

---

# **decisionengine**

***Release 1.7.1.dev20+gc4fc5997***

**Fermilab**

**Oct 01, 2021**



# CONTENTS

<b>1</b>	<b>Release Notes</b>	<b>3</b>
<b>2</b>	<b>Developer Documentation</b>	<b>25</b>
<b>3</b>	<b>Jenkins CI pipeline</b>	<b>27</b>
<b>4</b>	<b>Source code</b>	<b>33</b>
<b>5</b>	<b>Indices and tables</b>	<b>97</b>
	<b>Python Module Index</b>	<b>99</b>
	<b>Index</b>	<b>103</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 a restricted allocation of cycles



## RELEASE NOTES

### 1.1 Release Notes

HEPCloud's Decision Engine release notes.

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

#### 1.1.1 Release 1.7.0

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 length 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

## 1.1.2 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]

### 1.1.3 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)

### 1.1.4 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 pycpg2cffi (#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)  
1f8e44d : 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

## 1.1.5 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)

## 1.1.6 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)

## 1.1.7 Release 1.4.0

In this release:

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

### Issues fixed in this release

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

### Full list of commits since version 1.3

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

d836abf : next RC

926944a : Fix coveralls reporting (#198)

b95c323 : Updating base Dockerfile (#199)

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

2d791a7 : Test configuration policies. (#197)

236e27a : Ensure items are returned in a stable order (#202)  
e974f5f : add pylint and pycodestyle (#203)  
fbc7616 : 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.

## **1.1.8 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)

## 1.1.9 Release 1.2.0

In this release:

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

### decisionengine

3dfe167 : Jenkins pipeline improvements (#106)  
22a7073 : pull request for review request 137 (#105)  
cafffb2 : Make it possible to run code directly (for tests), and (#100)  
802e98b : replace psycog2 witt psycopg2-binary (#101)  
573ce8f : Jenkins pipeline improvements (#99)  
9d08835 : Run coveralls even under failed state (#97)  
bc1df4b : Add tests for PostgreSQL datasource (#71)  
c1ac391 : Fix missing py-modules.html (#96)  
8dbfdee : Setup gh-pages doc workflow (#94)  
cd4a01a : Doc (#93)  
673080d : set version to 1.2.0 (for now). Supply conf file that corresponds to (#91)  
f912225 : Db (#92)  
dc8b68a : Add reaper to the RPC (#83) (#90)  
29ade91 : adding .Jenkinsfile with Jenkins pipeline configuration (#86)  
c1dfe5c : Don't exclude E1004 from pylint, do exclude line breaks (#89)  
440f949 : Fix varname (#88)  
313d135 : Compress (#87)  
6b8dc4b : Revert "Add reaper to the RPC (#83)"

dbea8e5 : Update utils.sh so pytest will complete.  
e848316 : Update to postgresql11  
7f4b805 : Add reaper to the RPC (#83)  
0ba2c51 : remove astpp module and dependencies it pulls in (#81)  
6b8eab9 : don't track test coverage of tests (#80)  
0da18ec : made reaper.py executable  
aca24a3 : make reaper.py executable, make symbolic link to it from /usr/bin (#72)  
0202acf : Implementation of data reaper (#70)  
16b6be1 : Simple changes for Python 3 deployment (#69)  
fd2418c : Fix warnings caught by PEP-8 Speaks.  
d16359b : Python 3 (and other) simplifications.  
3c7b6b7 : Only run Github Actions for python3.6 (#68)  
453cbba : Update README.md  
b27ed53 : remove unnecessary (and actually 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/decisionengine\\_modules/issues/224](https://github.com/HEPCloud/decisionengine_modules/issues/224) (#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 coverage  
6d8a5f5 : decisionengine\_modules/NERSC: Make Nersc API call backward-compatible with old config (#196)  
a7e0af9 : Only run Github Actions for python3.6 (#24)

### 1.1.10 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 omission.  
24e0795 : Apply clang-format  
17c17f9 : Remove JSON dependency.  
faa0b22 : Massive cleanup.  
07b555f : Updates to Github Actions to allow building with python3.6  
fef6c11 : Fix errors when running pylint.sh multiple times  
da6f077 : Autopep8 -i fixes

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

17396da : logicengine: get rid of 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



## 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

### 3.1 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.

### 3.1.1 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

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

### 4.1 Welcome to decisionengine's documentation!

#### 4.1.1 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_combine_one_level()  
decisionengine.framework.config.tests.test_de_std.test_combine_one_level_skip_proxies()  
decisionengine.framework.config.tests.test_de_std.test_error_on_duplicate_keys()  
decisionengine.framework.config.tests.test_de_std.test_jpath()
```

#### **decisionengine.framework.config.tests.test\_policies module**

```
decisionengine.framework.config.tests.test_policies.test_channel_config_dir(tmp_path,  
                                                                              monkeypatch)  
decisionengine.framework.config.tests.test_policies.test_global_config_dir(tmp_path,  
                                                                              monkeypatch)  
decisionengine.framework.config.tests.test_policies.test_global_config_file(tmp_path,  
                                                                              monkeypatch)  
decisionengine.framework.config.tests.test_policies.test_valid_dir(tmp_path)
```

## Module contents

### Submodules

#### decisionengine.framework.config.ChannelConfigHandler module

Manager of channel configurations.

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

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

Bases: object

**\_load\_channel**(*channel\_name*, *path*)

**get\_channels**()

**load\_all\_channels**()

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

Any cached configurations will be dropped prior to reloading.

**load\_channel**(*channel\_name*)

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

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

<cached channel config. dir>/{channel\_name}.jsonnet

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

**print\_channel\_config**(*channel*)

decisionengine.framework.config.ChannelConfigHandler.**\_check\_keys**(*channel\_conf\_dict*)

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

decisionengine.framework.config.ChannelConfigHandler.**\_make\_de\_logger**(*global\_config*)

#### decisionengine.framework.config.ValidConfig module

ValidConfig represents a valid JSON document.

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

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

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

Bases: collections.UserDict

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

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

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

```
dump()
```

Print dictionary data to a valid JSON string.

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

## decisionengine.framework.config.policies module

Decision-engine default configuration policies.

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

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

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

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

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

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

1. If the ‘parent\_dir’ argument is provided, the returned path object will correspond to ‘{parent\_dir}/config.d’.
2. If the ‘CHANNEL\_CONFIG\_PATH’ environment variable has been set, the returned path object will correspond to \${CHANNEL\_CONFIG\_PATH}.
3. If neither 1 or 2 apply, the returned path object corresponds to ‘{global\_config\_dir()}/config.d’ (see documentation for ‘global\_config\_dir()’).

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

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

Retrieve global configuration dir as pathlib.Path object.

This is the directory that houses the ‘decision\_engine.jsonnet’ global configuration file.

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

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

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

If supplied, the ‘parent\_dir’ is assumed to be the full path corresponding to a directory containing the ‘decision\_engine.jsonnet’ file. If not provided, the global configuration directory is determined based on the behavior of the ‘global\_config\_dir()’ function.

An exception is raised if no ‘decision\_engine.jsonnet’ file is found.

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

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



## Module contents

### decisionengine.framework.dataspace package

#### Subpackages

### decisionengine.framework.dataspace.datasources package

#### Subpackages

### decisionengine.framework.dataspace.datasources.sqlalchemy\_ds package

#### Submodules

### decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.datasource\_api module

The datasource layer for our abstraction

**class** decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.datasource\_api.**SQLAlchemyDS**(*config\_dict*)  
Bases: *decisionengine.framework.dataspace.datasource.DataSource*

A DecisionEngine data source via the SQL Alchemy ORM

```

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

```

Exceptions should be caught and logged by the caller.

**\_abc\_impl** = <\_abc.\_abc\_data object>

**close()**

Close all connections to the database

**Returns** None

**connect()**

Create a pool of database connections

**Returns** None

**create\_tables()**

Create database tables

**Returns** None

**delete\_data\_older\_than**(*days*)

Delete data older than interval

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

**Returns** None

**duplicate\_datablock**(*taskmanager\_id, generation\_id, new\_generation\_id*)

For the given *taskmanager\_id*, make a copy of the datablock with given *generation\_id*, set the *generation\_id* for the datablock copy

**Parameters**

- **taskmanager\_id** (*str/uuid*) – id of taskmanager to retrieve
- **generation\_id** (*int*) – generation id to clone
- **new\_generation\_id** (*int*) – generation id to create

**Returns** None

**get\_datablock**(*taskmanager\_id, generation\_id*)

Return the entire datablock from the dataproduct table for the given *taskmanager\_id*, *generation\_id*

**Parameters**

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

**Returns** with all set keys and their associated values

**Return type** dict

**get\_dataproduct**(*taskmanager\_id, generation\_id, key*)

Return the data from the dataproduct table for the given *taskmanager\_id*, *generation\_id*, *key*

**Parameters**

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

**Returns** The possibly binary value stored earlier

**Return type** obj

**get\_dataproducts**(*taskmanager\_id, key=None*)

Return list of all data products associated with *taskmanager\_id*

**Parameters**

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

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

**Return type** tuple

**get\_header**(*taskmanager\_id, generation\_id, key*)

Return the header from the header table for the given *taskmanager\_id*, *generation\_id*, *key*

**Parameters**

- **taskmanager\_id** (*str/uuid*) – id of taskmanager to retrieve

- **generation\_id** (*int*) – generation id to locate
- **key** (*str*) – key for the value

**Returns**

**fields in order are:** taskmanager.taskmanager\_id, header.taskmanager\_id,  
header.generation\_id, header.key, header.create\_time, header.expiration\_time,  
header.scheduled\_create\_time, header.creator, header.schema\_id

**Return type** tuple

**get\_last\_generation\_id**(*taskmanager\_name*, *taskmanager\_id=None*)

Return last generation id for current task manager or taskmanager w/ task\_manager\_id.

**Parameters**

- **taskmanager\_name** (*str*) – name of taskmanager to retrieve
- **taskmanager\_id** (*str/uuid*) – id of taskmanager to retrieve

**Returns** the largest generation stored within the database

**Return type** int

**get\_metadata**(*taskmanager\_id*, *generation\_id*, *key*)

Return the metadata from the metadata table for the given taskmanager\_id, generation\_id, key

**Parameters**

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

**Returns**

**fields in order are:** taskmanager.taskmanager\_id, metadata.taskmanager\_id, meta-  
data.generation\_id, metadata.key, metadata.state, metadata.generation\_time, meta-  
data.missed\_update\_count

**Return type** tuple

**get\_schema**(*table=None*)

Given the table name return it's schema

**get\_taskmanager**(*taskmanager\_name*, *taskmanager\_id=None*)

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

If multiples match, find highest primary key.

**Parameters**

- **taskmanager\_name** (*str*) – name of taskmanager to retrieve
- **taskmanager\_id** (*str/uuid*) – id of taskmanager to retrieve

**Returns** the matching row, column names as keys

**Return type** dict

**get\_taskmanagers**(*taskmanager\_name=None*, *start\_time=None*, *end\_time=None*)

Find taskmanagers that meet our search

**Parameters**

- **taskmanager\_name** (*str*) – name of taskmanager to retrieve

- **start\_time** (*datetime*) – Datetime to confine against
- **end\_time** (*datetime*) – Datetime to confine against

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

**Return type** list

**insert**(*taskmanager\_id, generation\_id, key, value, header, metadata*)

Insert data into respective tables for the given taskmanager\_id, generation\_id, key

**Parameters**

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

**Returns** None

**reset\_connections**()

Reset the connection to the database. So long as self.engine isn't undef, the engine can still make new connections if new db actions happen. It just won't 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: sqlalchemy.orm.decl_api.Base
```

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

Existing code appears to depend on column order.

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

```
generation_id
```

```
id
```

```
key
```

```
taskmanager
```

```
taskmanager_id
```

```
value
```

```
class decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Header(**kwargs)
    Bases: sqlalchemy.orm.decl_api.Base
```

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

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

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

Existing code appears to depend on column order.

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

create\_time

creator

expiration\_time

generation\_id

id

key

scheduled\_create\_time

schema\_id

taskmanager

taskmanager\_id

```
class decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Metadata(**kwargs)  
    Bases: sqlalchemy.orm.decl\_api.Base
```

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

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

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

Existing code appears to depend on column order.

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

generation\_id

generation\_time

id

key

missed\_update\_count

state

**taskmanager**

**taskmanager\_id**

**class** decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema.**Schema**(\*\*kwargs)  
 Bases: [sqlalchemy.orm.decl\\_api.Base](#)

This table may not be in use

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

**\_sa\_class\_manager** = {'schema': <sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'schema\_id': <sqlalchemy.orm.attributes.InstrumentedAttribute object>}

**schema**

**schema\_id**

**class** decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema.**Taskmanager**(\*\*kwargs)  
 Bases: [sqlalchemy.orm.decl\\_api.Base](#)

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

**changes cascade on:** Header Metadata Dataproduct

Existing code appears to depend on column order.

**\_sa\_class\_manager** = {'datestamp': <sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'name': <sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'sequence\_id': <sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'task\_dataproduct': <sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'task\_header': <sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'task\_metadata': <sqlalchemy.orm.attributes.InstrumentedAttribute object>, 'taskmanager\_id': <sqlalchemy.orm.attributes.InstrumentedAttribute object>}

**datestamp**

**name**

**sequence\_id**

**task\_dataproduct**

**task\_header**

**task\_metadata**

**taskmanager\_id**

## decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.utils module

Code not written by us

decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.utils.**add\_engine\_pidguard**(engine)

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

decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.utils.**clone\_model**(model, \*\*kwargs)

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

decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.utils.**orm\_as\_dict**(obj)

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

## Module contents

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

**class** decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.**SQLAlchemyDS**(*config\_dict*)  
Bases: *decisionengine.framework.dataspace.datasource.DataSource*

A DecisionEngine data source via the SQL Alchemy ORM

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

Exceptions should be caught and logged by the caller.

**\_abc\_impl** = <\_abc.\_abc\_data object>

**close()**

Close all connections to the database

**Returns** None

**connect()**

Create a pool of database connections

**Returns** None

**create\_tables()**

Create database tables

**Returns** None

**delete\_data\_older\_than(days)**

Delete data older than interval

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

**Returns** None

**duplicate\_datablock(taskmanager\_id, generation\_id, new\_generation\_id)**

For the given taskmanager\_id, make a copy of the datablock with given generation\_id, set the generation\_id for the datablock copy

**Parameters**

- **taskmanager\_id** (*str/uuid*) – id of taskmanager to retrieve
- **generation\_id** (*int*) – generation id to clone
- **new\_generation\_id** (*int*) – generation id to create

**Returns** None



**get\_datablock**(*taskmanager\_id*, *generation\_id*)

Return the entire datablock from the dataproduct table for the given taskmanager\_id, generation\_id

**Parameters**

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

**Returns** with all set keys and their associated values

**Return type** dict

**get\_dataproduct**(*taskmanager\_id*, *generation\_id*, *key*)

Return the data from the dataproduct table for the given taskmanager\_id, generation\_id, key

**Parameters**

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

**Returns** The possibly binary value stored earlier

**Return type** obj

**get\_dataproducts**(*taskmanager\_id*, *key=None*)

Return list of all data products associated with with taskmanager\_id

**Parameters**

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

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

**Return type** tuple

**get\_header**(*taskmanager\_id*, *generation\_id*, *key*)

Return the header from the header table for the given taskmanager\_id, generation\_id, key

**Parameters**

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

**Returns**

**fields in order are:** taskmanager.taskmanager\_id, header.taskmanager\_id,  
header.generation\_id, header.key, header.create\_time, header.expiration\_time,  
header.scheduled\_create\_time, header.creator, header.schema\_id

**Return type** tuple

**get\_last\_generation\_id**(*taskmanager\_name*, *taskmanager\_id=None*)

Return last generation id for current task manager or taskmanager w/ task\_manager\_id.

**Parameters**

- **taskmanager\_name** (*str*) – name of taskmanager to retrieve
- **taskmanager\_id** (*str/uuid*) – id of taskmanager to retrieve

**Returns** the largest generation stored within the database

**Return type** int

**get\_metadata**(*taskmanager\_id, generation\_id, key*)

Return the metadata from the metadata table for the given taskmanager\_id, generation\_id, key

**Parameters**

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

**Returns**

**fields in order are:** taskmanager.taskmanager\_id, metadata.taskmanager\_id, 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, 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:*

*\_pytest.fixtures.Fixture*  
→  
*psycpg2.extensions.c*

Fixture factory for PostgreSQL.

**Parameters** **request** – fixture request object

**Returns** postgresql client

`decisionengine.framework.dataspace.datasources.tests.fixtures.PG_DE_DB_WITH_SCHEMA(PG_DE_DB_WITHOUT_SCHEMA)`  
 Load our PG schema into the database via this fixture so pytest knows the limitations on parallel usage of this database scope.

`decisionengine.framework.dataspace.datasources.tests.fixtures.PG_PROG(request: _pytest.fixtures.FixtureRequest, tmpdir_factory: _pytest.tmpdir.TempdirFactory) → Iterator[pytest_postgresql.executor.PostgreSQLExecutor]`

Process fixture for PostgreSQL.

**Parameters** `request` – fixture request object

**Returns** tcp executor

`decisionengine.framework.dataspace.datasources.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.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.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.datasources.tests.fixtures.mock_data_block()`  
 This fixture replaces the standard datablock implementation.

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

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

## decisionengine.framework.dataspace.datasources.tests.test\_datasource\_api module

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

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

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

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

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

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

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

**decisionengine.framework.dataspace.datasources.tests.test\_postgresql module**`decisionengine.framework.dataspace.datasources.tests.test_postgresql.test_generate_insert_query()`**Module contents****Submodules****decisionengine.framework.dataspace.datasources.null module****class** `decisionengine.framework.dataspace.datasources.null.NullDataSource(config_dict)`Bases: `decisionengine.framework.dataspace.datasource.DataSource`

Implementation of data source ABC that does nothing

`_abc_impl = <_abc._abc_data object>`**close()**

Close all connections to the database

**connect()**

Create a pool of database connections

**create\_tables()**

Create database tables

**delete\_data\_older\_than(*days*)**

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

**duplicate\_datablock(*taskmanager\_id, generation\_id, new\_generation\_id*)**

For the given taskmanager\_id, make a copy of the datablock with given generation\_id, set the generation\_id for the datablock copy

**Parameters**

- **taskmanager\_id** (string) – taskmanager\_id for generation to be retrieved
- **generation\_id** (int) – generation\_id of the data
- **new\_generation\_id** (int) – generation\_id of the new datablock created

**get\_datablock(*taskmanager\_id, generation\_id*)**

Return the entire datablock from the dataproduct table for the given taskmanager\_id, generation\_id

**Parameters**

- **taskmanager\_id** (string) – taskmanager\_id for generation to be retrieved
- **generation\_id** (int) – generation\_id of the data

**get\_dataproduct(*taskmanager\_id, generation\_id, key*)**

Return the data from the dataproduct table for the given taskmanager\_id, generation\_id, key

**Parameters**

- **taskmanager\_id** (string) – taskmanager\_id for generation to be retrieved
- **generation\_id** (int) – generation\_id of the data
- **key** (string) – key for the value

**get\_dataproducts**(*taskmanager\_id*, *key=None*)

Return list of all data products associated with with taskmanager\_id

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

**get\_header**(*taskmanager\_id*, *generation\_id*, *key*)

Return the header from the header table for the given taskmanager\_id, generation\_id, key

**Parameters**

- **taskmanager\_id** (string) – taskmanager\_id for generation to be retrieved
- **generation\_id** (int) – generation\_id of the data
- **key** (string) – key for the value

**get\_last\_generation\_id**(*taskmanager\_name*, *taskmanager\_id=None*)

Return last generation id for current task manager or taskmanager w/ task\_manager\_id.

