Metrics & Reports

Table of Contents

Data Collection

Why is an element not updating even though the data exists in the data source?

Issue

I created new dimensions in anticipation of new data that must be capured in Metric Insights. The data is now available in the data source, but Metric Insights is not fetching this new data during *data collection trigger* run or when attempting to collect the data manually. What is going?

Resolution

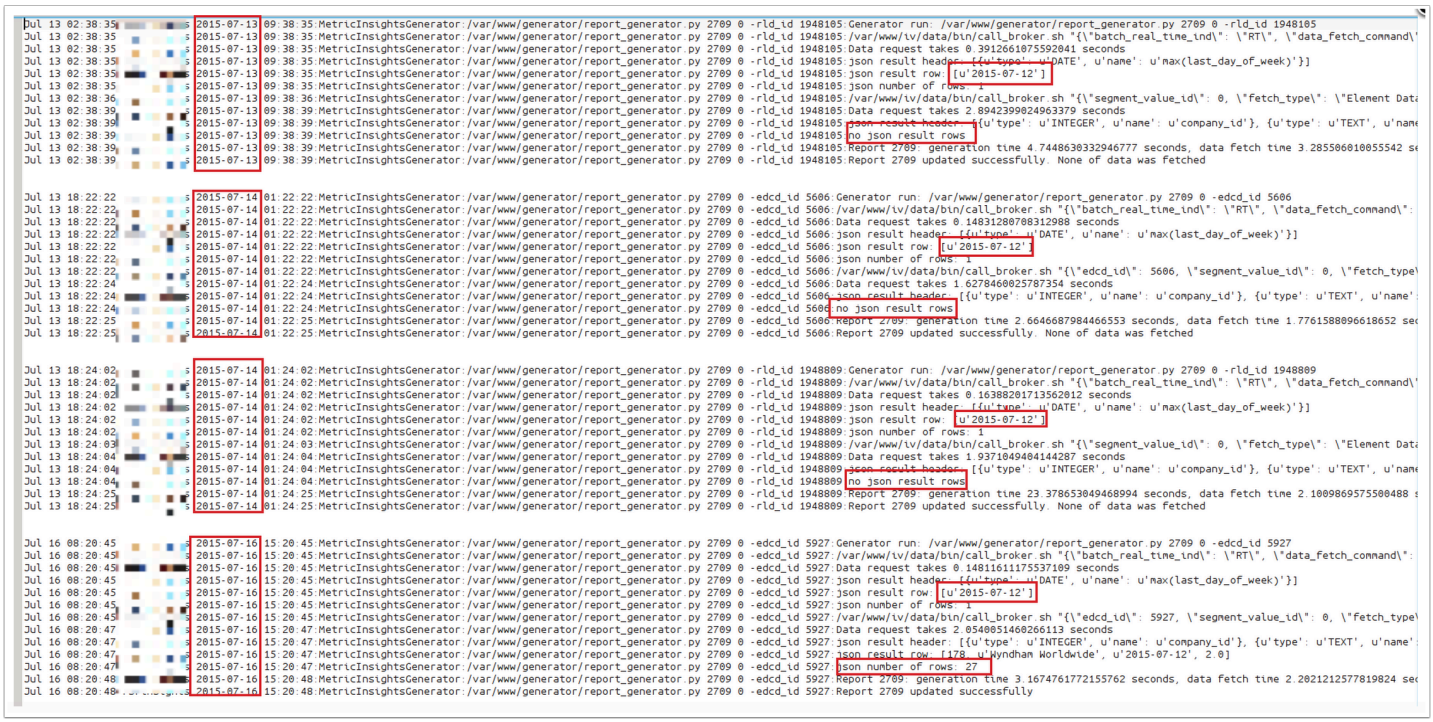
This is probably a data availability issue at the data source. Unless there is an error in Metric Insights during data collection, MI collects whatever it can read from the data source. In this instance, there was no new data being collected on Monday for the prior week. The data was not available until days later!

A good way to check is to look at /var/log/**mi.debug**. This shows whether data is being returned during data collection and even shows whether a collection is via the MI trigger or a manual call (see image below).

Manual calls are identified as "**-edcd-id** XXXXXX " and trigger calls are "**-rld_id** XXXXXX"

Notice that the calls are made daily, both via the trigger and manually. Each time, the correct measurement_time is used (2015-07-12). However no data is returned - **no json result rows**. Finally, on Thursday, the new data is fetched - **json number of rows: 27**, and you can see the actual data point that was fetched in the row above this line. This clearly shows that the data in question was never available in the source on Monday (for MI to collect anyway), but much later in the week, on Thursday.

The solution then, is to reschedule the data collection trigger to run when the data has loaded into the data source. You can also set a dependency to have MI check for data in the source first before attempting to collect any data (see http://kb.metricinsights.com/m/43678/l/615921-how-do-you-trigger-data-collection-when-data-jobs-at-the-source-are-finished). For any additional questions, please contact support@metricinsights.com.


Jul 13 02:38:35 2015-07-13 09:38:35:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948105:Generator run: /var/www/generator/report_generator.py 2709 0 -rld_id 1948105
Jul 13 02:38:35 2015-07-13 09:38:35:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948105:/var/www/iv/data/bin/call_broker.sh "{\"batch_real_time_ind\": \"RT\", \"data_fetch_command\"
Jul 13 02:38:35 2015-07-13 09:38:35:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948105:Data request takes 0.3912661075592041 seconds
Jul 13 02:38:35 2015-07-13 09:38:35:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948105:json result header: [{u'type': u'DATE', u'name': u'max(last_day_of_week)'}]
Jul 13 02:38:35 2015-07-13 09:38:35:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948105:json result row: [u'2015-07-12']
Jul 13 02:38:35 2015-07-13 09:38:35:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948105:json number of rows: 1
Jul 13 02:38:36 2015-07-13 09:38:36:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948105:/var/www/iv/data/bin/call_broker.sh "{\"segment_value_id\": 0, \"fetch_type\": \"Element Data
Jul 13 02:38:39 2015-07-13 09:38:39:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948105:Data request takes 2.8942399024963379 seconds
Jul 13 02:38:39 2015-07-13 09:38:39:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948105:json result header: [{u'type': u'INTEGER', u'name': u'company_id'}, {u'type': u'TEXT', u'name
Jul 13 02:38:39 2015-07-13 09:38:39:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948105:no json result rows
Jul 13 02:38:39 2015-07-13 09:38:39:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948105:Report 2709: generation time 4.74486303329467777 seconds, data fetch time 3.285506010055542 se
Jul 13 02:38:39 2015-07-13 09:38:39:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948105:Report 2709 updated successfully. None of data was fetched
Jul 13 18:22:22 2015-07-14 01:22:22:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5606:Generator run: /var/www/generator/report_generator.py 2709 0 -edcd_id 5606
Jul 13 18:22:22 2015-07-14 01:22:22:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5606:/var/www/iv/data/bin/call_broker.sh "{\"batch_real_time_ind\": \"RT\", \"data_fetch_command\":
Jul 13 18:22:22 2015-07-14 01:22:22:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5606:Data request takes 0.14831280708312988 seconds
Jul 13 18:22:22 2015-07-14 01:22:22:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5606:json result header: [{u'type': u'DATE', u'name': u'max(last_day_of_week)'}]
Jul 13 18:22:22 2015-07-14 01:22:22:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5606:json result row: [u'2015-07-12']
Jul 13 18:22:22 2015-07-14 01:22:22:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5606:json number of rows: 1
Jul 13 18:22:22 2015-07-14 01:22:22:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5606:/var/www/iv/data/bin/call_broker.sh "{\"edcd_id\": 5606, \"segment_value_id\": 0, \"fetch_type\
Jul 13 18:22:24 2015-07-14 01:22:24:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5606:Data request takes 1.6278460025787354 seconds
Jul 13 18:22:24 2015-07-14 01:22:24:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5606:json result header: [{u'type': u'INTEGER', u'name': u'company_id'}, {u'type': u'TEXT', u'name':
Jul 13 18:22:24 2015-07-14 01:22:24:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5606:no json result rows
Jul 13 18:22:25 2015-07-14 01:22:25:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5606:Report 2709: generation time 2.6646687984466553 seconds, data fetch time 1.7761588896618652 se
Jul 13 18:22:25 2015-07-14 01:22:25:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5606:Report 2709 updated successfully. None of data was fetched
Jul 13 18:24:02 2015-07-14 01:24:02:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948809:Generator run: /var/www/generator/report_generator.py 2709 0 -rld_id 1948809
Jul 13 18:24:02 2015-07-14 01:24:02:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948809:/var/www/iv/data/bin/call_broker.sh "{\"batch_real_time_ind\": \"RT\", \"data_fetch_command\"
Jul 13 18:24:02 2015-07-14 01:24:02:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948809:Data request takes 0.16388201713562012 seconds
Jul 13 18:24:02 2015-07-14 01:24:02:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948809:json result header: [{u'type': u'DATE', u'name': u'max(last_day_of_week)'}]
Jul 13 18:24:02 2015-07-14 01:24:02:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948809:json result row: [u'2015-07-12']
Jul 13 18:24:02 2015-07-14 01:24:02:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948809:json number of rows: 1
Jul 13 18:24:03 2015-07-14 01:24:03:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948809:/var/www/iv/data/bin/call_broker.sh "{\"segment_value_id\": 0, \"fetch_type\": \"Element Data
Jul 13 18:24:04 2015-07-14 01:24:04:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948809:Data request takes 1.9371049484144287 seconds
Jul 13 18:24:04 2015-07-14 01:24:04:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948809:json result header: [{u'type': u'INTEGER', u'name': u'company_id'}, {u'type': u'TEXT', u'name
Jul 13 18:24:04 2015-07-14 01:24:04:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948809:no json result rows
Jul 13 18:24:25 2015-07-14 01:24:25:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948809:Report 2709: generation time 23.378653049468994 seconds, data fetch time 2.1009869575500488 s
Jul 13 18:24:25 2015-07-14 01:24:25:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -rld_id 1948809:Report 2709 updated successfully. None of data was fetched
Jul 16 08:20:45 2015-07-16 15:20:45:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5927:Generator run: /var/www/generator/report_generator.py 2709 0 -edcd_id 5927
Jul 16 08:20:45 2015-07-16 15:20:45:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5927:/var/www/iv/data/bin/call_broker.sh "{\"batch_real_time_ind\": \"RT\", \"data_fetch_command\":
Jul 16 08:20:45 2015-07-16 15:20:45:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5927:Data request takes 0.14811611175537109 seconds
Jul 16 08:20:45 2015-07-16 15:20:45:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5927:json result header: [{u'type': u'DATE', u'name': u'max(last_day_of_week)'}]
Jul 16 08:20:45 2015-07-16 15:20:45:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5927:json result row: [u'2015-07-12']
Jul 16 08:20:45 2015-07-16 15:20:45:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5927:json number of rows: 1
Jul 16 08:20:45 2015-07-16 15:20:45:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5927:/var/www/iv/data/bin/call_broker.sh "{\"edcd_id\": 5927, \"segment_value_id\": 0, \"fetch_type\
Jul 16 08:20:47 2015-07-16 15:20:47:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5927:Data request takes 2.0540051460266113 seconds
Jul 16 08:20:47 2015-07-16 15:20:47:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5927:json result header: [{u'type': u'INTEGER', u'name': u'company_id'}, {u'type': u'TEXT', u'name':
Jul 16 08:20:47 2015-07-16 15:20:47:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5927:json result row: [178, u'Wyndham Worldwide', u'2015-07-12', 2.0]
Jul 16 08:20:47 2015-07-16 15:20:47:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5927:json number of rows: 27
Jul 16 08:20:48 2015-07-16 15:20:48:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5927:Report 2709: generation time 3.1674761772155762 seconds, data fetch time 2.2021212577819824 se
Jul 16 08:20:48 2015-07-16 15:20:48:MetricInsightsGenerator:/var/www/generator/report_generator.py 2709 0 -edcd_id 5927:Report 2709 updated successfully

Report validation fails when declaring variable in SQL Statement field with the error: "DataCollector returned error: Must declare the scalar variable."

Issue

I've created a report that uses variables to collect data and have them declared in the SQL Statement field. But when validating the report statement an error occurs: *"DataCollector returned error: Must declare the scalar variable "@OvrBgtMinAmt".* I am able to run this same query successfully from a SQL Server client however. Help!

Resolution

The short answer is to remove the semicolon after the DECLARE statement.

The long answer: there are several things affecting this issue, the most important being that most JDBC drivers do not allow multiple statements to be sent to the database in one JDBC query.

For example, MySQL, PostgreSQL, and Oracle all fail on the following query when run as a single JDBC statement:

```
create temporary table foo as select 1 i union select 2;
select * from foo
```

This is actually a good thing, because it helps protect against SQL injection attacks. Comparatively, SQL Server is one of the few JDBC drivers that actually allows sending multiple statements in one JDBC call.

Since Metric Insight users often want to run sql scripts like what's shown above for some light ETL procedures, MI was designed to parse out sql scripts separated by semicolons and run each query separately, all sharing the same connection and database session. This allows for users to do multiple ETL queries before getting to the actual metric query.

Unfortunately, not all JDBC drivers and databases are created equal, even though things like this work in MySQL in MI's batch mode processing:

```
set @foop :=3 ; select @foop

(MI first sends `set @foop :=3`, then sends another `select @foop` to the database)
```

SQL Server does NOT like this. It will yell when running "select @foop" in it's own jdbc call, even though the declare statement was run earlier. However, SQL Server is fine with this query:

```
declare @foop int
set @foop = 3
select @foop
```

where each of these statements is separated by new lines or whitespace. So, in order for this to work with SQL Server, **do not include any semicolons** to separate statements. The above syntax will work in Metric Insights.

Timing Rules for Element Dependencies within a Data Collection Trigger

Question

Our daily-reporting-refresh trigger has 8 elements. There are some dependencies between those elements. For example, one element brings in data from an external DB while other elements query against that data, creating a dependency.

Should we create new triggers to handle the dependent elements and set up trigger dependencies? Or is MI aware of the dependencies and therefore runs the elements accordingly?

