User-defined Metrics

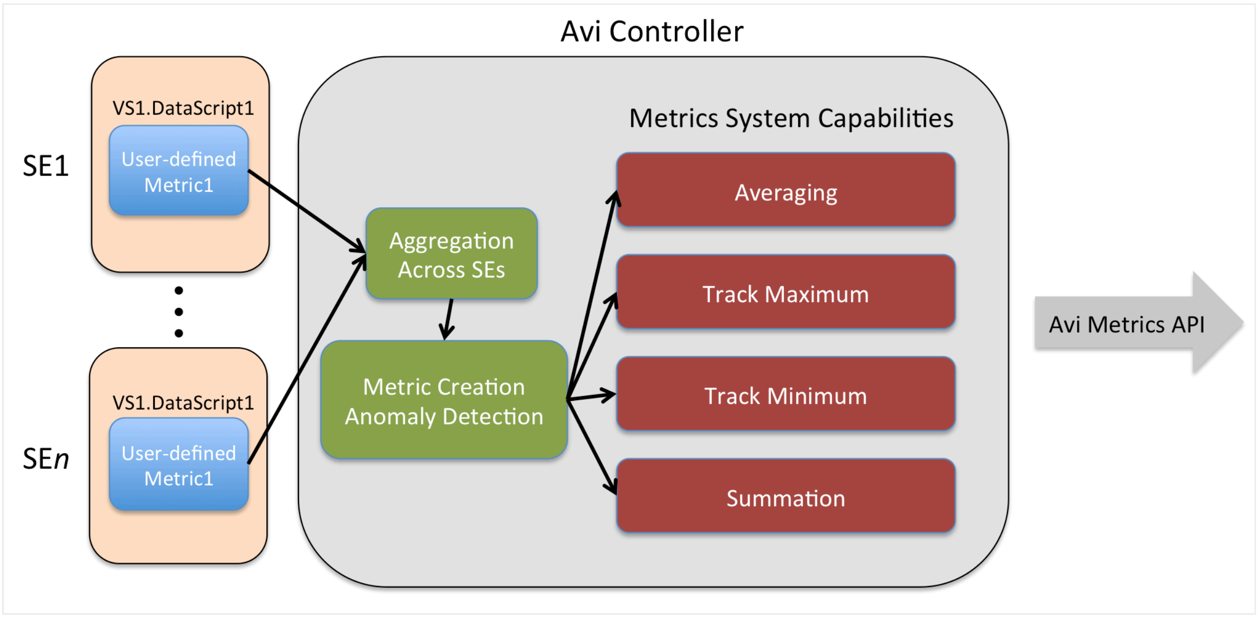
Avi Technical Reference (v18.2)

User-defined Metrics

view online

Avi Vantage provides a tremendous amount of consumable metrics data. There are situations where an organization may want some additional data unique to their applications or environment. To satisfy that need, starting with Avi Vantage 18.2.3, three new DataScript functions interact with the Avi metrics subsystem to enable an organization to create, manipulate and access their own metrics. Avi REST API calls can be used to incorporate that custom data into the customer's monitoring systems.

About User-Defined Metrics



A user-defined metric is ...

- An integer and can never be negative.
- Created and subsequently manipulated by a DataScript running in the Service Engines upon which the virtual service has been placed.
- Collected and aggregated by the Controller(s). The Controller can detect anomalies in the values of these metrics.
- Stored over time so that various metrics system capabilities can be applied, including averaging, min/max tracking, and summation.
- Able to be queried via the Avi Metrics API.

Any given user-defined metric is scoped to the virtual service that creates it via a DataScript. Since a DataScript can be used by more than one virtual service, it is the Controller's duty to keep identically named metrics in different virtual services isolated one from the other.

A user-defined metric's interpretation is left completely up to the user. Its value is based on conditions detected within the DataScript in which the metric is referenced. When a metric crosses some user-defined threshold, any number of actions ? or no action ? may be triggered within the DataScript. For example: * Use the `avi.vs.log()` DataScript function to simply report the value of the metric. * Use the `avi.vs.rate_limit( type, string_to_limit, [defer_action=False] )` DataScript function to rate-limit the virtual service.

That said, most of the time, metrics will be extracted via calls to the Avi REST API, rather than by the DataScripts themselves.

User-Defined Metric Types

Avi Vantage supports two *types* of user-defined metrics. One refers to a metric's type using one of two static integer values:

1. avi.vs.analytics.METRICTYPE_COUNTER ? A metric of this type can be incremented or cleared.
2. avi.vs.analytics.METRICTYPE_GAUGE ? A metric of this type can be incremented, decremented, cleared, or set.

DataScript Functions for User-Defined Metrics

The DataScript functions that manipulate the above metrics are:

- avi.vs.analytics.counter(metric_name, [operation], [value])
- avi.vs.analytics.gauge(metric_name, [operation], [value])
- avi.vs.analytics.get_metric(metric_name, metric_type)

Click on the above links to learn more about each of these.

