskmanager_table	(decisionengine.framework.dataspace.datasource.DataSource attribute), 93	test_channel_invalid_modules_no_keys() (in module decisionengine.framework.config.tests.test_config), 63
Terminated	(decisionengine.framework.engine.DecisionEngine.StopState attribute), 101	test_channel_invalid_modules_string() (in module decisionengine.framework.config.tests.test_config), 63
test()	(decisionengine.framework.tests.ModuleProgramOptions.AcquireWithConfig module method), 123	test_channel_loading() (in module decisionengine.framework.config.tests.test_config), 64
test()	(decisionengine.framework.tests.ModuleProgramOptions.AcquireWithSampleConfig module method), 123	test_channel_module_missing_all() (in module decisionengine.framework.config.tests.test_config), 64
test()	(decisionengine.framework.tests.ModuleProgramOptions.ConfigureTempFile module method), 123	test_channel_module_missing_module() (in module decisionengine.framework.config.tests.test_config), 64
test()	(decisionengine.framework.tests.ModuleProgramOptions.DescribeAll module method), 123	test_channel_names() (in module decisionengine.framework.config.tests.test_config), 64
test_acquire_for_sources()	(in module decisionengine.framework.tests.test_module_program_options), 128	test_channel_no_config_files() (in module decisionengine.framework.config.tests.test_config), 64
test_allow_duplicate_keys_same_values()	(in module decisionengine.framework.config.tests.test_de_std), 64	test_channel_no_modules() (in module decisionengine.framework.config.tests.test_config), 64
test_allow_duplicate_source_proxy_keys()	(in module decisionengine.framework.config.tests.test_de_std), 64	test_check_metrics_env_no_webserver() (in module decisionengine.framework.engine.tests.test_startup), 97
test_by_nonsense_is_err()	(in module decisionengine.framework.modules.tests.test_de_logger), 111	test_check_metrics_env_var_set() (in module decisionengine.framework.engine.tests.test_startup), 97
test_by_size()	(in module decisionengine.framework.modules.tests.test_de_logger), 111	test_check_metrics_env_var_unset() (in module decisionengine.framework.engine.tests.test_startup), 97
test_by_time()	(in module decisionengine.framework.modules.tests.test_de_logger), 111	test_client_can_get_de_server_reaper_status()
test_can_import()	(in module decisionengine.tests.test_framework_package), 134	
test_change_port()	(in module decisionengine.framework.engine.tests.test_startup), 97	
test_channel_config_dir()	(in module decisionengine.framework.config.tests.test_policies), 65	

(in module decisionengine.framework.tests.test_reaper), 128	129
test_client_can_get_de_server_status() (in module decisionengine.framework.tests.test_sample_config), 128	test_client_non_real_channel() (in module decisionengine.framework.tests.test_sample_config), 129
test_client_can_get_products_no_channels() (in module decisionengine.framework.tests.test_start_with_bad_channel), 129	test_client_print_product() (in module decisionengine.framework.tests.test_client_server), 127
test_client_can_kill_one_channel() (in module decisionengine.framework.tests.test_sample_config), 128	test_client_set_loglevel() (in module decisionengine.framework.tests.test_client_server), 127
test_client_can_restart_all_channels() (in module decisionengine.framework.tests.test_sample_config), 128	test_client_status_msg_to_logger() (in module decisionengine.framework.tests.test_client_server), 127
test_client_can_restart_one_channel() (in module decisionengine.framework.tests.test_sample_config), 128	test_client_with_no_command_says_use_help() (in module decisionengine.framework.engine.tests.test_client_only), 97
test_client_can_start_one_channel_added_after_starting() (in module decisionengine.framework.tests.test_empty_config), 127	test_client_with_no_server() (in module decisionengine.framework.engine.tests.test_client_only), 97
test_client_cannot_wait_on_bad_state() (in module decisionengine.framework.tests.test_client_errors), 126	test_client_with_no_server_verbose() (in module decisionengine.framework.engine.tests.test_client_only), 97
test_client_err_returned_as_rc() (in module decisionengine.framework.engine.tests.test_client_only), 97	test_combine_one_level() (in module decisionengine.framework.config.tests.test_de_std), 64
test_client_err_returned_as_rc() (in module decisionengine.framework.engine.tests.test_query_tool_only), 97	test_combine_one_level_skip_proxies() (in module decisionengine.framework.config.tests.test_de_std), 64
test_client_err_returned_verbose_as_rc() (in module decisionengine.framework.engine.tests.test_client_only), 97	test_combined_channels() (in module decisionengine.framework.tests.test_combined_channels), 127
test_client_err_returned_verbose_as_rc() (in module decisionengine.framework.engine.tests.test_query_tool_only), 97	test_combined_channels_3g() (in module decisionengine.framework.tests.test_combined_channels), 127
test_client_get_loglevel() (in module decisionengine.framework.tests.test_client_server), 127	test_compound_fact_with_spaces() (in module decisionengine.framework.logicengine.tests.test_facts), 105
test_client_help() (in module decisionengine.framework.engine.tests.test_client_only), 97	test_compress() (in module decisionengine.framework.dataspace.tests.test_datablock_zlib), 87
test_client_logger_level() (in module decisionengine.framework.tests.test_sample_config),	test_conditional_fact() (in module decisionengine.framework.logicengine.tests.test_fail_on_error), 106
	test_config_templates() (in module decisionengine.framework.tests.test_module_program_options), 128
	test_configuration_with_fact_using_function() (in module decisionengine.framework.engine.tests.test_client_only), 97