**Parameters**

- **taskmanager\_name** (string) – task manager name
- **taskmanager\_id** (string) – task manager id

**get\_metadata**(*taskmanager\_id*, *generation\_id*, *key*)

Return the metadata from the metadata table for the given taskmanager\_id, generation\_id, key

**Parameters**

- **taskmanager\_id** (string) – taskmanager\_id for generation to be retrieved
- **generation\_id** (int) – generation\_id of the data
- **key** (string) – key for the value

**get\_schema**(*table=None*)

Given the table name return it's schema

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

**get\_taskmanager**(*taskmanager\_name*, *taskmanager\_id=None*)

Retrieve TaskManager :type taskmanager\_name: **string** :arg taskmanager\_name: name of taskmanager to retrieve :type taskmanager\_id: **string** :arg taskmanager\_id: id of taskmanager to retrieve

**get\_taskmanagers**(*taskmanager\_name=None*, *start\_time=None*, *end\_time=None*)

Retrieve TaskManagers :type taskmanager\_name: **string** :arg taskmanager\_name: name of taskmanager to retrieve :type taskmanager\_id: **string** :arg taskmanager\_id: id of taskmanager to retrieve

**insert**(*taskmanager\_id*, *generation\_id*, *key*, *value*, *header*, *metadata*)

Insert data into respective tables for the given taskmanager\_id, generation\_id, key

**Parameters**

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

**reset\_connections**()

Drop any cached connections and reconnect to the database

**store\_taskmanager**(*name, taskmanager\_id, datestamp=None*)

Store TaskManager :type taskmanager\_name: **string** :arg taskmanager\_name: name of taskmanager to retrieve :type taskmanager\_id: **string** :arg taskmanager\_id: id of taskmanager to retrieve :type datestamp: **datetime** :arg datestamp: datetime of created object, defaults to 'now'

**update**(*taskmanager\_id, generation\_id, key, value, header, metadata*)

Update the data in respective tables for the given taskmanager\_id, generation\_id, key

**Parameters**

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

## decisionengine.framework.dataspace.datasources.postgresql module

**class** decisionengine.framework.dataspace.datasources.postgresql.**Postgresql**(*config\_dict*)

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

Implementation of postgresql data source

**\_\_query**(*query\_string, values=None, cursor\_factory=None*)

**\_abc\_impl** = <\_abc.\_abc\_data object>

**\_delete**(*sql\_query, values=None*)

**\_insert**(*table\_name\_or\_sql\_query, record=None*)

**\_insert\_returning\_result**(*table\_name\_or\_sql\_query, record=None*)

**\_remove**(*sql\_query, values=None*)

**\_select**(*query\_string, values=None, cursor\_factory=None*)

**\_select\_dictresult**(*sql\_query, values=None*)

**\_select\_getresult**(*sql\_query, values=None*)

**\_select\_tuple**(*sql\_query, values*)

**\_update**(*query\_string, values=None*)

**\_update\_returning\_result**(*query\_string, values=None*)

**close**()

Close all connections to the database

**connect**()

Create a pool of database connections

**create\_tables**()

Create database tables

**delete\_data\_older\_than**(*days*)

Delete data older than days interval :type days: **int** :arg days: remove data older than days interval



**duplicate\_datablock**(*taskmanager\_id, generation\_id, new\_generation\_id*)

For the given taskmanager\_id, make a copy of the datablock with given generation\_id, set the generation\_id for the datablock copy

**Parameters**

- **taskmanager\_id** (string) – taskmanager\_id for generation to be retrieved
- **generation\_id** (int) – generation\_id of the data
- **new\_generation\_id** (int) – generation\_id of the new datablock created

**get\_connection**()

**get\_datablock**(*taskmanager\_id, generation\_id*)

Return the entire datablock from the dataproduct table for the given taskmanager\_id, generation\_id

**Parameters**

- **taskmanager\_id** (string) – taskmanager\_id for generation to be retrieved
- **generation\_id** (int) – generation\_id of the data

**get\_dataproduct**(*taskmanager\_id, generation\_id, key*)

Return the data from the dataproduct table for the given taskmanager\_id, generation\_id, key

**Parameters**

- **taskmanager\_id** (string) – taskmanager\_id for generation to be retrieved
- **generation\_id** (int) – generation\_id of the data
- **key** (string) – key for the value

**get\_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, 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'

```
tables = {'dataproduuct': ['taskmanager_id TEXT', 'generation_id INT', 'key TEXT',  
'value BLOB'], 'header': ['taskmanager_id TEXT', 'generation_id INT', 'key TEXT',  
'create_time REAL', 'expiration_time REAL', 'scheduled_create_time REAL', 'creator  
TEXT', 'schema_id INT'], 'metadata': ['taskmanager_id TEXT', 'generation_id INT',  
'key TEXT', 'state TEXT', 'generation_time REAL', 'missed_update_count INT'],  
'schema': ['schema_id INT', 'schema BLOB']}
```

**update**(*taskmanager\_id, generation\_id, key, value, header, metadata*)

Update the data in respective tables for the given taskmanager\_id, generation\_id, key

**Parameters**

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

decisionengine.framework.dataspace.datasources.postgresql.**generate\_insert\_query**(*table\_name, keys*)

Generate insert query given table name and list of fields

**Parameters**

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

**Keys** list**Return type** str - insert query**Module contents****decisionengine.framework.dataspace.tests package****Submodules****decisionengine.framework.dataspace.tests.fixtures module**

decisionengine.framework.dataspace.tests.fixtures.**PG\_DE\_DB\_WITHOUT\_SCHEMA**(request: *\_pytest.fixtures.FixtureRequest*) → *psycpg2.extensions.connection*

Fixture factory for PostgreSQL.

**Parameters** **request** – fixture request object**Returns** postgresql client

decisionengine.framework.dataspace.tests.fixtures.**PG\_DE\_DB\_WITH\_SCHEMA**(*PG\_DE\_DB\_WITHOUT\_SCHEMA*)  
Load our PG schema into the database via this fixture so pytest knows the limitations on parallel usage of this database scope.

decisionengine.framework.dataspace.tests.fixtures.**PG\_PROG**(request: *\_pytest.fixtures.FixtureRequest*,  
*tmpdir\_factory*: *\_pytest.tmpdir.TempdirFactory*) → *Iterator[pytest\_postgresql.executor.PostgreSQLExecutor]*

Process fixture for PostgreSQL.

**Parameters** **request** – fixture request object**Returns** tcp executor

decisionengine.framework.dataspace.tests.fixtures.**SQLALCHEMY\_PG\_WITH\_SCHEMA**(*PG\_DE\_DB\_WITHOUT\_SCHEMA*)  
Get a blank database from `pytest_postgresql`. Then setup the SQLAlchemy style URL with that DB. The SQLAlchemyDS will create the schema as needed.

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

decisionengine.framework.dataspace.tests.fixtures.**datasource**(request)  
This parameterized fixture will setup up various datasources.

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

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

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

This parameterized fixture will setup up various datasources. Add datasource objects to DATA-SOURCES\_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\_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\_store\_taskmanager(dataspace)  
Can we make new entries

decisionengine.framework.dataspace.tests.test\_dataspace.test\_update(dataspace)  
Do updates work as expected

decisionengine.framework.dataspace.tests.test\_dataspace.test\_update\_bad(dataspace)  
Do updates fail to work on bogus taskmanager as expected

## Module contents

### Submodules

#### decisionengine.framework.dataspace.datablock module

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

Bases: object

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

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

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

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

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

**Return type** *DataBlock*

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

**Return type** dict

**get\_dataproducts**(key=None)

**get\_header**(key)  
Return the Header associated with the key in the database

**Return type** *Header*

**get\_metadata**(key)  
Return the metadata associated with the key in the database

**Return type** *Metadata*

**get\_taskmanager**(taskmanager\_name, taskmanager\_id=None)  
Retrieve TaskManager :type taskmanager\_name: string :arg taskmanager\_name: name of taskmanager to retrieve :type taskmanager\_id: string :arg taskmanager\_id: id of taskmanager to retrieve :rtype: :obj: dict

The dictionary returned looks like : { 'datestamp': datetime.datetime(2017, 12, 20, 17, 37, 17, 503210, tzinfo=psycpg2.tz.FixedOffsetTimezone(offset=-360, name=None)),  
'sequence\_id': 135L, 'name': 'AWS\_Calculations', 'taskmanager\_id': '77B16EB5-C79E-45B0-B1B1-37E846692E1D' }

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

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

**keys**()

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

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

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

Put data into the DataBlock

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

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

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

Bases: collections.UserDict

**\_abc\_impl** = <\_abc.\_abc\_data object>

**default\_data\_lifetime** = 1800

**is\_valid**()

Check if the Header has minimum required information

**required\_keys** = {'create\_time', 'creator', 'expiration\_time',  
'scheduled\_create\_time', 'schema\_id', 'taskmanager\_id'}

```
exception decisionengine.framework.dataspace.datablock.InvalidMetadataError
```

Bases: Exception

Errors due to invalid Metadata

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

Bases: collections.UserDict

**\_abc\_impl** = <\_abc.\_abc\_data object>

**required\_keys** = {'generation\_id', 'generation\_time', 'missed\_update\_count', 'state',  
'taskmanager\_id'}

**set\_state**(*state*)

Set the state for the Metadata

**valid\_states** = {'END\_CYCLE', 'METADATA\_UPDATE', 'NEW', 'START\_BACKUP'}



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

Bases: object

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

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

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

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

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

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

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

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

**Parameters** **zbytes** – compressed bytes

**Returns** returns python object

## decisionengine.framework.dataspace.datasource module

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

Bases: object

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

```
abstract close()
```

Close all connections to the database

```
abstract connect()
```

Create a pool of database connections

```
abstract create_tables()
```

Create database tables

```
dataprodukt_table = 'dataprodukt'
```

Name of the dataprodukt 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 dataprodukt 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.**DataSet**(config)

Bases: object

DataSet 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*)

A DE Server using a private database

decisionengine.framework.engine.tests.fixtures.**PG\_DE\_DB\_WITHOUT\_SCHEMA**(*request: \_pytest.fixtures.FixtureRequest*)  
→  
psycpg2.extensions.connection

Fixture factory for PostgreSQL.

**Parameters** **request** – fixture request object

**Returns** postgresql client

decisionengine.framework.engine.tests.fixtures.**PG\_DE\_DB\_WITH\_SCHEMA**(*PG\_DE\_DB\_WITHOUT\_SCHEMA*)  
Load our PG schema into the database via this fixture so pytest knows the limitations on parallel usage of this database scope.

decisionengine.framework.engine.tests.fixtures.**PG\_PROG**(*request: \_pytest.fixtures.FixtureRequest, tmpdir\_factory: \_pytest.tmpdir.TempdirFactory*) →  
Iterator[pytest\_postgresql.executor.PostgreSQLExecutor]

Process fixture for PostgreSQL.

**Parameters** **request** – fixture request object

**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\_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.test_change_port()
decisionengine.framework.engine.tests.test_startup.test_default_config()
```

## Module contents

### Submodules

### decisionengine.framework.engine.DecisionEngine module

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

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

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

```
    _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\_until**(*state, timeout=None*)**block\_while**(*state, timeout=None*)**get\_logger**()**handle\_sighup**(*signum, frame*)**reaper\_start**(*delay*)**reaper\_status**()**reaper\_stop**()**rm\_channel**(*channel, maybe\_timeout*)**rpc\_block\_while**(*state\_str, timeout=None*)**rpc\_get\_channel\_log\_level**(*channel*)**rpc\_get\_log\_level**()**rpc\_kill\_channel**(*channel, timeout=None*)**rpc\_print\_product**(*product, columns=None, query=None, types=False, format=None*)**rpc\_print\_products**()**rpc\_query\_tool**(*product, format=None, start\_time=None*)**rpc\_reaper\_start**(*delay=0*)

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

**rpc\_reaper\_status**()**rpc\_reaper\_stop**()**rpc\_rm\_channel**(*channel, maybe\_timeout*)**rpc\_set\_channel\_log\_level**(*channel, log\_level*)

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

**rpc\_show\_config**(*channel*)

Show the configuration for a channel.

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

class decisionengine.framework.engine.DecisionEngine.RequestHandler(request, client_address,
                                                                    server)
    Bases: xmlrpc.server.SimpleXMLRPCRequestHandler
    rpc_paths = ('/RPC2',)

class decisionengine.framework.engine.DecisionEngine.StopState(value)
    Bases: enum.Enum
    An enumeration.
    Clean = 2
    NotFound = 1
    Terminated = 3

decisionengine.framework.engine.DecisionEngine._channel_preamble(name)
decisionengine.framework.engine.DecisionEngine._create_de_server(global_config,
                                                                    channel_config_loader)
    Create the DE server with the passed global configuration and config manager
decisionengine.framework.engine.DecisionEngine._get_de_conf_manager(global_config_dir,
                                                                    channel_config_dir,
                                                                    options)

decisionengine.framework.engine.DecisionEngine._get_global_config(config_file, options)
decisionengine.framework.engine.DecisionEngine._start_de_server(server)
    Start the DE server and listen forever

decisionengine.framework.engine.DecisionEngine.main(args=None)
    If args is None, sys.argv will be used instead If args is a list, it will be used instead of sys.argv (for unit testing)
decisionengine.framework.engine.DecisionEngine.parse_program_options(args=None)
    If args is a list, it will be used instead of sys.argv
```



**decisionengine.framework.engine.Workers module**

**class** decisionengine.framework.engine.Workers.**Worker**(*task\_manager*, *logger\_config*)

Bases: multiprocessing.context.Process

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

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

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

**get\_consumes()**

**get\_produces()**

**get\_state\_name()**

**run()**

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

**wait\_until**(*state*, *timeout=None*)

**wait\_while**(*state*, *timeout=None*)

**class** decisionengine.framework.engine.Workers.**Workers**

Bases: object

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

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

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

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

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

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

**class** Access(*workers*, *lock*)

Bases: object

**\_update\_channel\_states()**

**access()**

**unguarded\_access()**

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

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

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

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

`decisionengine.framework.engine.de_client.execute_command_from_args(argsparsed, de_socket)`  
argsparsed should be from create\_parser in this file

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

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

## decisionengine.framework.engine.de\_query\_tool module

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

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

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

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

Calls the proper function for the arguments passed to de\_query\_tool.

### Parameters

- **argsparsed** (*Namespace*) – Should be from create\_parser in this file.
- **de\_socket** (*ServerProxy*) – RPC Server Proxy.

**Returns** Output of the command.

**Return type** str

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

Main function for de\_query\_tool

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

**Returns** Query result

**Return type** str

## Module contents

**decisionengine.framework.logicengine package**

**Subpackages**

**decisionengine.framework.logicengine.tests package**

**Submodules**

### **decisionengine.framework.logicengine.tests.test\_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_rule_that_does_not_fire(myengine)
    Rules that do not fire do not create entries in the returned actions and newfacts.
decisionengine.framework.logicengine.tests.test_simple_configuration.test_rule_that_fires(myengine)
```

## **Module contents**

### **Submodules**

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

```
class decisionengine.framework.logicengine.BooleanExpression.BooleanExpression(expr)
    Bases: object
    evaluate(d)
        Return the evaluated Boolean value of this expression in the context of the given data 'd'.
exception decisionengine.framework.logicengine.BooleanExpression.LogicError
    Bases: TypeError
decisionengine.framework.logicengine.BooleanExpression.function_name_from_call(callnode)
decisionengine.framework.logicengine.BooleanExpression.maybe_fail_on_error(expr)
```

**decisionengine.framework.logicengine.FactLookup module**

**class** decisionengine.framework.logicengine.FactLookup.**FactLookup**(*fact\_names, rules\_cfg*)

Bases: object

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

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

As an example, consider the following configuration:

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

In the above, the first fact to be evaluated will always be the top-level facts (i.e. those not encapsulated by the 'rules' table). The rules labeled 'publish\_1' and 'publish\_2' both rely on the 'should\_publish' fact in their expressions, and they in turn create their own facts with the same name. FactLookup ensures that 'publish\_1' and 'publish\_2' will both use the evaluated fact from the top-level 'facts' table.

**rule\_for**(*fact\_name*)

Selects rule required to evaluate fact with the supplied name.

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

**Return type** str

**Returns** Rule name

**sorted\_rules**(*rules\_cfg*)

Rules sorted according to rule dependencies.

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

**Return type** list

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

**decisionengine.framework.logicengine.LogicEngine module**

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

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

**\_create\_facts\_dataframe**(*newfacts*)

Convert newfacts dict in format below to dataframe with columns ['rule\_name', 'fact\_name', 'fact\_value']

facts dict format: 'newfacts': {

    '**publish\_glidein\_requests**': { 'allow\_hpc\_new': True, 'allow\_foo': True

    }, 'dummy\_rule': {

        'dummy\_new\_fact': True

```
    }  
}
```

**consumes()**

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

**evaluate(*db*)**

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

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

**evaluate\_facts(*db*)**

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

**Return type** dict

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

**produces()**

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

Assembles logic-engine configuration to unconditionally execute all publishers.

## decisionengine.framework.logicengine.Rule module

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

Bases: object

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

**evaluate(*evaluated\_facts*)**

Evaluates a compiled expression given the supplied facts.

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

**Return type** bool

## decisionengine.framework.logicengine.RuleEngine module

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

Bases: object

Engine responsible for evaluating logic-engine rules.

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

**execute(*evaluated\_facts*)**

Evaluates all rules given the supplied facts.

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

**Return type** tuple

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

## Module contents

### decisionengine.framework.managers package

#### Submodules

#### decisionengine.framework.managers.ChannelManager module

Channel Manager

**class** decisionengine.framework.managers.ChannelManager.**Channel**(*channel\_dict*)

Bases: object

Decision Channel. Instantiates runners according to channel configuration

**class** decisionengine.framework.managers.ChannelManager.**ChannelManager**(*name, dataspace, generation\_id, channel\_dict, global\_config, subscribe\_queue, channel\_subscribed*)

Bases: *decisionengine.framework.managers.ComponentManager.ComponentManager*

Channel Manager: Runs decision cycle for transforms and publishers

**check\_for\_new\_data\_products()**

Check the incoming data product queue, insert into datablock, and update source\_new\_data appropriately

**decision\_cycle()**

Decision cycle to be run periodically (by trigger)

**do\_backup()**

Duplicate current data block and return its copy

**Return type** DataBlock

**register\_with\_sources**(*channel\_id, channel\_name, all\_sources*)

Sends registration information for the sources that the channel is interested in to the SourceSubscription-Manager

**reset\_source\_flags**(*sources*)

Sets self.source\_new\_data to False for the sources which are being used by this decision cycle

**Parameters** **sources** (list) – list of source names that are to be set back to the “not-updated” state

**run()**

Channel Manager main loop

**run\_logic\_engine**(*data\_block=None*)

Run Logic Engine.

**Parameters** **data\_block** (DataBlock) – data block

**run\_publishers**(*actions, facts, data\_block=None*)

Run Publishers in main process.

**Parameters** **data\_block** (DataBlock) – data block

**run\_transform**(*transform, data\_block*)

Run a transform

**Parameters**

- **transform** (*TaskRunner*) – source TaskRunner
- **data\_block** (DataBlock) – data block

**run\_transforms**(*data\_block=None*)

Run transforms. So far in main process.