User-Defined Metric Lifecycle

User-defined metrics are always aged and automatically destroyed if not used for a period of more than 2 hours. Every time a counter or gauge is read or updated, its age is refreshed. Aging provides automatic garbage collection of user-defined metrics.

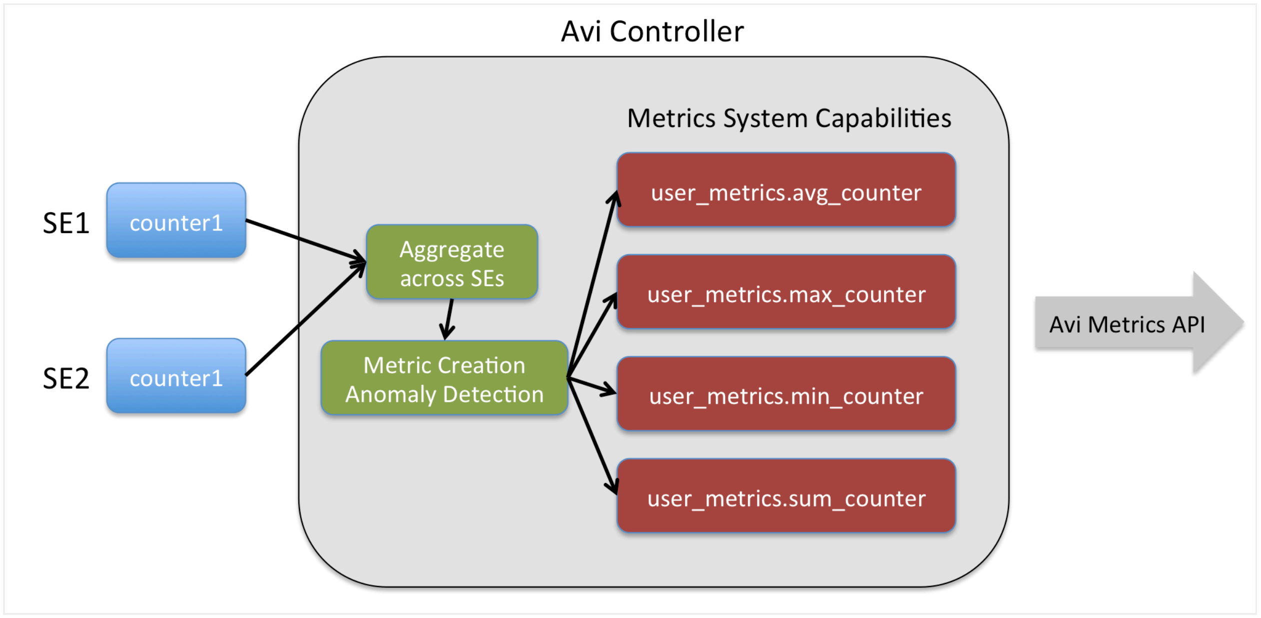
Accessing User-Defined Metrics

The Avi Controller provides access to these metrics via Avi metrics REST API calls. Refer to the REST API Guide for more details.

In a call such as the following:
`api/analytics/metrics/virtualservice/<vs_uuid>?metric_id=user_metrics.sum_counter&obj_id=<obj_id>&step=300&limit=1`

`obj_id` is either the ID of the counter or ?*? to identify all the active counter metrics.

Aggregated values that can be queried for avi.vs.analytics.METRICTYPE_COUNTER



- user_metrics.sum_counter ? Sum of counter metric values reported over a given period.
  Example: latest 5-minute sample of aggregated user metric having a `counter_id` equal to "Foo" can be fetched using following API:

```
api/analytics/metrics/virtualservice/<vs_uuid>?metric_id=user.sum_counter&obj_id=Foo&step=300&limit=1
```

