heka-py-raven Documentation

Release

Rob Miller, Victor Ng

December 06, 2013

Contents

heka-py-raven is a plugin extension for *heka-py <http://github.com/mozilla-services/heka-py>*. heka-py-raven provides logging extensions to capture stacktraces and some frame information such as local variables to faciliate debugging.

The plugin acts as a thin wrapper around the Raven <https://github.com/dcramer/raven> library for Sentry.

More information about how Mozilla Services is using heka (including what is being used for a router and what endpoints are in use / planning to be used) can be found on the Read The Docs page.

This version of heka-py-raven is compatible with version 3 of the Sentry protocol. This should make it compatible with Sentry 5.1->5.4 and Raven clients upto, but not including Raven 3.5.

To install heka-py-raven use:

```
pip install heka-py-raven[protocol_v3]
```

CHAPTER 1

Welcome to heka-py-raven's documentation!

1.1 Configuration

Configuration is normally handled through heka-py's configuration system using INI configuration files. A raven plugin must use the *heka_raven.raven_plugin:config_plugin* as the provider of the plugin.

The heka-py-raven plugin exports a name of 'raven' which is bound into the client.

In the following example, we will bind a method *raven* into the heka-py client so that we can send stacktrace information to the hekad server.

```
[heka_plugin_ravensection]
provider=heka_raven.raven_plugin:config_plugin
dsn = udp://username:password@sentryhost.com:9001/2
```

Alternatively, if loading heka-py's configuration by means of a *dict*, the plugin can be loaded with the example *dict* that follows, which will also bind the method *raven* to the heka-py client.

```
{
    'stream_class': 'heka.streams.StdOutStream',
    'plugins' : {
        'raven' : ['heka_raven.raven_plugin:config_plugin',
                   {'dsn': "udp://username:password@sentryhost.com:9001/2"}]
    }
}
```

You may also set 2 optional settings :

- **logger: The name that heka-py will use when logging messages. By** default this is set by the heka-py client.
- severity: The default severity of the error. Default severity level is 3 as defined by *heka.client:SEVERITY.ERROR* <https://github.com/mozilla-services/heka-py/blob/master/heka/client.py>

1.2 Usage

Obtaining a client should probably be done through your framework. Please refer to the heka documentation for complete details.

That said, if you are impatient you can obtain a client using *get_client*. We strongly suggest you do not do this though.

```
from heka.holder import get_client
get_client('myapp',
        {
         'stream_class': 'heka.streams.StdOutStream',
          'plugins' : {
              'raven' : ['metlog_raven.raven_plugin:config_plugin',
                         {'dsn': "udp://username:password@sentryhost.com:9001/2"}]
                       }
        })
```

Note that the above sender configuration will only route messages to stdout so that you can verify that logging is happening during development.

Logging exceptions is passive. The raven plugin will not rethrow an exception automatically if you invoke the method on the client directly. This prevents you from seeing exceptions raised from within the plugin code.

If you use the decorator syntax, the plugin will automatically rethrow the exception for you.

The raven plugin provides 2 ways to send messages. You can use decorator syntax, or access the plugin through the standard client.

Using the example configuration listed above, the following snippet will log catcha n exception and fire it off to details.

```
from heka.holder import get_client

client = get_client('some_client_name',
             {
                'stream_class': 'heka.streams.StdOutStream',
                 'plugins' : {
                      'raven' : ['heka.raven_plugin:config_plugin',
                                 {'dsn': "udp://username:password@sentryhost.com:9001/2"}]
                                }
             })
try:
    do_some_exception_throwing_thing()
except:
    client.raven('something bad happened')

    # re-raise the exception so someone can properly handle
    # the error
    raise
```

or you can use the decorator syntax

```
from heka.holder import get_client
from heka_raven.raven_plugin import capture_stack

client = get_client('some_client_name',
             {
                'stream_class': 'heka.senders.StdOutStream',
                 'plugins' : {
                      'raven' : ['heka_raven.raven_plugin:config_plugin',
                                 {'dsn': "udp://username:password@sentryhost.com:9001/2"}]
```

```
                                    }
            })

@capture_stack
def some_function(foo, bar):
    # Some code here that throws exceptions
    do_some_exception_throwing_thing()

some_function('foo', 'bar')
```

1.3 Compatibility

This version of heka-py-raven has only been tested to work against Raven 2.0.6 and Sentry 5.0.13. Other versions may work for you, but they have not been tested.

1.4 Data structure

The raven plugin will send quite a lot of information. Important keys to note:

- The type of the message will be set to 'stacktrace'
- The severity is set to 3 (SEVERITY.ERROR)

In the context of determining the source of the error, the 'fields' section of the heka-py blob has 2 keys which are of particular interest.