Answer

Here are the system timing rules for data collection triggers:

1. Elements that are sourced from an external DB ***or*** the Dashboard DB are processed first.

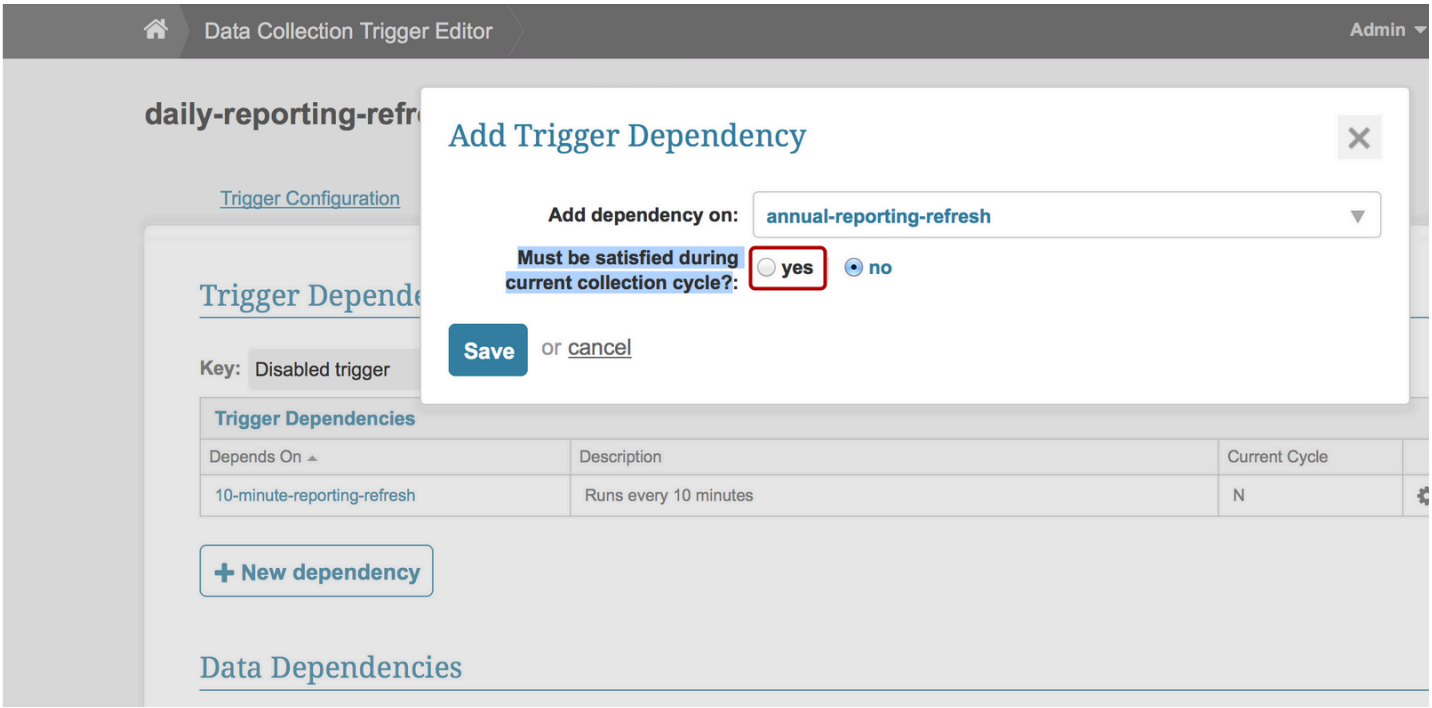
2. Elements that are sourced from ***Existing Metrics*** *or* ***Reports*** are processed last.

If your dependencies are more complex than that, then you will need to set up one or more data collection triggers and therefore explicitely set trigger dependencies. Please contact support@metricinsights.com for any additional questions.

What is the purpose of the "current collection cycle" setting for data/trigger dependencies?

This setting is used to prevent Triggers and Dependencies from getting out of sync.

Issue



A user has the following issue:

A Daily Trigger that should not run until that day's data has been processed first actually ran. Without the ***Must be satisfied during current collection cycle*** setting enabled, however, the following sequence of events was possible (and indeed, was actually happening):

Set-up:

A. Daily trigger is set to start as early as possible, provided that the data dependency has been satisfied.

B. Data dependency is usually satisfied at 6am.

What can happen if the Trigger and its Dependency get out of sync:

1. Data Dependency is satisfied on 11/1 at 6am.

2. Daily Trigger should start immediately afterwards, but fails for some reason. Subsequently, there is *no run* on 11/1.

3. The next cycle starts on 11/2 at midnight. Data Dependency has been satisfied since the last run, so the Daily Trigger starts immediately.

4. Data Dependency is satisfied on 11/2 at 6am.

5. Since the Daily Trigger has already run on 11/2 it does not run again that day after 6am.

6. The next cycle starts on 11/3 at midnight. Data Dependency has been satisfied since the last run, so the Daily Trigger starts immediately.

7. etc.

In this scenario, data collection proceeds as if the data dependency does not actually exist. What results is a data collection that has fallen out of sync with its data dependency.

Resolution

To prevent the Trigger from falling out of sync with its Data Dependency, you must set ***Must be satisfied during current collection cycle*** to ***yes***. The scenario described above then becomes impossible. In other words, the Daily Trigger cannot run on 11/2 at midnight until the Data Dependency has been satisfied ***on that day*** *at 6am.*

In what order are items processed during data collection processing?

Here you may find the information on the way the elements are being collected by MI to be processed by triggers.

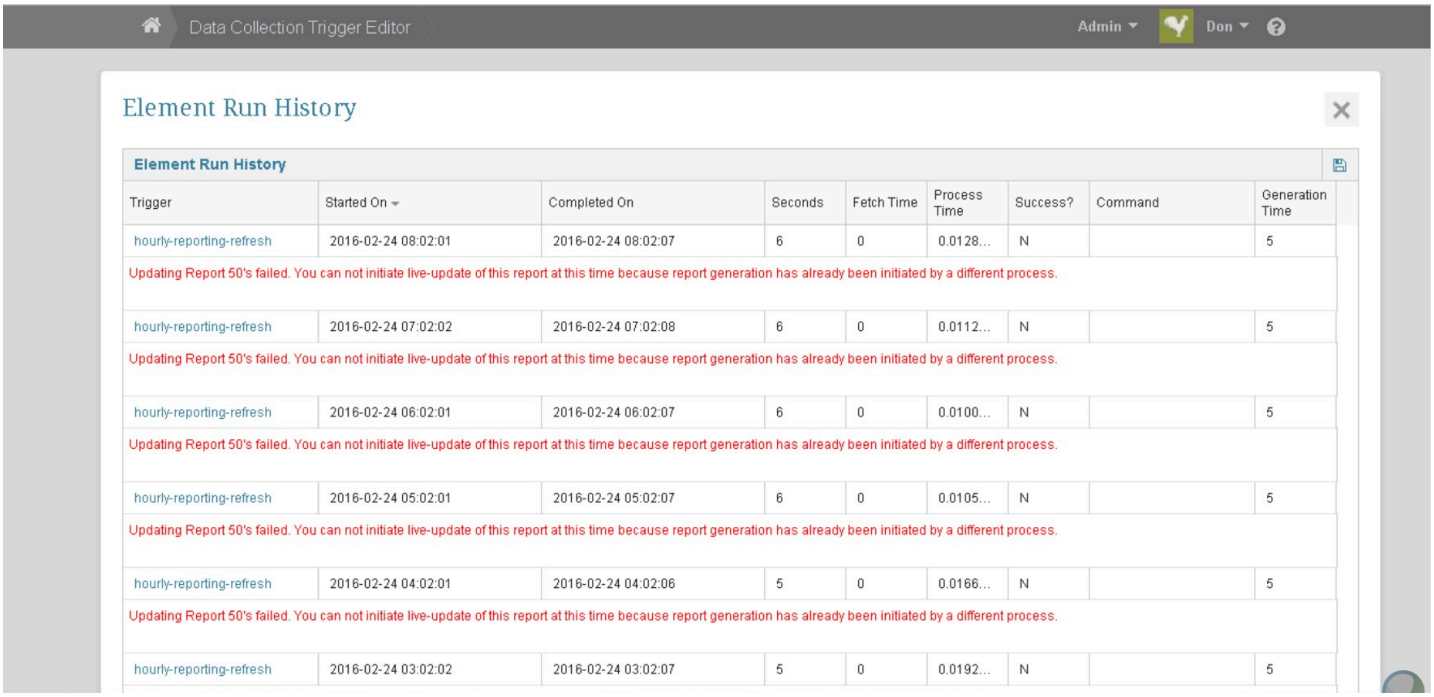
Assume you have 10 elements which use external Data Sources and 20 elements which depend on the internal elements. They will be processed depending on the functional criteria, then on multi- or single-threading and finally on external/internal data sourcing. The general order of processed elements will be based on the following stages:

Stage	Trigger work
1. Datasets	collects data for datasets sourced from sql/ plugin
2. Expired Reports	clears stored data for reports with *expire* settings
3. Expired external reports	clears stored data for external reports with the following expiresettings http://grabilla.com/0750f-c3f56681-32b7-4f2b-b199-fe24d6c8e1a3.png
4. Calendars	collects data for calendars sourced from sql/ plugin
5. Elements sourced from dataset	collects data for metrics and reports sourced from datasets
6. Elements	collects data for metrics/reports sourced from sql/plugin
7. External Images	collects images / PDFs for external reports `
8. Composite Metrics	collects data for metrics sourced from existing metrics
9. Elements sourced from Existing Reports	collects data for metrics/reports sourced from existing reports saved as an sql table
10. History for bulk-fetch dimensioned Metrics	collects data for new dimension values that are bulk-fetched for metrics sourced from sql/plugin, and also for those dimensions that are fetched with *'%<single-fetch>%'* in the data fetch command

Stage	Trigger work
11. Change reports	collects data for change reports
12. Aggregate Metrics	collects data for aggregate metrics
13. Totals for Aggregate Metrics	collects data for total dimensions values of aggregate metrics
14. Single Report Metrics	collects data for metrics sourced from single reports
15. Totals for Single Report Metrics	collects data for total dimension values of metrics sourced from single reports
16. Totals for Metrics	collects data for total dimensions values of metrics sourced from sql/plugin/datasets/ existing reports/external
17. Composite Reports	updates report sourced from existing metrics
18. Totals for Composite Metrics	collects data for total dimension values of composite metrics
19. Regeneration queue	regenerates charts that are in the queue; this includes updating charts that are affected by group annotations
20. Multi-Metrics	regenerates charts for multi-metrics

Why am I getting the error, "You can not initiate live-update of this report at this time because report generation has already been initiated by a different process"?

Issue



One of my reports failed to update earlier today. When looking at the Run History, I see the following error for each time the data collection trigger ran:

Updating Report 50's failed. You can not initiate live-update of this report at this time because report generation has already been initiated by a different process.

I then tried to update my report manually by clicking the **Update Live Report** button but that also shows the same error. What's going on? And where do I find this "different process"?

Resolution

The error is saying that your report basically never finished during the initial data collection. This rarely happens, and in most cases the data collection is finished or aborted. When it does happen, you have to fix the *stuck* report directly in the databse. To fix, please follow the steps below (you will need ssh access to the Metric Insights server):

1. SSH to the Metrc Insights server

2. At the command line enter: *mysql dashboard* (note, if the system database has been renamed previously, replace *dashboard* with the actual name)

3. Enter the following mysql statement: *delete from generator_lock;*

4. You should be able to manually update your report now by clicking **Update Live Report** in the Report Editor. If it's still stuck, please contact support@metricinsights.com.

I'm getting an "Error parsing DataCollector json result" while validating, collecting data, and updating charts. Why?

ISSUE

While validating, collecting data, and updating charts, I am getting an *"Error parsing DataCollector json result"* error. What is going on here?

RESOLUTION

There are a couple of reasons why this error can occur:

A.) The permissions for */opt/mi/iv/data/temp/datacollector/data* are wrong. Update the permissions by running this command directly on the Metric Insights server:

chmod 777 /var/www/iv/data/temp/datacollector/data.

For example:

root@mi-server:/var/www/iv/data/temp/datacollector# ls -ld data

drwxr-xr-x 2 root root 4096 Jan 20 14:03 data

root@mi-server:/var/www/iv/data/temp/datacollector# chmod 777 data

A permissions issue can be determined by checking /var/log/mi.error. You'll see an error similar to:

/var/www/generator/report_validator.py data_fetch 508:**Error parsing incoming json data**:
Oct 25 09:47:10 mi-server 2013-10-25 09:47:10:MetricInsightsGenerator:/var/www/generator/report_validator.py data_fetch 508:
Warning: file_put_contents(**/var/www/iv/data/temp/datacollector/data**/1382662030_97059400): failed to open stream: **Permission denied** in /var/www/iv/engine/libs/Custom/Broker/Abstract.php on line 267

B.) The */opt/mi/datacollector/lib/logback.xml* file is pointing to the wrong logs directory for **insight.log** and **query.log**. The logback.xml file must be able to locate the aforementioned logs in order to update for each action taken in Metric Insights. To fix, open logback.xml in an editor (vi, vim) and locate the tags **<file>** and **<fileNamePattern>**. These two should be pointing to:

/var/log/insight in Metric Insights version 3.x

or

/opt/mi/datacollector/logs in Metric Insights version 4.x.

For example, the <file> tag may have been something like:

<file>${mi.insightd_dir}/logs/query.log</file>

but ${mi.insightd_dir} is not yet a declared variable for version 3.x. Either declare that variable, or explicitly point it to the right directory based on the version of MI installed.

This error was found by viewing /var/log/mi.error. You'll see an error similar to:

2016-04-26 13:09:07:MetricInsightsGenerator:/var/www/generator/report_generator.py 1820 0 -edcd_id 11957:**Error parsing incoming json data:**
Apr 26 13:09:07 mi-server 2016-04-26 13:09:07:MetricInsightsGenerator:/var/www/generator/report_generator.py 1820 0 -edcd_id 11957:13:09:03,574 |-INFO in ch.qos.logback.classic.LoggerContext[default] - Could NOT find resource [logback.groovy] 13:09:03,576 |-INFO in ch.qos.logback.classic.LoggerContext[default] - Could NOT find resource [logback-test.xml] 13:09:03,576 |-INFO in ch.qos.logback.classic.LoggerContext[default] - **Found resource [logback.xml]** at [file:/var/www/datacollector/lib/logback.xml] 13:09:04,151

How do you trigger data collection when data jobs at the source are finished?