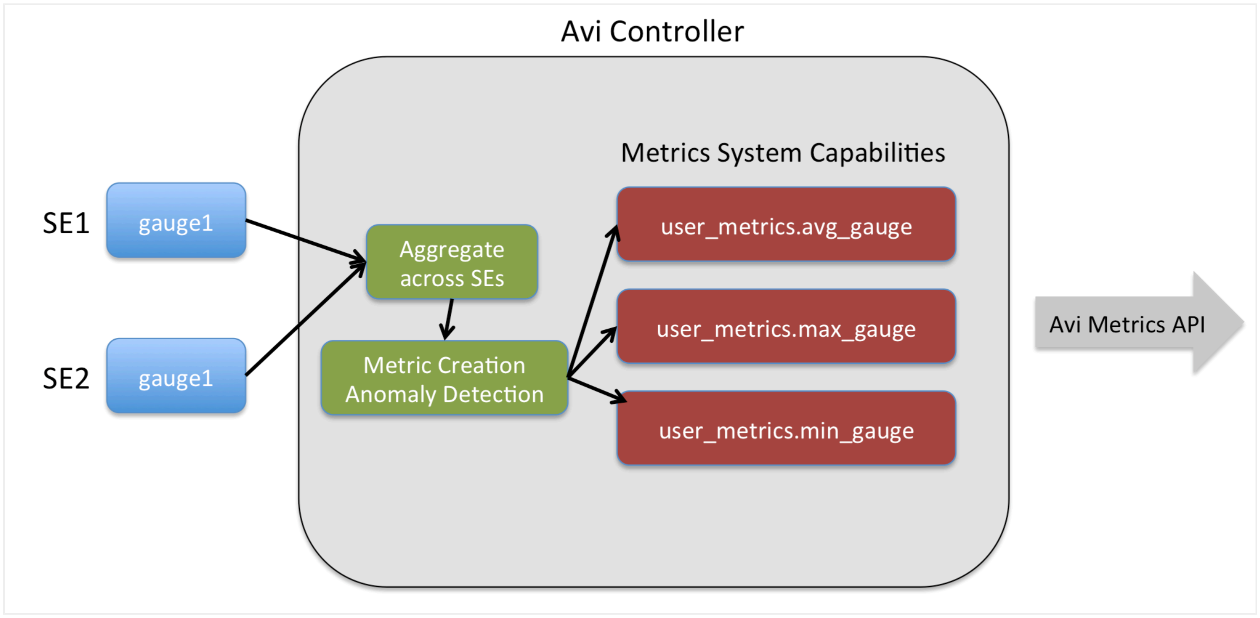
- user_metrics.avg_counter ? Time averaged per-second values of counter metric over a given period.
  Example: latest 5-minute sample of averaged per-second rate of counter metric Foo can be requested using this API call:

```
api/analytics/metrics/virtualservice/<vs_uuid>?metric_id=user.avg_counter&obj_id=Foo&step=300&limit=1
```

- user_metrics.max_counter and user_metrics.min_counter ? Maximum/minimum per-second reported values of counter metric Foo over a given period.
  Example: the per-second max and min values can be requested using these API calls:

```
api/analytics/metrics/virtualservice/<vs_uuid>?metric_id=user.max_counter&obj_id=Foo&step=300&limit=1
  api/analytics/metrics/virtualservice/<vs_uuid>?metric_id=user.min_counter&obj_id=Foo&step=300&limit=1
```

Aggregated values that can be queried for avi.vs.analytics.METRICTYPE_GAUGE



- user_metrics.avg_gauge ? Average of raw counter value of type gauge reported over given period.
  Example: fetch counter 'foo' via API

```
/api/analytics/metrics/virtualservice/?metric_id=user_metrics.avg_gauge&limit=1&step=300&obj_id=foo
```

- user_metrics,max_gauge ? Maximum raw counter value of type gauge reported over given period.
  Example: fetch counter 'foo' via API

```
/api/analytics/metrics/virtualservice/?metric_id=user_metrics.max_gauge&limit=1&step=300&obj_id=foo
```

- user_metrics.min_gauge ? Minimum raw counter value of type gauge reported over given period.
  Example: fetch counter 'foo' via API

```
/api/analytics/metrics/virtualservice/?metric_id=user_metrics.min_gauge&limit=1&step=300&obj_id=foo
```

API Examples

Example 1 ? Provide the average number of 502 responses coming from an application

The following DataScript statement increments by 1 a counter named abs_502 every time a 502 error is detected.

```
if avi.http.status() == "502" then
   avi.vs.analytics.counter("abs_502", avi.vs.analytics.INCR, 1)
    end
```

The metrics data for this user-defined value can be retrieved by querying the Avi metrics REST API for the specific virtual service. Example formats:

```
api/analytics/metrics/virtualservice/<vs_uuid>?metric_id=user_metrics.avg_counter&obj_id=abs_502
```

Example 2: Track NTTP Requests by TLS/SSL Protocol Version

```
-- HTTP_REQ
if avi.ssl.protocol() == "TLSv1.2" then
    avi.vs.analytics.counter("tls1_2", avi.vs.analytics.INCR, 1)
elseif avi.ssl.protocol() == "TLSv1.1" then
    avi.vs.analytics.counter("tls1_1", avi.vs.analytics.INCR, 1)
elseif avi.ssl.protocol() == "TLSv1" then
    avi.vs.analytics.counter("tls1", avi.vs.analytics.INCR, 1)
elseif avi.ssl.protocol() == "SSLv3" then
    avi.vs.analytics.counter("ssl3", avi.vs.analytics.INCR, 1)
end
```

The metrics data for these four user-defined values can be retrieved by querying the metrics API for the specific virtual service. Example formats:

- `/api/analytics/metrics/virtualservice/<vs_uuid>?metric_id=user_metrics.sum_counter&obj_id=tls1_2`
- `/api/analytics/metrics/virtualservice/<vs_uuid>?metric_id=user_metrics.sum_counter&obj_id=tls1_1`
- `/api/analytics/metrics/virtualservice/<vs_uuid>?metric_id=user_metrics.sum_counter&obj_id=tls1`
- `/api/analytics/metrics/virtualservice/<vs_uuid>?metric_id=user_metrics.sum_counter&obj_id=ssl3`