Metrics and Schema Mappings

The problem we are trying to resolve is that today we don't have a transparent way of mapping from one telemetry framework data sets to other telemetry framework data sets. There is a multitude of write plugins that normalize data from one set to the other, however they are implementation specific (per framework/protocol they implement). We'd like to leverage a common description language to establish a methodology for normalizing data sets between frameworks, i.e. a way of describing mapping from one data set to another. Then use generic write plugins that can parse the appropriate schema, do the real mapping on the fly and submit the information to the relevant end point (unfortunately there still might need to be specific write plugins/utilities for authentication with the relevant end point).

The description language of choice is JSON.

To visualize the schema you can copy paste them to http://chris.photobooks.com/json/

To help work through the schema proposals, we will work through a use case to normalize data from the internal collectd format to the ceilometer format.

Ceilometer use case

Ceilometer provides the possibility to submit samples via the REST API to allow users to send custom samples into this service. The samples that can be sent to Telemetry are not limited to the actual existing meters. There is a possibility to provide data for any new, customer defined counter by filling out all the required fields of the POST request. If the sample corresponds to an existing meter, then the fields like `meter-type` and meter name should be matched accordingly.

A standard meter definition looks like:

```
metric:
  - name: 'meter name'
    event_type: 'event name'
    type: 'type of meter eg: gauge, cumulative or delta'
    unit: 'name of unit eg: MB'
    volume: 'path to a measurable value eg: $.payload.size'
    resource_id: 'path to resource id eg: $.payload.id'
    project_id: 'path to project id eg: $.payload.owner'
```

The `meter.yaml` file, it contains the sample definitions for all the meters that Telemetry can collect.
The required fields for sending a sample using the command-line client are:

- ID of the corresponding resource. (`--resource-id`)
- Name of meter. (`--meter-name`)
- Type of meter. (`--meter-type`)

  Predefined meter types:

  - Gauge
  - Delta
  - Cumulative
- Unit of meter. (`--meter-unit`)
- Volume of sample. (`--sample-volume`)

To post a list of new Samples to Telemetry.

Parameters:	• **direct** (str) – a flag indicates whether the samples will be posted directly to storage or not. • **samples** (list(OldSample)) – a list of samples within the request body.

Ceilometer OldSample schema

```
{
    "counter_name": "instance",
    "counter_type": "gauge",
    "counter_unit": "instance",
    "counter_volume": 1.0,
    "message_id": "5460acce-4fd6-480d-ab18-9735ec7b1996",
    "project_id": "35b17138-b364-4e6a-a131-8f3099c5be68",
    "recorded_at": "2015-01-01T12:00:00",
    "resource_id": "bd9431c1-8d69-4ad3-803a-8d4a6b89fd36",
    "resource_metadata": {
        "name1": "value1",
        "name2": "value2"
    },
    "source": "openstack",
    "timestamp": "2015-01-01T12:00:00",
    "user_id": "efd87807-12d2-4b38-9c70-5f5c2ac427ff"
}
```

Currently data normalization or mappings are managed manually in the ceilometer write plugin:

Units for meter samples are set manually in https://github.com/openstack/collectd-ceilometer-plugin/blob/master/collectd_ceilometer/units.py:

```
# Unit mappings in alphabetical order

UNITS = {

    'apache.apache_idle_workers':   'Workers',

    'apache.apache_bytes':          'B/s',

    'apache.apache_requests':       'Req/s',

    'apache.apache_scoreboard':     'Slots',

    'apache.apache_connections':    'Connections',



    'apcups.timeleft':      's',

    'apcups.temperature':   '°C',

    'apcups.percent':       'Load',

    'apcups.charge':        'Ah',

    'apcups.frequency':     'Hz',

    'apcups.voltage':       'V',


...
```

and the meter code itself can be found here: https://github.com/openstack/collectd-ceilometer-plugin/blob/master/collectd_ceilometer/meters/base.py. Field s in the ceilometer write plugin at the moment are mapped as follows:

As you can see, the meter sample aren't mapped to look like the meter samples submitted from an OpenStack service where there is overlap.

So where there is overlap in meters in terms of the samples collected by collectd and published to ceilometer, we would like to describe the mapping in a generic way.

