
Coralogix Python SDK

Release 2.0.1

May 27, 2021

Table of Contents

1	Installation	3
2	Configuration	5
3	Implementation	7
4	Configuration under uWSGI	9
5	Framework integrations	11
5.1	Django	11
5.2	Flask	12

This package provides logging suites integrated with Coralogix logs analytics platform.

CHAPTER 1

Installation

Install the Coralogix python logger with pip:

```
$ pip install coralogix_logger
```

or directly from the sources:

```
$ git clone https://github.com/coralogix/python-coralogix-sdk.git
$ cd sdk-python
$ python setup.py install
```


CHAPTER 2

Configuration

Using Coralogix Python SDK requires these mandatory parameters:

- **PRIVATE KEY** - A unique ID which represents your company. This ID will be sent to your mail once you register to Coralogix.
- **APPLICATION NAME** - The name of your main application. For example, a company named “*SuperData*” could insert the “*SuperData Production*” string parameter; or if they want to debug their test environment, they might insert the “*SuperData – Test*”.
- **SUBSYSTEM NAME** - Your application probably has multiple subsystems. For example: Backend servers, Middleware, Frontend servers etc. In order to help you examine and filter the data you need, inserting the subsystem parameter is vital.

CHAPTER 3

Implementation

Adding Coralogix logging handler in your logging system:

```
import logging
# For version 1.*
from coralogix.coralogix_logger import CoralogixLogger
# For version 2.*
from coralogix.handlers import CoralogixLogger

PRIVATE_KEY = "[YOUR_PRIVATE_KEY_HERE]"
APP_NAME = "[YOUR_APPLICATION_NAME]"
SUB_SYSTEM = "[YOUR_SUBSYSTEM_NAME]"

# Get an instance of Python standard logger.
logger = logging.getLogger("Python Logger")
logger.setLevel(logging.DEBUG)

# Get a new instance of Coralogix logger.
coralogix_handler = CoralogixLogger(PRIVATE_KEY, APP_NAME, SUB_SYSTEM)

# Add coralogix logger as a handler to the standard Python logger.
logger.addHandler(coralogix_handler)

# Send message
logger.info("Hello World!")
```

Also, you can configure the SDK with dictConfig for Python logging library:

```
import logging

PRIVATE_KEY = '[YOUR_PRIVATE_KEY_HERE]'
APP_NAME = '[YOUR_APPLICATION_NAME]'
SUB_SYSTEM = '[YOUR_SUBSYSTEM_NAME]'

logging.config.dictConfig({
```

(continues on next page)

(continued from previous page)

```
'version': 1,
'disable_existing_loggers': False,
'formatters': {
    'default': {
        'format': '[%(asctime)s]: %(levelname)s: %(message)s',
    }
},
'handlers': {
    'coralogix': {
        'class': 'coralogix.handlers.CoralogixLogger',
        'level': 'DEBUG',
        'formatter': 'default',
        'private_key': PRIVATE_KEY,
        'app_name': APP_NAME,
        'subsystem': SUB_SYSTEM,
    }
},
'root': {
    'level': 'DEBUG',
    'handlers': [
        'coralogix',
    ]
},
'loggers': {
    'backend': {
        'level': 'DEBUG',
        'handlers': [
            'coralogix',
        ]
    }
}
})
```

Configuration under uWSGI

By default uWSGI does not enable threading support within the Python interpreter core. This means it is not possible to create background threads from Python code. As the Coralogix logger relies on being able to create background threads (for sending logs), this option is required.

You can enable threading either by passing **--enable-threads** to uWSGI command line:

```
$ uwsgi wsgi.ini --enable-threads
```

Another option is to enable threads in your wsgi.ini file:

wsgi.ini:

```
...
enable-threads = true
...
```

Framework integrations

5.1 Django

To enable Django logging to Coralogix you'll need to add the following lines to your **settings.py**:

```
...  
  
LOGGING = {  
    'version': 1,  
    'disable_existing_loggers': False,  
    'formatters': {  
        'default': {  
            'format': '[%(asctime)s]: %(levelname)s: %(message)s',  
        }  
    },  
    'handlers': {  
        'coralogix': {  
            'class': 'coralogix.handlers.CoralogixLogger',  
            'level': 'DEBUG',  
            'formatter': 'default',  
            'private_key': '[YOUR_PRIVATE_KEY_HERE]',  
            'app_name': '[YOUR_APPLICATION_NAME]',  
            'subsystem': '[YOUR_SUBSYSTEM_NAME]',  
        }  
    },  
    'root': {  
        'level': 'DEBUG',  
        'handlers': [  
            'coralogix',  
        ]  
    },  
    'loggers': {  
        'backend': {  
            'level': 'DEBUG',
```

(continues on next page)

(continued from previous page)

```

        'handlers': [
            'coralogix',
        ]
    }
}

```

5.2 Flask

To enable Flask logging to Coralogix you can use the following code template:

```

#!/usr/bin/env python
# -*- coding: utf-8 -*-

from logging.config import dictConfig
from flask import Flask

dictConfig({
    'version': 1,
    'disable_existing_loggers': False,
    'formatters': {
        'default': {
            'format': '[%(asctime)s]: %(levelname)s: %(message)s',
        }
    },
    'handlers': {
        'coralogix': {
            'class': 'coralogix.handlers.CoralogixLogger',
            'level': 'DEBUG',
            'formatter': 'default',
            'private_key': '[YOUR_PRIVATE_KEY_HERE]',
            'app_name': '[YOUR_APPLICATION_NAME]',
            'subsystem': '[YOUR_SUBSYSTEM_NAME]',
        }
    },
    'root': {
        'level': 'DEBUG',
        'handlers': [
            'coralogix',
        ]
    },
    'loggers': {
        'backend': {
            'level': 'DEBUG',
            'handlers': [
                'coralogix',
            ]
        }
    }
})

app = Flask(__name__)

...

```