ISSUE

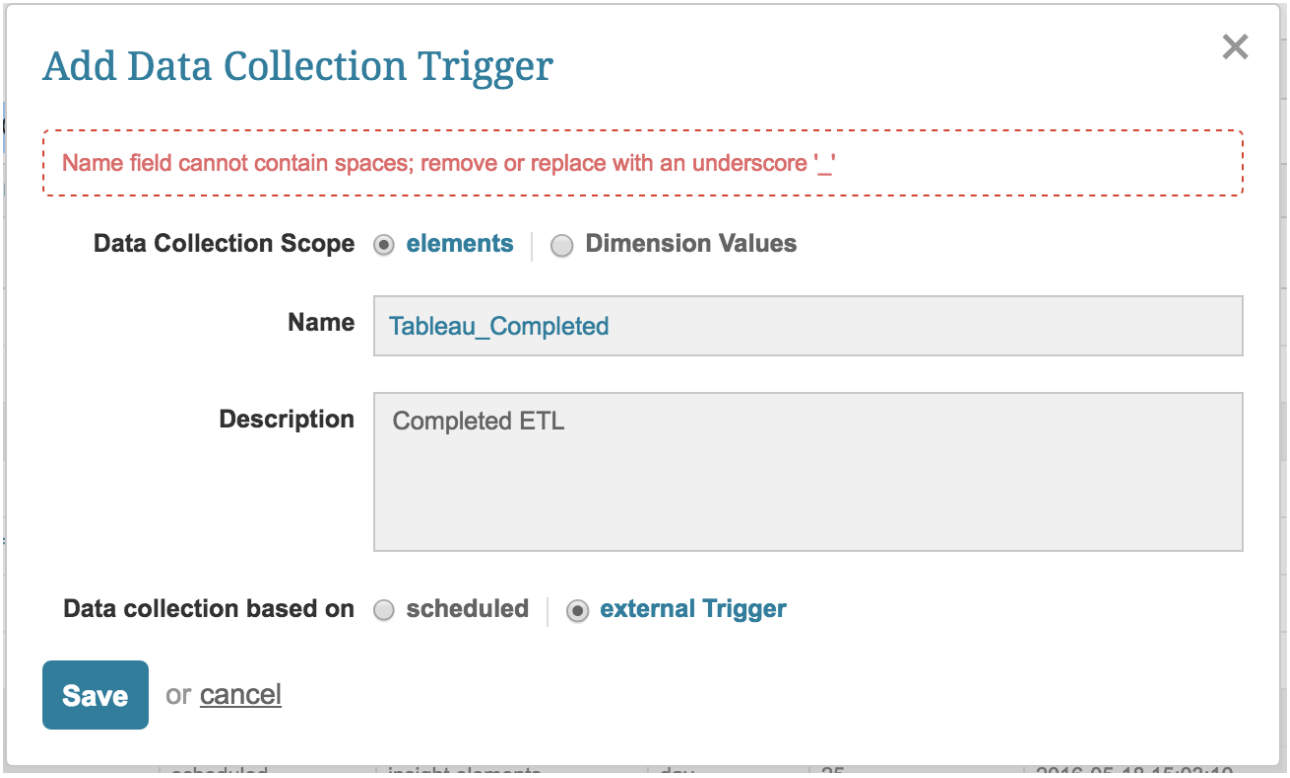
I don't want to collect data from my plug-in data sources (e.g., MicroStrategy or Tableau) until my sources have completed loading. How can I tell Metric Insights that my sources have been loaded and that collection can begin?

RESOLUTION

There are a few ways to accomplish this.

1.) One way is to define a data refresh trigger that is externally triggered (see image below).

To learn more please visit this article: http://help.metricinsights.com/m/Automating_the_Collection_of_Data/l/104472-establish-an-externally-triggered-data-collection-trigger



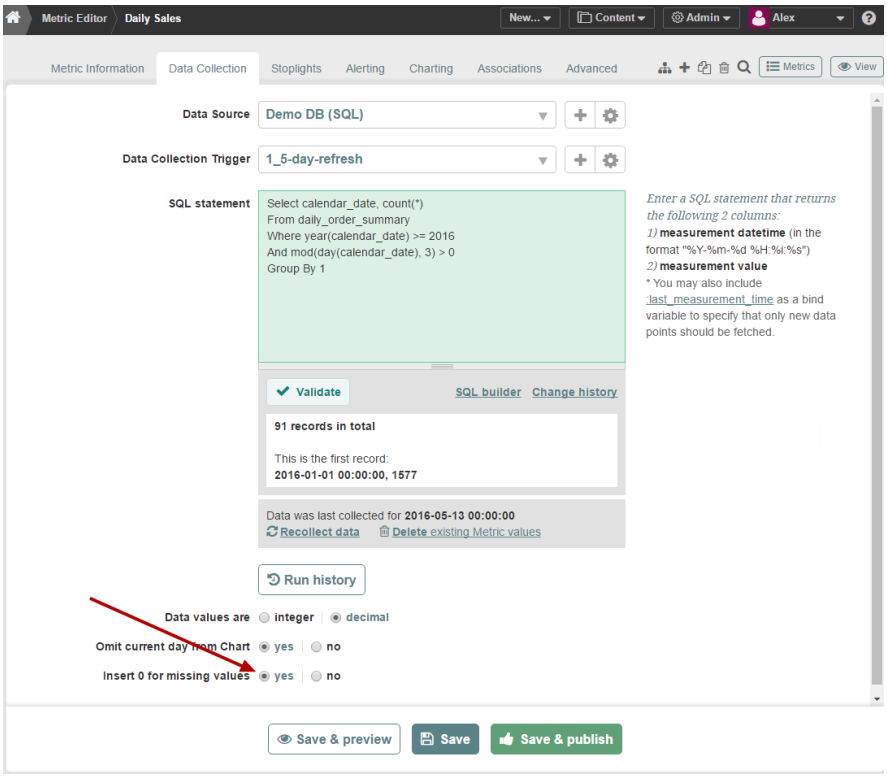
2.) Another way is to define a data collection dependency. Your data collection trigger can be defined to check on a regular schedule (once per day, etc.) but is attached to a data dependency. The data dependency runs against your data source and can check for count of total records, calendar_date, etc. and returns a *true/false* if satisfied. This is a defined query which provides some assurance that the data jobs are completed and sources are loaded. The

dependency can run regularly (e.g. every minute) within a certain time period after you anticipate your data jobs to have completed (e.g. every minute after 7am through 12pm).

You can find more details how to setup a data dependency in this article http://help.metricinsights.com/m/Automating_the_Collection_of_Data/l/104474-set-up-data-dependencies-for-a-data-collection-trigger

How does 'Insert 0s for missing values' work?

RESOLUTION



Assume you have metric M with 'Insert 0s for missing values' = YES. Assume, its last point is 2016-07-01, and you are going to recollect data from 2016-01-01.

Zeroed values are generated for interval between last point and now (i.e., from 2016-07-01 to 2016-08-05 in the example). If the incoming dataset has holes between 2016-07-01 and 2016-08-05, then these holes will be filled with 0s.

The system will not insert zeros for holes that occurred before the last measurement time. If you want the system to to fill such holes, you will need to delete historical data first, thereby forcing the system to reset the last measurement time.

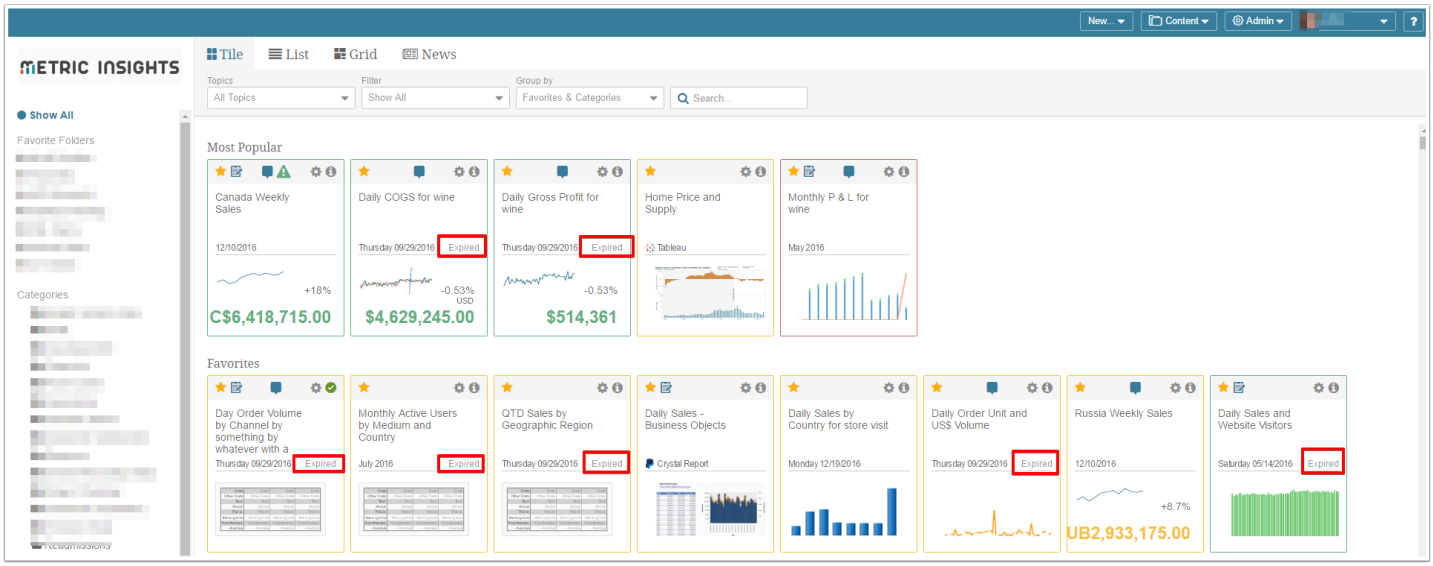
The same algorithm works for segmented metrics as well. Each segment value is considered separately (last point for particular segment value is used).

Algorithm is limited by 10K points for single generation. If last point for daily metric is 1900-01-01, then interval will be from 1900-01-01 till 1927-05-20.

Expired elements

Question

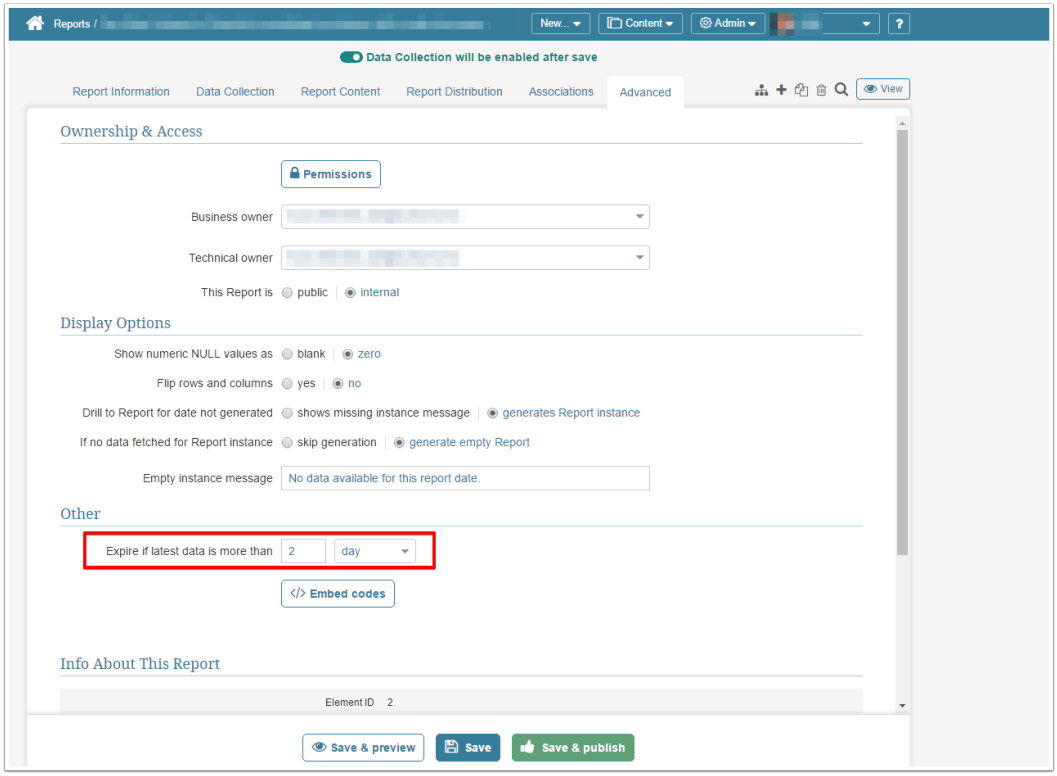
Some elements marked as "expired" on Homepage:



1. What is the definition of expire for a tile and is there a criteria that user can set up?
2. For example, Metric marked as "expired". Does it stop the process of updating new data for this element?

Answer

1. To understand how the system determines that your data is expired, go to your Metric/ Report Editor -> Advanced tab -> Other section. You'll see a setting "Expire if latest data is more than" that is unique for each Metric/Report.
If element marked as "expired" with setting “expire if latest data is more than 2 days”, it means the most recent data is prior to 2 days ago.



2. "Expired" mark just states and visualizes that element doesn't have latest data for the period that user set in "Expire if latest data is more than" setting. Element is still fetching data based on datacollection trigger.

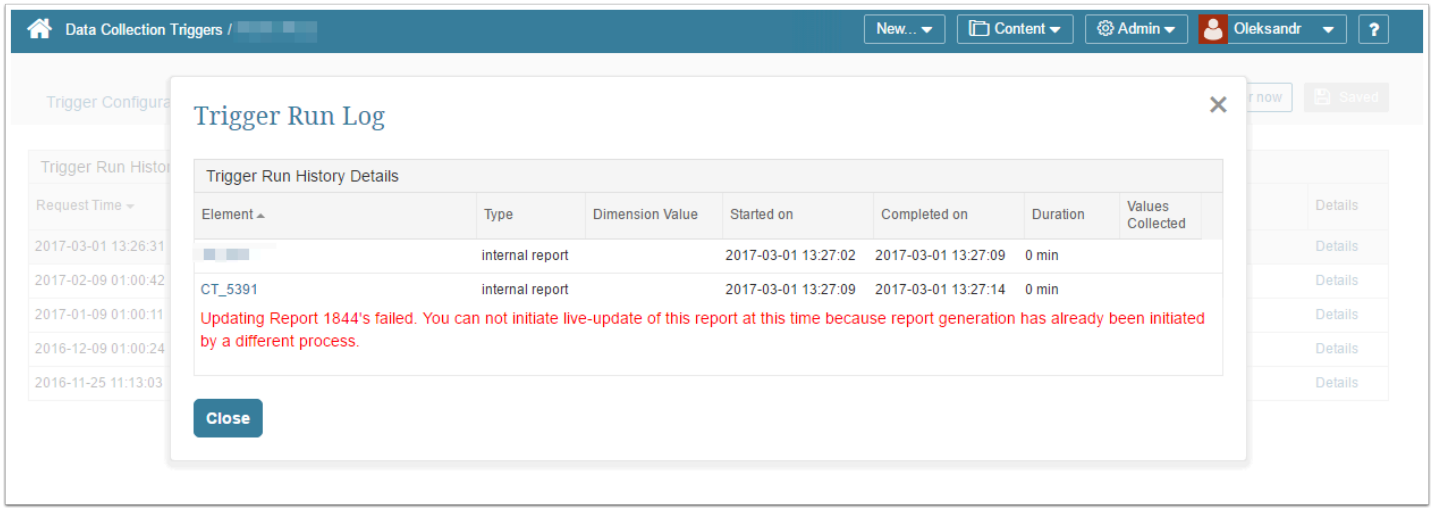
Why am I getting a "Updating Report failed. You can not initiate live-update of this report at this time because report generation has already been initiated by a different process" error?