**Parameters** **data\_block** (DataBlock) – data block**wait\_for\_all**(*events\_done*)

Wait for all sources or transforms to finish

**Parameters** **events\_done** (list) – list of events to wait for**wait\_for\_all\_sources**(*channel\_sources*)

Wait for all sources this channel is interested in to finish their execution

**Parameters** **channel\_sources** (list) – list of sources that need to be polled for completion**wait\_for\_any\_source**(*channel\_sources*)

Waits for any source this channel is interested in to post an updated data block before allowing continuation

**Parameters** **channel\_sources** (list) – list of sources to be watched for updated data**wait\_for\_registration**(*channel\_id*)**class** decisionengine.framework.managers.ChannelManager.**TaskRunner**(*conf\_dict*)

Bases: object

Provides interface to loadable modules and events for synchronization of execution

decisionengine.framework.managers.ChannelManager.**\_make\_runners\_for**(*configs*)**decisionengine.framework.managers.ComponentManager module**

Decision Engine ComponentManager (Base class for ChannelManager and SourceManager)

**class** decisionengine.framework.managers.ComponentManager.**ComponentManager**(*name,*  
*generation\_id,*  
*global\_config*)

Bases: object

Base class for decisionengine components such as Sources and Channels

**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

**get\_loglevel**()**get\_state**()**get\_state\_name**()**get\_state\_value**()



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

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

**take\_offline**(*current\_data\_block*)

offline and stop this component manager

`decisionengine.framework.managers.ComponentManager.create_runner(module_name, class_name,  
parameters)`

Create instance of dynamically loaded module

## decisionengine.framework.managers.SourceManager module

Source Manager

**class** `decisionengine.framework.managers.SourceManager.Source`(*name*, *source\_dict*)

Bases: object

Decision Source. Instantiates Source runners according to the provided Source configuration

**class** `decisionengine.framework.managers.SourceManager.SourceManager`(*name*, *generation\_id*,  
*source\_config*,  
*global\_config*,  
*data\_block\_queue*)

Bases: `decisionengine.framework.managers.ComponentManager.ComponentManager`

Source Manager: Runs decision cycle for transforms and publishers

**data\_block\_send**(*source\_name*, *source\_id*, *data*, *header*)

**run**()

**class** `decisionengine.framework.managers.SourceManager.SourceRunner`(*conf\_dict*)

Bases: object

Provides interface to loadable modules and events for synchronization of execution

## decisionengine.framework.managers.SourceSubscriptionManager module

Source Subscription Manager

**class**

`decisionengine.framework.managers.SourceSubscriptionManager.SourceSubscriptionManager`

Bases: `threading.Thread`

This implements the communication between Sources and Channels

**get\_new\_subscriptions**()

**run**()

Method representing the thread's activity.

You may override this method in a subclass. The standard `run()` method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the `args` and `kwargs` arguments, respectively.

**send\_data\_product\_to\_subscribed**(*new\_source\_info*)

```
class decisionengine.framework.managers.SourceSubscriptionManager.Subscription(channel_manager_id,
                                                                              chan-
                                                                              nel_manager_name,
                                                                              source_names)
```

Bases: object

## Module contents

**decisionengine.framework.modules package**

## Subpackages

**decisionengine.framework.modules.tests package**

## Submodules

**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\_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.Module module

```
class decisionengine.framework.modules.Module.Module(set_of_parameters)
    Bases: object
    A skeleton of a module
    get_data_block()
    get_parameters()
    set_data_block(data_block)
decisionengine.framework.modules.Module.consumes(**kwargs)
decisionengine.framework.modules.Module.produces(**kwargs)
decisionengine.framework.modules.Module.verify_products(producer, data)
```

### decisionengine.framework.modules.Publisher module

```
class decisionengine.framework.modules.Publisher.Parameter(name, type=None, default=None,
                                                         comment=None)
    Bases: object
class decisionengine.framework.modules.Publisher.Publisher(set_of_parameters)
    Bases: decisionengine.framework.modules.Module.Module
    _consumes = {}
    publish(data_block=None)
```

**shutdown()**

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

decisionengine.framework.modules.Publisher.**describe**(cls, program\_options=<class 'decisionengine.framework.modules.describe.ModuleProgramOptions'>)

decisionengine.framework.modules.Publisher.**supports\_config**(\*args)

### decisionengine.framework.modules.Source module

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

Bases: object

**class** decisionengine.framework.modules.Source.**Source**(set\_of\_parameters)

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

**\_produces** = {}

**acquire()**

**post\_create**(global\_config)

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.SourceProxy module

Fill in data from another channel data block

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

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

**\_get\_data**(data\_block, key)

**\_supported\_config** = {'Dataproducts': (<class 'list'>, None, 'List of data products to retrieve.'), 'max\_attempts': (<class 'int'>, 10, 'Number of attempts allowed to fetch products.'), 'retry\_interval': (<class 'int'>, 60, 'Number of seconds to wait between retries.'), 'source\_channel': (<class 'str'>, None, 'Channel from which to retrieve data products.')}

**acquire()**

Overrides Source class method

**post\_create**(global\_config)

## decisionengine.framework.modules.Transform module

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

Bases: object

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

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

```
_consumes = {}
```

```
_produces = {}
```

```
transform()
```

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

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

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

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

## decisionengine.framework.modules.de\_logger module

Logger to use in all modules

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

Reset the logconf.pylogconfig dictionary

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

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

```
decisionengine.framework.modules.de_logger.set_logging(log_level, file_rotate_by,
                                                         rotation_time_unit='D', rotation_interval=1,
                                                         max_backup_count=6,
                                                         max_file_size=200000000,
                                                         log_file_name='/tmp/decision_engine_logs/decisionengine.log')
```

### 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.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.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 convenience wrappers are provided.

Additionally you may use the .lock attribute for *with* block to lock the state during specific operations.

**get()**

This function is a minimally locking check to fetch the state.

**get\_state\_value()**

**has\_value**(state)

**inactive()**

**property lock**

**probably\_running()**

**set**(state)

This function will lock (and possibly block) to ensure a consistent change to the state value.

This function can be blocked using the .lock to force state sync between threads if need be.

**should\_stop()**

**wait\_until**(state, timeout=None)

**wait\_while**(state, timeout=None)

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

Bases: enum.Enum

An enumeration.

**ACTIVE** = 2

**BOOT** = 0

**ERROR** = 7

**IDLE** = 1

```
OFFLINE = 6
SHUTDOWN = 5
SHUTTINGDOWN = 4
STEADY = 3
```

## decisionengine.framework.taskmanager.TaskManager module

Task Manager

```
class decisionengine.framework.taskmanager.TaskManager.Channel(channel_dict, channel_name)
    Bases: object
```

Decision Channel. Instantiates workers according to channel configuration

```
class decisionengine.framework.taskmanager.TaskManager.TaskManager(name, generation_id,
                                                                    channel_dict, global_config)
    Bases: decisionengine.framework.managers.ComponentManager.ComponentManager
```

Task Manager

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

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

    Return type DataBlock
```

```
get_consumes()
```

```
get_produces()
```

```
run()
    Task Manager main loop
```

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

    Parameters data_block (DataBlock) – data block
```

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

    Parameters data_block (DataBlock) – data block
```

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

    Parameters src (Worker) – source Worker
```

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

    Parameters
    • transform (Worker) – source Worker
    • data_block (DataBlock) – data block
```

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



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

**set\_to\_shutdown()**

**start\_sources**(*data\_block=None*)

Start sources, each in a separate thread

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

**wait\_for\_all**(*events\_done*)

Wait for all sources or transforms to finish

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

**wait\_for\_any**(*events\_done*)

Wait for any sources to finish

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

**class** `decisionengine.framework.taskmanager.TaskManager.Worker`(*conf\_dict*, *base\_class*,  
*channel\_name*)

Bases: object

Provides interface to loadable modules and events to synchronise execution

`decisionengine.framework.taskmanager.TaskManager._create_module_instance`(*config\_dict*,  
*base\_class*,  
*channel\_name*)

Create instance of dynamically loaded module

`decisionengine.framework.taskmanager.TaskManager._find_only_one_subclass`(*module*, *base\_class*)  
Search through module looking for only one subclass of the supplied *base\_class*

`decisionengine.framework.taskmanager.TaskManager._make_workers_for`(*configs*, *base\_class*,  
*channel\_name*)

## decisionengine.framework.taskmanager.module\_graph module

Ensure no circularities in produces and consumes.

`decisionengine.framework.taskmanager.module_graph._consumed_products`(\**worker\_lists*)

`decisionengine.framework.taskmanager.module_graph._produced_products`(\**worker\_lists*)

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

Ensures no circularities among data products.

## Module contents

### decisionengine.framework.tests package

#### Submodules

### decisionengine.framework.tests.ABTransform module

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

Bases: `decisionengine.framework.modules.Transform.Transform`

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

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

```
class decisionengine.framework.tests.BATransform.BATransform(module_parameters, *args,
                                                             **kwargs)
    Bases: decisionengine.framework.modules.Transform.Transform
    _consumes = {'A': None}
    _produces = {'B': None}
```

#### decisionengine.framework.tests.ErrorOnAcquire module

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

#### decisionengine.framework.tests.FailingPublisher module

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

#### decisionengine.framework.tests.FailingSourceNOP module

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

#### decisionengine.framework.tests.FailingSourceProxy module

```
class decisionengine.framework.tests.FailingSourceProxy.FailingSourceProxy(config)
    Bases: decisionengine.framework.modules.SourceProxy.SourceProxy
    acquire()
        Overrides Source class method
```

**decisionengine.framework.tests.ModuleProgramOptions module**

```

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

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

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

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

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

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

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

decisionengine.framework.tests.ModuleProgramOptions._expected_config_template(name)
decisionengine.framework.tests.ModuleProgramOptions._expected_config_template_with_comments(name)
decisionengine.framework.tests.ModuleProgramOptions._expected_help(name)
decisionengine.framework.tests.ModuleProgramOptions._expected_source_help(name,
has_sample_config=False)

decisionengine.framework.tests.ModuleProgramOptions._normalize(string)
decisionengine.framework.tests.ModuleProgramOptions._run_as_main(name, *program_options)

```

**decisionengine.framework.tests.PublisherNOP module**

```

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

```

**decisionengine.framework.tests.PublisherWithMissingConsumes module**

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

**decisionengine.framework.tests.SourceAlias module****decisionengine.framework.tests.SourceNOP module**

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

**decisionengine.framework.tests.SourceWithSampleConfigNOP module**

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

**decisionengine.framework.tests.SupportsConfigPublisher module**

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

**decisionengine.framework.tests.TransformNOP module**

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

**decisionengine.framework.tests.TransformWithMissingProducesConsumes module**

**class** `decisionengine.framework.tests.TransformWithMissingProducesConsumes.TransformWithMissingProducesConsumes`  
 Bases: `decisionengine.framework.modules.Transform.Transform`  
**transform**(*data\_block*)

**decisionengine.framework.tests.WorkingSourceProxy module**

**class** `decisionengine.framework.tests.WorkingSourceProxy.WorkingSourceProxy`(*config*)  
 Bases: `decisionengine.framework.modules.SourceProxy.SourceProxy`  
**acquire**()  
 Overrides Source class method

**decisionengine.framework.tests.fixtures module**

defaults for pytest

`decisionengine.framework.tests.fixtures.DE_SERVER`(*conf\_path=None, conf\_override=None, channel\_conf\_path=None, channel\_conf\_override=None, host='127.0.0.1', port=None*)

A DE Server using a private database

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

Fixture factory for PostgreSQL.

**Parameters** **request** – fixture request object

**Returns** postgresql client

`decisionengine.framework.tests.fixtures.PG_DE_DB_WITH_SCHEMA`(*PG\_DE\_DB\_WITHOUT\_SCHEMA*)  
 Load our PG schema into the database via this fixture so pytest knows the limitations on parallel usage of this database scope.

`decisionengine.framework.tests.fixtures.PG_PROG`(*request: \_pytest.fixtures.FixtureRequest, tmpdir\_factory: \_pytest.tmpdir.TmpdirFactory*) →  
`Iterator[pytest_postgresql.executor.PostgreSQLExecutor]`

Process fixture for PostgreSQL.

**Parameters** **request** – fixture request object

**Returns** tcp executor

`decisionengine.framework.tests.fixtures.SQLALCHEMY_PG_WITH_SCHEMA`(*PG\_DE\_DB\_WITHOUT\_SCHEMA*)  
 Get a blank database from `pytest_postgresql`. Then setup the SQLAlchemy style URL with that DB. The SQLAlchemyDS will create the schema as needed.

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

### decisionengine.framework.tests.test\_client\_errors module

Fixture based DE Server tests of the sample config

`decisionengine.framework.tests.test_client_errors.test_client_cannot_wait_on_bad_state(deserver)`  
Verify wait is for a valid state

### decisionengine.framework.tests.test\_client\_server module

Fixture based DE Server for the de-client tests

`decisionengine.framework.tests.test_client_server.test_client_print_product(deserver)`  
`decisionengine.framework.tests.test_client_server.test_client_print_product_columns(deserver)`  
`decisionengine.framework.tests.test_client_server.test_client_print_product_columns_query(deserver)`  
`decisionengine.framework.tests.test_client_server.test_client_print_product_json(deserver)`  
`decisionengine.framework.tests.test_client_server.test_client_print_product_not_real(deserver)`  
`decisionengine.framework.tests.test_client_server.test_client_print_product_not_string(deserver)`  
Make sure the public API is protected against bad values  
`decisionengine.framework.tests.test_client_server.test_client_print_product_query(deserver)`  
`decisionengine.framework.tests.test_client_server.test_client_print_product_types(deserver)`  
`decisionengine.framework.tests.test_client_server.test_client_print_product_vertical(deserver)`  
`decisionengine.framework.tests.test_client_server.test_client_status_msg_to_stdout(deserver)`  
Make sure the actual client console call goes to stdout

### decisionengine.framework.tests.test\_defaults module

Fixture based DE Server tests of the sample config

`decisionengine.framework.tests.test_defaults.test_client_can_get_de_server_show_channel_logger_level(deserver)`  
Verify unknown channel has NOTSET  
`decisionengine.framework.tests.test_defaults.test_client_de_config_is_json(deserver)`  
Verify config can be fetched in json format  
`decisionengine.framework.tests.test_defaults.test_global_channel_log_level_in_config(deserver)`  
Verify `global_channel_log_level` setting exists

### decisionengine.framework.tests.test\_error\_on\_acquire module

`decisionengine.framework.tests.test_error_on_acquire.test_source_only_channel(deserver)`

### **decisionengine.framework.tests.test\_module\_program\_options module**

```
decisionengine.framework.tests.test_module_program_options.test_acquire_for_sources()
decisionengine.framework.tests.test_module_program_options.test_config_templates()
decisionengine.framework.tests.test_module_program_options.test_descriptions()
decisionengine.framework.tests.test_module_program_options.test_help()
decisionengine.framework.tests.test_module_program_options.test_module_alias()
```

### **decisionengine.framework.tests.test\_query\_tool\_server module**

Fixture based DE Server for the de-query-tool tests

```
decisionengine.framework.tests.test_query_tool_server.test_query_tool_csv(deserver)
decisionengine.framework.tests.test_query_tool_server.test_query_tool_default(deserver)
decisionengine.framework.tests.test_query_tool_server.test_query_tool_invalid_product(deserver)
decisionengine.framework.tests.test_query_tool_server.test_query_tool_json(deserver)
decisionengine.framework.tests.test_query_tool_server.test_query_tool_since(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_start_delay(deserver)
    Verify reaper can start with delay
decisionengine.framework.tests.test_reaper.test_client_can_get_de_server_reaper_status(deserver)
    Verify reaper status
decisionengine.framework.tests.test_reaper.test_client_can_get_de_server_reaper_stop(deserver)
    Verify reaper can stop
```

### **decisionengine.framework.tests.test\_restart\_channel module**

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

### **decisionengine.framework.tests.test\_sample\_config module**

Fixture based DE Server tests of the defaults

```
decisionengine.framework.tests.test_sample_config.test_client_can_double_set_de_server_channel_log_level
    Verify set log level to current level isn't an error
decisionengine.framework.tests.test_sample_config.test_client_can_get_de_server_channel_config(deserver)
    Verify config has expected items
decisionengine.framework.tests.test_sample_config.test_client_can_get_de_server_channel_log_level(deserver)
    Verify can fetch log level for a channel
```

`decisionengine.framework.tests.test_sample_config.test_client_can_get_de_server_show_config(deserver)`  
Verify config has expected items

`decisionengine.framework.tests.test_sample_config.test_client_can_get_de_server_show_logger_level(deserver)`  
Verify can fetch log level

`decisionengine.framework.tests.test_sample_config.test_client_can_get_de_server_status(deserver)`  
Verify channel enters stable state

`decisionengine.framework.tests.test_sample_config.test_client_can_get_products(deserver)`  
Verify client can get channel products

`decisionengine.framework.tests.test_sample_config.test_client_can_get_products_no_channels(deserver)`  
Verify client can get channel products even when none are run

`decisionengine.framework.tests.test_sample_config.test_client_can_kill_one_channel(deserver)`  
Verify client can kill a single channel

`decisionengine.framework.tests.test_sample_config.test_client_can_kill_one_channel_force(deserver)`  
Verify client can kill a single channel with force

`decisionengine.framework.tests.test_sample_config.test_client_can_kill_one_channel_timeout(deserver)`  
Verify client can kill a single channel with timeout

`decisionengine.framework.tests.test_sample_config.test_client_can_set_de_server_channel_log_level(deserver)`  
Verify set log level for a channel

`decisionengine.framework.tests.test_sample_config.test_client_can_start_all_channel(deserver)`  
Verify client can start all channel

`decisionengine.framework.tests.test_sample_config.test_client_can_start_one_channel(deserver)`  
Verify client can start a single channel

`decisionengine.framework.tests.test_sample_config.test_client_can_stop_channels(deserver)`  
Verify client can stop channels

`decisionengine.framework.tests.test_sample_config.test_client_can_stop_one_channel(deserver)`  
Verify client can stop a single channel

`decisionengine.framework.tests.test_sample_config.test_client_can_stop_server(deserver)`  
Verify de-client can run `-stop`

`decisionengine.framework.tests.test_sample_config.test_client_cannot_double_start(deserver)`  
Verify client cannot double start channels

`decisionengine.framework.tests.test_sample_config.test_client_get_channel_log_fails_cleanly(deserver)`  
Verify graceful fail on bogus channel

`decisionengine.framework.tests.test_sample_config.test_client_get_non_real_channel(deserver)`  
Verify config for missing channel does what it should

`decisionengine.framework.tests.test_sample_config.test_client_set_channel_log_fails_cleanly(deserver)`  
Verify graceful fail on bogus channel

`decisionengine.framework.tests.test_sample_config.test_client_start_non_real_channel(deserver)`  
Verify start for missing channel does what it should

`decisionengine.framework.tests.test_sample_config.test_client_stop_non_real_channel(deserver)`  
Verify stop for missing channel does what it should

`decisionengine.framework.tests.test_sample_config.test_client_wait_timeout_works(deserver)`  
Verify channel enters stable state and timeout works too



### decisionengine.framework.tests.test\_source\_proxy module

Fixture based tests of the SourceProxy module.

`decisionengine.framework.tests.test_source_proxy.test_stop_failing_source_proxy(deserver_fail)`

`decisionengine.framework.tests.test_source_proxy.test_working_source_proxy(deserver)`

### 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_mock_data_block)`  
*cap*

Verify client can get channel products even when none are run

### decisionengine.framework.tests.test\_start\_with\_no\_channels module

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

`decisionengine.framework.tests.test_start_with_no_channels.deserver_mock_data_block(mock_data_block)`

`decisionengine.framework.tests.test_start_with_no_channels.test_start_from_nothing(deserver_mock_data_block)`

## Module contents

### decisionengine.framework.util package

#### Submodules

### decisionengine.framework.util.fs module

`decisionengine.framework.util.fs.files_with_extensions(dir_path, *extensions)`

Return all files in `dir_path` that match the provided extensions.

If no extensions are given, then all files in `dir_path` are returned.

Results are sorted by channel name to ensure stable output.

### decisionengine.framework.util.metrics module

