OpenStack APIs: Fault Messages Verbosity

OpenStack Oslo Project > Blueprint proposal

1) There might be failure scenarios in which the information that is given within the standard API fault message is limited while using the message and detailed message only.
2) The discussed feature is aim to allow adding more readable "log" entries to the fault message, allowing expanding the detailed message attribute.
3) In each OpenStack project, any flow that will need to propagate more informative error description will be able to do so using these "log" entries.
4) The verbosity level will be set by the user/client itself by setting a dedicated request header, whose value will be "info", "debug" or "none" (the last one will be the default if the header won't be existed).
5) Further detailed design within the Oslo project will be made upon blueprint approval.
6) Example:
    a. Assume a case of a fault message regarding internal failure that ends with error code 500:

{"computeFault": { "code":500, "details":"Internal Server Error", "message":"Some Failure"}}

If the use would not have set a request header "X-verbosity" or set it to "none" – he/she would have got the response above.

b. But, if the user would have set the request header "X-verbosity" to "info", he/she could have got the following response:

```
{
    "computeFault":
    {
        "code":500,
        "details":"Internal Server Error",
        "message":"Some Failure",
        "fault_logs":
        [
            {
                "error_code":"error code 1",
                "message":"some message 1"
            },
            {
                "error_code":"error code 2",
                "message":"some message 2"
            }
        ]
    }
}
```

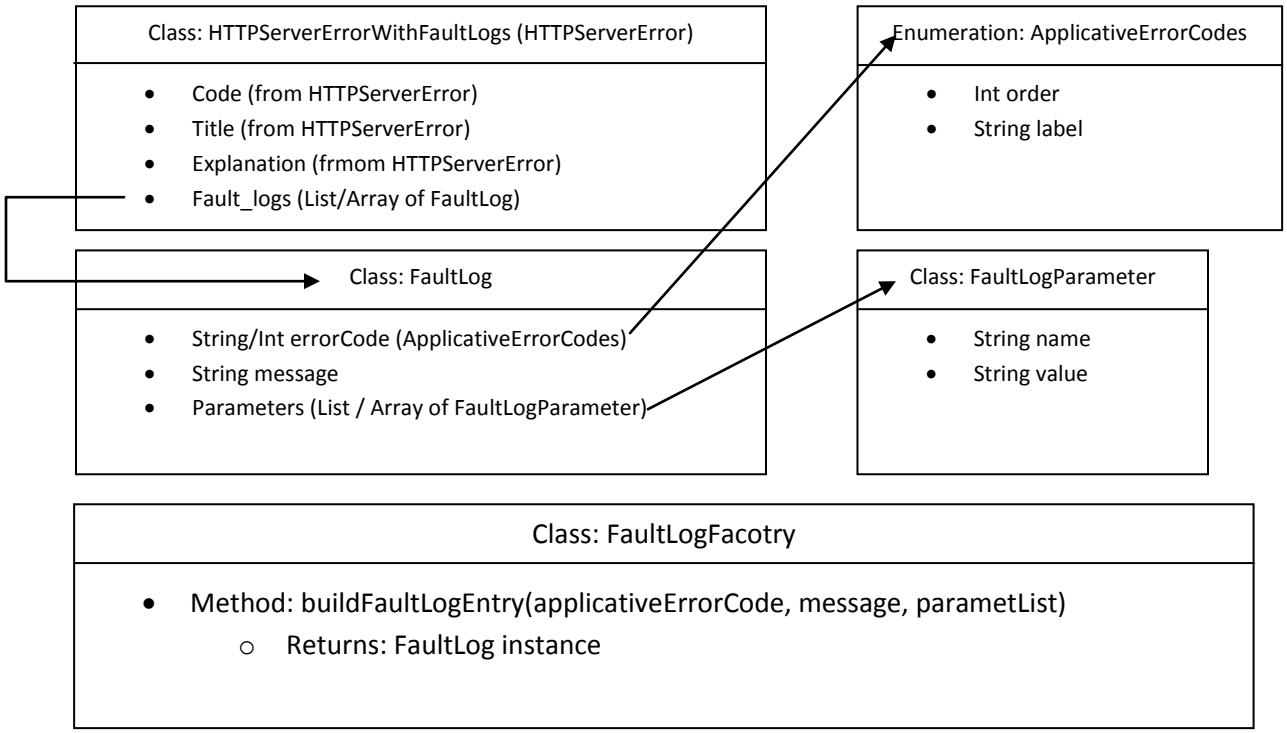
c. But, if the user would have set the request header “X-verbosity” to “debug”, he/she could have got even more failure information, for example specifying even wrong parameters list:

```
{
    “computeFault”:
    {
        “code”:500,
        “details”:”Internal Server Error”,
        “message”:”Some Failure”,
        “fault_logs”:
        [
            {
                “error_code”:”error code 1”,
                “message”:”some message 1”
                “parameters”:
                [
                    {“name”:”paramName1”, “value”:”paramValue1”},
                    {“name”:”paramName2”, “value”:”paramValue2”}
                ]
            },
            {
                “error_code”:”error code 2”,
                “message”:”some message 2”
                “parameters”:
                [
                    {“name”:”paramName3”, “value”:”paramValue3”}
                ]
            }
        ]
    }
}
```