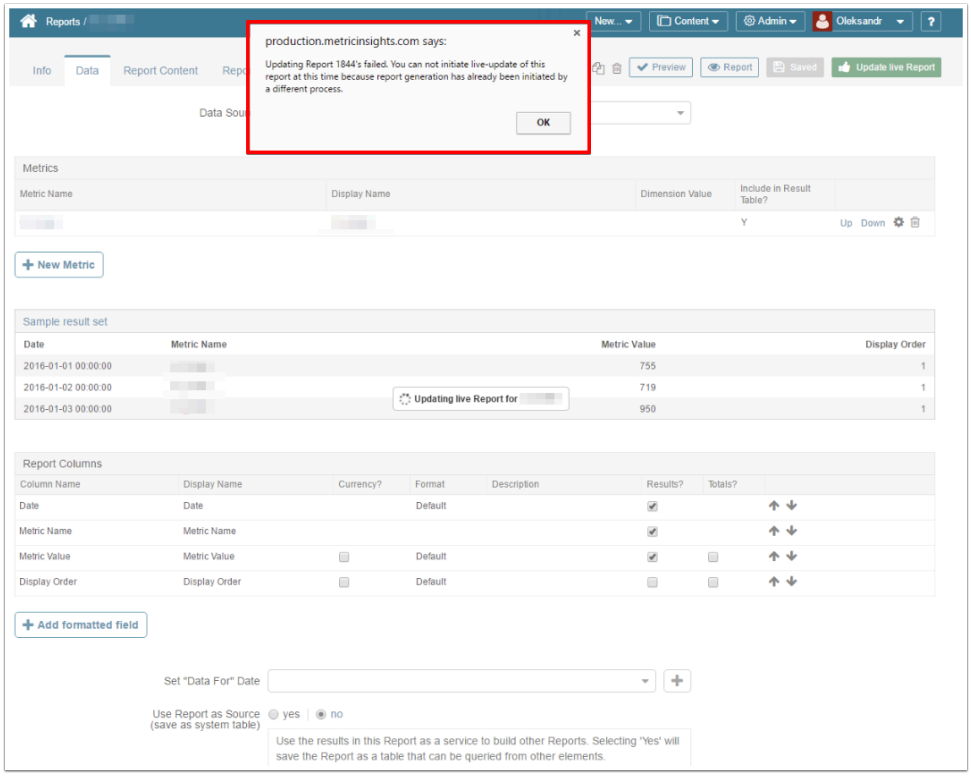
Issue

I see the following error during the most recent *data collection trigger* run for a report. When attempting to collect the data manually, I am getting the same error:

Updating Report failed. You can not initiate live-update of this report at this time because report generation has already been initiated by a different process.

What does this mean and why is this happening?





Resolution

This error is caused by a hanging data collection thread for that report. Normally, the thread closes upon completion, but this error indicates the thread is still open. Therefore, any subsequent data collection attempts, whether through the trigger or a manual collection, is prevented from executing to prevent data duplication.

Please follow the steps below to end the running thread:
1.) SSH to the Metric Insights server with "root" privileges.

2.) Run the following command to search for running processes:

```
ps -ef | grep report_generator
```

The output lists the PID (second column) and the element ID (number listed after *report_generator.py*)

```
root@[redacted]# ps -ef | grep report_generator
www-data  3780  2843  0  2016 ?        00:00:00 sh -c /opt/mi/generator/report_generator.py 1035 0 -rld_id "11259489" -user_id 0
www-data  3781  3780  0  2016 ?        00:00:45 /opt/mi/.python/bin/python /opt/mi/generator/report_generator.py 1035 0 -rld_id 11259489 -user_id 0
www-data 18079 18076  0 00:35 ?        00:00:00 sh -c /opt/mi/generator/report_generator.py 1844 0 -rld_id "11557647" -user_id 0
www-data 18081 18079  0 00:35 ?        00:00:01 /opt/mi/.python/bin/python /opt/mi/generator/report_generator.py 1844 0 -rld_id 11557647 -user_id 0
root     21137 58569  0 14:44 pts/0    00:00:00 grep report_generator
```

In this example, the PID is 18081 and the element ID for the report is 1844.

3.) Run following command to kill the running process in question by referencing the PID:

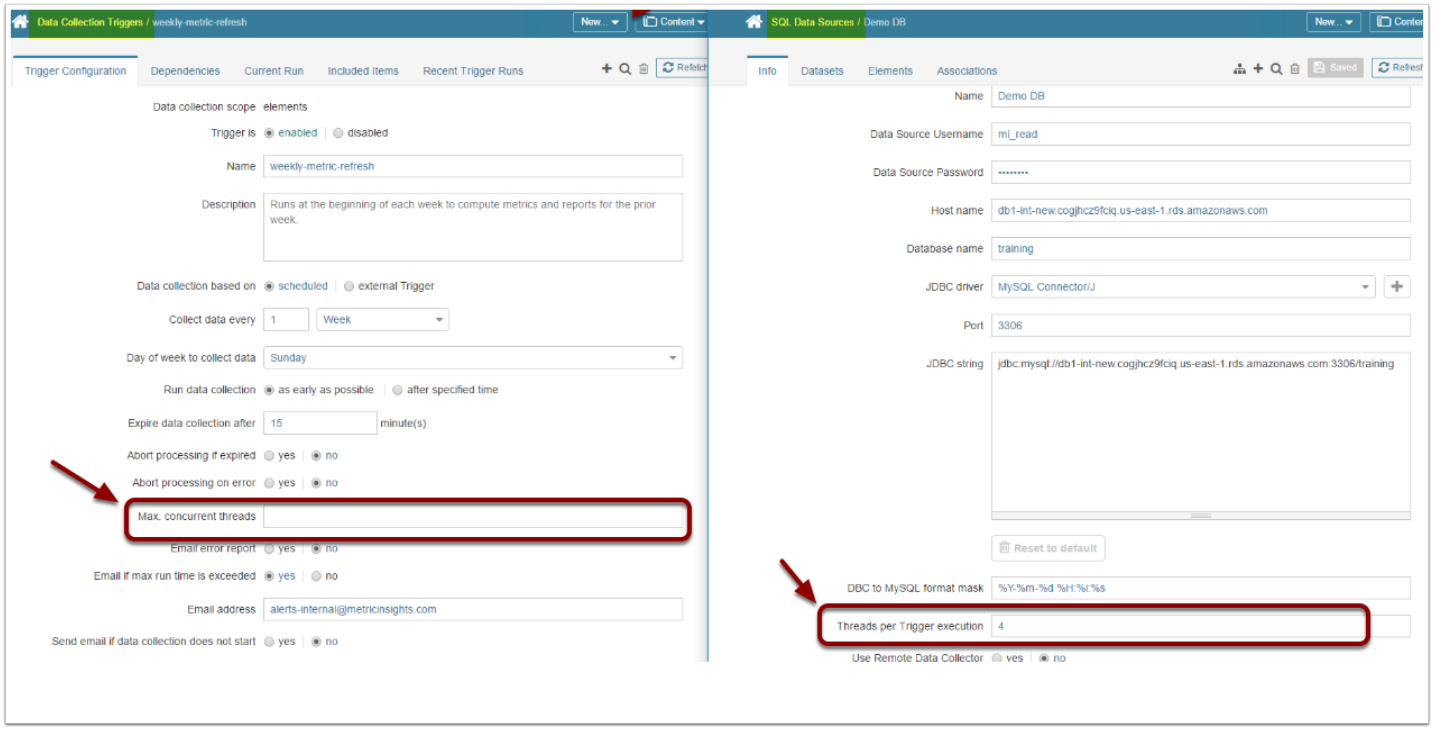
```
kill -9
```

Killing this long running thread allows the system to complete the data collection and automatically update the database. You should see a completion time in the report's Run History and you can proceed with running data collection again. For any additional questions, please contact support@metricinsights.com.

How do "Max Concurrent Threads" (Data Collection) and "Threads per Trigger execution" (Data Source) work together?

Question

How do the settings *Max Concurrent Threads* (Data Collection Trigger setting) and *Threads per Trigger execution* (Data Source setting) work together?



Answer

The data collection process for an element against a data source is executed via **threads**. This process may be single-threaded or across multiple threads. Threads are executed simultaneously and independently from each other. As seen in the images above, the maximum number of threads that get executed can be configured in the Data Collection Trigger and Data Source Editors. To understand how they both work in tandem, let's first go over each setting.

Max Concurrent Threads

By default, the **Max Concurrent Threads** field is left blank for *Data Collection Triggers*. The system treats this blank state as “unlimited number of threads allowed to be executed upon run.” Generally you can leave this setting as is, however, if you have many elements collecting

data for a given time period, you may need to limit the number of threads executed to lessen the load on the system.

For example, by setting the option to 4 means the trigger can only collect data for 4 elements at a time (one thread for each element). If there are 12 elements to be processed, then the trigger will collect data for 4 elements at a time (4, 4, 4). Comparatively, if you set this option to 1 for the same trigger, then the trigger will collect data for 1 element at a time, taking longer to complete.

Setting this option to 1 might be beneficial unless each element returns a large amount of data (thousands of rows) during data collection. In short, setting this option requires a bit of fine-tuning and understanding what the elements are collecting.

Threads per Trigger execution

The **Threads per Trigger execution** at the *Data Source* level functions the same way: to allow some number of threads to be executed at any given time for that trigger. However, whereas a blank setting for Data Collection Triggers equals "unlimited number of threads," for the Data Source, a blank setting equals "1 thread only”

So, what happens when both settings have different values? (see Diagram 1 below)

Let's assume you have data collection trigger *T* which collects data for a set of elements. All these elements source data from Data Source 1 (*DS1*) and Data Source 2 (*DS2*). For this example, DS1 is set to use *3 Threads per Trigger execution*, while DS2 is set to use *4 threads*. That means T will spawn 3 + 4 = *7 threads upon data collection run*.

Now assume T is set to use *2 Max Concurrent Threads*. That means only 2 (out of the 7 possible) threads can be active at any given time. While 2 threads are open, the other 5 are on standby. When one of the two active threads completes, an opening appears and the next thread in line is executed. This pattern continues until all elements have been collected.

Note, these collection jobs are all managed by a thread manager. This manager makes sure that:

1. All items (elements, calendars, datasets, etc.) are being grouped and updated in a specific order
   - To understand how triggers prioritize element execution by element type, please see this article: http://kb.metricinsights.com/m/43678/l/478480-in-what-order-are-items-processed
2. The number of allowed threads defined at the Data Source level does not exceed the number of maximum allowed threads on the Data Collection Trigger side. This is important because if left blank, again, the Data Source defaults to 1 thread but unlimited for the Trigger.

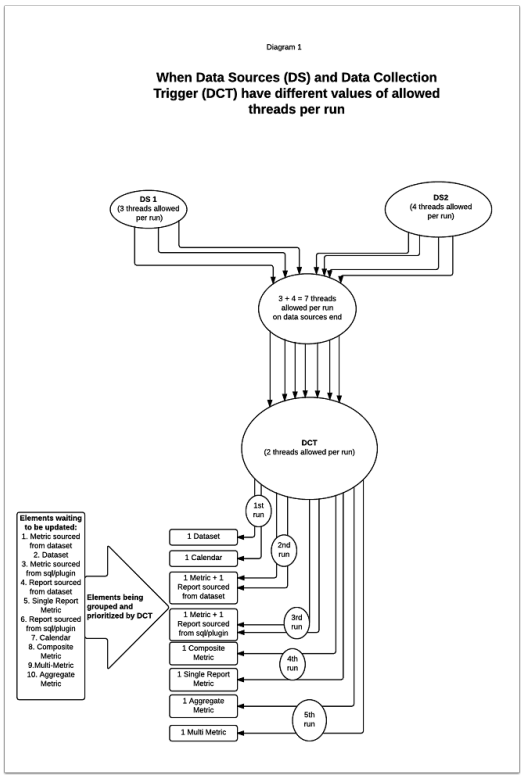
The thread manager compares the sum of allowed threads for all data sources involved (Total) with the number of maximum allowed threads per data collection trigger (Max). If the Total exceeds the Max, the thread manager allows the Max number of threads to run concurrently. If

the Total is less than the Max, then the thread manager allows only the Total number of threads to run.

To paint this example again (see Diagram 2 below), let's say the thread setting for Data Sources and Data Collection Triggers are left blank (default settings). Now, let's assume you have 6 elements which use the same trigger but gets data from 3 different data sources (DS1, DS2, DS3). With the default thread settings this means each data source allows a maximum of *1 thread each* per run while the trigger allows *unlimited threads*.

The thread manager groups the 6 elements, prioritizes them by type, then checks how many threads are allowed for each data source. With the default settings and 3 data sources, this means 3 threads are allowed to run concurrently (1 thread x 3 datasources = 3 total threads). That means that at any given time during the collection, only 3 elements are updated at a time, with the other elements forming a queue.