```
class decisionengine.framework.util.metrics.Counter(name, documentation, labelnames=(),
                                                    namespace="", subsystem="", unit="", registry=<prometheus_client.registry.CollectorRegistry
                                                    object>, _labelvalues=None)
```

Bases: prometheus\_client.metrics.Counter

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

Bases: prometheus\_client.metrics.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)
```

```
class decisionengine.framework.util.metrics.Histogram(name, documentation, labelnames=(),
                                                       namespace="", subsystem="", unit="", registry=<prometheus_client.registry.CollectorRegistry
                                                       object>, _labelvalues=None, buckets=(0.005,
                                                       0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75,
                                                       1.0, 2.5, 5.0, 7.5, 10.0, inf))
```

Bases: prometheus\_client.metrics.Histogram

```
class decisionengine.framework.util.metrics.Summary(name, documentation, labelnames=(),
                                                    namespace="", subsystem="", unit="", registry=<prometheus_client.registry.CollectorRegistry
                                                    object>, _labelvalues=None)
```

Bases: prometheus\_client.metrics.Summary

### 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.singleton module

```
class decisionengine.framework.util.singleton.ScopedSingleton
    Bases: decisionengine.framework.util.singleton.Singleton
    Singleton pattern using Metaclass with weak refs
    _instances = <WeakValueDictionary>

class decisionengine.framework.util.singleton.ScopedSingletonABC(name, bases, namespace,
                                                                **kwargs)
    Bases: abc.ABCMeta, decisionengine.framework.util.singleton.ScopedSingleton

class decisionengine.framework.util.singleton.Singleton
    Bases: type
    Singleton pattern using Metaclass with strong refs
    _instances = {}

class decisionengine.framework.util.singleton.SingletonABC(name, bases, namespace, **kwargs)
    Bases: abc.ABCMeta, decisionengine.framework.util.singleton.Singleton
```

## decisionengine.framework.util.sockets module

```
decisionengine.framework.util.sockets.get_random_port()
```

## decisionengine.framework.util.subclasses module

```
decisionengine.framework.util.subclasses._derived_class(cls, base_class)
    Only matches subclasses that are not equal to the base class.

decisionengine.framework.util.subclasses.all_subclasses(module, base_class)
    Return all of a module's subclasses of the given base class.
```

## Module contents

### Submodules

## decisionengine.framework.about module

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

## decisionengine.framework.version module

### Module contents

## decisionengine.tests package

### Submodules

### decisionengine.tests.test\_framework\_package module

Make sure decisionengine.framework is a valid python package

`decisionengine.tests.test_framework_package.test_can_import()`

### Module contents

### Module contents

## 4.2 Indices and tables

- [genindex](#)
- [modindex](#)
- [search](#)

## INDICES AND TABLES

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



## PYTHON MODULE INDEX

### d

[decisionengine](#), 96  
[decisionengine.framework](#), 95  
[decisionengine.framework.about](#), 95  
[decisionengine.framework.config](#), 37  
[decisionengine.framework.config.ChannelConfigHandler](#), 64  
[decisionengine.framework.config.policies](#), 36  
[decisionengine.framework.config.tests](#), 35  
[decisionengine.framework.config.tests.test\\_config](#), 33  
[decisionengine.framework.config.tests.test\\_de\\_std](#), 34  
[decisionengine.framework.config.tests.test\\_policies](#), 34  
[decisionengine.framework.config.ValidConfig](#), 35  
[decisionengine.framework.dataspace](#), 65  
[decisionengine.framework.dataspace.datablock](#), 59  
[decisionengine.framework.dataspace.datasource](#), 61  
[decisionengine.framework.dataspace.datasources](#), 55  
[decisionengine.framework.dataspace.datasources.null](#), 50  
[decisionengine.framework.dataspace.datasources.postgresql](#), 52  
[decisionengine.framework.dataspace.datasources.sqlalchemy\\_us](#), 44  
[decisionengine.framework.dataspace.datasources.sqlalchemy\\_us.datasource\\_api](#), 37  
[decisionengine.framework.dataspace.datasources.sqlalchemy\\_us.db\\_schema](#), 41  
[decisionengine.framework.dataspace.datasources.sqlalchemy\\_us.utils](#), 43  
[decisionengine.framework.dataspace.datasources.tests](#), 50  
[decisionengine.framework.dataspace.datasources.tests.fixtures](#), 47  
[decisionengine.framework.dataspace.datasources.tests.test\\_datasource\\_api](#), 48  
[decisionengine.framework.dataspace.datasources.tests.test\\_datasources](#), 50  
[decisionengine.framework.dataspace.dataspace](#), 63  
[decisionengine.framework.dataspace.maintain](#), 64  
[decisionengine.framework.dataspace.tests](#), 59  
[decisionengine.framework.dataspace.tests.fixtures](#), 55  
[decisionengine.framework.dataspace.tests.test\\_datablock](#), 57  
[decisionengine.framework.dataspace.tests.test\\_datablock\\_zl](#), 57  
[decisionengine.framework.dataspace.tests.test\\_datasource](#), 57  
[decisionengine.framework.dataspace.tests.test\\_dataspace](#), 58  
[decisionengine.framework.dataspace.tests.test\\_Reaper](#), 56  
[decisionengine.framework.engine](#), 70  
[decisionengine.framework.engine.de\\_client](#), 70  
[decisionengine.framework.engine.de\\_query\\_tool](#), 70  
[decisionengine.framework.engine.DecisionEngine](#), 66  
[decisionengine.framework.engine.tests](#), 66  
[decisionengine.framework.engine.tests.fixtures](#), 65  
[decisionengine.framework.engine.tests.test\\_client\\_only](#), 66  
[decisionengine.framework.engine.tests.test\\_query\\_tool\\_only](#), 66  
[decisionengine.framework.engine.tests.test\\_startup](#), 66  
[decisionengine.framework.engine.Workers](#), 69  
[decisionengine.framework.logicengine](#), 75  
[decisionengine.framework.logicengine.BooleanExpression](#), 72  
[decisionengine.framework.logicengine.FactLookup](#), 73  
[decisionengine.framework.logicengine.LogicEngine](#), 73

decisionengine.framework.logicengine.Rule,	74	78
decisionengine.framework.logicengine.RuleEngine,	74	78
decisionengine.framework.logicengine.tests,	72	79
decisionengine.framework.logicengine.tests.test_cassandra,	71	81
decisionengine.framework.logicengine.tests.test_decisionengine,	71	82
decisionengine.framework.logicengine.tests.test_dispatcher_interfaces,	71	85
decisionengine.framework.logicengine.tests.test_facts,	71	85
decisionengine.framework.logicengine.tests.test_fail_on_error,	71	86
decisionengine.framework.logicengine.tests.test_pandasfact,	72	84
decisionengine.framework.logicengine.tests.test_rulewithinegativefact,	72	85
decisionengine.framework.logicengine.tests.test_simpleconfigframework,	72	86
decisionengine.framework.managers,	75	86
decisionengine.framework.managers.ChannelManager,	75	86
decisionengine.framework.managers.ComponentManager,	76	86
decisionengine.framework.managers.SourceManager,	77	86
decisionengine.framework.managers.SourceSubscriptionManager,	77	86
decisionengine.framework.modules,	83	87
decisionengine.framework.modules.de_logger,	81	87
decisionengine.framework.modules.describe,	82	87
decisionengine.framework.modules.logging_config_dirs,	82	88
decisionengine.framework.modules.Module,	79	88
decisionengine.framework.modules.print_description,	82	88
decisionengine.framework.modules.Publisher,	79	88
decisionengine.framework.modules.Source,	80	88
decisionengine.framework.modules.SourceProxy,	80	88
decisionengine.framework.modules.tests,	79	90
decisionengine.framework.modules.tests.test_dispatcher,	78	90
decisionengine.framework.modules.tests.test_Module,	78	90
decisionengine.framework.modules.tests.test_module_dispatcher,	79	90
decisionengine.framework.modules.tests.test_Publisher,	78	91
decisionengine.framework.modules.tests.test_Source,	78	91
decisionengine.framework.modules.tests.test_Transform,	78	91
decisionengine.framework.modules.tests.test_translate_product,	78	91
decisionengine.framework.modules.Transform,	78	91
decisionengine.framework.modules.translate_product_name,	78	91
decisionengine.framework.taskmanager,	85	93
decisionengine.framework.taskmanager.module_graph,	85	93
decisionengine.framework.taskmanager.ProcessingState,	85	93
decisionengine.framework.taskmanager.TaskManager,	85	93
decisionengine.framework.tests,	93	93
decisionengine.framework.tests.ABTransform,	93	93
decisionengine.framework.tests.BATransform,	93	93
decisionengine.framework.tests.ErrorOnAcquire,	93	93
decisionengine.framework.tests.FailingPublisher,	93	93
decisionengine.framework.tests.FailingSourceNOP,	93	93
decisionengine.framework.tests.FailingSourceProxy,	93	93
decisionengine.framework.tests.fixtures,	93	93
decisionengine.framework.tests.ModuleProgramOptions,	93	93
decisionengine.framework.tests.PublisherNOP,	93	93
decisionengine.framework.tests.PublisherWithMissingConsumer,	93	93
decisionengine.framework.tests.SourceAlias,	93	93
decisionengine.framework.tests.SourceNOP,	93	93
decisionengine.framework.tests.SourceWithSampleConfigNOP,	93	93
decisionengine.framework.tests.SupportsConfigPublisher,	93	93
decisionengine.framework.tests.test_client_errors,	93	93
decisionengine.framework.tests.test_client_server,	93	93
decisionengine.framework.tests.test_defaults,	93	93
decisionengine.framework.tests.test_error_on_acquire,	93	93
decisionengine.framework.tests.test_module_program_options,	93	93
decisionengine.framework.tests.test_query_tool_server,	93	93



91  
decisionengine.framework.tests.test\_reaper,  
91  
decisionengine.framework.tests.test\_restart\_channel,  
91  
decisionengine.framework.tests.test\_sample\_config,  
91  
decisionengine.framework.tests.test\_source\_proxy,  
93  
decisionengine.framework.tests.test\_start\_with\_bad\_channels,  
93  
decisionengine.framework.tests.test\_start\_with\_no\_channels,  
93  
decisionengine.framework.tests.TransformNOP,  
88  
decisionengine.framework.tests.TransformWithMissingProducesConsumes,  
89  
decisionengine.framework.tests.WorkingSourceProxy,  
89  
decisionengine.framework.util, 95  
decisionengine.framework.util.fs, 93  
decisionengine.framework.util.metrics, 94  
decisionengine.framework.util.reaper, 94  
decisionengine.framework.util.singleton, 95  
decisionengine.framework.util.sockets, 95  
decisionengine.framework.util.subclasses, 95  
decisionengine.framework.version, 95  
decisionengine.tests, 96  
decisionengine.tests.test\_framework\_package,  
96



# INDEX

## Symbols

\_DEFAULT\_MULTIPROC\_MODE (decisionengine.framework.util.metrics.Gauge attribute), 94  
 \_\_determine\_multiprocess\_mode\_existence() (decisionengine.framework.util.metrics.Gauge method), 94  
 \_\_insert() (decisionengine.framework.dataspace.datablock.Datablock method), 59  
 \_\_query() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 52  
 \_\_update() (decisionengine.framework.dataspace.datablock.Datablock method), 59  
 \_abc\_impl (decisionengine.framework.config.ValidConfig.ValidConfig attribute), 35  
 \_abc\_impl (decisionengine.framework.dataspace.datablock.Datablock.Header attribute), 60  
 \_abc\_impl (decisionengine.framework.dataspace.datablock.Datablock.Metadata attribute), 60  
 \_abc\_impl (decisionengine.framework.dataspace.datasources.DataSource attribute), 61  
 \_abc\_impl (decisionengine.framework.dataspace.datasources.null.NullDataSource attribute), 50  
 \_abc\_impl (decisionengine.framework.dataspace.datasources.postgresql.Postgresql attribute), 52  
 \_abc\_impl (decisionengine.framework.dataspace.datasources.sqlalchemy\_us.SQLAlchemyDS attribute), 44  
 \_abc\_impl (decisionengine.framework.dataspace.datasources.sqlalchemy\_us.datasource\_api.SQLAlchemyDS attribute), 37  
 \_channel\_config\_dir() (in module decisionengine.framework.config.tests.test\_config), 33  
 \_channel\_preamble() (in module decisionengine.framework.engine.DecisionEngine), 68  
 \_check\_keys() (in module decisionengine.framework.config.ChannelConfigHandler), 35  
 \_check\_override() (in module decisionengine.framework.engine.tests.test\_startup), 66  
 \_config\_from\_file() (in module decisionengine.framework.config.ValidConfig), 36  
 \_consumed\_products() (in module decisionengine.framework.taskmanager.module\_graph), 85  
 \_consumes (decisionengine.framework.modules.Publisher.Publisher attribute), 79  
 \_consumes (decisionengine.framework.modules.Transform.Transform attribute), 81  
 \_consumes (decisionengine.framework.tests.ABTransform.ABTransform attribute), 85  
 \_consumes (decisionengine.framework.tests.BATransform.BATransform attribute), 86  
 \_consumes (decisionengine.framework.tests.FailingPublisher.FailingPublisher attribute), 86  
 \_consumes (decisionengine.framework.tests.PublisherNOP.PublisherNOP attribute), 87  
 \_consumes (decisionengine.framework.tests.TransformNOP.TransformNOP attribute), 88  
 \_consumes\_not\_subset() (in module decisionengine.framework.tests.test\_start\_with\_bad\_channels), 93  
 \_create\_de\_server() (in module decisionengine.framework.engine.DecisionEngine), 68  
 create\_facts\_dataframe() (decisionengine.framework.logicengine.LogicEngine.LogicEngine method), 73  
 \_create\_module\_instance() (in module decisionengine.framework.taskmanager.TaskManager), 85  
 \_dataframe\_to\_column\_names() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 66  
 \_dataframe\_to\_csv() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 66  
 \_dataframe\_to\_json() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 67  
 \_dataframe\_to\_table() (decisionengine.framework.engine.DecisionEngine.DecisionEngine

`method`), 67  
`_dataframe_to_vertical_tables()` (decisionengine.framework.engine.DecisionEngine.method), 35  
`method`), 67  
`_delete()` (decisionengine.framework.dataspace.datasources.postgresql.Postgresql.method), 52  
`_derived_class()` (in module decisionengine.framework.util.subclasses), 95  
`_dispatch()` (decisionengine.framework.engine.DecisionEngine.method), 67  
`_expected_acquire_result()` (in module decisionengine.framework.tests.ModuleProgramOptions), 87  
`_expected_circularity()` (in module decisionengine.framework.tests.test\_start\_with\_bad\_channels), 93  
`_expected_config_template()` (in module decisionengine.framework.tests.ModuleProgramOptions), 87  
`_expected_config_template_with_comments()` (in module decisionengine.framework.tests.ModuleProgramOptions), 87  
`_expected_help()` (in module decisionengine.framework.tests.ModuleProgramOptions), 87  
`_expected_source_help()` (in module decisionengine.framework.tests.ModuleProgramOptions), 87  
`_find_only_one_subclass()` (in module decisionengine.framework.taskmanager.TaskManager), 85  
`_get_data()` (decisionengine.framework.modules.SourceProxy.SourceProxy.method), 80  
`_get_de_conf_manager()` (in module decisionengine.framework.engine.DecisionEngine), 68  
`_get_global_config()` (in module decisionengine.framework.engine.DecisionEngine), 68  
`_global_config_file()` (in module decisionengine.framework.config.tests.test\_config), 33  
`_insert()` (decisionengine.framework.dataspace.datasources.postgresql.Postgresql.method), 52  
`_insert_returning_result()` (decisionengine.framework.dataspace.datasources.postgresql.Postgresql.method), 52  
`_instances` (decisionengine.framework.util.singleton.ScopedSingleton.attribute), 95  
`_instances` (decisionengine.framework.util.singleton.Singleton.attribute), 95  
`_load_channel()` (decisionengine.framework.config.ChannelConfigHandler.ChannelConfigHandler.method), 35  
`_make_de_logger()` (in module decisionengine.framework.config.ChannelConfigHandler), 35  
`_make_runners_for()` (in module decisionengine.framework.managers.ChannelManager), 76  
`_make_workers_for()` (in module decisionengine.framework.taskmanager.TaskManager), 85  
`_missing_consumes()` (in module decisionengine.framework.tests.test\_start\_with\_bad\_channels), 93  
`_missing_produces()` (in module decisionengine.framework.tests.test\_start\_with\_bad\_channels), 93  
`_normalize()` (in module decisionengine.framework.tests.ModuleProgramOptions), 87  
`_par_default()` (in module decisionengine.framework.modules.describe), 82  
`_par_type()` (in module decisionengine.framework.modules.describe), 82  
`_print_comment()` (in module decisionengine.framework.modules.print\_description), 82  
`_print_type()` (in module decisionengine.framework.modules.print\_description), 82  
`_print_value()` (in module decisionengine.framework.modules.print\_description), 82  
`_produced_products()` (in module decisionengine.framework.taskmanager.module\_graph), 85  
`_produces` (decisionengine.framework.modules.Source.Source.attribute), 80  
`_produces` (decisionengine.framework.modules.Transform.Transform.attribute), 81  
`_produces` (decisionengine.framework.tests.ABTransform.ABTransform.attribute), 86  
`_produces` (decisionengine.framework.tests.BATransform.BATransform.attribute), 86  
`_produces` (decisionengine.framework.tests.ErrorOnAcquire.ErrorOnAcquire.attribute), 86  
`_produces` (decisionengine.framework.tests.SourceNOP.SourceNOP.attribute), 88  
`_produces` (decisionengine.framework.tests.SourceWithSampleConfigNOP.SourceWithSampleConfigNOP.attribute), 88  
`_produces` (decisionengine.framework.tests.TransformNOP.TransformNOP.attribute), 88  
`_reaper_loop()` (decisionengine.framework.taskmanager.TaskManager.method), 85

nengine.framework.dataspace.maintain.Reaper  
 method), 64  
 \_remove() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql  
 method), 52  
 \_reset\_config() (in module decisio-  
 nengine.framework.modules.de\_logger),  
 81  
 \_run\_as\_main() (in module decisio-  
 nengine.framework.tests.ModuleProgramOptions),  
 87  
 \_sa\_class\_manager (decisio-  
 nengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema.Dataproduct  
 attribute), 41  
 \_sa\_class\_manager (decisio-  
 nengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema.Header  
 attribute), 41  
 \_sa\_class\_manager (decisio-  
 nengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema.Metadata  
 attribute), 42  
 \_sa\_class\_manager (decisio-  
 nengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema.Schema  
 attribute), 43  
 \_sa\_class\_manager (decisio-  
 nengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema.Taskmanager  
 attribute), 43  
 \_sa\_registry (decisio-  
 nengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema.Base  
 attribute), 41  
 \_select() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql  
 method), 52  
 \_select\_dictresult() (decisio-  
 nengine.framework.dataspace.datasources.postgresql.Postgresql  
 method), 52  
 \_select\_getresult() (decisio-  
 nengine.framework.dataspace.datasources.postgresql.Postgresql  
 method), 52  
 \_select\_tuple() (decisio-  
 nengine.framework.dataspace.datasources.postgresql.Postgresql  
 method), 52  
 \_setitem() (decisionengine.framework.dataspace.datablock.DataBlock  
 method), 59  
 \_spec\_from\_file\_name() (in module decisio-  
 nengine.framework.modules.print\_description),  
 82  
 \_start\_de\_server() (in module decisio-  
 nengine.framework.engine.DecisionEngine),  
 68  
 \_supported\_config (decisio-  
 nengine.framework.modules.SourceProxy.SourceProxy  
 attribute), 80  
 \_supported\_config (decisio-  
 nengine.framework.tests.SourceWithSampleConfigNOP.SourceWithSampleConfigNOP  
 attribute), 88  
 \_supported\_config (decisio-

nengine.framework.tests.SupportsConfigPublisher.SupportsConfig  
 attribute), 88  
 \_update() (decisionengine.framework.dataspace.datasources.postgresql.P  
 method), 52  
 \_update\_channel\_states() (decisio-  
 nengine.framework.engine.Workers.Workers  
 method), 69  
 \_update\_returning\_result() (decisio-  
 nengine.framework.dataspace.datasources.postgresql.Postgresql  
 method), 52  
 Any\_ds.db\_schema.Dataproduct  
 ABTransform (class in decisio-  
 nengine.framework.tests.ABTransform), 85  
 access() (decisionengine.framework.engine.Workers.Workers  
 method), 69  
 acquire() (decisionengine.framework.modules.Source.Source  
 method), 80  
 acquire() (decisionengine.framework.modules.SourceProxy.SourceProxy  
 method), 80  
 acquire() (decisionengine.framework.tests.ErrorOnAcquire.ErrorOnAcqu  
 method), 86  
 acquire() (decisionengine.framework.tests.FailingSourceProxy.FailingSou  
 method), 88  
 acquire() (decisionengine.framework.tests.SourceNOP.SourceNOP  
 method), 88  
 acquire() (decisionengine.framework.tests.SourceWithSampleConfigNOP.  
 method), 88  
 acquire() (decisionengine.framework.tests.WorkingSourceProxy.Working  
 method), 89  
 AcquireWithConfig (class in decisio-  
 nengine.framework.tests.ModuleProgramOptions),  
 87  
 AcquireWithSampleConfig (class in decisio-  
 nengine.framework.tests.ModuleProgramOptions),  
 87  
 ACTIVE (decisionengine.framework.taskmanager.ProcessingState.State  
 attribute), 83  
 add\_engine\_pidguard() (in module decisio-  
 nengine.framework.dataspace.datasources.sqlalchemy\_ds.utils),  
 43  
 all\_subclasses() (in module decisio-  
 nengine.framework.util.subclasses), 95