Proposal 1

Looking at a combine Schema. This schema describes the collectd internal types, and the ceilometer fields that reference them

```
{
        "$schema": "http://json-schema.org/draft-04/schema#",
        "title": "collectd Base Schema",
        "description": "A schema that defines the collectd meters and events",
        "definitions": {
            "value list": {
                    "description": "Statistics in collectd consist of a value list",
                        "type": "object",
                        "properties": {
                                "values" : {
                                        "type": "array",
                                        "anyOf" : [
                                                { "type" : "object", "properties" : { "absolute" : { "type" :
"number" } } },
                                                { "type" : "object", "properties" : { "counter" : { "type" :
"number" } } },
                                                { "type" : "object", "properties" : { "derive" : { "type" :
"number" } } },
                                                { "type" : "object", "properties" : { "gauge" : { "type" :
"number" } } }
                                        ]
                                },
                                "value length" : {
                                        "description": "The number of values in the data set",
                                        "type": "number"
                                },
                                "time" : {
                                        "description": "Time stamp at which the value was collected",
                                        "type": "number"
                                },
                                "Interval" : {
                                        "description": "interval at which to expect a new value",
                                        "type": "number"
                                },
                                "host" : {
                                        "description": "used to identify the host",
                                        "type": "string"
                                },
                                "plugin" : {
                                        "description": "used to identify the plugin",
                                        "type": "string"
                                },
                                "plugin instance" : {
                                        "description": "used to group a set of values together",
                                        "type": "string"
                                },
                                "type" : {
                                        "description": "unit used to measure a value",
                                        "type": "string"
                                },
                                "type instance " : {
                                        "description": "used to distinguish between values that have an
identical type",
                                        "type": "string"
                                },
                                "metadata" : {
                                        "description": "an opaque data structure that enables the passing of
additional information about a value list",
                                        "type": "string"
                                }
                        }
```

```
                },
                "notifications": {
                        "description": "Notifications in collectd are generic messages",
                        "type": "object",
                        "properties": {
                                "severity" : {
                                                "description": "can be one of OKAY, WARNING, and FAILURE",
                                                "type": "string"
                                },
                                "time" : {
                                        "description": "Time stamp at which the event was collected",
                                        "type": "number"
                                },
                                "message" : {
                                        "description": "The notification message",
                                        "type": "string"
                                },
                                "host" : {
                                        "description": "used to identify the host",
                                        "type": "string"
                                },
                                "plugin" : {
                                        "description": "used to identify the plugin",
                                        "type": "string"
                                },
                                "plugin instance" : {
                                        "description": "used to group a set of values together",
                                        "type": "string"
                                },
                                "type" : {
                                        "description": "unit used to measure a value",
                                        "type": "string"
                                },
                                "type instance " : {
                                        "description": "used to distinguish between values that have an
identical type",
                                        "type": "string"
                                },
                                "metadata" : {
                                        "description": "an opaque data structure that enables the passing of
additional information about a value list",
                                        "type": "string"
                                }
                        }
                }
        },
        "counter_name": [
                        {"$ref": "#/definitions/value list/plugin"},
            {"$ref": "#/definitions/value list/plugin instance"}

        ],
        "counter_unit": [
            {"$ref": "#/definitions/value list/type_instance"},
                {"$ref": "#/definitions/value list/type"}
        ],
        "counter_volume": {
                "$ref": "#/definitions/value list/values"
        },
        "resource_id": [
                {"$ref": "#/definitions/value list/host"},
                {"$ref": "#/definitions/value list/plugin"},
                {"$ref": "#/definitions/value list/plugin instance"}
        ],
        "timestamp": {
                "$ref": "#/definitions/value list/time"
        },
        "resource_metadata": {
        "$ref": "#/definitions/value list/metadata"
    },
        "source": {
            "type": "string"
```

```
        },
        "user_id": {
            "type": "string"
        }
}
```

Issues:

Proposal 1 is too high level and doesn't get into the internals of what meters get mapped to what and what units should be associated with the meters /events - in essence it doesn't improve on what's there today for overlapping meters. However **is good for new meters**.

Proposal 2

Involves 2 schema:

1. The final message format schema
2. The mapping schema

The final message format schema for the ceilometer sample is:

```
{
    "counter_name": "instance",
    "counter_type": "gauge",
    "counter_unit": "instance",
    "counter_volume": 1.0,
    "message_id": "5460acce-4fd6-480d-ab18-9735ec7b1996",
    "project_id": "35b17138-b364-4e6a-a131-8f3099c5be68",
    "recorded_at": "2015-01-01T12:00:00",
    "resource_id": "bd9431c1-8d69-4ad3-803a-8d4a6b89fd36",
    "resource_metadata": {
        "name1": "value1",
        "name2": "value2"
    },
    "source": "openstack",
    "timestamp": "2015-01-01T12:00:00",
    "user_id": "efd87807-12d2-4b38-9c70-5f5c2ac427ff"
}
```

