sgevents Documentation

Release 0.1

Mike Boers

February 25, 2016

Contents

sgevents is a Shotgun event-log daemon.

This project aims to encapsulate the process of polling the Shotgun event log, smooth out as many edge cases as we can transparently deal with, and provide assistance in dealing with the rest of them.

Finally, it provides a method of registering event handlers in a manner to be well behaved in development environments.

CHAPTER 1

Contents

1.1 Python API Reference

1.1.1 The Event Log

1.1.2 The Event Object

1.2 Reverse Engineering

Herein lie things we have learned that were not obvious about the event log.

1.2.1 Retired entities are `null`

The `entity` field is `null` for retired entities, as determined when you read the log, not when the log was created. Ergo, if you are processing a backlog of log events, you can find yourself dealing with "Change" or even "New" events with a null entity field.

It is still possible to read the entity via a `$FROM` filter:

```
>>> sg.find_one('Task', [('$FROM$EventLogEntry.entity.id', 'is', 2017315)], [], retired_only=True)
{'type': 'Task', 'id': 67519}
```

This is exposed via `Event.find_retired_entity()`.

1.2.2 Multi-Entity changes have no `new_value`

Changes to a `multi_entity` do not have a `new_value` and/or `old_value` in the `meta`. Instead, there is a list of entities added and removed in `added` and `removed` (respectively) in `meta`.

1.2.3 Back-references invade metadata

The `meta` field is a mixture of referee and referrer data when it is the backref of a `multi_entity` field.

E.g. if you set the the `Task.entity` to be a `Shot`, `Shot.tasks` will be updated to include that `Task`. The event pertaining to `Shot.tasks` will contain a few oddities which match the `Task`:

- `entity_type`

- `entity_id`
- `field_data_type`

If you combine this with the retired entity rule, you see that you can't even derive what the referenced entity was. There are two backup plans:

1. You can lookup the other log entry via `original_event_log_entry_id` to figure out what the entity was set to.
2. Use the entity type formatted into the event type, and the `$FROM$` syntax to select the entity with `retired_only=True`.

1.2.4 `"Reading"` meta-type

The `type` field is of the format `{domain}_{entity_type}_{subtype}`, in which `domain` is the string `"Shotgun"` for anything triggered by Shotgun, `entity_type` is the type of the entity that the event's action was upon, and `subtype` is the action itself, e.g. `"New"`, `"Change"`, etc..

We have seen at least one "meta" `entity_type` in which it does not match the actual type of the `entity` field: `"Reading"`.

`"Reading"` appears to be a meta-type which is describing when a human has read a note attached to an entity:

```
{
    "attribute_name": "read_by_current_user",
    "entity": {
        "id": 9174,
        "name": "<note goes here>",
        "type": "Note"
    },
    "event_type": "Shotgun_Reading_Change",
    "id": 2020717,
    "meta": {
        "attribute_name": "read_by_current_user",
        "new_value": "read",
        "old_value": null,
        "type": "attribute_change"
    },
    "type": "EventLogEntry",
    "user": {
        "id": 78,
        "name": "<name of human>",
        "type": "HumanUser"
    }
}
```