**B**  
 Base (class in decisio-  
 nengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_sch  
 41  
 BATransform (class in decisio-  
 nengine.framework.tests.BATransform), 86  
 block\_until() (decisio-  
 nengine.framework.engine.DecisionEngine.DecisionEngine  
 method), 67

block\_while() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 67

BooleanExpression (class in decisionengine.framework.logicengine.BooleanExpression), 72

BOOT (decisionengine.framework.taskmanager.ProcessingState attribute), 83

**C**

Channel (class in decisionengine.framework.managers.ChannelManager), 75

Channel (class in decisionengine.framework.taskmanager.TaskManager), 84

channel\_config\_dir() (in module decisionengine.framework.config.policies), 36

ChannelConfigHandler (class in decisionengine.framework.config.ChannelConfigHandler), 35

ChannelManager (class in decisionengine.framework.managers.ChannelManager), 75

check\_for\_new\_data\_products() (decisionengine.framework.managers.ChannelManager.ChannelManager method), 75

Clean (decisionengine.framework.engine.DecisionEngine.StopState attribute), 68

clone\_model() (in module decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.utils), 43

close() (decisionengine.framework.dataspace.datasource.DataSource method), 61

close() (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 50

close() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 52

close() (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.SQLAlchemyDS method), 37

close() (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.SQLAlchemyDS method), 44

close() (decisionengine.framework.dataspace.dataspace.DataSpace method), 63

ComponentManager (class in decisionengine.framework.managers.ComponentManager), 76

compress() (in module decisionengine.framework.dataspace.datablock), 61

config() (in module decisionengine.framework.config.tests.test\_de\_std), 34

config() (in module decisionengine.framework.dataspace.tests.test\_Reaper), 56

ConfigTemplate (class in decisionengine.framework.tests.ModuleProgramOptions), 87

connect() (decisionengine.framework.dataspace.datasource.DataSource method), 61

connect() (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 50

connect() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 52

connect() (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.SQLAlchemyDS method), 37

connect() (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.SQLAlchemyDS method), 44

console\_scripts\_main() (in module decisionengine.framework.engine.de\_client), 70

console\_scripts\_main() (in module decisionengine.framework.engine.de\_query\_tool), 70

consumes() (decisionengine.framework.logicengine.LogicEngine.LogicEngine method), 74

consumes() (in module decisionengine.framework.modules.Module), 79

consumes() (in module decisionengine.framework.modules.Publisher), 80

consumes() (in module decisionengine.framework.modules.Transform), 81

Counter (class in decisionengine.framework.util.metrics), 94

create\_parser() (in module decisionengine.framework.engine.de\_client), 70

create\_parser() (in module decisionengine.framework.engine.de\_query\_tool), 70

create\_runner() (in module decisionengine.framework.managers.ComponentManager), 77

create\_tables() (decisionengine.framework.dataspace.datasource.DataSource method), 61

create\_tables() (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 50

create\_tables() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 52

create\_tables() (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.datasource.SQLAlchemyDS method), 37

create\_tables() (decisionengine.framework.dataspace.dataspace.DataSpace method), 61



engine.framework.dataspace.datasources.sqlalchemyDS (class in decisionengine.framework.engine.DecisionEngine),  
 method), 44  
 create\_time (decisionengine.framework.dataspace.datasources.sqlalchemyDS attribute), 42  
 creator (decisionengine.framework.dataspace.datasources.sqlalchemyDS attribute), 42  
**D**  
 data\_block\_put() (decisionengine.framework.managers.ComponentManager.ComponentManager method), 76  
 data\_block\_send() (decisionengine.framework.managers.SourceManager.SourceManager method), 77  
 DataBlock (class in decisionengine.framework.dataspace.datablock), 59  
 Dataproduct (class in decisionengine.framework.dataspace.datasources.sqlalchemyDS), 41  
 dataproduct\_table (decisionengine.framework.dataspace.datasource.DataSource attribute), 61  
 DataSource (class in decisionengine.framework.dataspace.datasource), 61  
 datasource() (in module decisionengine.framework.dataspace.datasources.tests.fixtures), 48  
 datasource() (in module decisionengine.framework.dataspace.tests.fixtures), 55  
 DataSpace (class in decisionengine.framework.dataspace.dataspace), 63  
 dataspace() (in module decisionengine.framework.dataspace.tests.fixtures), 55  
 DataSpaceConfigurationError, 64  
 DataSpaceConnectionError, 64  
 DataSpaceError, 64  
 DataSpaceExistsError, 64  
 datestamp (decisionengine.framework.dataspace.datasources.sqlalchemyDS.us\_db\_schema.TaskManager attribute), 43  
 decision\_cycle() (decisionengine.framework.managers.ChannelManager.ChannelManager method), 75  
 decision\_cycle() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 84  
 decisionengine module, 96  
 decisionengine.framework.config module, 37  
 decisionengine.framework.config.ChannelConfigHandler module, 35  
 decisionengine.framework.config.policies module, 36  
 decisionengine.framework.config.tests module, 35  
 decisionengine.framework.config.tests.test\_config module, 33  
 decisionengine.framework.config.tests.test\_de\_std module, 34  
 decisionengine.framework.config.tests.test\_policies module, 34  
 decisionengine.framework.config.ValidConfig module, 35  
 decisionengine.framework.dataspace module, 65  
 decisionengine.framework.dataspace.datablock module, 59  
 decisionengine.framework.dataspace.datasource module, 61  
 decisionengine.framework.dataspace.datasources module, 55  
 decisionengine.framework.dataspace.datasources.null module, 50  
 decisionengine.framework.dataspace.datasources.postgresql module, 52  
 decisionengine.framework.dataspace.datasources.sqlalchemy module, 44  
 decisionengine.framework.dataspace.datasources.sqlalchemy module, 37  
 decisionengine.framework.dataspace.datasources.sqlalchemy module, 41  
 decisionengine.framework.dataspace.datasources.sqlalchemy module, 43  
 decisionengine.framework.dataspace.datasources.tests module, 50  
 decisionengine.framework.dataspace.datasources.tests.fixture module, 47  
 decisionengine.framework.dataspace.datasources.tests.test module, 48  
 decisionengine.framework.dataspace.datasources.tests.test module, 50  
 decisionengine.framework.dataspace.dataspace module, 63  
 decisionengine.framework.dataspace.maintain

module, 64	module, 71
decisionengine.framework.dataspace.tests	decisionengine.framework.logicengine.tests.test_facts
module, 59	module, 71
decisionengine.framework.dataspace.tests.fixture	decisionengine.framework.logicengine.tests.test_fail_on_error
module, 55	module, 71
decisionengine.framework.dataspace.tests.test_decisionengine	decisionengine.framework.logicengine.tests.test_pandas_fact
module, 57	module, 72
decisionengine.framework.dataspace.tests.test_decisionengine	decisionengine.framework.logicengine.tests.test_rule_with_data
module, 57	module, 72
decisionengine.framework.dataspace.tests.test_decisionengine	decisionengine.framework.logicengine.tests.test_simple_correlation
module, 57	module, 72
decisionengine.framework.dataspace.tests.test_dataspace	decisionengine.framework.managers
module, 58	module, 78
decisionengine.framework.dataspace.tests.test_decisionengine	decisionengine.framework.managers.ChannelManager
module, 56	module, 75
decisionengine.framework.engine	decisionengine.framework.managers.ComponentManager
module, 70	module, 76
decisionengine.framework.engine.de_client	decisionengine.framework.managers.SourceManager
module, 70	module, 77
decisionengine.framework.engine.de_query_tool	decisionengine.framework.managers.SourceSubscriptionManager
module, 70	module, 77
decisionengine.framework.engine.DecisionEngine	decisionengine.framework.modules
module, 66	module, 83
decisionengine.framework.engine.tests	decisionengine.framework.modules.de_logger
module, 66	module, 81
decisionengine.framework.engine.tests.fixture	decisionengine.framework.modules.describe
module, 65	module, 82
decisionengine.framework.engine.tests.test_client_side_engine	decisionengine.framework.modules.logging_configDict
module, 66	module, 82
decisionengine.framework.engine.tests.test_query_side_engine	decisionengine.framework.modules.Module
module, 66	module, 79
decisionengine.framework.engine.tests.test_startup	decisionengine.framework.modules.print_description
module, 66	module, 82
decisionengine.framework.engine.Workers	decisionengine.framework.modules.Publisher
module, 69	module, 79
decisionengine.framework.logicengine	decisionengine.framework.modules.Source
module, 75	module, 80
decisionengine.framework.logicengine.BooleanExpression	decisionengine.framework.modules.SourceProxy
module, 72	module, 80
decisionengine.framework.logicengine.FactLookup	decisionengine.framework.modules.tests
module, 73	module, 79
decisionengine.framework.logicengine.LogicEngine	decisionengine.framework.modules.tests.test_de_logger
module, 73	module, 78
decisionengine.framework.logicengine.Rule	decisionengine.framework.modules.tests.test_Module
module, 74	module, 78
decisionengine.framework.logicengine.RuleEngine	decisionengine.framework.modules.tests.test_module_decorator
module, 74	module, 79
decisionengine.framework.logicengine.tests	decisionengine.framework.modules.tests.test_Publisher
module, 72	module, 78
decisionengine.framework.logicengine.tests.test_decisionengine	decisionengine.framework.modules.tests.test_Source
module, 71	module, 78
decisionengine.framework.logicengine.tests.test_decisionengine	decisionengine.framework.modules.tests.test_Transform
module, 71	module, 78
decisionengine.framework.logicengine.tests.test_decisionengine	decisionengine.framework.modules.tests.test_translate_procedure
module, 71	module, 78



module, 79	module, 91
decisionengine.framework.modules.Transform	decisionengine.framework.tests.test_reaper
module, 81	module, 91
decisionengine.framework.modules.translate_producer	decisionengine.framework.tests.test_restart_channel
module, 82	module, 91
decisionengine.framework.taskmanager	decisionengine.framework.tests.test_sample_config
module, 85	module, 91
decisionengine.framework.taskmanager.module_graph	decisionengine.framework.tests.test_source_proxy
module, 85	module, 93
decisionengine.framework.taskmanager.ProcessingStation	decisionengine.framework.tests.test_start_with_bad_channel
module, 83	module, 93
decisionengine.framework.taskmanager.TaskManager	decisionengine.framework.tests.test_start_with_no_channels
module, 84	module, 93
decisionengine.framework.tests	decisionengine.framework.tests.TransformNOP
module, 93	module, 88
decisionengine.framework.tests.ABTransform	decisionengine.framework.tests.TransformWithMissingProduce
module, 85	module, 89
decisionengine.framework.tests.BATransform	decisionengine.framework.tests.WorkingSourceProxy
module, 86	module, 89
decisionengine.framework.tests.ErrorOnAcquire	decisionengine.framework.util
module, 86	module, 95
decisionengine.framework.tests.FailingPublisher	decisionengine.framework.util.fs
module, 86	module, 93
decisionengine.framework.tests.FailingSourceNOP	decisionengine.framework.util.metrics
module, 86	module, 94
decisionengine.framework.tests.FailingSourceProxy	decisionengine.framework.util.reaper
module, 86	module, 94
decisionengine.framework.tests.fixtures	decisionengine.framework.util.singleton
module, 89	module, 95
decisionengine.framework.tests.ModuleProgramOptions	decisionengine.framework.util.sockets
module, 87	module, 95
decisionengine.framework.tests.PublisherNOP	decisionengine.framework.util.subclasses
module, 87	module, 95
decisionengine.framework.tests.PublisherWithMissingConsumers	decisionengine.framework.version
module, 88	module, 95
decisionengine.framework.tests.SourceAlias	decisionengine.tests
module, 88	module, 96
decisionengine.framework.tests.SourceNOP	decisionengine.tests.test_framework_package
module, 88	module, 96
decisionengine.framework.tests.SourceWithSampleConfigNOP	decisionengine.tests.test_framework_package
module, 88	module, 96
decisionengine.framework.tests.SupportsConfigPublisher	decisionengine.tests.test_framework_package
module, 88	module, 96
decisionengine.framework.tests.test_client_errors	decisionengine.tests.test_framework_package
module, 90	module, 96
decisionengine.framework.tests.test_client_server	decisionengine.tests.test_framework_package
module, 90	module, 96
decisionengine.framework.tests.test_defaults	decisionengine.tests.test_framework_package
module, 90	module, 96
decisionengine.framework.tests.test_error_on_acquire	decisionengine.tests.test_framework_package
module, 90	module, 96
decisionengine.framework.tests.test_module_program_options	decisionengine.tests.test_framework_package
module, 91	module, 96
decisionengine.framework.tests.test_query_tool_server	decisionengine.tests.test_framework_package

`delete_data_older_than()` (decisionengine.framework.dataspace.datasources.postgresql.PostgreSQLDataSource.delete\_data\_older\_than method), 52  
`delete_data_older_than()` (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.DataSource.delete\_data\_older\_than method), 38  
`delete_data_older_than()` (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.DataSource.delete\_data\_older\_than method), 37  
`delete_data_older_than()` (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.DataSource.delete\_data\_older\_than method), 44  
`delete_data_older_than()` (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.DataSource.delete\_data\_older\_than method), 44  
`Describe` (class in decisionengine.framework.tests.ModuleProgramOptions), 87  
`describe()` (in module decisionengine.framework.modules.Publisher), 80  
`describe()` (in module decisionengine.framework.modules.Source), 80  
`describe()` (in module decisionengine.framework.modules.Transform), 81  
`DescribeAlias` (class in decisionengine.framework.tests.ModuleProgramOptions), 87  
`DEServer()` (in module decisionengine.framework.engine.tests.fixtures), 65  
`DEServer()` (in module decisionengine.framework.tests.fixtures), 89  
`deserver_mock_data_block()` (in module decisionengine.framework.tests.test\_restart\_channel), 91  
`deserver_mock_data_block()` (in module decisionengine.framework.tests.test\_start\_with\_no\_channels), 93  
`do_backup()` (decisionengine.framework.managers.ChannelManager.ChannelManager.do\_backup method), 75  
`do_backup()` (decisionengine.framework.taskmanager.TaskManager.TaskManager.do\_backup method), 84  
`dump()` (decisionengine.framework.config.ValidConfig.ValidConfig.dump method), 36  
`duplicate()` (decisionengine.framework.dataspace.datablock.DataBlock.duplicate method), 59  
`duplicate_datablock()` (decisionengine.framework.dataspace.datasource.DataSource.duplicate\_datablock method), 61  
`duplicate_datablock()` (decisionengine.framework.dataspace.datasources.null.NullDataSource.duplicate\_datablock method), 50  
`duplicate_datablock()` (decisionengine.framework.dataspace.datasources.postgresql.PostgreSQLDataSource.duplicate\_datablock method), 52  
`duplicate_datablock()` (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.DataSource.duplicate\_datablock method), 38  
`duplicate_datablock()` (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.DataSource.duplicate\_datablock method), 44  
`duplicate_datablock()` (decisionengine.framework.dataspace.dataspace.DataSpace.duplicate\_datablock method), 63  
**E**  
`ensure_no_circularities()` (in module decisionengine.framework.taskmanager.module\_graph), 85  
`ERROR` (decisionengine.framework.taskmanager.ProcessingState.State attribute), 83  
`ErrorOnAcquire` (class in decisionengine.framework.tests.ErrorOnAcquire), 86  
`evaluate()` (decisionengine.framework.logicengine.BooleanExpression.BooleanExpression.evaluate method), 72  
`evaluate()` (decisionengine.framework.logicengine.LogicEngine.LogicEngine.evaluate method), 74  
`evaluate()` (decisionengine.framework.logicengine.Rule.Rule.evaluate method), 74  
`evaluate_facts()` (decisionengine.framework.logicengine.LogicEngine.LogicEngine.evaluate\_facts method), 74  
`execute()` (decisionengine.framework.logicengine.RuleEngine.RuleEngine.execute method), 74  
`execute_command_from_args()` (in module decisionengine.framework.engine.de\_client), 70  
`execute_command_from_args()` (in module decisionengine.framework.engine.de\_query\_tool), 70  
`expiration_time` (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema.Schema attribute), 42  
**F**  
`FactLookup` (class in decisionengine.framework.logicengine.FactLookup), 73  
`FailingPublisher` (class in decisionengine.framework.tests.FailingPublisher), 86  
`FailingSourceProxy` (class in decisionengine.framework.tests.FailingSourceProxy), 86  
`files_with_extensions()` (in module decisionengine.framework.util.fs), 93  
`function_name_from_call()` (in module decisionengine.framework.logicengine.BooleanExpression), 72