Proposed Classes/Methods/API

- The general idea is to have a singleton FaultLogFactory instance (for each OpenStack service its own singleton instance, see below class def) that the developer will be able to refer to and call the buildFaultLogEntry method. The developer will know in what scenario the fault case exists and provide an applicative error code , a message and maybe a list relevant parameters.
- The developer will be able to send a list of several fault log entries (as much as needed) within a single fault message to the user.

- The proposed infrastructure will allow a better verbosity in a more complicated fault scenarios in the system.
- Suggested new classes involved are extending the class HTTPServerError (exc.py) as described (this can easily be done in equivalent way on HTTPClientError (exc.py)):



- Code Sample
  - Assume you manage a resource allocation process;
  - There are two reasons because of which the allocation process may fail – lack of space OR another needed resource is missing.
  - The process should fail only if both reasons occur;
  - The developer may want to inform the user with both errors in the same fault message

```
import oslo.FaultLogFacotry

import oslo.FaultLog

...

applicative_error_codes = Enum('IncefficientSpace', 'AnotherResourceMissing')

...
```

```
try:

      # some code

except Error:

      faultLog1 =
      fault_log_facotry.buildFaultLogEntry(applicative_error_codes.Incefficie
      ntSpace, 'No more space')

      faultLog2 =
      fault_log_facotry.buildFaultLogEntry(applicative_error_codes.AnotherRes
      ourceMissing, 'Another resource is missing', ['resource'='some resource
      type'])

      faultLogList = [faultLog1, faultLog2]

      raise webob.exc.HTTPServerErrorWithFaultLogs(faultLogList)
```

- Class HTTPServerErrorWithFaultLogs will have its own **init method**, that beside HTTPServerError.init, an array of pre-created list of Fault_logs will also be initiated.

- Once the developer will initiate a new HTTPServerErrorWithFaultLogs, it will first construct a list of FaultLogs. Each FaultLog will be initiated by calling FaultLogFacotry. buildFaultLogEntry with the relevant ApplicativeErrorCodes (one of pre-defined applicative error codes) and, if needed, a list of FaultLogParameters (list of {name=value}s)

- The idea is that every OpenStack project will be able to import HTTPServerErrorWithFaultLogs, and the developer will be able to call raise webob.exc. HTTPServerErrorWithFaultLogs(faultLogList)

- Use case example:
    - Assume you manage a resource allocation process;
    - There are two reasons because of which the allocation process may fail – lack of space OR another needed resource is missing.
    - The process should fail only if both reasons occur;
    - The developer may want to inform the user with both errors in the same fault message