developer.skatelescope.org Documentation

Release 0.1.0-beta

Marco Bartolini

Oct 20, 2021

HOME

CHAPTER

ONE

LOGGING TRANSACTION IDS

A transaction context handler is available to inject ids fetched from the the skuid service into logs. The transaction id will be logged on entry and exit of the context handler. In the event of an exception, the transaction id will be logged with the exception stack trace. The ID generated depends on whether or not the SKUID_URL environment variable is set.

1.1 Example

```
from ska_ser_log_transactions import transaction

def command(self, parameter_json):
    parameters = json.reads(parameter_json)
    with transaction('My Command', parameters) as transaction_id:
        # ...
        parameters['transaction_id'] = transaction_id
        device.further_command(json.dumps(parameters))
        # ...
```

1.2 Asynchronous Example

```
from ska_ser_log_transactions import async_transaction

async def command(self, parameter_json):
    parameters = json.reads(parameter_json)
    async with async_transaction('My Command', parameters) as transaction_id:
        # ...
        parameters['transaction_id'] = transaction_id
        await device.further_command(json.dumps(parameters))
        # ...
```

1.3 Custom logger example

By default the context handler logs to the `ska.transaction` logger with default formatting. A custom logger can be used by passing it in via an optional argument.

```
import logging
from ska_ser_log_transactions import transaction

custom_logger = logging.getLogger(__name__)

def command(self, parameter_json):
    parameters = json.reads(parameter_json)
    with transaction('My Command', parameters, logger=custom_logger) as transaction_
↪id:
        # ...
        parameters['transaction_id'] = transaction_id
        device.further_command(json.dumps(parameters))
        # ...
```

1.4 Log messages

Log message formats:

- On Entry:
  - Transaction[id]: Enter[name] with parameters [arguments] marker[marker]
- On Exit:
  - Transaction[id]: Exit[name] marker[marker]
- On exception:
  - Transaction[id]: Exception[name] marker[marker] – Stacktrace –

The marker can be used to match entry/exception/exit log messages.

1.4.1 Example ska formatted logs for successful transaction

```
1|2020-10-01T12:49:31.119Z|INFO|Thread-210|log_entry|transactions.py
↪#154||Transaction[txn-local-20201001-981667980]: Enter[Command] with parameters [{}
↪] marker[52764]
1|2020-10-01T12:49:31.129Z|INFO|Thread-210|log_exit|transactions.py
↪#154||Transaction[txn-local-20201001-981667980]: Exit[Command] marker[52764]
```

1.4.2 Example ska formatted logs for failed transaction

```
1|2020-10-01T12:51:35.588Z|INFO|Thread-204|log_entry|transactions.py
↪#154||Transaction[txn-local-20201001-354400050]: Enter[Transaction thread [7]] with␣
↪parameters [{}] marker[21454]
1|2020-10-01T12:51:35.598Z|ERROR|Thread-204|log_exit|transactions.py
↪#149||Transaction[txn-local-20201001-354400050]: Exception[Transaction thread [7]]␣
↪marker[21454]
Traceback (most recent call last):
  File "python_file.py", line 27, in thread_with_transaction_exception
    raise RuntimeError("An exception has occurred")
RuntimeError: An exception has occurred
1|2020-10-01T12:51:35.601Z|INFO|Thread-204|log_exit|transactions.py
↪#154||Transaction[txn-local-20201001-354400050]: Exit[Transaction thread [7]]␣
↪marker[21454]
```

CHAPTER

TWO

REQUIREMENTS

The system used for development needs to have Python 3 and `pip` installed.