## G

Gauge (class in <i>decisionengine.framework.util.metrics</i> ), 94	<i>method</i> ), 63
generate_insert_query() (in module <i>decisionengine.framework.dataspace.datasources.postgresql</i> ), 54	get_dataproduct() (decisionengine.framework.dataspace.datasource.DataSource <i>method</i> ), 61
generation_id (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Metadata attribute), 41	get_dataproduct() (decisionengine.framework.dataspace.datasources.null.NullDataSource <i>method</i> ), 50
generation_id (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Metadata attribute), 42	get_dataproduct() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql <i>method</i> ), 53
generation_id (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Metadata attribute), 42	get_dataproduct() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource <i>method</i> ), 38
generation_time (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Metadata attribute), 42	get_dataproduct() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.SQLAlchemyDS <i>method</i> ), 45
get() (decisionengine.framework.dataspace.datablock.DataBlock <i>method</i> ), 59	get_dataproduct() (decisionengine.framework.dataspace.dataspace.DataSpace <i>method</i> ), 63
get() (decisionengine.framework.taskmanager.ProcessingState.ProcessingState <i>method</i> ), 83	get_dataproducts() (decisionengine.framework.dataspace.datablock.DataBlock <i>method</i> ), 59
get_channels() (decisionengine.framework.config.ChannelConfigHandler.ChannelConfigHandler <i>method</i> ), 35	get_dataproducts() (decisionengine.framework.dataspace.datasource.DataSource <i>method</i> ), 62
get_connection() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql <i>method</i> ), 53	get_dataproducts() (decisionengine.framework.dataspace.datasources.null.NullDataSource <i>method</i> ), 50
get_consumes() (decisionengine.framework.engine.Workers.Worker <i>method</i> ), 69	get_dataproducts() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql <i>method</i> ), 53
get_consumes() (decisionengine.framework.taskmanager.TaskManager.TaskManager <i>method</i> ), 84	get_dataproducts() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource <i>method</i> ), 38
get_data_block() (decisionengine.framework.modules.Module.Module <i>method</i> ), 79	get_dataproducts() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.SQLAlchemyDS <i>method</i> ), 45
get_datablock() (decisionengine.framework.dataspace.datasource.DataSource <i>method</i> ), 61	get_dataproducts() (decisionengine.framework.dataspace.dataspace.DataSpace <i>method</i> ), 63
get_datablock() (decisionengine.framework.dataspace.datasources.null.NullDataSource <i>method</i> ), 50	get_header() (decisionengine.framework.dataspace.datablock.DataBlock <i>method</i> ), 59
get_datablock() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql <i>method</i> ), 53	get_header() (decisionengine.framework.dataspace.datasource.DataSource <i>method</i> ), 62
get_datablock() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource_api.SQLAlchemyDS <i>method</i> ), 38	get_header() (decisionengine.framework.dataspace.datasources.null.NullDataSource <i>method</i> ), 51
get_datablock() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.SQLAlchemyDS <i>method</i> ), 44	get_header() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql <i>method</i> ), 53
get_datablock() (decisionengine.framework.dataspace.dataspace.DataSpace <i>method</i> ), 63	get_header() (decisionengine.framework.dataspace.datasources.sqlalchemy_ds.datasource <i>method</i> ), 38

method), 38

get\_header() (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.SQLAlchemyDS method), 45

get\_header() (decisionengine.framework.dataspace.DataSpace method), 63

get\_last\_generation\_id() (decisionengine.framework.dataspace.datasource.DataSource method), 62

get\_last\_generation\_id() (decisionengine.framework.datasources.null.NullDataSource method), 51

get\_last\_generation\_id() (decisionengine.framework.datasources.postgresql.PostgreSQL method), 53

get\_last\_generation\_id() (decisionengine.framework.datasources.sqlalchemy\_ds SQLAlchemyDS method), 39

get\_last\_generation\_id() (decisionengine.framework.datasources.sqlalchemy\_ds SQLAlchemyDS method), 45

get\_last\_generation\_id() (decisionengine.framework.dataspace.DataSpace method), 63

get\_logger() (decisionengine.framework.engine.DecisionEngine method), 67

get\_logger() (in module decisionengine.framework.modules.de\_logger), 81

get\_loglevel() (decisionengine.framework.managers.ComponentManager.ComponentManager method), 76

get\_metadata() (decisionengine.framework.dataspace.datablock.DataBlock method), 59

get\_metadata() (decisionengine.framework.dataspace.datasource.DataSource method), 62

get\_metadata() (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 51

get\_metadata() (decisionengine.framework.dataspace.datasources.postgresql.PostgreSQL method), 53

get\_metadata() (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds SQLAlchemyDS method), 39

get\_metadata() (decisionengine.framework.dataspace.datasource.DataSource method), 46

get\_metadata() (decisionengine.framework.dataspace.DataSpace method), 51

method), 63

get\_new\_subscriptions() (decisionengine.framework.managers.SourceSubscriptionManager.SourceSubscriptionManager method), 77

get\_parameters() (decisionengine.framework.modules.Module.Module method), 79

get\_produces() (decisionengine.framework.engine.Workers.Worker method), 69

get\_produces() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 84

get\_random\_port() (in module decisionengine.framework.util.sockets), 95

get\_schema() (decisionengine.framework.dataspace.datasource.DataSource method), 62

get\_schema() (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 51

get\_schema() (decisionengine.framework.dataspace.datasources.postgresql.PostgreSQL method), 53

get\_schema() (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds SQLAlchemyDS method), 39

get\_schema() (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds SQLAlchemyDS method), 46

get\_state() (decisionengine.framework.managers.ComponentManager.ComponentManager method), 76

get\_state\_name() (decisionengine.framework.engine.Workers.Worker method), 69

get\_state\_name() (decisionengine.framework.managers.ComponentManager.ComponentManager method), 76

get\_state\_value() (decisionengine.framework.managers.ComponentManager.ComponentManager method), 76

get\_state\_value() (decisionengine.framework.taskmanager.ProcessingState.ProcessingState method), 83

get\_taskmanager() (decisionengine.framework.dataspace.datablock.DataBlock method), 59

get\_taskmanager() (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds SQLAlchemyDS method), 39

get\_taskmanager() (decisionengine.framework.dataspace.datasource.DataSource method), 46

get\_taskmanager() (decisionengine.framework.dataspace.DataSpace method), 51

[get\\_taskmanager\(\)](#) (decisionengine.framework.dataspace.datasources.postgresql.PostgreSQLDataSource attribute), 54  
[get\\_taskmanager\(\)](#) (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.api.SQLAlchemyDS attribute), 39  
[get\\_taskmanager\(\)](#) (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.SQLAlchemyDS attribute), 46  
[get\\_taskmanager\(\)](#) (decisionengine.framework.dataspace.dataspace.DataSpace attribute), 64  
[get\\_taskmanagers\(\)](#) (decisionengine.framework.datasource.DataSource method), 62  
[get\\_taskmanagers\(\)](#) (decisionengine.framework.datasources.null.NullDataSource method), 51  
[get\\_taskmanagers\(\)](#) (decisionengine.framework.datasources.postgresql.PostgreSQLDataSource method), 54  
[get\\_taskmanagers\(\)](#) (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.api.SQLAlchemyDS method), 39  
[get\\_taskmanagers\(\)](#) (decisionengine.framework.dataspace.dataspace.DataSpace method), 46  
[get\\_taskmanagers\(\)](#) (decisionengine.framework.dataspace.dataspace.DataSpace method), 60  
[get\\_taskmanagers\(\)](#) (decisionengine.framework.dataspace.datablock.DataBlock method), 60  
[global\\_config\\_dir\(\)](#) (in module decisionengine.framework.config.policies), 36  
[global\\_config\\_file\(\)](#) (in module decisionengine.framework.config.policies), 36  

## H

[handle\\_sighup\(\)](#) (decisionengine.engine.DecisionEngine.DecisionEngine method), 67  
[has\\_value\(\)](#) (decisionengine.framework.taskmanager.ProcessingState.ProcessingState method), 83  

## Header

[Header](#) (class in decisionengine.framework.dataspace.datablock), 60  

## Header

[Header](#) (class in decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema), 41  

## header\_table

[header\\_table](#) (decisionengine.framework.datasource.DataSource attribute), 62  

## Help

[Help](#) (class in decisionengine.framework.tests.ModuleProgramOptions), 87



`load_sample_data_into_datasource()` (in module `decisionengine.framework.dataspace.tests.fixtures`), 56  
`lock` (`decisionengine.framework.taskmanager.ProcessingState.ProcessingState` property), 83  
`log_setup()` (in module `decisionengine.framework.modules.tests.test_de_logger`), 78  
`logic_engine_with_fact()` (in module `decisionengine.framework.logicengine.tests.test_fail_on_error`), 71  
`LogicEngine` (class in `decisionengine.framework.logicengine.LogicEngine`), 73  
`LogicError`, 72  
**M**  
`main()` (in module `decisionengine.framework.engine.de_client`), 70  
`main()` (in module `decisionengine.framework.engine.de_query_tool`), 70  
`main()` (in module `decisionengine.framework.engine.DecisionEngine`), 68  
`main()` (in module `decisionengine.framework.util.reaper`), 94  
`main_wrapper()` (in module `decisionengine.framework.modules.describe`), 82  
`make_db()` (in module `decisionengine.framework.logicengine.tests.test_facts`), 71  
`mark_demented()` (`decisionengine.framework.dataspace.dataspace.DataSpace` method), 64  
`mark_expired()` (`decisionengine.framework.dataspace.datablock.DataBlock` method), 60  
`mark_expired()` (`decisionengine.framework.dataspace.dataspace.DataSpace` method), 64  
`maybe_fail_on_error()` (in module `decisionengine.framework.logicengine.BooleanExpression`), 72  
`Metadata` (class in `decisionengine.framework.dataspace.datablock`), 60  
`Metadata` (class in `decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema`), 42  
`metadata` (`decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema.Base` attribute), 41  
`metadata_table` (`decisionengine.framework.dataspace.datasource.DataSource` attribute), 63  
`MIN_RETENTION_INTERVAL_DAYS` (`decisionengine.framework.dataspace.maintain.Reaper` attribute), 64  
`MIN_SECONDS_BETWEEN_RUNS` (`decisionengine.framework.dataspace.maintain.Reaper` attribute), 64  
`missed_update_count` (`decisionengine.framework.dataspace.datasources.sqlalchemy_ds.db_schema` attribute), 42  
`mock_data_block()` (in module `decisionengine.framework.dataspace.datasources.tests.fixtures`), 48  
`module`  
`decisionengine`, 96  
`decisionengine.framework`, 95  
`decisionengine.framework.about`, 95  
`decisionengine.framework.config`, 37  
`decisionengine.framework.config.ChannelConfigHandler`, 35  
`decisionengine.framework.config.policies`, 36  
`decisionengine.framework.config.tests`, 35  
`decisionengine.framework.config.tests.test_config`, 33  
`decisionengine.framework.config.tests.test_de_std`, 34  
`decisionengine.framework.config.tests.test_policies`, 34  
`decisionengine.framework.config.ValidConfig`, 35  
`decisionengine.framework.dataspace`, 65  
`decisionengine.framework.dataspace.datablock`, 59  
`decisionengine.framework.dataspace.datasource`, 61  
`decisionengine.framework.dataspace.datasources`, 55  
`decisionengine.framework.dataspace.datasources.null`, 50  
`decisionengine.framework.dataspace.datasources.postgre`, 52  
`decisionengine.framework.dataspace.datasources.sqlalch`, 44  
`decisionengine.framework.dataspace.datasources.sqlalch`, 37  
`decisionengine.framework.dataspace.datasources.sqlalch`, 41  
`decisionengine.framework.dataspace.datasources.sqlalch`, 43  
`decisionengine.framework.dataspace.datasources.tests`, 50

decisionengine.framework.dataspace.datasources.tests.fixtures,  
 47 decisionengine.framework.logicengine.tests.test\_cascade,  
 decisionengine.framework.dataspace.datasources.tests.test\_datasource\_api,  
 48 decisionengine.framework.logicengine.tests.test\_constraints,  
 decisionengine.framework.dataspace.datasources.tests.test\_postgresql,  
 50 decisionengine.framework.logicengine.tests.test\_duplicates,  
 decisionengine.framework.dataspace.dataspace, 71  
 63 decisionengine.framework.logicengine.tests.test\_facts,  
 decisionengine.framework.dataspace.maintain, 71  
 64 decisionengine.framework.logicengine.tests.test\_fail\_on\_error,  
 decisionengine.framework.dataspace.tests, 71  
 59 decisionengine.framework.logicengine.tests.test\_pandas,  
 decisionengine.framework.dataspace.tests.fixtures, 72  
 55 decisionengine.framework.logicengine.tests.test\_rule\_weights,  
 decisionengine.framework.dataspace.tests.test\_tablelock, 71  
 57 decisionengine.framework.logicengine.tests.test\_simple\_rules,  
 decisionengine.framework.dataspace.tests.test\_tablelock\_zlib, 71  
 57 decisionengine.framework.managers, 78  
 decisionengine.framework.dataspace.tests.test\_dataspaces, decisionengine.framework.managers.ChannelManager,  
 57 75  
 decisionengine.framework.dataspace.tests.test\_dataspaces, decisionengine.framework.managers.ComponentManager,  
 58 76  
 decisionengine.framework.dataspace.tests.test\_deaper, decisionengine.framework.managers.SourceManager,  
 56 77  
 decisionengine.framework.engine, 70 decisionengine.framework.managers.SourceSubscriptionManager,  
 decisionengine.framework.engine.de\_client, 77  
 70 decisionengine.framework.modules, 83  
 decisionengine.framework.engine.de\_query\_tool, decisionengine.framework.modules.de\_logger,  
 70 81  
 decisionengine.framework.engine.DecisionEngine, decisionengine.framework.modules.describe,  
 66 82  
 decisionengine.framework.engine.tests, 66 decisionengine.framework.modules.logging\_configDict,  
 decisionengine.framework.engine.tests.fixtures, 82  
 65 decisionengine.framework.modules.Module,  
 decisionengine.framework.engine.tests.test\_client\_tool\_only, 78  
 66 decisionengine.framework.modules.print\_description,  
 decisionengine.framework.engine.tests.test\_query\_tool\_only, 80  
 66 decisionengine.framework.modules.Publisher,  
 decisionengine.framework.engine.tests.test\_startup, 79  
 66 decisionengine.framework.modules.Source,  
 decisionengine.framework.engine.Workers, 80  
 69 decisionengine.framework.modules.SourceProxy,  
 decisionengine.framework.logicengine, 75 80  
 decisionengine.framework.logicengine.BooleanExpression, decisionengine.framework.modules.tests,  
 72 79  
 decisionengine.framework.logicengine.FactLookup, decisionengine.framework.modules.tests.test\_de\_logger,  
 73 78  
 decisionengine.framework.logicengine.LogicEngine, decisionengine.framework.modules.tests.test\_Module,  
 73 78  
 decisionengine.framework.logicengine.Rule, decisionengine.framework.modules.tests.test\_module\_decision,  
 74 79  
 decisionengine.framework.logicengine.RuleEngine, decisionengine.framework.modules.tests.test\_Publisher,  
 74 78  
 decisionengine.framework.logicengine.tests, decisionengine.framework.modules.tests.test\_Source,

78	91
decisionengine.framework.modules.tests.test_Transform	decisionengine.framework.tests.test_query_tool_server,
78	91
decisionengine.framework.modules.tests.test_translate_engine_iframe	decisionengine.framework.tests.test_reaper,
79	91
decisionengine.framework.modules.Transform,	decisionengine.framework.tests.test_restart_channel,
81	91
decisionengine.framework.modules.translate_producer_name	decisionengine.framework.tests.test_sample_config,
82	91
decisionengine.framework.taskmanager, 85	decisionengine.framework.tests.test_source_proxy,
decisionengine.framework.taskmanager.module_graph,	93
85	decisionengine.framework.tests.test_start_with_bad_chan
decisionengine.framework.taskmanager.ProcessingState,	93
83	decisionengine.framework.tests.test_start_with_no_chan
decisionengine.framework.taskmanager.TaskManager,	93
84	decisionengine.framework.tests.TransformNOP,
decisionengine.framework.tests, 93	88
decisionengine.framework.tests.ABTransform,	decisionengine.framework.tests.TransformWithMissingPro
85	89
decisionengine.framework.tests.BATransform,	decisionengine.framework.tests.WorkingSourceProxy,
86	89
decisionengine.framework.tests.ErrorOnAcquire,	decisionengine.framework.util, 95
86	decisionengine.framework.util.fs, 93
decisionengine.framework.tests.FailingPublisher,	decisionengine.framework.util.metrics, 94
86	decisionengine.framework.util.reaper, 94
decisionengine.framework.tests.FailingSourceNOP,	decisionengine.framework.util.singleton,
86	95
decisionengine.framework.tests.FailingSourceProxy,	decisionengine.framework.util.sockets, 95
86	decisionengine.framework.util.subclasses,
decisionengine.framework.tests.fixtures,	95
89	decisionengine.framework.version, 95
decisionengine.framework.tests.ModuleProgramOptions,	decisionengine.tests, 96
87	decisionengine.tests.test_framework_package,
decisionengine.framework.tests.PublisherNOP,	96
87	Module (class in decisio-
decisionengine.framework.tests.PublisherWithMissingConsumes,	engine.framework.modules.Module), 79
88	ModuleProgramOptions (class in decisio-
decisionengine.framework.tests.SourceAlias,	engine.framework.modules.describe), 82
88	mydata() (in module decisio-
decisionengine.framework.tests.SourceNOP,	engine.framework.logicengine.tests.test_pandas_fact),
88	72
decisionengine.framework.tests.SourceWithSampleConfigNOP,	(in module decisio-
88	engine.framework.logicengine.tests.test_cascaded_rules),
decisionengine.framework.tests.SupportsConfigPublisher,	
88	myengine() (in module decisio-
decisionengine.framework.tests.test_client_errors,	engine.framework.logicengine.tests.test_pandas_fact),
90	72
decisionengine.framework.tests.test_client_myengine(),	(in module decisio-
90	engine.framework.logicengine.tests.test_rule_with_negated_fact
decisionengine.framework.tests.test_defaults,	72
90	myengine() (in module decisio-
decisionengine.framework.tests.test_error_on_acquire,	engine.framework.logicengine.tests.test_simple_configuration),
90	72
decisionengine.framework.tests.test_module_program_options,	



## N

`name` (`decisionengine.framework.dataspace.datasources.sqlalchemy_db_schema.TaskManager`), 48

`NotFound` (`decisionengine.framework.engine.DecisionEngine.StopState`), 68

`NullDataSource` (class in `decisionengine.framework.dataspace.datasources.null`), 50

## O

`OFFLINE` (`decisionengine.framework.taskmanager.ProcessingState.StillProcessing`), 89

`orm_as_dict()` (in module `decisionengine.framework.dataspace.datasources.sqlalchemy_ds_utils`), 43

## P

`Parameter` (class in `decisionengine.framework.modules.describe`), 82

`Parameter` (class in `decisionengine.framework.modules.Publisher`), 79

`Parameter` (class in `decisionengine.framework.modules.Source`), 80

`Parameter` (class in `decisionengine.framework.modules.Transform`), 81

`parse_program_options()` (in module `decisionengine.framework.engine.DecisionEngine`), 68

`passthrough_configuration()` (in module `decisionengine.framework.logicengine.LogicEngine`), 74

`PG_DE_DB_WITH_SCHEMA()` (in module `decisionengine.framework.dataspace.datasources.tests.fixtures`), 47

`PG_DE_DB_WITH_SCHEMA()` (in module `decisionengine.framework.dataspace.tests.fixtures`), 55

`PG_DE_DB_WITH_SCHEMA()` (in module `decisionengine.framework.engine.tests.fixtures`), 65

`PG_DE_DB_WITH_SCHEMA()` (in module `decisionengine.framework.tests.fixtures`), 89

`PG_DE_DB_WITHOUT_SCHEMA()` (in module `decisionengine.framework.dataspace.datasources.tests.fixtures`), 47

`PG_DE_DB_WITHOUT_SCHEMA()` (in module `decisionengine.framework.dataspace.tests.fixtures`), 55

`PG_DE_DB_WITHOUT_SCHEMA()` (in module `decisionengine.framework.engine.tests.fixtures`), 65

`PG_DE_DB_WITHOUT_SCHEMA()` (in module `decisionengine.framework.tests.fixtures`), 89

`PG_PROG()` (in module `decisionengine.framework.dataspace.datasources.tests.fixtures`), 48

`PG_PROG()` (in module `decisionengine.framework.dataspace.tests.fixtures`), 55

`PG_PROG()` (in module `decisionengine.framework.engine.tests.fixtures`), 65

`PG_PROG()` (in module `decisionengine.framework.tests.fixtures`), 89

`post_create()` (in module `decisionengine.framework.modules.Source.Source`), 80

`post_create()` (in module `decisionengine.framework.modules.SourceProxy.SourceProxy`), 80

`Postgresql` (class in `decisionengine.framework.dataspace.datasources.postgresql`), 52