Metric Insights recommends leaving these fields empty to start with. *Then, based on system specs, the number of elements to collect data for, and dataset size, the number of threads allowed per run can be adjusted over time.*



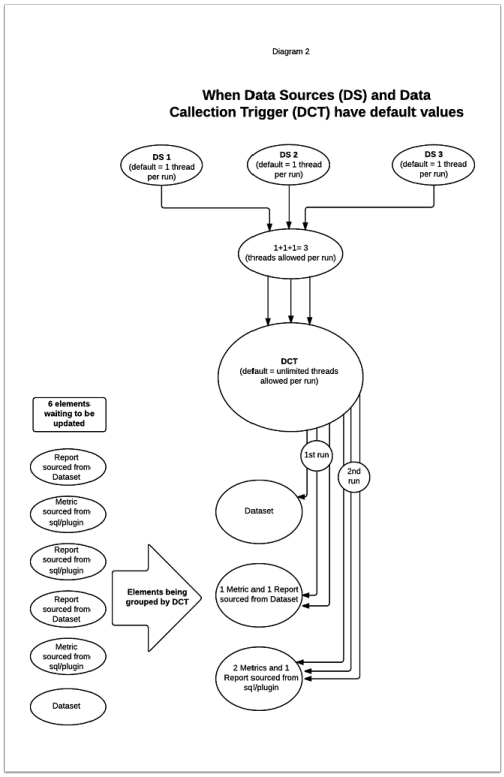
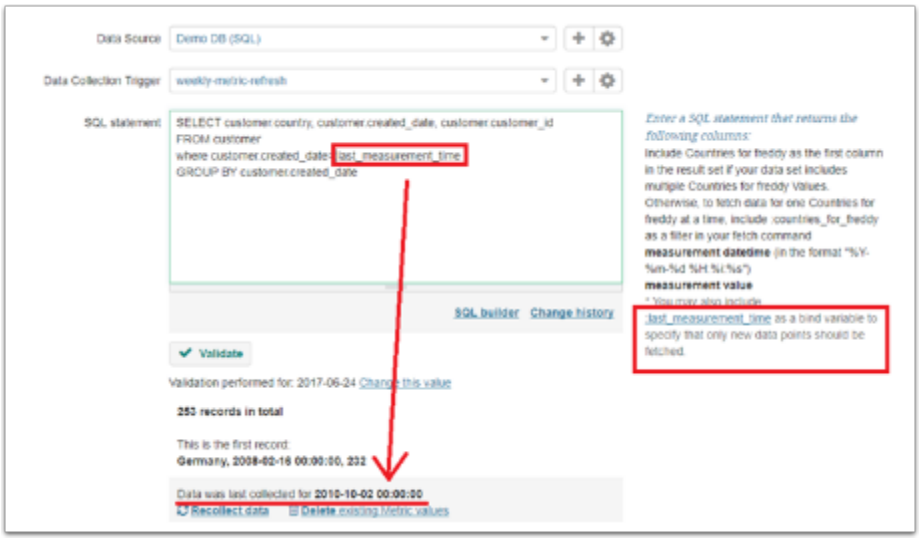


Diagram 2
When Data Sources (DS) and Data Callection Trigger (DCT) have default values
DS 1 (default = 1 thread per run)
DS 2 (default = 1 thread per run)
DS 3 (default = 1 thread per run)
1+1+1= 3 (threads allowed per run)
DCT (default = unlimited threads allowed per run)
6 elements waiting to be updated
Report sourced from Dataset
Metric sourced from sql/plugin
Report sourced from sql/plugin
Report sourced from Dataset
Metric sourced from sql/plugin
Dataset
Elements being grouped by DCT
1st run
2nd run
Dataset
1 Metric and 1 Report sourced from Dataset
2 Metrics and 1 Report sourced from sql/plugin

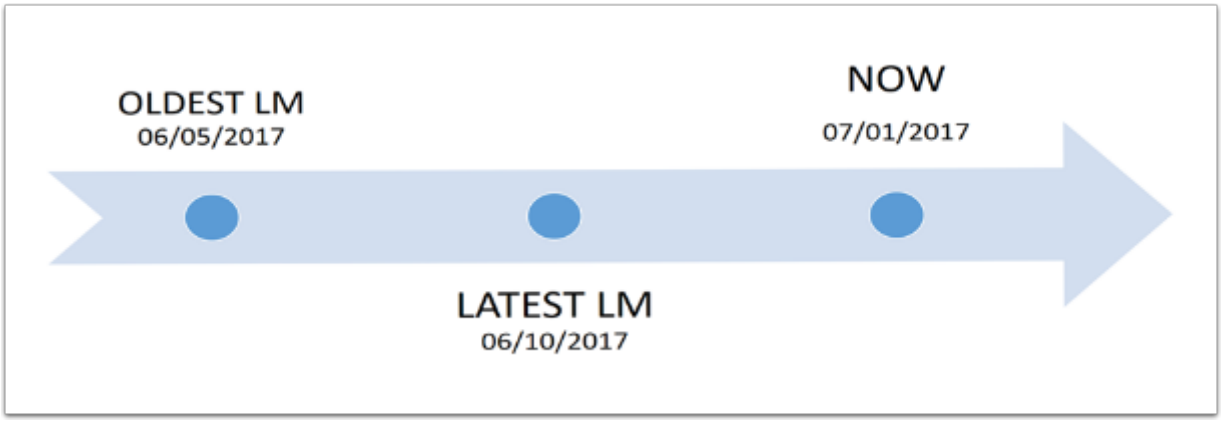
Using the :last_measurement_time bind parameter calculation dependency

Using the *:last_measurement_time* bind parameter can be perplexing especially for dimensioned metrics. Read on to understand how the system applies the bind parameter across dimensions during data collection.

For metrics, you must specify the time period you are collecting data for in the query. For example, if you state the exact time period in your where clause, like <where date_time between '2017-07-01' and '2017-07-07'>, the system fetches data for the period specified. However, you cannot use exact dates when collecting data going forward at set intervals. To handle this dynamically, the :last_measurement_time (LM) bind parameter is used to structure a query in MI to collect data going forward, without specifying exact dates. This bind parameter represents the last date collected by MI that is stored in the system database.



For undimensioned metrics, the LM always represents the last date Metric Insights has collected data for. That's simple enough. However, for a dimensioned metric, the last date collected for may vary across dimension values. The LM can either be the oldest date across all dimension values (OLDEST LM) or it can be the latest date across all dimension values (LATEST LM). The timeline below represents visually what this looks like in reference to the current date.



How the system determines which LM to use.

There are several variables and additional user settings the system considers when identifying which LM to use:

- *single-fetch* and *batch-fetch* variables
- Two interval options set in the metric editor:
  - *On data collection also re-run last*
  - *Batch load data going back no more than.*
- Manual data collection via the *Recollect Data* button or scheduled data collection

<*Single fetch*> is used to determine a specific dimension value the system should update and collect data for. <*batch-fetch*> allows the system to perform a bulk collection of all values with one query.

On data collection also re-run last and *Batch load data going back no more than* are two interval options the system considers when identifying a specific date to collect data from. These settings can be configured in the Metrics Editor > Data tab

These intervals represent the period the data is to be re-fetched for. They can be expressed in number of days, weeks or months – depending on the measurement interval chosen while creating the element.

· For instance, if 5 is set in *On data collection also re-run last* interval, the data collector will also check for updates going back 5 days from the LM.

· *Batch load data going back no more than* interval is ALWAYS disregarded when it comes to single-fetched data.

Now, with a dimensioned metric, please refer to the attached file to see how the LM date is selected depending on the conditions involved.

Table_1_for_the_last_measurement_doc.pdf

Alerts

Data Quality: Alert When Missing Data

⚠ New feature: How does 'Insert 0s for missing values' work? provides an updated way to alert on missing data.

There are two popular scenarios for alerting with there is missing data or a data source fails to update. In both cases, we start by pulling in your source data. Then we create a metric to implement some logic and set an alert rule.

Use Case #1: Alert me when I'm missing data / Alert me on NULL's

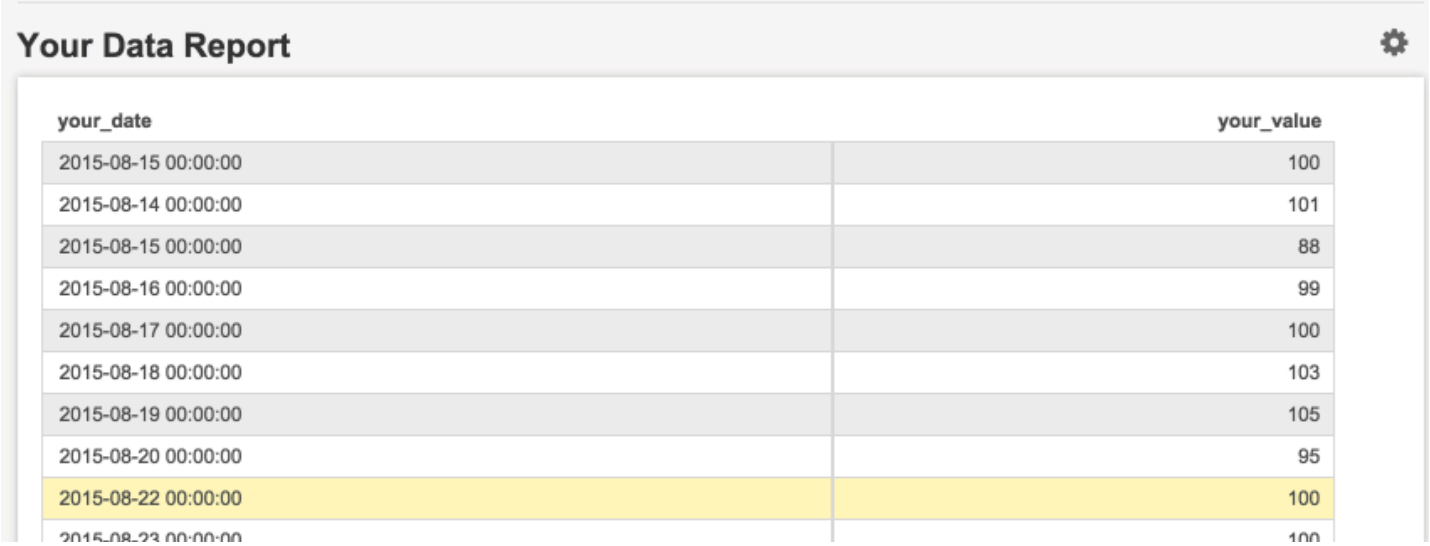
your_date	your_value
8/15/15	100
8/14/15	101
8/15/15	88
8/16/15	99
8/17/15	100
8/18/15	103
8/19/15	105
8/20/15	95
8/22/15	100
8/23/15	100
8/24/15	100
8/25/15	99
8/26/15	103
8/28/15	95
8/29/15	100
8/30/15	100
8/31/15	100

Values are missing on 8-21 and 8-27. If we create a metric on this data then nothing will be charted for those days. I want to be alerted when I'm missing data like this. If my data source returned a 0 then it would chart (and I could alert on it) so I'll have to insert the missing values in Metric Insights. The first step is to create a report and pull the source data into the Metric Insights Database.

Create a Report and Save as MySQL Table

Create a Report that pulls in your data. Use the data source for your source system and the appropriate data collection trigger (schedule).

View Report Data

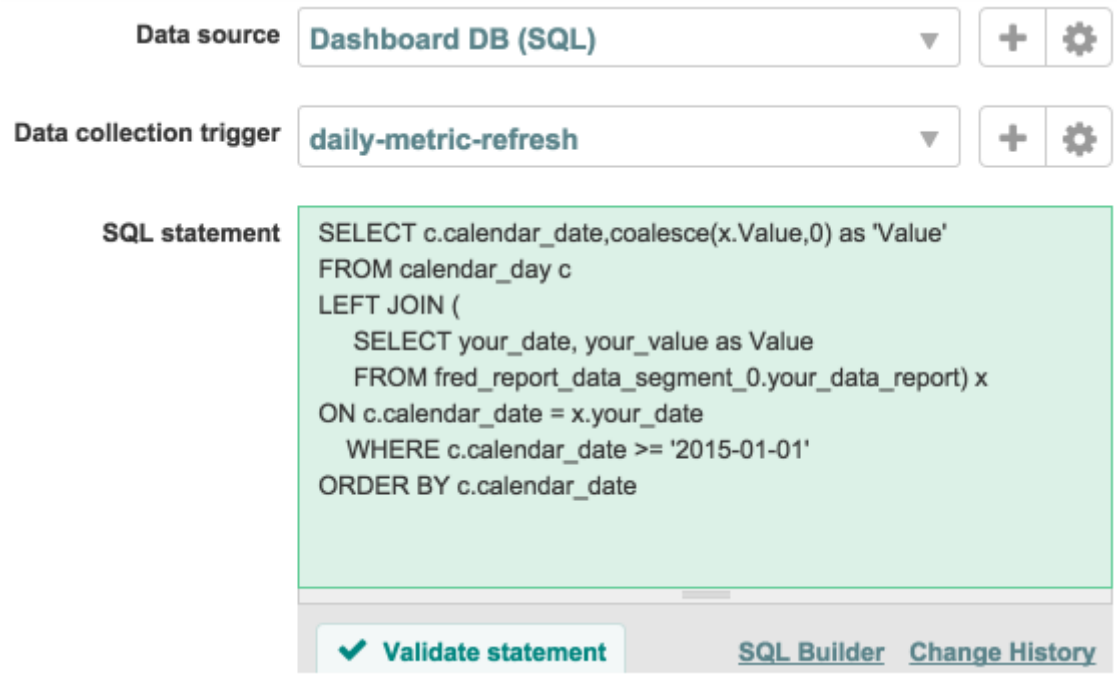


Your Data Report

your_date	your_value
2015-08-15 00:00:00	100
2015-08-14 00:00:00	101
2015-08-15 00:00:00	88
2015-08-16 00:00:00	99
2015-08-17 00:00:00	100
2015-08-18 00:00:00	103
2015-08-19 00:00:00	105
2015-08-20 00:00:00	95
2015-08-22 00:00:00	100
2015-08-23 00:00:00	100

Here is what my report looks like.

Create a Metric the inserts Zeros and Alerts



Use the Dashboard Database so that we can query both the system's 'calendar_date' table and your report's table. This query only pulls data from 2015 and newer (the calendar table goes back to 1980).

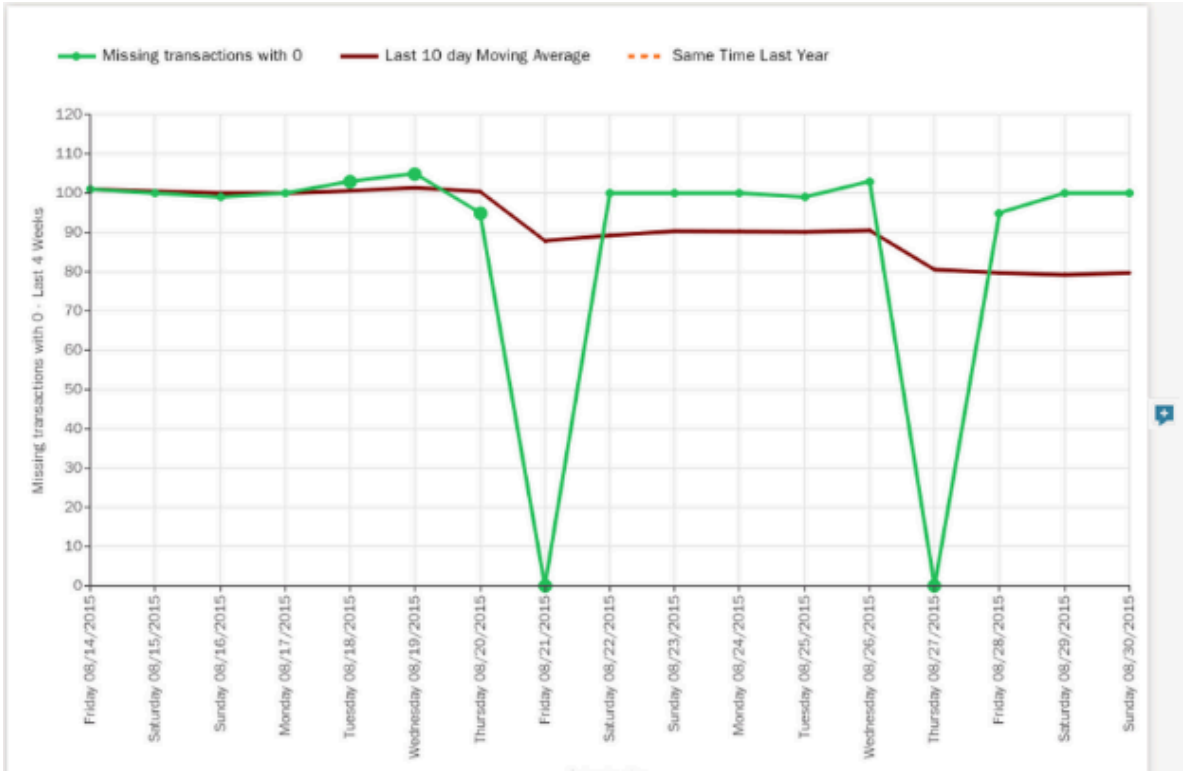
SQL:

```
SELECT c.calendar_date,coalesce(x.Value,0) as 'Value'
FROM calendar_day c
LEFT JOIN (
   SELECT your_date, your_value as Value
   FROM fred_report_data_segment_0.your_data_report) x
ON c.calendar_date = x.your_date
  WHERE c.calendar_date >= '2015-01-01'
ORDER BY c.calendar_date
```