The proposed mapping schema is:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "collectd Base Schema",
    "description": "A schema that defines the collectd meters and events",
    "definitions": {
        "collectd_field": {
            "description": "collectd_fields",
            "type": "object",
            "properties": {
                "anyOf" : {
                    "Interval" : {
                        "description": "interval at which to expect a new value",
                        "type": "number"
                    },
                    "host" : {
                        "description": "used to identify the host",
                        "type": "string"
                    },
```

```
                    "message" : {
                        "description": "The notification message",
                        "type": "string"
                    },
                    "metadata" : {
                        "description": "an opaque data structure that enables the passing of additional
information about a value list",
                        "type": "string"
                    },
                    "plugin" : {
                        "description": "used to identify the plugin",
                        "type": "string"
                    },
                    "plugin instance" : {
                        "description": "used to group a set of values together",
                        "type": "string"
                    },
                    "severity" : {
                        "description": "can be one of OKAY, WARNING, and FAILURE",
                        "type": "string"
                    },
                    "time" : {
                        "description": "Time stamp at which the value was collected",
                        "type": "number"
                    },
                    "type" : {
                        "description": "unit used to measure a value",
                        "type": "string"
                    },
                    "type instance " : {
                        "description": "used to distinguish between values that have an identical type",
                        "type": "string"
                    },
                    "values" : {
                        "type": "object",
                        "anyOf" : {
                            "absolute" : { "type" : "string" } ,
                            "counter" : { "type" : "string" },
                            "derive" : { "type" : "string" },
                            "gauge" : { "type" : "string" },
                                                "actual_values" : { "type" : "string" }
                        }
                    },
                    "value length" : {
                        "description": "The number of values in the data set",
                        "type": "number"
                    }
                }
            }
        },
        "mappings": {
            "description": "mapping pair",
            "type": "object",
            "properties": {
                "oneOf": {
                "$ref": "#/definitions/collectd_field",
                "collectd_string": { "type": "string" }
                },
                "new_string": { "type": "string" },
                                "new_value": { "type": "string" }
            }
        },
        "mappings_with_conversion": {
            "description": "Mappings from collectd timestamps to other framework timestamps",
            "type": "object",
            "properties": {
                "transform": {
                    "description": "array of mappings from collectd to other frameworks",
                    "type": "array",
                    "items": {
                        "$ref": "#/definitions/mapping",
```

```
                        "conversion": "string"
                    }
                }
            },
            "required": [ "mappings" ]
        }
    },
    "field_mappings": {
        "description": "array of field_mappings",
        "type": "array",
        "items": {
                    "set": {
                            "description": "set of associated of field_mappings",
                                "type": "array",
                                "anyOf":{
                                        "$ref": "#/definitions/mappings",
                                        "$ref1": "#/definitions/mappings_with_conversion"
                                }
                        }
        }
    }
}
```

Taking the following ceilometer meter as an example:

```
  - name: 'compute.node.cpu.percent'

    event_type: 'compute.metrics.update'

    type: 'gauge'

    unit: 'percent'

    volume: $.payload.metrics[?(@.name='cpu.percent')].value * 100

    resource_id: $.payload.host + "_" + $.payload.nodename

    timestamp: $.payload.metrics[?(@.name='cpu.percent')].timestamp

    metadata:

      event_type: $.event_type

      host: $.publisher_id

      source: $.payload.metrics[?(@.name='cpu.percent')].source
```

Using the schema above:

```
{
    "field_mappings": [
            "set": [
                    {
                            "new_string": "counter_name",
                        "collectd_field": {
                                "plugin": "cpu"
                            },
                            "conversion": "cpu"
                        },
                        {
                                "new_string": "counter_type",
                                "collectd_field": {
                                        "values": {
                                                "gauge": "cumulative"
                                        }
                                }
                        },
                        {
                                "new_string": "counter_volume",
                                "collectd_field": {
                                        "values": {
                                                "actual_values": "values_array"
                                        }
                                }
                                "conversion": "None"
                        },
                        {
                                "new_string": "timestamp",
                                "collectd_string": "timestamp"
                        },
                        {
                                "new_string": "source",
                                "new_value": "collectd"
                        },
                        {
                                "new_string": "resource_id",
                                "collectd_field": {
                                        "values": {
                                                "host": "uuid",
                                                "plugin instance": "plugin_instance"
                                        }
                                }
                        }
                ]
        ]
}
```

Key_words: values_array.

References

http://docs.openstack.org/admin-guide/telemetry-data-collection.html

http://docs.openstack.org/developer/ceilometer/webapi/v2.html#samples-and-statistics