`print_channel_config()` (in module `decisionengine.framework.config.ChannelConfigHandler.ChannelConfigHandler`), 35

`print_consumes()` (in module `decisionengine.framework.modules.print_description`), 82

`print_produces()` (in module `decisionengine.framework.modules.print_description`), 82

`print_supported_config()` (in module `decisionengine.framework.modules.print_description`), 82

`probably_running()` (in module `decisionengine.framework.taskmanager.ProcessingState.ProcessingState`), 83

`process_args()` (in module `decisionengine.framework.modules.describe.ModuleProgramOptions`), 82

`ProcessingState` (class in `decisionengine.framework.taskmanager.ProcessingState`), 83

`produces()` (in module `decisionengine.framework.logicengine.LogicEngine.LogicEngine`), 74

`produces()` (in module `decisionengine.framework.modules.Module`), 79

`produces()` (in module `decisionengine.framework.modules.Source`), 80

`produces()` (in module `decisionengine.framework.modules.Transform`), 81

`ProductRetriever` (class in `decisionengine.framework.dataspace.datablock`), 60

`publish()` (in module `decisionengine.framework.modules.Publisher.Publisher`), 89

method), 79  
 publish() (decisionengine.framework.tests.FailingPublisher.publish() method), 86  
 publish() (decisionengine.framework.tests.PublisherNOP.PublisherNOP method), 87  
 Publisher (class in decisionengine.framework.modules.Publisher), 79  
 PublisherNOP (class in decisionengine.framework.tests.PublisherNOP), 87  
 PublisherWithMissingConsumes (class in decisionengine.framework.tests.PublisherWithMissingConsumes), method), 75  
 88  
 put() (decisionengine.framework.dataspace.datablock.DataBlock method), 60  
**R**  
 reap() (decisionengine.framework.dataspace.maintain.Reaper method), 64  
 Reaper (class in decisionengine.framework.dataspace.maintain), 64  
 reaper() (in module decisionengine.framework.dataspace.tests.test\_Reaper), 56  
 reaper\_start() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 67  
 reaper\_status() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 67  
 reaper\_stop() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 67  
 register\_with\_sources() (decisionengine.framework.managers.ChannelManager.ChannelManager method), 75  
 registry (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.schema.Base attribute), 41  
 RequestHandler (class in decisionengine.framework.engine.DecisionEngine), 68  
 required\_keys (decisionengine.framework.dataspace.datablock.Header attribute), 60  
 required\_keys (decisionengine.framework.dataspace.datablock.Metadata attribute), 60  
 reset\_connections() (decisionengine.framework.dataspace.datasource.DataSource method), 63  
 reset\_connections() (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 51  
 reset\_connections() (decisionengine.framework.dataspace.datasources.postgresql.Postgresql method), 54  
 reset\_connections() (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.datasource method), 40  
 reset\_connections() (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.SQLAlchemy method), 47  
 reset\_source\_flags() (decisionengine.framework.managers.ChannelManager.ChannelManager method), 75  
 retention\_interval (decisionengine.framework.dataspace.maintain.Reaper property), 64  
 rm\_channel() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 67  
 rpc\_block\_while() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 67  
 rpc\_get\_channel\_log\_level() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 67  
 rpc\_get\_log\_level() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 67  
 rpc\_kill\_channel() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 67  
 rpc\_paths (decisionengine.framework.engine.DecisionEngine.RequestHandler attribute), 68  
 rpc\_print\_product() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 67  
 rpc\_print\_products() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 67  
 rpc\_query\_tool() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 67  
 rpc\_reaper\_start() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 67  
 rpc\_reaper\_status() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 67  
 rpc\_reaper\_stop() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 67  
 rpc\_rm\_channel() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 67

[rpc\\_set\\_channel\\_log\\_level\(\)](#) (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 84  
[rpc\\_set\\_channel\\_log\\_level\(\)](#) (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 67  
[rpc\\_show\\_config\(\)](#) (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 84  
[rpc\\_show\\_config\(\)](#) (decisionengine.framework.managers.ChannelManager.ChannelManager method), 67  
[rpc\\_show\\_de\\_config\(\)](#) (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 75  
[rpc\\_show\\_de\\_config\(\)](#) (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 67  
[rpc\\_start\\_channel\(\)](#) (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 84  
[rpc\\_start\\_channel\(\)](#) (decisionengine.framework.managers.ChannelManager.ChannelManager method), 68  
[rpc\\_start\\_channels\(\)](#) (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 76  
[rpc\\_start\\_channels\(\)](#) (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 68  
[rpc\\_status\(\)](#) (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 84  
[rpc\\_status\(\)](#) (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 68  
[rpc\\_stop\(\)](#) (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 68  
[rpc\\_stop\\_channel\(\)](#) (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 68  
[rpc\\_stop\\_channel\(\)](#) (decisionengine.framework.managers.ChannelManager.ChannelManager method), 68  
[rpc\\_stop\\_channels\(\)](#) (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 68  
[rpc\\_stop\\_channels\(\)](#) (decisionengine.framework.managers.ChannelManager.ChannelManager method), 68  
[Rule](#) (class in decisionengine.framework.logicengine.Rule), 74  
[rule\\_for\(\)](#) (decisionengine.framework.logicengine.FactLookup.FactLookup method), 73  
[RuleEngine](#) (class in decisionengine.framework.logicengine.RuleEngine), 74  
[run\(\)](#) (decisionengine.framework.engine.Workers.Worker method), 69  
[run\(\)](#) (decisionengine.framework.managers.ChannelManager.ChannelManager method), 75  
[run\(\)](#) (decisionengine.framework.managers.SourceManager.SourceManager method), 77  
[run\(\)](#) (decisionengine.framework.managers.SourceSubscriptionManager.SourceSubscriptionManager method), 77  
[run\(\)](#) (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 84  
[run\\_logic\\_engine\(\)](#) (decisionengine.framework.managers.ChannelManager.ChannelManager method), 75  
[run\\_logic\\_engine\(\)](#) (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 84  
[run\\_publishers\(\)](#) (decisionengine.framework.managers.ChannelManager.ChannelManager method), 75  
[run\\_publishers\(\)](#) (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 76

set\_state() (decisionengine.framework.dataspace.datablock.Metadata method), 60  
 set\_to\_shutdown() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 85  
 should\_stop() (decisionengine.framework.taskmanager.ProcessingState.ProcessingState method), 83  
 SHUTDOWN (decisionengine.framework.taskmanager.ProcessingState attribute), 84  
 shutdown() (decisionengine.framework.modules.Publisher method), 79  
 SHUTTINGDOWN (decisionengine.framework.taskmanager.ProcessingState.State attribute), 84  
 Singleton (class in decisionengine.framework.util.singleton), 95  
 SingletonABC (class in decisionengine.framework.util.singleton), 95  
 sorted\_rules() (decisionengine.framework.logicengine.FactLookup.FactLookup method), 73  
 Source (class in decisionengine.framework.managers.SourceManager), 77  
 Source (class in decisionengine.framework.modules.Source), 80  
 SourceManager (class in decisionengine.framework.managers.SourceManager), 77  
 SourceNOP (class in decisionengine.framework.tests.SourceNOP), 88  
 SourceProxy (class in decisionengine.framework.modules.SourceProxy), 80  
 SourceRunner (class in decisionengine.framework.managers.SourceManager), 77  
 SourceSubscriptionManager (class in decisionengine.framework.managers.SourceSubscriptionManager), 77  
 SourceWithMissingProduces (class in decisionengine.framework.tests.FailingSourceNOP), 86  
 SourceWithSampleConfigNOP (class in decisionengine.framework.tests.SourceWithSampleConfigNOP), 88  
 spec\_if\_main() (in module decisionengine.framework.modules.print\_description), 82  
 SQLALCHEMY\_PG\_WITH\_SCHEMA() (in module decisionengine.framework.dataspace.datasources.tests.fixtures), 48  
 SQLALCHEMY\_PG\_WITH\_SCHEMA() (in module decisionengine.framework.dataspace.tests.fixtures), 55  
 SQLALCHEMY\_PG\_WITH\_SCHEMA() (in module decisionengine.framework.engine.tests.fixtures), 65  
 SQLALCHEMY\_PG\_WITH\_SCHEMA() (in module decisionengine.framework.tests.fixtures), 89  
 SQLALCHEMY\_TEMPFILE\_SQLITE() (in module decisionengine.framework.dataspace.datasources.tests.fixtures), 48  
 SQLALCHEMY\_TEMPFILE\_SQLITE() (in module decisionengine.framework.dataspace.tests.fixtures), 55  
 SQLALCHEMY\_TEMPFILE\_SQLITE() (in module decisionengine.framework.engine.tests.fixtures), 65  
 SQLALCHEMY\_TEMPFILE\_SQLITE() (in module decisionengine.framework.tests.fixtures), 89  
 SQLAlchemyDS (class in decisionengine.framework.dataspace.datasources.sqlalchemy\_ds), 44  
 SQLAlchemyDS (class in decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.datasource), 37  
 stop() (decisionengine.framework.dataspace.maintain.Reaper method), 64  
 start\_channel() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 68  
 start\_channels() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 68  
 start\_sources() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 85  
 State (class in decisionengine.framework.taskmanager.ProcessingState), 83  
 state (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.db attribute), 42  
 STEADY (decisionengine.framework.taskmanager.ProcessingState.State attribute), 84  
 stop() (decisionengine.framework.dataspace.maintain.Reaper method), 64  
 stop\_channels() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 68  
 stop\_worker() (decisionengine.framework.engine.DecisionEngine.DecisionEngine method), 68  
 StopState (class in decisionengine.framework.engine.DecisionEngine), 68  
 store\_taskmanager() (decisionengine.framework.dataspace.datablock.DataBlock method), 60  
 store\_taskmanager() (decisionengine.framework.taskmanager.TaskManager.TaskManager method), 85



nengine.framework.dataspace.datasource.DataSourceTaskManager (class in decisio-  
 method), 63 nengine.framework.taskmanager.TaskManager),  
 store\_taskmanager() (decision- 84  
 nengine.framework.dataspace.datasources.null.NullTaskManager (decision-  
 method), 51 nengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_sch  
 store\_taskmanager() (decision- attribute), 41  
 nengine.framework.dataspace.datasources.postgresql.DataSource (decision-  
 method), 54 nengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_sch  
 store\_taskmanager() (decision- attribute), 42  
 nengine.framework.dataspace.datasources.sqlalchemy\_source\_api.SQLAlchemyDS (decision-  
 method), 40 nengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_sch  
 store\_taskmanager() (decision- attribute), 43  
 nengine.framework.dataspace.datasources.sqlalchemy\_source\_api SQLAlchemyDS (decision-  
 method), 47 nengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_sch  
 store\_taskmanager() (decision- attribute), 41  
 nengine.framework.dataspace.dataspace.DataSpace taskmanager\_id (decision-  
 method), 64 nengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_sch  
 Subscription (class in decisio- attribute), 42  
 nengine.framework.managers.SourceSubscriptionTaskManager\_id (decision-  
 77 nengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_sch  
 Summary (class in decisio- attribute), 43  
 nengine.framework.util.metrics), 94 taskmanager\_id (decision-  
 supports\_config() (in module decisio- nengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_sch  
 nengine.framework.modules.describe), 82 attribute), 43  
 supports\_config() (in module decisio- taskmanager\_table (decision-  
 nengine.framework.modules.Publisher), nengine.framework.dataspace.datasource.DataSource  
 80 attribute), 63  
 supports\_config() (in module decisio- TaskRunner (class in decisio-  
 nengine.framework.modules.Source), 80 nengine.framework.managers.ChannelManager),  
 supports\_config() (in module decisio- 76  
 nengine.framework.modules.Transform), Terminated (decisionengine.framework.engine.DecisionEngine.StopState  
 81 attribute), 68  
 SupportsConfig (class in decisio- test() (decisionengine.framework.tests.ModuleProgramOptions.AcquireW  
 nengine.framework.tests.SupportsConfigPublisher), method), 87  
 88 test() (decisionengine.framework.tests.ModuleProgramOptions.AcquireW  
 method), 87  
 test() (decisionengine.framework.tests.ModuleProgramOptions.ConfigTer  
 method), 87  
 test() (decisionengine.framework.tests.ModuleProgramOptions.Describe  
 method), 87  
 test() (decisionengine.framework.tests.ModuleProgramOptions.Describe  
 method), 87  
 test() (decisionengine.framework.tests.ModuleProgramOptions.Help  
 method), 87  
 task\_dataproduct (decision- test\_acquire\_for\_sources() (in module decisio-  
 nengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema.Taskmanager  
 attribute), 43 nengine.framework.tests.test\_module\_program\_options),  
 task\_header (decision- 41  
 nengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema.Taskmanager  
 attribute), 43  
 task\_metadata (decision- nengine.framework.modules.tests.test\_de\_logger),  
 nengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema.Taskmanager  
 attribute), 43  
 Taskmanager (class in decisio- test\_by\_size() (in module decisio-  
 nengine.framework.dataspace.datasources.sqlalchemy\_ds.db\_schema), nengine.framework.modules.tests.test\_de\_logger),  
 43 test\_by\_time() (in module decisio-

```

    nengine.framework.modules.tests.test_de_logger),
78
test_can_import() (in module decisio-
    nengine.tests.test_framework_package), 96
test_change_port() (in module decisio-
    nengine.framework.engine.tests.test_startup),
66
test_channel_config_dir() (in module decisio-
    nengine.framework.config.tests.test_policies),
34
test_channel_empty_config() (in module decisio-
    nengine.framework.config.tests.test_config), 33
test_channel_empty_dictionary() (in module deci-
    sionengine.framework.config.tests.test_config),
33
test_channel_invalid_modules_list()
    (in module decisio-
    nengine.framework.config.tests.test_config),
33
test_channel_invalid_modules_no_keys()
    (in module decisio-
    nengine.framework.config.tests.test_config),
33
test_channel_invalid_modules_string()
    (in module decisio-
    nengine.framework.config.tests.test_config),
33
test_channel_loading() (in module decisio-
    nengine.framework.config.tests.test_config),
33
test_channel_module_missing_all()
    (in module decisio-
    nengine.framework.config.tests.test_config),
33
test_channel_module_missing_module()
    (in module decisio-
    nengine.framework.config.tests.test_config),
34
test_channel_module_missing_parameters()
    (in module decisio-
    nengine.framework.config.tests.test_config),
34
test_channel_names() (in module decisio-
    nengine.framework.config.tests.test_config),
34
test_channel_no_config_files() (in module deci-
    sionengine.framework.config.tests.test_config),
34
test_channel_no_modules() (in module decisio-
    nengine.framework.config.tests.test_config),
34
test_client_can_double_set_de_server_channel_log_level()
    (in module decisio-
    nengine.framework.tests.test_sample_config),
91
test_client_can_get_de_server_channel_config()
    (in module decisio-
    nengine.framework.tests.test_sample_config),
91
test_client_can_get_de_server_channel_log_level()
    (in module decisio-
    nengine.framework.tests.test_sample_config),
91
test_client_can_get_de_server_reaper_start_delay()
    (in module decisio-
    nengine.framework.tests.test_reaper), 91
test_client_can_get_de_server_reaper_status()
    (in module decisio-
    nengine.framework.tests.test_reaper), 91
test_client_can_get_de_server_reaper_stop()
    (in module decisio-
    nengine.framework.tests.test_reaper), 91
test_client_can_get_de_server_show_channel_logger_level()
    (in module decisio-
    nengine.framework.tests.test_defaults), 90
test_client_can_get_de_server_show_config()
    (in module decisio-
    nengine.framework.tests.test_sample_config),
91
test_client_can_get_de_server_show_logger_level()
    (in module decisio-
    nengine.framework.tests.test_sample_config),
92
test_client_can_get_de_server_status()
    (in module decisio-
    nengine.framework.tests.test_sample_config),
92
test_client_can_get_products()
    (in module decisio-
    nengine.framework.tests.test_sample_config),
92
test_client_can_get_products_no_channels()
    (in module decisio-
    nengine.framework.tests.test_sample_config),
92
test_client_can_get_products_no_channels()
    (in module decisio-
    nengine.framework.tests.test_start_with_bad_channels),
93
test_client_can_kill_one_channel()
    (in module decisio-
    nengine.framework.tests.test_sample_config),
92
test_client_can_kill_one_channel_force()
    (in module decisio-
    nengine.framework.tests.test_sample_config),
92
test_client_can_kill_one_channel_timeout()

```

(in module decisionengine.framework.tests.test_sample_config), 92	nengine.framework.tests.test_sample_config), 92
test_client_can_set_de_server_channel_log_level() (in module decisionengine.framework.tests.test_sample_config), 92	test_client_get_non_real_channel() (in module decisionengine.framework.tests.test_sample_config), 92
test_client_can_start_all_channel() (in module decisionengine.framework.tests.test_sample_config), 92	test_client_help() (in module decisionengine.framework.engine.tests.test_client_only), 66
test_client_can_start_one_channel() (in module decisionengine.framework.tests.test_sample_config), 92	test_client_print_product() (in module decisionengine.framework.tests.test_client_server), 90
test_client_can_stop_channels() (in module decisionengine.framework.tests.test_sample_config), 92	test_client_print_product_columns() (in module decisionengine.framework.tests.test_client_server), 90
test_client_can_stop_one_channel() (in module decisionengine.framework.tests.test_sample_config), 92	test_client_print_product_columns_query() (in module decisionengine.framework.tests.test_client_server), 90
test_client_can_stop_server() (in module decisionengine.framework.tests.test_sample_config), 92	test_client_print_product_json() (in module decisionengine.framework.tests.test_client_server), 90
test_client_cannot_double_start() (in module decisionengine.framework.tests.test_sample_config), 92	test_client_print_product_not_real() (in module decisionengine.framework.tests.test_client_server), 90
test_client_cannot_wait_on_bad_state() (in module decisionengine.framework.tests.test_client_errors), 90	test_client_print_product_not_string() (in module decisionengine.framework.tests.test_client_server), 90
test_client_de_config_is_json() (in module decisionengine.framework.tests.test_defaults), 90	test_client_print_product_query() (in module decisionengine.framework.tests.test_client_server), 90
test_client_err_returned_as_rc() (in module decisionengine.framework.engine.tests.test_client_only), 66	test_client_print_product_types() (in module decisionengine.framework.tests.test_client_server), 90
test_client_err_returned_as_rc() (in module decisionengine.framework.engine.tests.test_query_tool_only), 66	test_client_print_product_vertical() (in module decisionengine.framework.tests.test_client_server), 90
test_client_err_returned_verbose_as_rc() (in module decisionengine.framework.engine.tests.test_client_only), 66	test_client_set_channel_log_fails_cleanly() (in module decisionengine.framework.tests.test_sample_config), 92
test_client_err_returned_verbose_as_rc() (in module decisionengine.framework.engine.tests.test_query_tool_only), 66	test_client_start_non_real_channel() (in module decisionengine.framework.tests.test_sample_config), 92
test_client_get_channel_log_fails_cleanly() (in module decisionengine.framework.tests.test_client_server),	test_client_status_msg_to_stdout() (in module decisionengine.framework.tests.test_client_server),