Calendar Table

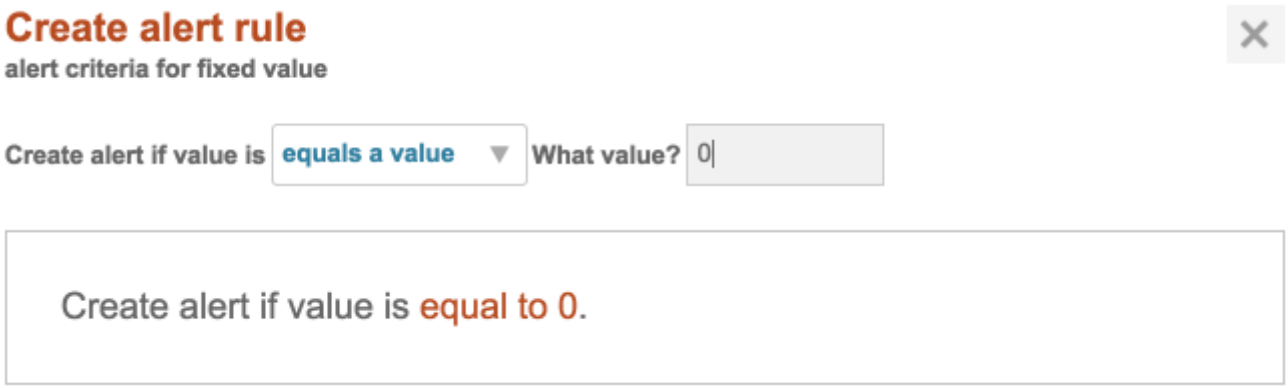
```
mysql> desc calendar_day ;
+-------------------+---------------+------+-----+---------+----------------+
| Field             | Type          | Null | Key | Default | Extra          |
+-------------------+---------------+------+-----+---------+----------------+
| day_id            | int(11)       | NO   | PRI | NULL    | auto_increment |
| name              | varchar(10)   | NO   |     | NULL    |                |
| day_of_week       | tinyint(4)    | NO   |     | NULL    |                |
| calendar_date     | date          | NO   | UNI | NULL    |                |
| week_id           | int(11)       | NO   | MUL | NULL    |                |
| month_id          | int(11)       | NO   | MUL | NULL    |                |
| quarter_id        | int(11)       | NO   | MUL | NULL    |                |
| year_id           | int(11)       | NO   | MUL | NULL    |                |
| fiscal_month_id   | int(11)       | YES  | MUL | NULL    |                |
| fiscal_quarter_id | int(11)       | YES  | MUL | NULL    |                |
| fiscal_year_id    | int(11)       | YES  | MUL | NULL    |                |
| is_dayoff_ind     | enum('Y','N') | YES  |     | N       |                |
+-------------------+---------------+------+-----+---------+----------------+
```

Metric Chart



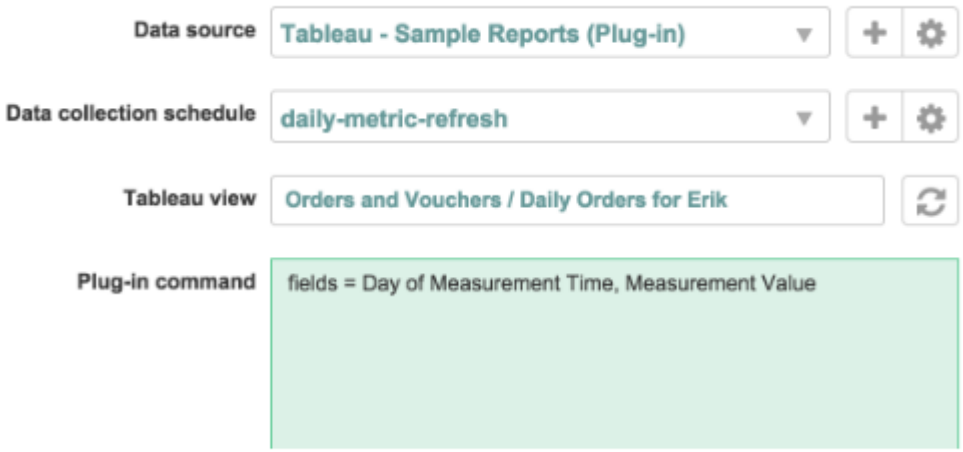
Here is the joined data on the metric's chart. On 8-21 and 8-27 we chart the inserted zeros.

Alert Rule if Value = 0



Then create an alert rule based on a fixed value.

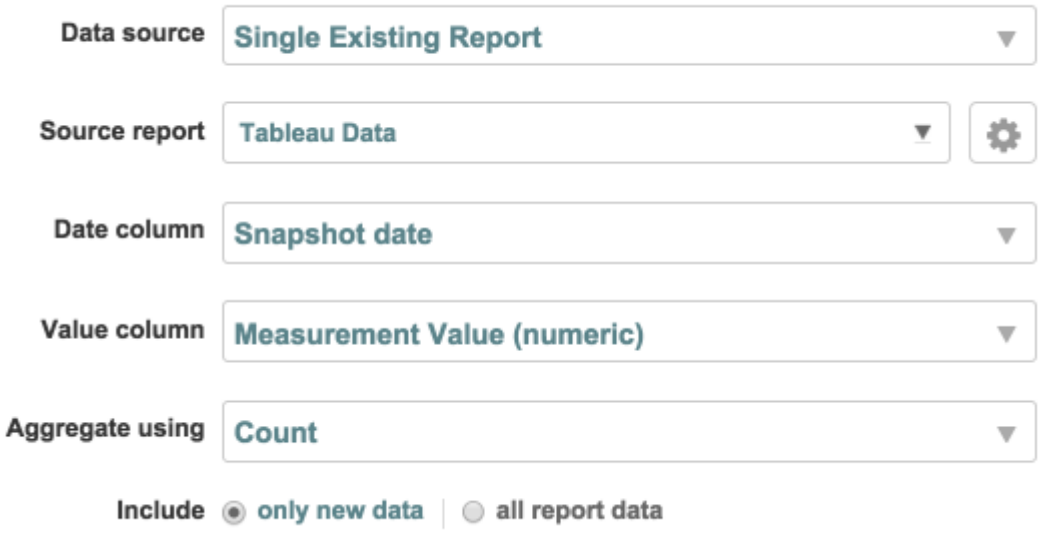
Use Case #2: Alert me when my Data Source Doesn't Update (Variation)



I have a Tableau view and I want to be alerted when it doesn't update. The strategy here is to create a report and pull in data from the data source (Tableau in this case but could be anywhere data is expected to load consistently).

Save as MySQL Table

Create Metric that Counts Report Records



Note: Using version 3.3's Single Existing Report Data Source (New). You can use the 'Existing Report' or 'Dashboard DB' to query the same data like this:

SELECT measurement_time, count(*)

FROM report_table_name

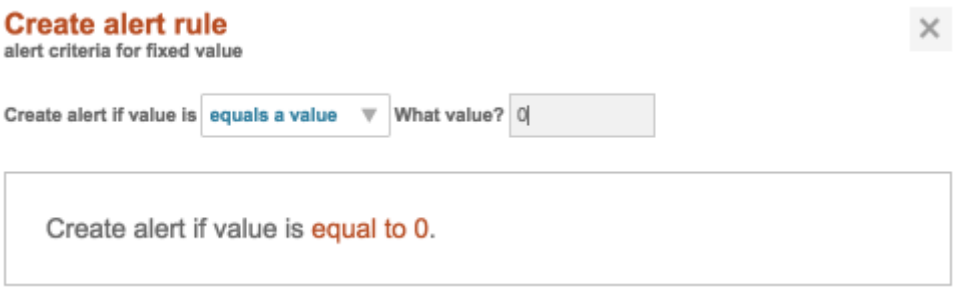
GROUP BY 1

'measurement_time' is the snapshot date Metric Insights applies when saving the report as a MySQL table. The count represents the numbers of records Metric Insights pulls into the report. If the count is zero than no data has been loaded. If you have false alerts check the timing of data collection.

Metric Chart

Remember the measurement_time column only has dates on a go-forward basis. When you create the metric you will only have one data point to start with. This chart tells is the report pulled in 89 records. That is good. All we care about is that there are records at all.

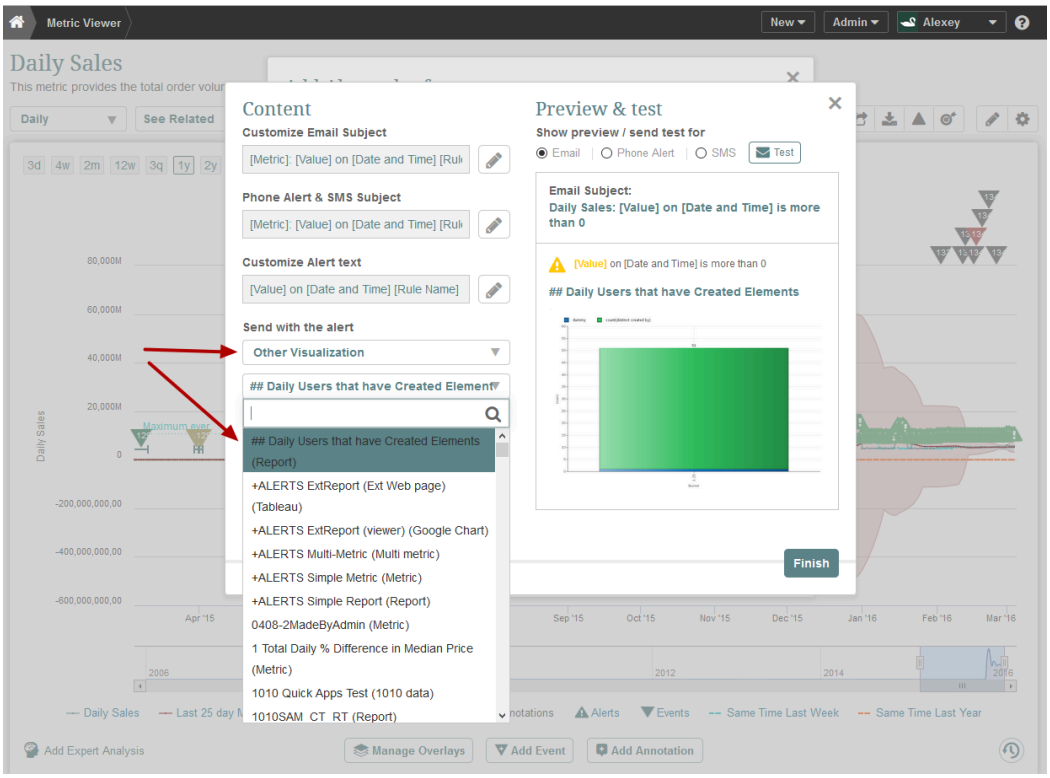
Alert if there are No Records



Create the alert rule to fire when there are no records (value =0). You can change the threshold if needed so that you are alerted when there are unusual values (statistical alerting).

Report/External Report/Metric linking with Metric for alerting

Description



In Metric Insights v.4.0 we moved away from conception of linking multiple Reports/External Reports to one Metric Alert rule. Now we can choose only one element to be linked with one Alert Rule.

On the screenshot you can see *Send with the alert* drop-down where you can select *Other Visualization* and choose Report/External Report/Metric from the list that would be sent to your email when Alert Rule is triggered.

Note. If your Metric with Alert Rule is dimensioned, you can choose visualization from elements that are not dimensioned or dimensioned by the same dimension as this Metric.

How do I re-issue an alert from the UI?

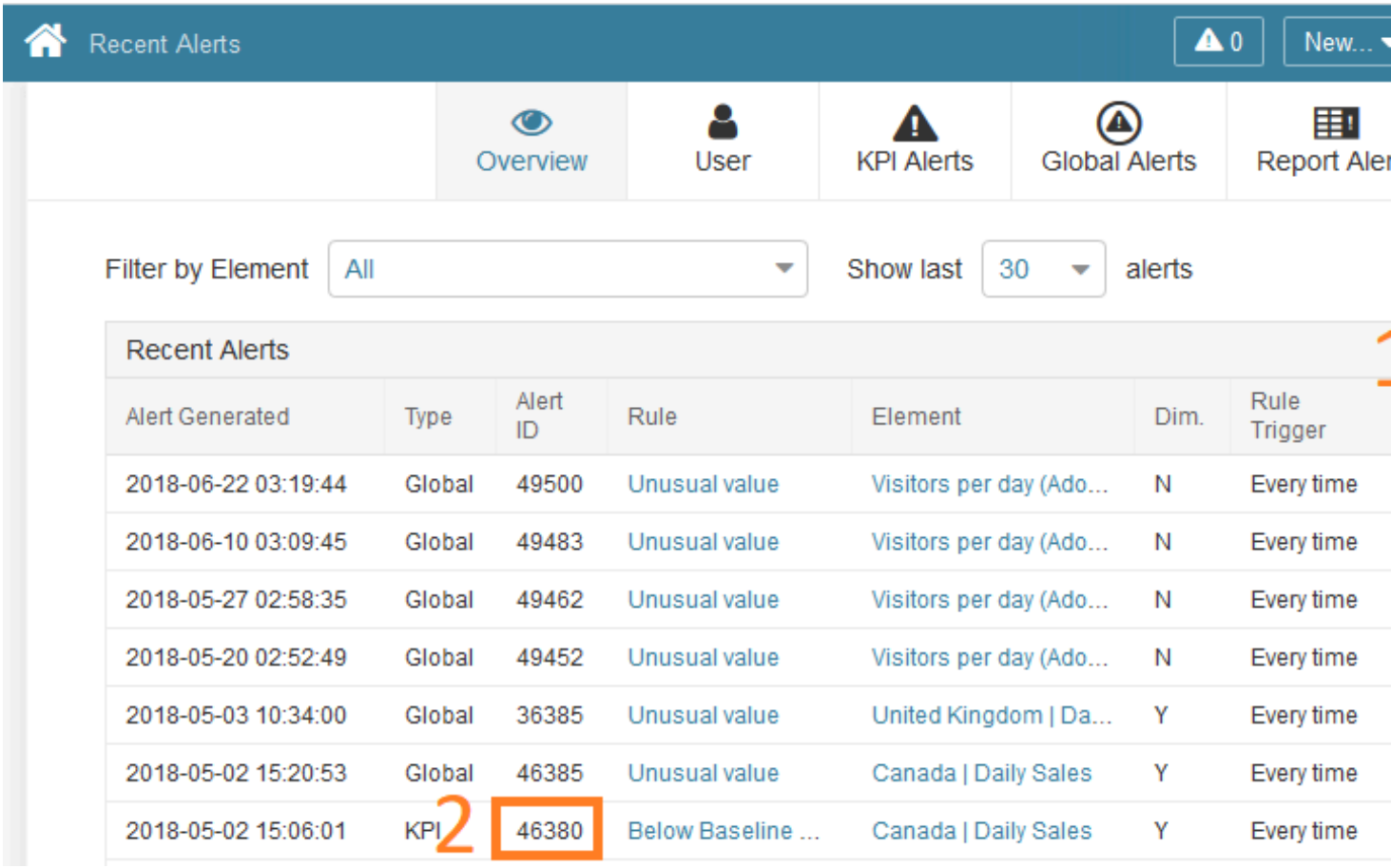
Question:

How do I re-issue an alert without having to delete/re-collect data?