- culprit: This is the function name that threw the exception
- frames: This is a list of stackframes captured. The frames are ordered from inner most to outer most frame. The frame that caused the exception will be at the end of the list.

Each frame is represented as a dictionary with the keys:

> abs_path: Absolute path to the current module context_line: source code of the function where the exception passed through filename: relative path filename of the current module function: function name that the exception passed through lineno: line number in the source module where the exception passed through module: module name pre_context: 2 source lines prior to the exception in the current module post_context: 2 source lines after the exception in the current module: vars: local variables at the time immediately after the exception has been caught

---

TODO: Replace with ProtobufExample

A sample of the JSON emitted is provided below to illustrate all the details that are captured. This sample below is generated by the included test suite.

```
{u'env_version': u'0.8',
 u'fields': {u'epoch_timestamp': 1336490211.8597479, 'msg': ''},
 u'logger': u'myapp',
 u'payload': u'eJzFVm1v2zYQ/iuEvtjBbL0rtY22QNd2n1ZgQNp+aBsIFEUpbGhSICnPWuH/viMppbFTLwXWbgYk66jnjnf3PI
 u'severity': 3,
 u'timestamp': u'2012-05-08T15:16:51.859750',
 u'type': u'sentry'}
```

Heka adds a single key 'msg' which is an optional string argument to attach to the stacktrace. The message string is included in both the Heka 'fields' dictionary as well as within the Raven binary blob in the 'extra' key.

The sentry data is packed into the payload in it's native zlib/base64 encoded format for simplicity. Unpacked, the raven encodes the following information.

```
{'checksum': 'ccd44e9a94c1fe48503b38dd95859c95',
 'culprit': 'test_heka.exception_call2',
 'event_id': '443e20a669b04ba38711f29e74376392',
 'extra': {'msg': ''},
 'level': 40,
 'message': 'ZeroDivisionError: integer division or modulo by zero',
 'modules': {},
 'project': 1,
 'sentry.interfaces.Exception': {'module': 'exceptions',
                                 'type': 'ZeroDivisionError',
                                 'value': 'integer division or modulo by zero'},
 'sentry.interfaces.Stacktrace': {'frames': [{'abs_path': '/Users/victorng/dev/heka-py-raven/heka_rav
                                             'context_line': '        exception_call1(5, 5)',
                                             'filename': 'tests/test_heka.py',
                                             'function': 'test_plugins_config',
                                             'lineno': 93,
                                             'module': 'test_heka',
                                             'post_context': ['    except:',
                                                              "        client.raven('some_logger_nar
                                             'pre_context': ['        return exception_call2(y, x, 4
                                                             '',
                                                             '    try:'],
                                             'vars': {'cfg_txt': '\n    [heka]\n    sender_class = h
                                                      'client': '<heka.client.HekaClient object at (
                                                      'client_from_text_config': '<function client_f
                                                      'exception_call1': '<function exception_call1
                                                      'exception_call2': '<function exception_call2
                                                      'json': "<module 'json' from '/System/Library/
                                            {'abs_path': '/Users/victorng/dev/heka-py-raven/heka_rav
                                             'context_line': '        return exception_call2(y, x, 4
                                             'filename': 'tests/test_heka.py',
                                             'function': 'exception_call1',
                                             'lineno': 90,
                                             'module': 'test_heka',
                                             'post_context': ['',
                                                              '    try:'],
                                             'pre_context': ['        return a + b + c / (a - b)',
                                                             '',
                                                             '    def exception_call1(x, y):'],
                                             'vars': {'exception_call2': '<function exception_call2
                                                      'x': 5,
                                                      'y': 5}},
                                            {'abs_path': '/Users/victorng/dev/heka-py-raven/heka_rav
                                             'context_line': '        return a + b + c / (a - b)',
                                             'filename': 'tests/test_heka.py',
                                             'function': 'exception_call2',
                                             'lineno': 87,
                                             'module': 'test_heka',
                                             'post_context': ['',
                                                              '    def exception_call1(x, y):'],
                                             'pre_context': ["    client = client_from_text_config(c
```

```
                                                    '',
                                                    '    def exception_call2(a, b, c):'],
                                        'vars': {'a': 5,
                                                 'b': 5,
                                                 'c': 42}}]},
 'server_name': 'Victors-MacBook-Air.local',
 'site': None,
 'time_spent': None,
 'timestamp': '2012-05-08T15:08:38.657850Z'}
```

Raven allows extension of the data by adding new keys to the sentry data through the 'extra' key, but this is not currently supported.

1.5 raven_plugin

CHAPTER 2

Indices and tables

- *genindex*
- *modindex*
- *search*