90	test_DataBlock_duplicate() (in module decisio-
test_client_stop_non_real_channel()	nengine.framework.dataspace.tests.test_datablock),
(in module decisio-	57
nengine.framework.tests.test_sample_config),	test_DataBlock_get_dataproducts()
92	(in module decisio-
test_client_wait_timeout_works()	nengine.framework.dataspace.tests.test_datablock),
(in module decisio-	57
nengine.framework.tests.test_sample_config),	test_DataBlock_get_header() (in module decisio-
92	nengine.framework.dataspace.tests.test_datablock),
test_client_with_no_command_says_use_help()	57
(in module decisio-	test_DataBlock_get_metadata() (in module decisio-
nengine.framework.engine.tests.test_client_only),	nengine.framework.dataspace.tests.test_datablock),
66	57
test_client_with_no_server() (in module decisio-	test_DataBlock_get_taskmanager()
nengine.framework.engine.tests.test_client_only),	(in module decisio-
66	nengine.framework.dataspace.tests.test_datablock),
test_client_with_no_server_verbose()	57
(in module decisio-	test_DataBlock_is_expired() (in module decisio-
nengine.framework.engine.tests.test_client_only),	nengine.framework.dataspace.tests.test_datablock),
66	57
test_combine_one_level() (in module decisio-	test_DataBlock_is_expired_with_key()
nengine.framework.config.tests.test_de_std),	(in module decisio-
34	nengine.framework.dataspace.tests.test_datablock),
test_combine_one_level_skip_proxies()	57
(in module decisio-	test_DataBlock_key_management()
nengine.framework.config.tests.test_de_std),	(in module decisio-
34	nengine.framework.dataspace.tests.test_datablock),
test_compound_fact_with_spaces()	57
(in module decisio-	test_DataBlock_key_management_change_name()
nengine.framework.logicengine.tests.test_facts),	(in module decisio-
71	nengine.framework.dataspace.tests.test_datablock),
test_compress() (in module decisio-	57
nengine.framework.dataspace.tests.test_datablock),	test_DataBlock_mark_expired() (in module decisio-
57	nengine.framework.dataspace.tests.test_datablock),
test_conditional_fact() (in module decisio-	57
nengine.framework.logicengine.tests.test_fail_on_test),	test_DataBlock_no_key_by_name()
71	(in module decisio-
test_config_templates() (in module decisio-	nengine.framework.dataspace.tests.test_datablock),
nengine.framework.tests.test_module_program_options),	57
91	test_DataBlock_to_str() (in module decisio-
test_configuration_with_fact_using_function()	nengine.framework.dataspace.tests.test_datablock),
(in module decisio-	57
nengine.framework.logicengine.tests.test_construction),	test_dataspace_config_finds_bad()
71	(in module decisio-
test_configuration_with_numpy_facts()	nengine.framework.dataspace.tests.test_dataspace),
(in module decisio-	58
nengine.framework.logicengine.tests.test_construction),	test_default_config() (in module decisio-
71	nengine.framework.engine.tests.test_startup),
test_create_tables() (in module decisio-	66
nengine.framework.dataspace.datasources.tests.test_default_top),	test_default_top_construction() (in module decisio-
48	nengine.framework.logicengine.tests.test_construction),
test_DataBlock_constructor() (in module decisio-	71
nengine.framework.dataspace.tests.test_datablock),	test_delete_data_older_than_arg()
57	(in module decisio-



```

engine.framework.dataspace.datasources.tests.test_fail_small_reaper() (in module decisio-
48 engine.framework.dataspace.tests.test_Reaper),
test_descriptions() (in module decisio- 56
engine.framework.tests.test_module_program_options),
91 test_fail_small_run_interval()
(in module decisio-
engine.framework.dataspace.tests.test_Reaper),
test_duplicate_datablock() (in module decisio- 56
engine.framework.dataspace.datasources.tests.test_datasource_api),
48 test_fail_start_two_reapers() (in module decisio-
engine.framework.dataspace.tests.test_Reaper),
test_duplicate_datablock() (in module decisio- 56
engine.framework.dataspace.tests.test_dataspace),
58 test_fail_wrong_config_key() (in module decisio-
engine.framework.dataspace.tests.test_Reaper),
71 test_duplicate_fact_names() (in module decisio-
engine.framework.logicengine.tests.test_duplicate_fact_names),
test_empty_config() (in module decisio- 71
engine.framework.config.tests.test_config),
34 test_false_fact_with_spaces() (in module decisio-
engine.framework.logicengine.tests.test_fail_on_error),
test_error_on_duplicate_keys() (in module deci- 71
sionengine.framework.config.tests.test_de_std),
34 test_false_literal_fact() (in module decisio-
engine.framework.logicengine.tests.test_fail_on_error),
test_exclusive_options() (in module decisio- 71
engine.framework.engine.tests.test_client_only),
66 test_generate_insert_query() (in module decisio-
engine.framework.dataspace.datasources.tests.test_postgresql),
50 test_get_datablock() (in module decisio-
engine.framework.dataspace.datasources.tests.test_datasource_
48
test_get_datablock() (in module decisio-
engine.framework.dataspace.tests.test_dataspace),
test_fact_using_numpy_array() (in module decisio- 58
engine.framework.logicengine.tests.test_facts),
71 test_get_dataproduct() (in module decisio-
engine.framework.dataspace.datasources.tests.test_datasource_
71
test_fact_using_numpy_function()
(in module decisio-
engine.framework.logicengine.tests.test_facts), test_get_dataproduct() (in module decisio-
71 engine.framework.dataspace.tests.test_dataspace),
58 test_fact_with_fail_on_error()
(in module decisio-
engine.framework.logicengine.tests.test_facts), test_get_dataproduct() (in module decisio-
71 engine.framework.dataspace.tests.test_dataspace),
58 test_fact_with_misspecified_attribute()
test_get_dataproduct_not_exist()
(in module decisio-
engine.framework.logicengine.tests.test_fail_on_error), (in module decisio-
71 engine.framework.dataspace.datasources.tests.test_datasource_
48
test_fact_with_nested_names() (in module decisio- test_get_dataproduct_not_exist()
engine.framework.logicengine.tests.test_facts), (in module decisio-
71 engine.framework.dataspace.tests.test_dataspace),
58 test_fail_bad_config() (in module decisio-
engine.framework.dataspace.tests.test_Reaper), test_get_dataproducts() (in module decisio-
56 engine.framework.dataspace.datasources.tests.test_datasource_
49
test_fail_missing_config() (in module decisio- test_get_dataproducts() (in module decisio-
engine.framework.dataspace.tests.test_Reaper), 56
engine.framework.dataspace.tests.test_dataspace),
58 test_fail_missing_config_key()
(in module decisio-
engine.framework.dataspace.tests.test_Reaper), test_get_dataproducts_not_exist()
56 (in module decisio-
engine.framework.dataspace.datasources.tests.test_datasource_
49
test_fail_on_error() (in module decisio- 49
engine.framework.logicengine.tests.test_fail_on_error), test_get_dataproducts_not_exist()
71 (in module decisio-

```

```

nengine.framework.dataspace.tests.test_dataspace)
58 test_get_taskmanagers() (in module decisio-
nengine.framework.dataspace.datasources.tests.test_datasource_
test_get_header() (in module decisio-
49 nengine.framework.dataspace.datasources.tests.test_datasource_
test_get_header() (in module decisio-
58 nengine.framework.dataspace.tests.test_dataspace)
test_get_taskmanagers_not_exist()
58 (in module decisio-
test_get_header_not_exist() (in module decisio-
nengine.framework.dataspace.datasources.tests.test_datasource_
49 nengine.framework.dataspace.datasources.tests.test_datasource_api),
test_get_taskmanagers_not_exist()
test_get_header_not_exist() (in module decisio-
58 nengine.framework.dataspace.tests.test_dataspace),
nengine.framework.dataspace.tests.test_dataspace),
58 test_get_last_generation_id() (in module decisio-
nengine.framework.dataspace.datasources.tests.test_datasource_
49 nengine.framework.tests.test_defaults),
test_get_last_generation_id() (in module decisio-
58 nengine.framework.dataspace.tests.test_dataspace),
nengine.framework.config.tests.test_policies),
34 test_get_last_generation_id_not_exist()
test_global_channel_log_level_in_config()
nengine.framework.config.tests.test_policies),
49 nengine.framework.config.tests.test_policies),
test_has_config() (in module decisio-
nengine.framework.dataspace.datasources.tests.test_datasource_
49 nengine.framework.dataspace.tests.test_dataspace)
test_has_config() (in module decisio-
58 nengine.framework.dataspace.tests.test_dataspace),
test_get_metadata() (in module decisio-
58 nengine.framework.dataspace.datasources.tests.test_datasource_
49 nengine.framework.dataspace.tests.test_dataspace)
test_has_methods_we_expect() (in module decisio-
57 nengine.framework.dataspace.tests.test_datasource),
test_get_metadata() (in module decisio-
58 nengine.framework.dataspace.tests.test_dataspace)
test_Header_constructor() (in module decisio-
57 nengine.framework.dataspace.tests.test_datablock),
test_get_metadata_not_exist() (in module decisio-
57 nengine.framework.dataspace.datasources.tests.test_datasource_
49 nengine.framework.dataspace.tests.test_datablock),
test_get_metadata_not_exist() (in module decisio-
58 nengine.framework.dataspace.tests.test_dataspace)
test_help() (in module decisio-
nengine.framework.tests.test_module_program_options),
91 test_get_taskmanager_exists() (in module decisio-
nengine.framework.dataspace.datasources.tests.test_datasource_
49 nengine.framework.tests.test_index_error() (in module decisio-
nengine.framework.logicengine.tests.test_fail_on_error),
71 test_get_taskmanager_exists() (in module decisio-
nengine.framework.dataspace.tests.test_dataspace)
test_insert() (in module decisio-
58 nengine.framework.dataspace.datasources.tests.test_datasource_
49 nengine.framework.dataspace.tests.test_dataspace),
test_get_taskmanager_not_exists()
58 test_insert() (in module decisio-
nengine.framework.config.tests.test_de_std),
34 test_get_taskmanager_not_exists()
test_jpath() (in module decisio-
58 nengine.framework.config.tests.test_de_std),
34 test_just_stop_no_error() (in module decisio-

```

<i>nengine.framework.dataspace.tests.test_Reaper</i> ), 56	<i>nengine.framework.tests.test_query_tool_server</i> ), 91
<i>test_loop_of_start_stop_in_clumps()</i> (in module <i>decisionengine.framework.dataspace.tests.test_Reaper</i> ), 56	<i>test_query_tool_since()</i> (in module <i>decisionengine.framework.tests.test_query_tool_server</i> ), 91
<i>test_Metadata_constructor()</i> (in module <i>decisionengine.framework.dataspace.tests.test_datablock</i> ), 57	<i>test_query_tool_with_no_server()</i> (in module <i>decisionengine.framework.engine.tests.test_query_tool_only</i> ), 66
<i>test_Metadata_set_state()</i> (in module <i>decisionengine.framework.dataspace.tests.test_datablock</i> ), 57	<i>test_query_tool_with_no_server_verbose()</i> (in module <i>decisionengine.framework.engine.tests.test_query_tool_only</i> ), 66
<i>test_minimal_jsonnet_right_extension()</i> (in module <i>decisionengine.framework.config.tests.test_config</i> ), 34	<i>test_reap_default_state()</i> (in module <i>decisionengine.framework.dataspace.tests.test_Reaper</i> ), 56
<i>test_minimal_jsonnet_wrong_extension()</i> (in module <i>decisionengine.framework.config.tests.test_config</i> ), 34	<i>test_reaper_can_reap()</i> (in module <i>decisionengine.framework.dataspace.tests.test_Reaper</i> ), 56
<i>test_misspecified_fact()</i> (in module <i>decisionengine.framework.logicengine.tests.test_fail_on_error</i> ), 72	<i>test_reset_connections()</i> (in module <i>decisionengine.framework.dataspace.datasources.tests.test_datasource</i> ), 49
<i>test_module_alias()</i> (in module <i>decisionengine.framework.tests.test_module_program_options</i> ), 91	<i>test_restart_channel()</i> (in module <i>decisionengine.framework.tests.test_restart_channel</i> ), 91
<i>test_module_structure()</i> (in module <i>decisionengine.framework.modules.tests.test_Module</i> ), 78	<i>test_rule_that_does_not_fire()</i> (in module <i>decisionengine.framework.logicengine.tests.test_cascaded_rules</i> ), 71
<i>test_multiple_consumes_declarations()</i> (in module <i>decisionengine.framework.modules.tests.test_module_decorators</i> ), 79	<i>test_rule_that_does_not_fire()</i> (in module <i>decisionengine.framework.logicengine.tests.test_pandas_fact</i> ), 72
<i>test_multiple_produces_declarations()</i> (in module <i>decisionengine.framework.modules.tests.test_module_decorators</i> ), 79	<i>test_rule_that_does_not_fire()</i> (in module <i>decisionengine.framework.logicengine.tests.test_rule_with_negated_fact</i> ), 72
<i>test_publisher_structure()</i> (in module <i>decisionengine.framework.modules.tests.test_Publisher</i> ), 78	<i>test_rule_that_does_not_fire()</i> (in module <i>decisionengine.framework.logicengine.tests.test_simple_configuration</i> ), 72
<i>test_query_tool_csv()</i> (in module <i>decisionengine.framework.tests.test_query_tool_server</i> ), 91	<i>test_rule_that_fires()</i> (in module <i>decisionengine.framework.logicengine.tests.test_cascaded_rules</i> ), 71
<i>test_query_tool_default()</i> (in module <i>decisionengine.framework.tests.test_query_tool_server</i> ), 91	<i>test_rule_that_fires()</i> (in module <i>decisionengine.framework.logicengine.tests.test_pandas_fact</i> ), 72
<i>test_query_tool_help()</i> (in module <i>decisionengine.framework.engine.tests.test_query_tool_only</i> ), 66	<i>test_rule_that_fires()</i> (in module <i>decisionengine.framework.logicengine.tests.test_rule_with_negated_fact</i> ), 72
<i>test_query_tool_invalid_product()</i> (in module <i>decisionengine.framework.tests.test_query_tool_server</i> ), 91	<i>test_rule_that_fires()</i> (in module <i>decisionengine.framework.logicengine.tests.test_simple_configuration</i> ), 72
<i>test_query_tool_json()</i> (in module <i>decisionengine.framework.tests.test_query_tool_server</i> ), 91	<i>test_simple_fact()</i> (in module <i>decisionengine.framework.logicengine.tests.test_simple_configuration</i> ), 72

```

    nengine.framework.logicengine.tests.test_facts), test_translate_all() (in module decisio-
71 nengine.framework.modules.tests.test_translate_product_name),
test_source_fail_can_be_fixed() 79
    (in module decisio- test_translate_illegal_characters()
nengine.framework.dataspace.tests.test_Reaper), (in module decisio-
56 nengine.framework.modules.tests.test_translate_product_name),
test_source_only_channel() (in module decisio- 79
nengine.framework.tests.test_error_on_acquire), test_translate_none() (in module decisio-
90 nengine.framework.modules.tests.test_translate_product_name),
test_source_structure() (in module decisio- 79
nengine.framework.modules.tests.test_Source), test_translate_simple() (in module decisio-
78 nengine.framework.modules.tests.test_translate_product_name),
test_start_delay() (in module decisio- 79
nengine.framework.dataspace.tests.test_Reaper), test_translate_with_underscores()
56 (in module decisio-
nengine.framework.modules.tests.test_translate_product_name),
test_start_from_nothing() (in module decisio- 79
nengine.framework.tests.test_start_with_no_channels),
93 test_trivial_configuration() (in module decisio-
nengine.framework.dataspace.tests.test_Reaper), 71
56 test_true_fact() (in module decisio-
nengine.framework.dataspace.tests.test_Reaper), 72
test_start_stop_stop() (in module decisio-
nengine.framework.dataspace.tests.test_Reaper), 72
56 test_true_literal_fact() (in module decisio-
nengine.framework.dataspace.tests.test_Reaper), 72
test_state_can_be_active() (in module decisio-
nengine.framework.dataspace.tests.test_Reaper), 72
56 test_update() (in module decisio-
nengine.framework.dataspace.datasources.tests.test_datasource_
test_state_sets_timer_and_uses_it() 49
    (in module decisio-
nengine.framework.dataspace.tests.test_Reaper), test_update() (in module decisio-
56 nengine.framework.dataspace.tests.test_dataspace),
test_stop_failing_source_proxy() 58
    (in module decisio-
nengine.framework.tests.test_source_proxy), test_update_bad() (in module decisio-
93 nengine.framework.dataspace.datasources.tests.test_datasource_
49
test_store_taskmanager() (in module decisio- test_update_bad() (in module decisio-
nengine.framework.dataspace.datasources.tests.test_datasource_ 58
49 nengine.framework.dataspace.tests.test_dataspace),
test_store_taskmanager() (in module decisio- test_valid_but_empty_config() (in module decisio-
nengine.framework.dataspace.tests.test_dataspace), nengine.framework.config.tests.test_config), 34
58 test_valid_dir() (in module decisio-
nengine.framework.config.tests.test_policies),
test_supports_config() (in module decisio- 34
nengine.framework.modules.tests.test_module_decorators), 79
79 test_working_source_proxy() (in module decisio-
nengine.framework.tests.test_source_proxy), 93
test_syntax_error() (in module decisio- test_wrong_configuration() (in module decisio-
nengine.framework.logicengine.tests.test_facts), 71
71
test_syntax_error_in_config_names_bad_file() test_wrong_product_names() (in module decisio-
    (in module decisio- nengine.framework.modules.tests.test_module_decorators),
34 79
test_transform_structure() (in module decisio- test_wrong_product_types() (in module decisio-
nengine.framework.modules.tests.test_Transform), nengine.framework.modules.tests.test_module_decorators),
78 79

```

[test\\_zdumps\(\)](#) (in module `decisionengine.framework.dataspace.tests.test_datablock_valid_config`), 60  
[test\\_zloads\(\)](#) (in module `decisionengine.framework.dataspace.tests.test_datablock_valid_value`), 57  
[Transform](#) (class in `decisionengine.framework.modules.Transform`), 81  
[transform\(\)](#) (decisionengine.framework.modules.Transform.Transform method), 81  
[transform\(\)](#) (decisionengine.framework.tests.TransformNOP.TransformNOP method), 88  
[transform\(\)](#) (decisionengine.framework.tests.TransformWithMissingProducersConsumes.TransformWithMissingProducersConsumes method), 89  
[TransformNOP](#) (class in `decisionengine.framework.tests.TransformNOP`), 88  
[TransformWithMissingProducersConsumes](#) (class in `decisionengine.framework.tests.TransformWithMissingProducersConsumes`), 89  
[translate\(\)](#) (in module `decisionengine.framework.modules.translate_product_name`), 82  
[translate\\_all\(\)](#) (in module `decisionengine.framework.modules.translate_product_name`), 82  
**U**  
[unguarded\\_access\(\)](#) (decisionengine.framework.engine.Workers.Workers method), 69  
[update\(\)](#) (decisionengine.framework.dataspace.datasource.DataSource method), 63  
[update\(\)](#) (decisionengine.framework.dataspace.datasources.null.NullDataSource method), 52  
[update\(\)](#) (decisionengine.framework.dataspace.datasource.workers.sql.PostgreSQL class in `decisionengine.framework.engine.Workers`), 69  
[update\(\)](#) (decisionengine.framework.dataspace.datasource.workers.sqlalchemy\_ds.datasource\_api.SQLAlchemyDS class in `decisionengine.framework.taskmanager.TaskManager`), 40  
[update\(\)](#) (decisionengine.framework.dataspace.datasources.sqlalchemy\_ds.SQLAlchemyDS method), 47  
[update\(\)](#) (decisionengine.framework.dataspace.dataspace.DataSpace.Workers (class in `decisionengine.framework.engine.Workers`), 69  
[Workers.Access](#) (class in `decisionengine.framework.engine.Workers`), 69  
[WorkingSourceProxy](#) (class in `decisionengine.framework.tests.WorkingSourceProxy`), 89  
**V**  
[valid\\_dir\(\)](#) (in module `decisionengine.framework.config.policies`), 36  
[valid\\_states](#) (decisionengine.framework.dataspace.datablock.Metadata

## Z

`zdumps()` (in module *decisionengine.framework.dataspace.datablock*),  
[61](#)

`zloads()` (in module *decisionengine.framework.dataspace.datablock*),  
[61](#)