<i>nengine.framework.logicengine.tests.test_construction</i> ),	86
105	<i>test_DataBlock_to_str()</i> (in module <i>decisionengine.framework.dataspace.tests.test_datablock</i> ),
<i>test_configuration_with_nmy_facts()</i>	<i>nengine.framework.dataspace.tests.test_datablock</i> ),
(in module <i>decisionengine.framework.logicengine.tests.test_construction</i> ),	86
<i>test_conflicting_source_configurations()</i>	<i>test_dataspace_config_finds_bad()</i>
(in module <i>decisionengine.framework.logicengine.tests.test_construction</i> ),	(in module <i>decisionengine.framework.dataspace.tests.test_dataspace</i> ),
105	87
<i>test_conflicting_source_configurations()</i>	<i>test_default_config()</i> (in module <i>decisionengine.framework.engine.tests.test_startup</i> ),
(in module <i>decisionengine.framework.tests.test_shared_sources</i> ),	97
129	<i>test_default_config()</i> (in module <i>decisionengine.framework.logicengine.tests.test_construction</i> ),
<i>test_create_tables()</i> (in module <i>decisionengine.framework.dataspace.datasources.tests.test_shared_sources</i> ),	105
80	<i>test_defaults()</i> (in module <i>decisionengine.framework.tests.test_defaults</i> ),
<i>test_DataBlock_constructor()</i> (in module <i>decisionengine.framework.dataspace.tests.test_datablock</i> ),	127
86	<i>test_delete()</i> (in module <i>decisionengine.framework.dataspace.tests.test_dataspace</i> ),
<i>test_DataBlock_duplicate()</i> (in module <i>decisionengine.framework.dataspace.tests.test_datablock</i> ),	87
86	<i>test_delete_data_older_than_arg()</i>
<i>test_DataBlock_get_dataproducts()</i>	(in module <i>decisionengine.framework.dataspace.datasources.tests.test_datasource</i> ),
(in module <i>decisionengine.framework.dataspace.tests.test_datablock</i> ),	80
86	<i>test_descriptions()</i> (in module <i>decisionengine.framework.tests.test_module_program_options</i> ),
<i>test_DataBlock_get_header()</i> (in module <i>decisionengine.framework.dataspace.tests.test_datablock</i> ),	128
86	<i>test_duplicate_datablock()</i> (in module <i>decisionengine.framework.dataspace.datasources.tests.test_datasource</i> ),
<i>test_DataBlock_get_metadata()</i> (in module <i>decisionengine.framework.dataspace.tests.test_datablock</i> ),	80
86	<i>test_duplicate_datablock()</i> (in module <i>decisionengine.framework.dataspace.tests.test_dataspace</i> ),
<i>test_DataBlock_get_taskmanager()</i>	87
(in module <i>decisionengine.framework.dataspace.tests.test_datablock</i> ),	<i>test_duplicate_fact_names()</i> (in module <i>decisionengine.framework.logicengine.tests.test_duplicate_fact_names</i> ),
86	105
<i>test_DataBlock_is_expired()</i> (in module <i>decisionengine.framework.dataspace.tests.test_datablock</i> ),	<i>test_dynamic_publisher()</i> (in module <i>decisionengine.framework.tests.test_dynamic_test_modules</i> ),
86	127
<i>test_DataBlock_is_expired_with_key()</i>	<i>test_dynamic_source()</i> (in module <i>decisionengine.framework.tests.test_dynamic_test_modules</i> ),
(in module <i>decisionengine.framework.dataspace.tests.test_datablock</i> ),	127
86	<i>test_dynamic_transform()</i> (in module <i>decisionengine.framework.tests.test_dynamic_test_modules</i> ),
<i>test_DataBlock_key_management()</i>	127
(in module <i>decisionengine.framework.dataspace.tests.test_datablock</i> ),	<i>test_empty_config()</i> (in module <i>decisionengine.framework.config.tests.test_config</i> ),
86	64
<i>test_DataBlock_key_management_change_name()</i>	<i>test_empty_source_structure()</i> (in module <i>decisionengine.framework.modules.tests.test_EmptySource</i> ),
(in module <i>decisionengine.framework.dataspace.tests.test_datablock</i> ),	110
86	<i>test_error_conditions()</i> (in module <i>decisionengine.framework.logicengine.tests.test_bool_function_name</i> ),
<i>test_DataBlock_no_key_by_name()</i>	104
(in module <i>decisionengine.framework.dataspace.tests.test_datablock</i> ),	<i>test_error_on_acquire()</i> (in module <i>decisionengine.framework.logicengine.tests.test_bool_function_name</i> ),
<i>nengine.framework.dataspace.tests.test_datablock</i> ),	