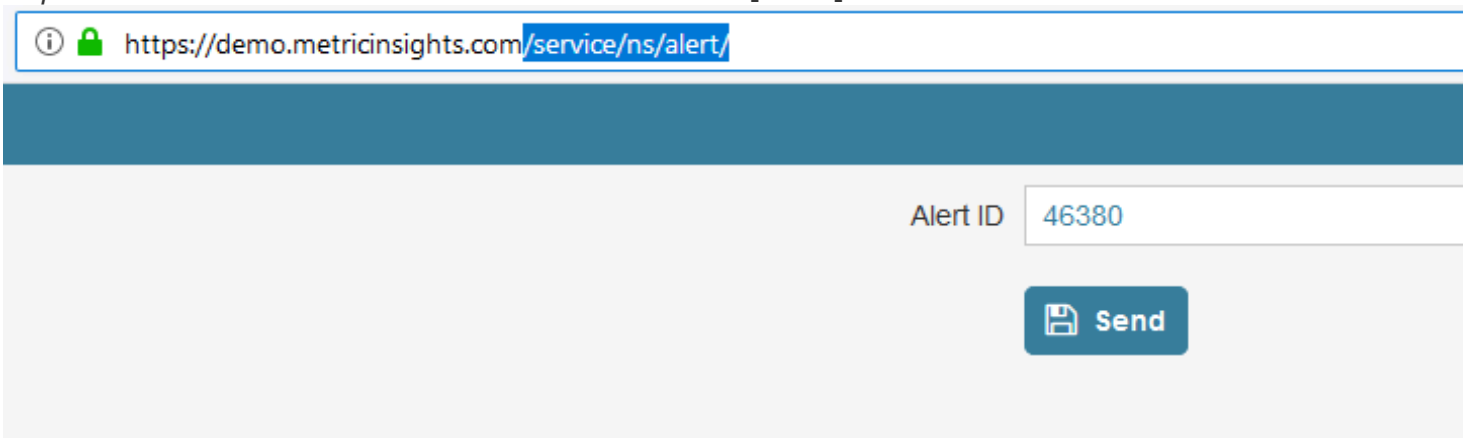
Answer:

To re-issue an alert without having to delete/re-collect data please follow the steps below:

1. Get the Alert ID number for the alert you want to re-issue: Content > Alerts > Overview > "Alert ID" column



2. Next, in another browser window, paste **/service/ns/alert/** after the hostname. You'll be presented with an 'Alert ID'. Enter the Alert ID number and press [Send]. For example: *https://<hostname>.com**/service/ns/alert/*** then hit [Send].



Deeper Alert Event Analysis

Question:

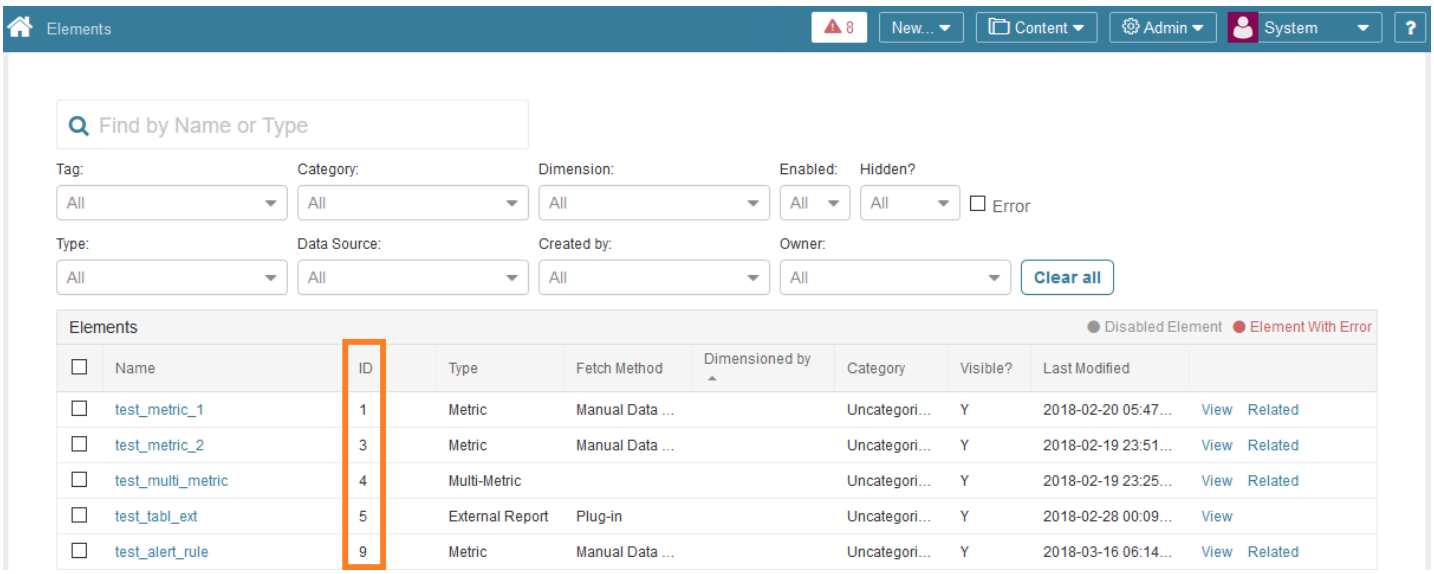
We have KPI Alerts going out but I need to do a deeper analysis of the alerts. How do I do this?

Solution:

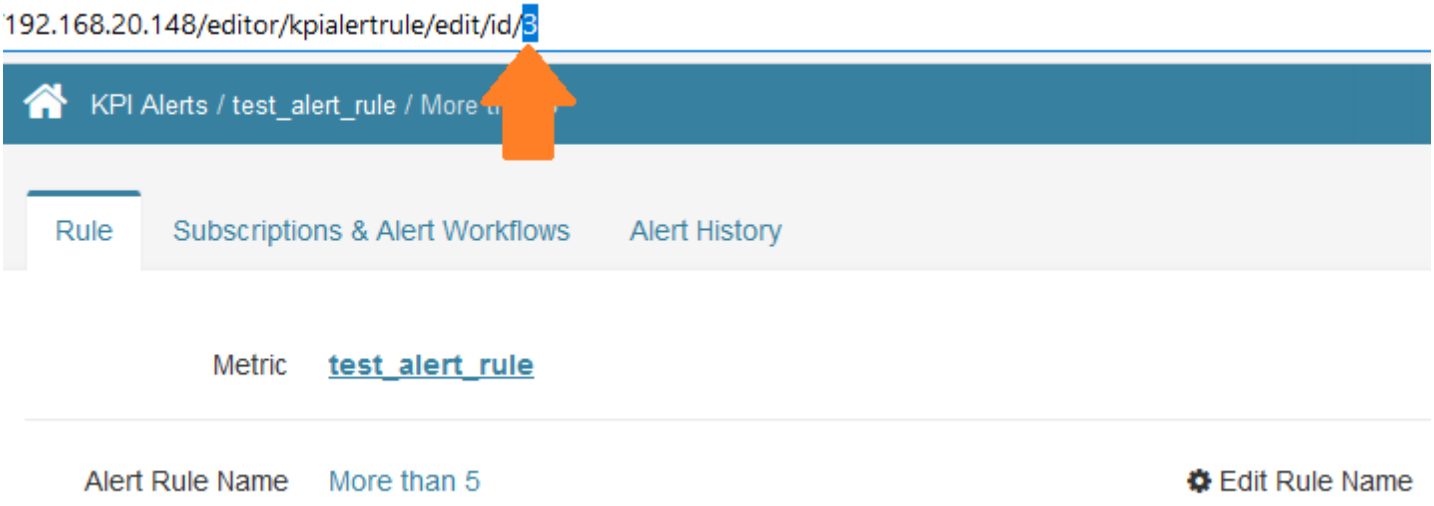
In the event you need to do a deeper analysis of alert events generated in Metric Insights (MI), you can query the MI database to tease out the information you need. The primary table holding alert event information is alert_event in the dashboard database. You can either create a dataset querying against the default 'Dashboard DB' data source, or query directly in MySQL if you have access to the db server.

For example, let's say you have a metric with a large number of alerts and you want to know how many alert events satisfied some condition. The dashboard.alert_event table contains all the fields needed to run this analysis. Here are the two primary fields you'll need to structure your queries:

1) **element_id** — This represents the Metric in question. You can also grab this from the URL when viewing the metric or from the ID column in the Elements List page as shown below:



2) **alert_rule_id** — This represents the Alert Rule that was created for the metric. You can also grab the ID from the URL by going to the Content menu > Alerts > KPI Alerts > clicking on the Alert Rule > grab the numeric value from the URL (../id/**some_number**):



After you've identified the Metric and Alert Rule you want to analyze by element_id and alert_rule_id, you can use the following fields to further structure your query:

- **is_first_time_only_ind** — 'Y' (yes) means the Alert Event is generated for the first time only when some defined condition is met per an Alert Rule. Subsequent data points that meet the condition do not generate an Alert Event until the value drops out of the condition.
- **is_frequently_ind** — 'Y' (yes) means the Alert Event is generated when some defined condition is met for a range of subsequent values (N out of M data points)
- **is_frequently_first_time_only_ind** — 'Y' (yes) means if the parameter *is_frequently_ind* is met for the first data point in the range, then an Alert Event is generated for only the first data point in that range.
- **alert_mask** — This is an indicator that combines the three previous parameters and is the result of a binary sum as shown below. Note, the parameter *alert_every_time* is always equal to 1 if enabled for the alert rule:

Row (condition) name in the table alert_event	TRUE	FALSE
+ alert_every_time	1	-
+ alert_first_time_only	2	0
+ alert_frequently	4	0
+ alert_frequently_first_time_only	8	0
= alert_mask	**Sum()**	

Example: The query below essentially states "find all alert events which have been generated for a metric with ID '123' and the *alert_first_time_only* condition is met":

```
SELECT COUNT(*)
FROM alert_event
WHERE element_id=123 AND alert_mask = 3;
```

How to obtain the archived MI alert records?

Issue

Is there a way to obtain a logbook for all alerts generated since day one?

Resolution

Alert records on the Alert History tab are kept based on system settings; the longer the measurement interval of the Metric is (associated with an Alert), the longer the Alert records can be seen on this tab.

Default history settings for element measurement intervals are the following:

- Minute - 4 days,
- Hourly - 7 days,
- Daily - 45 days,
- Weekly - 18 weeks,
- Monthly, Quarterly, Annual - 5 years.

After the time periods above, alert records are archived, are not shown in the Alert History tab anymore and move to another system table. The archived alert records are deleted from the archive only if all the related metrics are deleted or if the alert rule is deleted.

These records can be used e.g. form the list of aggregated number of alerts per metric for statistical purposes (using MI Dataset, that collects data by MySQL query). If you need to extract this kind of historical alert information, please contact support@metricinsights.com to get a specific MySQL query, according to your business case.

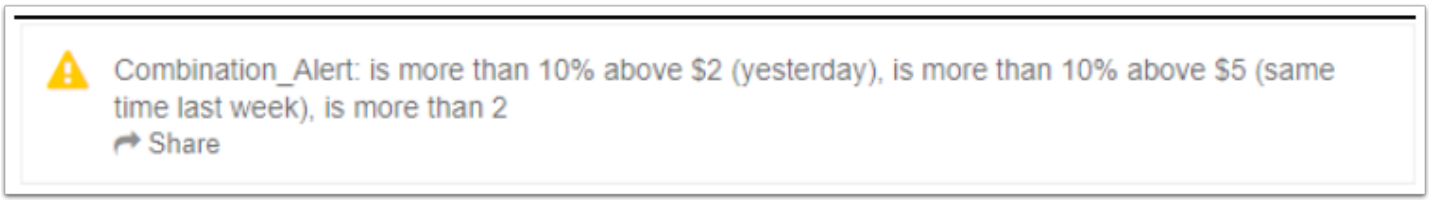
Note

Unfortunately, the historical MI alert notification content is not possible to collect in accessible format, as the contents are encoded before the notification is sent (via burst/SMS/CSV etc.). Furthermore, alert notification contents are stored in MI only for 7 days.

How to change Alert text for Combined Alert?