CHAPTER

THREE

INSTALL

3.1 From source

- Clone the repo

```
git clone git@gitlab.com:ska-telescope/ska-ser-log-transactions.git
```

- Install requirements

```
python3 -m pip install -r requirements.txt --extra-index-url https://nexus.engageska-
↪portugal.pt/repository/pypi/simple
```

- Install the package

```
python3 -m pip install .
```

3.2 From the Nexus PyPI

```
python3 -m pip install ska-ser-log-transactions --extra-index-url https://nexus.
↪engageska-portugal.pt/repository/pypi/simple
```

CHAPTER

FOUR

TESTING

- Install the test requirements

```
python3 -m pip install -r requirements-test.txt
```

- Run the tests

```
tox
```

- Lint

```
tox -e lint
```

CHAPTER

FIVE

WRITING DOCUMENTATION

The documentation generator for this project is derived from SKA's SKA Developer Portal repository

The documentation can be edited under `./docs/src`

5.1 Build the documentation

- Install the test requirements

```
python3 -m pip install -r requirements-test.txt
```

- Build docs

```
tox -e docs
```

The documentation can then be consulted by opening the file `./docs/build/html/index.html`

5.1.1 API documentation

This section details the public API for configuring application logging across the SKA project.

Python

The API for the configuration using Python is shown below.

Public API Documentation

Imports

Module init code.

ska_ser_log_transactions.**transaction**
alias of *ska_ser_log_transactions.transactions.Transaction*

ska_ser_log_transactions.**async_transaction**
alias of *ska_ser_log_transactions.transactions.AsyncTransaction*

Classes

class ska_ser_log_transactions.transactions.**TransactionBase**(*name: str, params: dict = {}, transaction_id: str = '', transaction_id_key: str = 'transaction_id', logger: Optional[<Mock id='140077651555984'>] = None*)

Transaction context handler.

Provides:

- Transaction ID::
  * Re-use existing transaction ID, if available
  * If no transaction ID, or empty or None, then generate a new ID
  * context handler returns the transaction ID used
- Log messages on entry, exit, and exception

```
def command(self, parameter_json):
    parameters = json.reads(parameter_json)
    with transaction('My Command', parameters) as transaction_id:
        # ...
        parameters['transaction_id'] = transaction_id
        device.further_command(json.dumps(pars))
        # ...

def command(self, parameter_json):
    parameters = json.reads(parameter_json)
    with transaction('My Command', parameters, transaction_id="123") as␣
↪transaction_id:
        # ...
        parameters['transaction_id'] = transaction_id
        device.further_command(json.dumps(pars))
        # ...

def command(self, parameter_json):
    parameters = json.reads(parameter_json)
    parameters["txn_id_key"] = 123
    with transaction('My Command', parameters, transaction_id_key="txn_id_key")
        as transaction_id:
        # ...
        parameters['transaction_id'] = transaction_id
        device.further_command(json.dumps(pars))
        # ...
```

Log message formats:

On Entry:

Transaction[id]: Enter[name] with parameters [arguments] marker[marker]

On Exit:

Transaction[id]: Exit[name] marker[marker]

On exception:

Transaction[id]: Exception[name] marker[marker]

Stacktrace

log_entry()

Log the entry message

log_exit(*exc_type*)

Log the exit message and exception if it occurs

Parameters **exc_type** (*exception_type*) – Exception type

class ska_ser_log_transactions.transactions.**Transaction**(*name: str, params: dict = {}, transaction_id: str = '', transaction_id_key: str = 'transaction_id', logger: Optional[<Mock id='140077651555984'>] = None*)

Bases: *ska_ser_log_transactions.transactions.TransactionBase*

class ska_ser_log_transactions.transactions.**AsyncTransaction**(*name: str, params: dict = {}, transaction_id: str = '', transaction_id_key: str = 'transaction_id', logger: Optional[<Mock id='140077651555984'>] = None*)

Bases: *ska_ser_log_transactions.transactions.TransactionBase*

class ska_ser_log_transactions.transactions.**TransactionIdGenerator**

TransactionIdGenerator retrieves a transaction id from skuid. Skuid may fetch the id from the service if the SKUID_URL is set or alternatively generate one.

CHAPTER

SIX

SKA TRANSACTION LOGGING

These are all the packages, functions and scripts that form part of the project.

- *API documentation*

PYTHON MODULE INDEX

S

INDEX