<code>nengine.framework.tests.test_error_on_acquire),</code>	85
127	<code>test_false_fact_with_spaces() (in module decisio-</code>
<code>test_error_on_bad_names() (in module decisio-</code>	<code>nengine.framework.logicengine.tests.test_fail_on_error),</code>
<code>nengine.framework.logicengine.tests.test_simple_configuration),</code>	106
106	<code>test_false_literal_fact() (in module decisio-</code>
<code>test_error_on_duplicate_keys() (in module decisio-</code>	<code>nengine.framework.logicengine.tests.test_fail_on_error),</code>
<code>engine.framework.config.tests.test_de_std),</code>	106
64	<code>test_get_datablock() (in module decisio-</code>
<code>test_exclusive_options() (in module decisio-</code>	<code>nengine.framework.dataspace.datasources.tests.test_datasource_</code>
<code>nengine.framework.engine.tests.test_client_only),</code>	81
97	<code>test_get_datablock() (in module decisio-</code>
<code>test_fact_using_numpy_array() (in module decisio-</code>	<code>nengine.framework.dataspace.tests.test_dataspace),</code>
<code>nengine.framework.logicengine.tests.test_facts),</code>	87
105	<code>test_get_dataproduct() (in module decisio-</code>
<code>test_fact_using_numpy_function()</code>	<code>nengine.framework.dataspace.datasources.tests.test_datasource_</code>
<code>(in module decisio-</code>	81
<code>nengine.framework.logicengine.tests.test_facts),</code>	<code>test_get_dataproduct() (in module decisio-</code>
105	<code>nengine.framework.dataspace.tests.test_dataspace),</code>
<code>test_fact_with_fail_on_error()</code>	87
<code>(in module decisio-</code>	<code>test_get_dataproduct_not_exist()</code>
<code>nengine.framework.logicengine.tests.test_facts),</code>	<code>(in module decisio-</code>
105	<code>nengine.framework.dataspace.datasources.tests.test_datasource_</code>
<code>test_fact_with_misspecified_attribute()</code>	81
<code>(in module decisio-</code>	<code>test_get_dataproduct_not_exist()</code>
<code>nengine.framework.logicengine.tests.test_fail_on_error),</code>	<code>(in module decisio-</code>
106	<code>nengine.framework.dataspace.tests.test_dataspace),</code>
<code>test_fact_with_nested_names() (in module decisio-</code>	87
<code>nengine.framework.logicengine.tests.test_facts),</code>	<code>test_get_dataproducts() (in module decisio-</code>
105	<code>nengine.framework.dataspace.datasources.tests.test_datasource_</code>
<code>test_fail_bad_config() (in module decisio-</code>	81
<code>nengine.framework.dataspace.tests.test_Reaper),</code>	<code>test_get_dataproducts() (in module decisio-</code>
85	<code>nengine.framework.dataspace.tests.test_dataspace),</code>
<code>test_fail_missing_config() (in module decisio-</code>	87
<code>nengine.framework.dataspace.tests.test_Reaper),</code>	<code>test_get_dataproducts_not_exist()</code>
85	<code>(in module decisio-</code>
<code>test_fail_missing_config_key()</code>	<code>nengine.framework.dataspace.datasources.tests.test_datasource_</code>
<code>(in module decisio-</code>	81
<code>nengine.framework.dataspace.tests.test_Reaper),</code>	<code>test_get_dataproducts_not_exist()</code>
85	<code>(in module decisio-</code>
<code>test_fail_on_error() (in module decisio-</code>	<code>nengine.framework.dataspace.tests.test_dataspace),</code>
<code>nengine.framework.logicengine.tests.test_fail_on_error),</code>	87
106	<code>test_get_header() (in module decisio-</code>
<code>test_fail_small_retain() (in module decisio-</code>	<code>nengine.framework.dataspace.datasources.tests.test_datasource_</code>
<code>nengine.framework.dataspace.tests.test_Reaper),</code>	81
85	<code>test_get_header() (in module decisio-</code>
<code>test_fail_small_run_interval()</code>	<code>nengine.framework.dataspace.tests.test_dataspace),</code>
<code>(in module decisio-</code>	87
<code>nengine.framework.dataspace.tests.test_Reaper),</code>	<code>test_get_header_not_exist() (in module decisio-</code>
85	<code>nengine.framework.dataspace.datasources.tests.test_datasource_</code>
<code>test_fail_start_two_reapers() (in module decisio-</code>	81
