Failure Documentation

Release

OpenStack Foundation

January 19, 2017

Contents

A python package that provides useful exception (aka failure) additions.

Contents:

CHAPTER 1

Classes

class failure.**Failure**(*exc_info=None*, *exc_args=None*, *exc_kwargs=None*, *exception_str=''*, *exc_type_names=None*, *cause=None*, *traceback_str=''*, *generated_on=None*)

Bases: failure._utils.StrMixin

An immutable object that represents failure.

Failure objects encapsulate exception information so that they can be re-used later to re-raise, inspect, examine, log, print, serialize, deserialize...

For those who are curious, here are a few reasons why the original exception itself *may* not be reraised and instead a reraised wrapped failure exception object will be instead. These explanations are *only* applicable when a failure object is serialized and deserialized (when it is retained inside the python process that the exception was created in the the original exception can be reraised correctly without issue).

- Traceback objects are not serializable/recreatable, since they contain references to stack frames at the location where the exception was raised. When a failure object is serialized and sent across a channel and recreated it is *not* possible to restore the original traceback and originating stack frames.
- The original exception *type* can not *always* be guaranteed to be found, certain nodes can run code that is not accessible/available when the failure is being deserialized. Even if it was possible to use pickle safely (which it is not) it would not *always* be possible to find the originating exception or associated code in this situation.
- The original exception *type* can not be guaranteed to be constructed in a *correct* manner. At the time of failure object creation the exception has already been created and the failure object can not assume it has knowledge (or the ability) to recreate the original type of the captured exception (this is especially hard if the original exception was created via a complex process via some custom exception __init__ method).
- The original exception *type* can not *always* be guaranteed to be constructed and/or imported in a *safe* manner. Importing *foreign* exception types dynamically can be problematic when not done correctly and in a safe manner; since failure objects can capture *any* exception it would be *unsafe* to try to import those exception types namespaces and modules on the receiver side dynamically (this would create similar issues as the pickle module has).

TODO(harlowja): use parts of http://bugs.python.org/issue17911 and the backport at https://pypi.python.org/pypi/traceback2/ to (hopefully) simplify the methods and contents of this object...

BASE_EXCEPTIONS = {2: ('exceptions.BaseException', 'exceptions.Exception'), 3: ('builtins.BaseException', 'builtins.Ex

Root exceptions of all other python exceptions (as a string).

See: https://docs.python.org/2/library/exceptions.html

SCHEMA = {'definitions': {'cause': {'additionalProperties': True, 'required': ['exception_str', 'traceback_str', 'exc_type_

Expected failure schema (in json schema format).

classmethod from_exc_info(*exc_info=None*, *retain_exc_info=True*, *cause=None*, *find_cause=True*)
Creates a failure object from a `sys.exc_info()` tuple.

classmethod from_exception(*exception*, *retain_exc_info=True*, *cause=None*, *find_cause=True*)
Creates a failure object from a exception instance.

classmethod validate(*data*)
Validate input data matches expected failure `dict` format.

matches(*other*)
Checks if another object is equivalent to this object.

Returns checks if another object is equivalent to this object

Return type boolean

exception
Exception value, or `None` if exception value is not present.

Exception value *may* be lost during serialization.

generated_on
Python major & minor version tuple this failure was generated on.

May be `None` if not provided during creation (or after if lost).

exception_str
String representation of exception.

exception_args
Tuple of arguments given to the exception constructor.

exception_kwargs
Dict of keyword arguments given to the exception constructor.

exception_type_names
Tuple of current exception type **names** (in MRO order).

exc_info
Exception info tuple or `None`.

See: **https://docs.python.org/2/library/sys.html#sys.exc_info for what** the contents of this tuple are (if none, then no contents can be examined).

traceback_str
Exception traceback as string.

static reraise_if_any(*failures*, *cause_cls_finder=None*)
Re-raise exceptions if argument is not empty.

If argument is empty list/tuple/iterator, this method returns None. If argument is converted into a list with a single `Failure` object in it, that failure is reraised. Else, a `WrappedFailure` exception is raised with the failure list as causes.

reraise(*cause_cls_finder=None*)
Re-raise captured exception (possibly trying to recreate).

check(**exc_classes*)
Check if any of `exc_classes` caused the failure.

Arguments of this method can be exception types or type names (strings **fully qualified**). If captured exception is an instance of exception of given type, the corresponding argument is returned, otherwise `None` is returned.

cause
Nested failure *cause* of this failure.

This property is typically only useful on 3.x or newer versions of python as older versions do **not** have associated causes.

Refer to **PEP 3134** and **PEP 409** and **PEP 415** for what this is examining to find failure causes.

pformat(*traceback=False*)
Pretty formats the failure object into a string.

iter_causes()
Iterate over all causes.

classmethod from_dict(*data*)
Converts this from a dictionary to a object.

to_dict(*include_args=True*, *include_kwargs=True*)
Converts this object to a dictionary.

Parameters

- **include_args** – boolean indicating whether to include the exception args in the output.
- **include_kwargs** – boolean indicating whether to include the exception kwargs in the output.

copy(*deep=False*)
Copies this object (shallow or deep).

Parameters **deep** – boolean indicating whether to do a deep copy (or a shallow copy).

CHAPTER 2

Helper functions

CHAPTER 3

Examples

CHAPTER 4

Indices and tables

- genindex
- modindex
- search