<code>nengine.framework.dataspace.tests.test_Reaper),</code>	<code>test_get_header_not_exist() (in module decisio-</code>
85	<code>nengine.framework.dataspace.tests.test_dataspace),</code>
<code>test_fail_wrong_config_key() (in module decisio-</code>	87
<code>nengine.framework.dataspace.tests.test_Reaper),</code>	<code>test_get_last_generation_id() (in module decisio-</code>



```

nengine.framework.dataspace.datasources.tests.test_datasource_api),
81                                     65
test_get_last_generation_id() (in module decisio- test_global_config_file() (in module decisio-
nengine.framework.dataspace.tests.test_dataspace), nengine.framework.config.tests.test_policies),
87                                     65
test_get_last_generation_id_not_exist() test_has_config() (in module decisio-
(in module decisio- nengine.framework.dataspace.datasources.tests.test_datasource_
nengine.framework.dataspace.datasources.tests.test_datasource_api),
81                                     81
test_get_last_generation_id_not_exist() test_has_config() (in module decisio-
(in module decisio- nengine.framework.dataspace.tests.test_dataspace),
nengine.framework.dataspace.tests.test_dataspace) 88
87 test_has_methods_we_expect() (in module decisio-
nengine.framework.dataspace.tests.test_datasource),
test_get_metadata() (in module decisio- 87
nengine.framework.dataspace.datasources.tests.test_datasource_api),
81 test_header_constructor() (in module decisio-
nengine.framework.dataspace.tests.test_datablock),
test_get_metadata() (in module decisio- 86
nengine.framework.dataspace.tests.test_dataspace) test_header_is_valid() (in module decisio-
87 nengine.framework.dataspace.tests.test_datablock),
test_get_metadata_not_exist() (in module decisio- 86
nengine.framework.dataspace.datasources.tests.test_datasource_api), (in module decisio-
81 nengine.framework.tests.test_module_program_options),
test_get_metadata_not_exist() (in module decisio- 128
nengine.framework.dataspace.tests.test_dataspace) test_index_error() (in module decisio-
87 nengine.framework.logicengine.tests.test_fail_on_error),
test_get_taskmanager_exists() (in module decisio- 106
nengine.framework.dataspace.datasources.tests.test_datasource_api), (in module decisio-
81 nengine.framework.dataspace.datasources.tests.test_datasource_
test_get_taskmanager_exists() (in module decisio- 81
nengine.framework.dataspace.tests.test_dataspace) test_insert() (in module decisio-
87 nengine.framework.dataspace.tests.test_dataspace),
test_get_taskmanager_not_exists() 88
(in module decisio- test_jpath() (in module decisio-
nengine.framework.dataspace.datasources.tests.test_datasource_api), nengine.framework.config.tests.test_de_std),
81                                     64
test_get_taskmanager_not_exists() test_just_stop_no_error() (in module decisio-
(in module decisio- nengine.framework.dataspace.tests.test_Reaper),
nengine.framework.dataspace.tests.test_dataspace), 85
88 test_loop_of_start_stop_in_clumps()
test_get_taskmanagers() (in module decisio- (in module decisio-
nengine.framework.dataspace.datasources.tests.test_datasource_api), nengine.framework.dataspace.tests.test_Reaper),
81                                     85
test_get_taskmanagers() (in module decisio- test_mark_expired() (in module decisio-
nengine.framework.dataspace.tests.test_dataspace), nengine.framework.dataspace.tests.test_dataspace),
88                                     88
test_get_taskmanagers_not_exist() test_Metadata_constructor() (in module decisio-
(in module decisio- nengine.framework.dataspace.tests.test_datablock),
nengine.framework.dataspace.datasources.tests.test_datasource_api),
81                                     86
test_get_taskmanagers_not_exist() test_Metadata_set_state() (in module decisio-
(in module decisio- nengine.framework.dataspace.tests.test_datablock),
nengine.framework.dataspace.tests.test_dataspace) 86
88 test_minimal_jsonnet_right_extension()
test_global_config_dir() (in module decisio- (in module decisio-
nengine.framework.config.tests.test_config),

```

64	85
test_minimal_jsonnet_wrong_extension() (in module decisio- nengine.framework.config.tests.test_config),	test_reset_connections() (in module decisio- nengine.framework.dataspace.datasources.tests.test_datasource_
64	81
test_missing_data_product_name_not_supported() (in module decisio- nengine.framework.modules.tests.test_EmptySource),	test_rule_that_does_not_fire() (in module decisio- nengine.framework.logicengine.tests.test_cascaded_rules),
110	105
test_misspecified_fact() (in module decisio- nengine.framework.logicengine.tests.test_fail_on_error),	test_rule_that_does_not_fire() (in module decisio- nengine.framework.logicengine.tests.test_pandas_fact),
106	106
test_module_alias() (in module decisio- nengine.framework.tests.test_module_program_options),	test_rule_that_does_not_fire() (in module decisio- nengine.framework.logicengine.tests.test_rule_with_negated_fact
128	106
test_module_structure() (in module decisio- nengine.framework.modules.tests.test_Module),	test_rule_that_does_not_fire() (in module decisio- nengine.framework.logicengine.tests.test_simple_configuration),
110	106
test_multiple_consumes_declarations() (in module decisio- nengine.framework.modules.tests.test_module_decorators),	test_rule_that_fires() (in module decisio- nengine.framework.logicengine.tests.test_cascaded_rules),
111	105
test_multiple_produces_declarations() (in module decisio- nengine.framework.modules.tests.test_module_decorators),	test_rule_that_fires() (in module decisio- nengine.framework.logicengine.tests.test_pandas_fact),
111	106
test_publisher_status() (in module decisio- nengine.framework.tests.test_publisher_status),	test_rule_that_fires() (in module decisio- nengine.framework.logicengine.tests.test_rule_with_negated_fact
128	106
test_publisher_status_board() (in module decisio- nengine.framework.tests.test_publisher_status_board),	test_rule_that_fires() (in module decisio- nengine.framework.logicengine.tests.test_simple_configuration),
128	106
test_publisher_structure() (in module decisio- nengine.framework.modules.tests.test_Publisher),	test_same_source_types_separate_channels() (in module decisio- nengine.framework.tests.test_same_source_types),
110	128
test_query_tool() (in module decisio- nengine.framework.tests.test_query_tool_server),	test_setup_queue_logging() (in module decisio- nengine.framework.modules.tests.test_QueueLogger),
128	110
test_query_tool_help() (in module decisio- nengine.framework.engine.tests.test_query_tool_only),	test_shared_source() (in module decisio- nengine.framework.tests.test_shared_sources),
97	129
test_query_tool_with_no_server() (in module decisio- nengine.framework.engine.tests.test_query_tool_only),	test_simple_fact() (in module decisio- nengine.framework.logicengine.tests.test_facts),
97	105
test_query_tool_with_no_server_verbose() (in module decisio- nengine.framework.engine.tests.test_query_tool_only),	test_source_fail_can_be_fixed() (in module decisio- nengine.framework.dataspace.tests.test_Reaper),
97	86
test_reap_default_state() (in module decisio- nengine.framework.dataspace.tests.test_Reaper),	test_source_structure() (in module decisio- nengine.framework.modules.tests.test_Source),
85	111
test_reaper_can_reap() (in module decisio- nengine.framework.dataspace.tests.test_Reaper),	test_start_delay() (in module decisio- nengine.framework.dataspace.tests.test_Reaper),

86	111
test_start_queue_logger() (in module decisio- nengine.framework.modules.tests.test_QueueLogger),	test_translate_with_underscores() (in module decisio- nengine.framework.modules.tests.test_translate_product_name),
110	111
test_start_stop() (in module decisio- nengine.framework.dataspace.tests.test_Reaper),	test_trivial_configuration() (in module decisio- nengine.framework.logicengine.tests.test_construction),
86	105
test_start_stop_stop() (in module decisio- nengine.framework.dataspace.tests.test_Reaper),	test_true_fact() (in module decisio- nengine.framework.logicengine.tests.test_fail_on_error),
86	106
test_state_can_be_active() (in module decisio- nengine.framework.dataspace.tests.test_Reaper),	test_true_literal_fact() (in module decisio- nengine.framework.logicengine.tests.test_fail_on_error),
86	106
test_state_sets_timer_and_uses_it() (in module decisio- nengine.framework.dataspace.tests.test_Reaper),	test_update() (in module decisio- nengine.framework.dataspace.datasources.tests.test_datasource_
86	81
test_status_during_startup() (in module decisio- nengine.framework.tests.test_status_during_startup),	test_update() (in module decisio- nengine.framework.dataspace.tests.test_dataspace),
129	88
test_stop_queue_logger() (in module decisio- nengine.framework.modules.tests.test_QueueLogger),	test_update_bad() (in module decisio- nengine.framework.dataspace.datasources.tests.test_datasource_
110	82
test_store_taskmanager() (in module decisio- nengine.framework.dataspace.datasources.tests.test_datasource_	test_update_bad() (in module decisio- nengine.framework.dataspace.tests.test_dataspace),
81	88
test_store_taskmanager() (in module decisio- nengine.framework.dataspace.tests.test_dataspace),	test_valid_but_empty_config() (in module decisio- nengine.framework.config.tests.test_config),
88	64
test_supports_config() (in module decisio- nengine.framework.modules.tests.test_module_decorators),	test_valid_dir() (in module decisio- nengine.framework.config.tests.test_policies),
111	65
test_syntax_error() (in module decisio- nengine.framework.logicengine.tests.test_facts),	test_verify_bad_broker() (in module decisio- nengine.framework.engine.tests.test_verify_redis_server),
105	98
test_syntax_error_in_config_names_bad_file() (in module decisio- nengine.framework.config.tests.test_config),	test_verify_bad_redis_server() (in module decisio- nengine.framework.engine.tests.test_verify_redis_server),
64	98
test_transform_structure() (in module decisio- nengine.framework.modules.tests.test_Transform),	test_verify_bad_url() (in module decisio- nengine.framework.engine.tests.test_verify_redis_server),
111	98
test_translate_all() (in module decisio- nengine.framework.modules.tests.test_translate_product_name),	test_verify_redis_server() (in module decisio- nengine.framework.engine.tests.test_verify_redis_server),
111	98
test_translate_illegal_characters() (in module decisio- nengine.framework.modules.tests.test_translate_product_name),	test_verify_redis_url() (in module decisio- nengine.framework.engine.tests.test_verify_redis_server),
111	98
test_translate_none() (in module decisio- nengine.framework.modules.tests.test_translate_product_name),	test_worker_logger_sized_rotation() (in module decisio- nengine.framework.engine.tests.test_ChannelWorkers),
111	96
test_translate_simple() (in module decisio- nengine.framework.modules.tests.test_translate_product_name),	test_worker_logger_sized_rotation() (in module decisio- nengine.framework.engine.tests.test_SourceWorkers),
	96

96  
test\_worker\_logger\_timed\_rotation() (in module decisionengine.framework.engine.tests.test\_ChannelWorkers), 96  
test\_worker\_logger\_timed\_rotation() (in module decisionengine.framework.engine.tests.test\_SourceWorkers), 96  
test\_worker\_logger\_wrong\_rotation\_method() (in module decisionengine.framework.engine.tests.test\_SourceWorkers), 96  
test\_worker\_name() (in module decisionengine.framework.engine.tests.test\_ChannelWorkers), 96  
test\_worker\_name() (in module decisionengine.framework.engine.tests.test\_SourceWorkers), 96  
test\_wrong\_configuration() (in module decisionengine.framework.logicengine.tests.test\_construction), 105  
test\_wrong\_product\_names() (in module decisionengine.framework.modules.tests.test\_module\_decorators), method), 118  
test\_wrong\_product\_types() (in module decisionengine.framework.modules.tests.test\_module\_decorators), method), 119  
test\_zdumps() (in module decisionengine.framework.dataspace.tests.test\_datablock\_zlib), 87  
test\_zloads() (in module decisionengine.framework.dataspace.tests.test\_datablock\_zlib), 87  
Transform (class in decisionengine.framework.modules.Transform), 113  
transform() (decisionengine.framework.modules.Transform.Transform method), 113  
transform() (decisionengine.framework.tests.DynamicTransform.DynamicTransform method), 122  
transform() (decisionengine.framework.tests.TransformNOP.TransformNOP method), 125  
transform() (decisionengine.framework.tests.TransformWithMissingProduceConsumes.TransformWithMissingProduceConsumes method), 125  
TransformNOP (class in decisionengine.framework.tests.TransformNOP), 125  
TransformWithMissingProduceConsumes (class in decisionengine.framework.tests.TransformWithMissingProduceConsumes method), 125  
transform\_with\_missing\_produce\_consumes() (in module decisionengine.framework.tests.TransformWithMissingProduceConsumes method), 125  
translate() (in module decisionengine.framework.modules.translate\_product\_name), 115  
translate\_all() (in module decisionengine.framework.modules.translate\_product\_name), 115  
U  
update() (decisionengine.framework.dataspace.datasource.DataSource method), 93  
update() (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 84  
update() (decisionengine.framework.dataspace.datasources.sqlalchemy\_datasource.DataSource method), 71  
update() (decisionengine.framework.dataspace.datasources.sqlalchemy\_datasource.DataSource method), 79  
update() (decisionengine.framework.dataspace.dataspace.DataSpace method), 94  
update() (decisionengine.framework.engine.SourceWorkers.SourceWorker method), 103  
update() (decisionengine.framework.taskmanager.PublisherStatus.PublisherStatus method), 118  
update() (decisionengine.framework.taskmanager.SourceProductCache.SourceProductCache method), 119  
V  
valid\_dir() (in module decisionengine.framework.config.policies), 67  
valid\_states (decisionengine.framework.dataspace.datablock.Metadata attribute), 90  
validated\_workflow() (in module decisionengine.framework.taskmanager.module\_graph), 121  
ValidConfig (class in decisionengine.framework.config.ValidConfig), 66  
value (decisionengine.framework.dataspace.datasources.sqlalchemy\_datasource.DataSource attribute), 72  
verify\_products() (in module decisionengine.framework.modules.Module), 112  
W  
wait\_for\_n\_writes() (in module decisionengine.framework.tests.WriteToDisk), 125  
wait\_until() (decisionengine.framework.taskmanager.ProcessingState.ProcessingState method), 117  
wait\_while() (decisionengine.framework.engine.ChannelWorkers.ChannelWorker method), 98

`wait_while()` (*decisionengine.framework.taskmanager.ProcessingState.ProcessingState* method), [117](#)

`Worker` (class in *decisionengine.framework.taskmanager.module\_graph*), [120](#)

`WriteToDisk` (class in *decisionengine.framework.tests.WriteToDisk*), [125](#)

## Z

`zdumps()` (in module *decisionengine.framework.dataspace.datablock*), [91](#)

`zloads()` (in module *decisionengine.framework.dataspace.datablock*), [91](#)