ISSUE

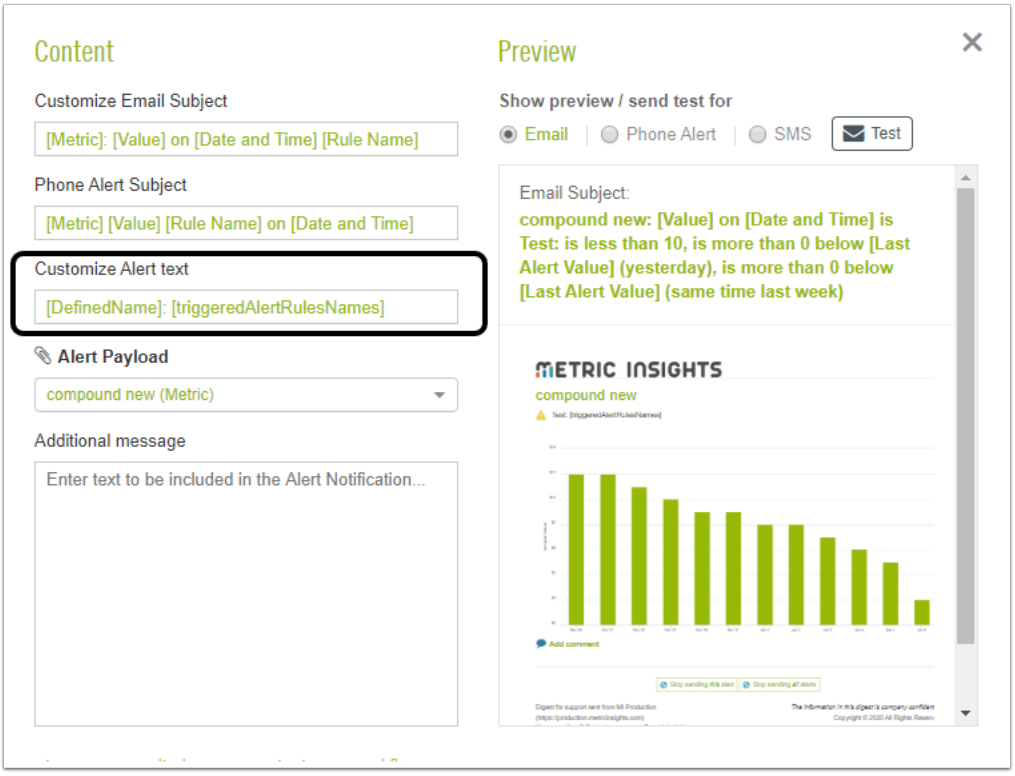
By default, the text that displayed in the metric viewer when an alert is triggered for Combined Alert contains only alerts condition and values that are present in alerts condition but does not contain actual value and alert triggered date what is most often required by the customer. For example:



So how can we add the necessary information (value and date) to make the text more informative?

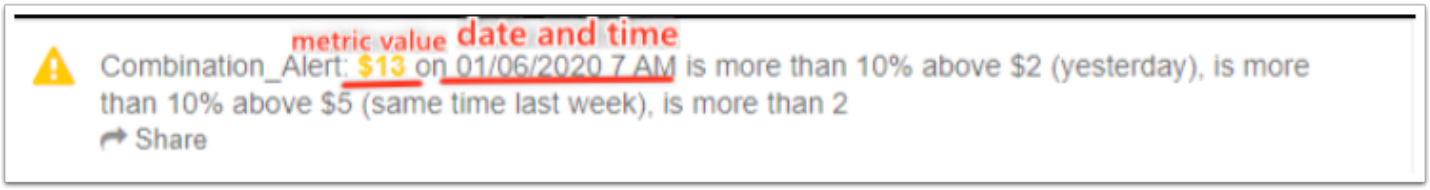
RESOLUTION

The text displayed in the metric viewer is configured in the alert editor - Content tab - Customize Alert text



This field contains only [DefinedName] and [triggeredAlertRulesNames] by default. But builder for this field allows us to add the values we need.

It remains only to determine where in the text we want to see the values we need. For example, add value and date after the rule name, but before the triggered rules. In this case, we will get the next result:

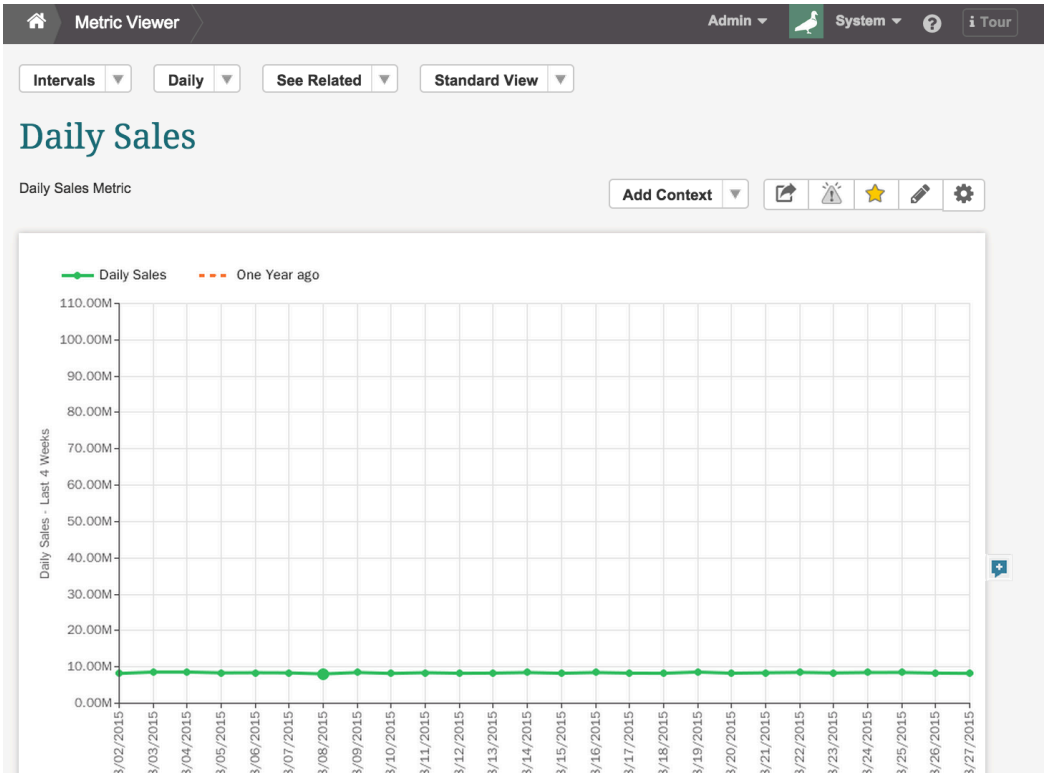


Charts

Y scale is too large for data being displayed

ISSUE

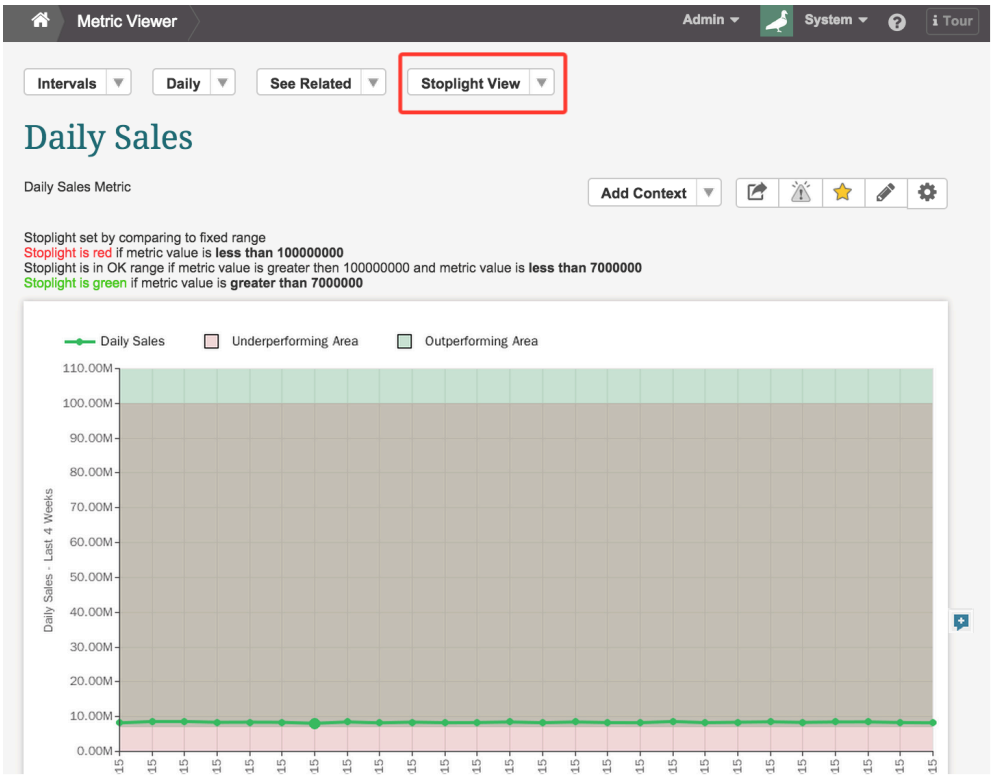
In my Metric Viewer, the y scale is way too big for the data being displayed - as if it were *zoomed out*. What's going on?

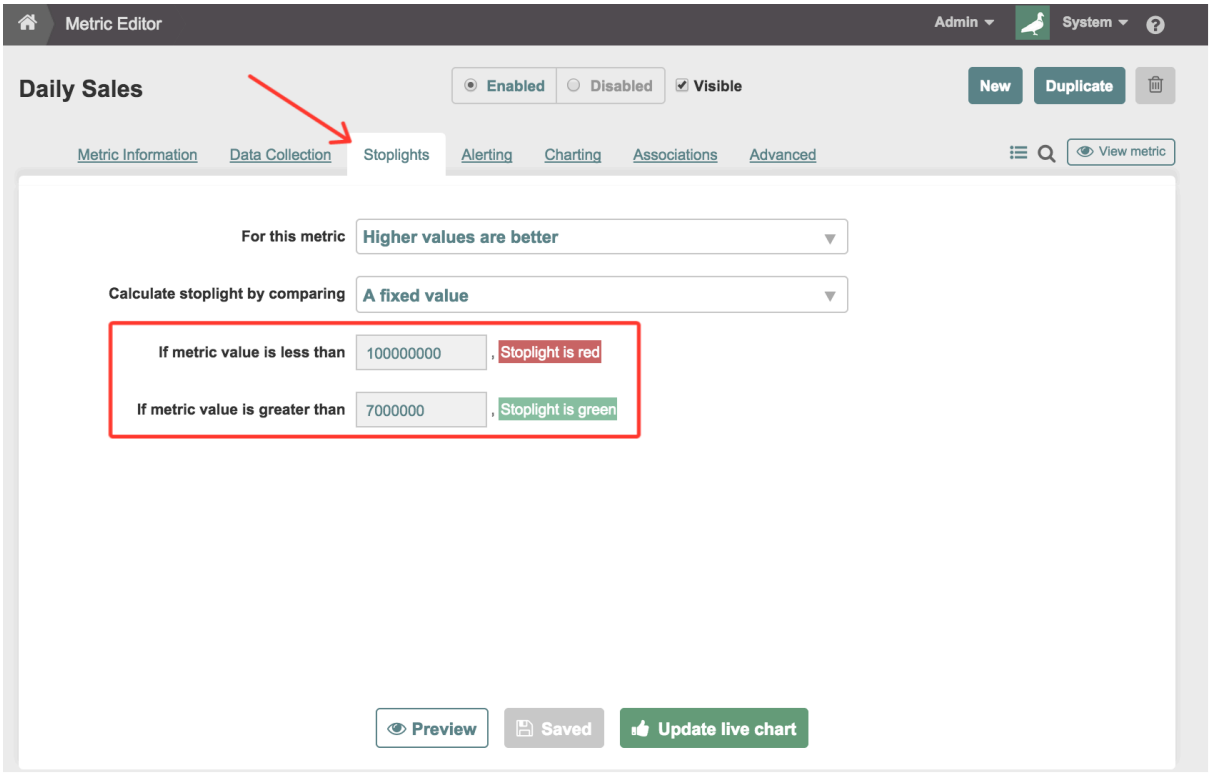


RESOLUTION

The issue is with the stoplight threshold that is set for that particular metric. If you switch the viewer to the Stoplight View, you'll see the threshold is set very high compared to the data that is charted (see image below).

Change the threshold lower to adjust the y scale accordingly, unless that is the actual threshold you want.


Metric Viewer
Admin
System
Tour
Intervals
Daily
See Related
Stoplight View
Daily Sales
Daily Sales Metric
Add Context
Stoplight set by comparing to fixed range
Stoplight is red if metric value is less than 100000000
Stoplight is in OK range if metric value is greater then 100000000 and metric value is less than 7000000
Stoplight is green if metric value is greater than 7000000
Daily Sales
Underperforming Area
Outperforming Area
110.00M
100.00M
90.00M
80.00M
70.00M
60.00M
50.00M
40.00M
30.00M
20.00M
10.00M
0.00M
Daily Sales - Last 4 Weeks

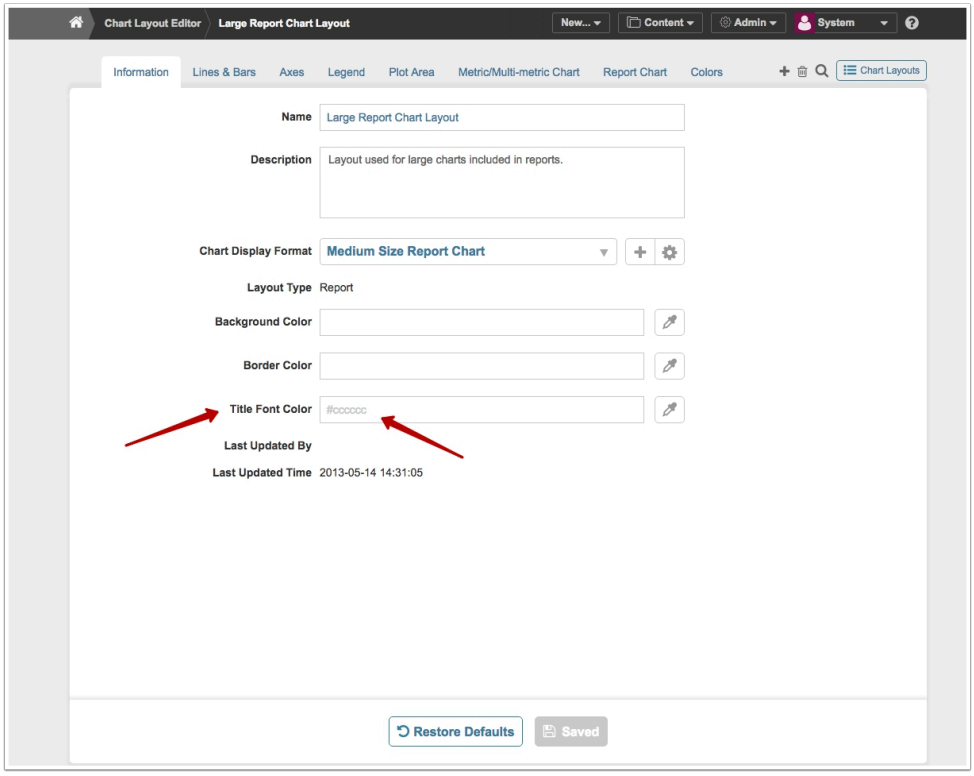

Metric Editor
Admin
System
Daily Sales
Enabled
Disabled
Visible
New
Duplicate
Metric Information
Data Collection
Stoplights
Alerting
Charting
Associations
Advanced
View metric
For this metric
Higher values are better
Calculate stoplight by comparing
A fixed value
If metric value is less than
100000000
Stoplight is red
If metric value is greater than
7000000
Stoplight is green
Preview
Saved
Update live chart

Changing color of chart legend

For chart legend text used color from Chart Layout settings
To to change it, follow the steps below:

1. Go to Admin -> Charting Options -> Report Chart Layouts ->Large Report Chart Layout

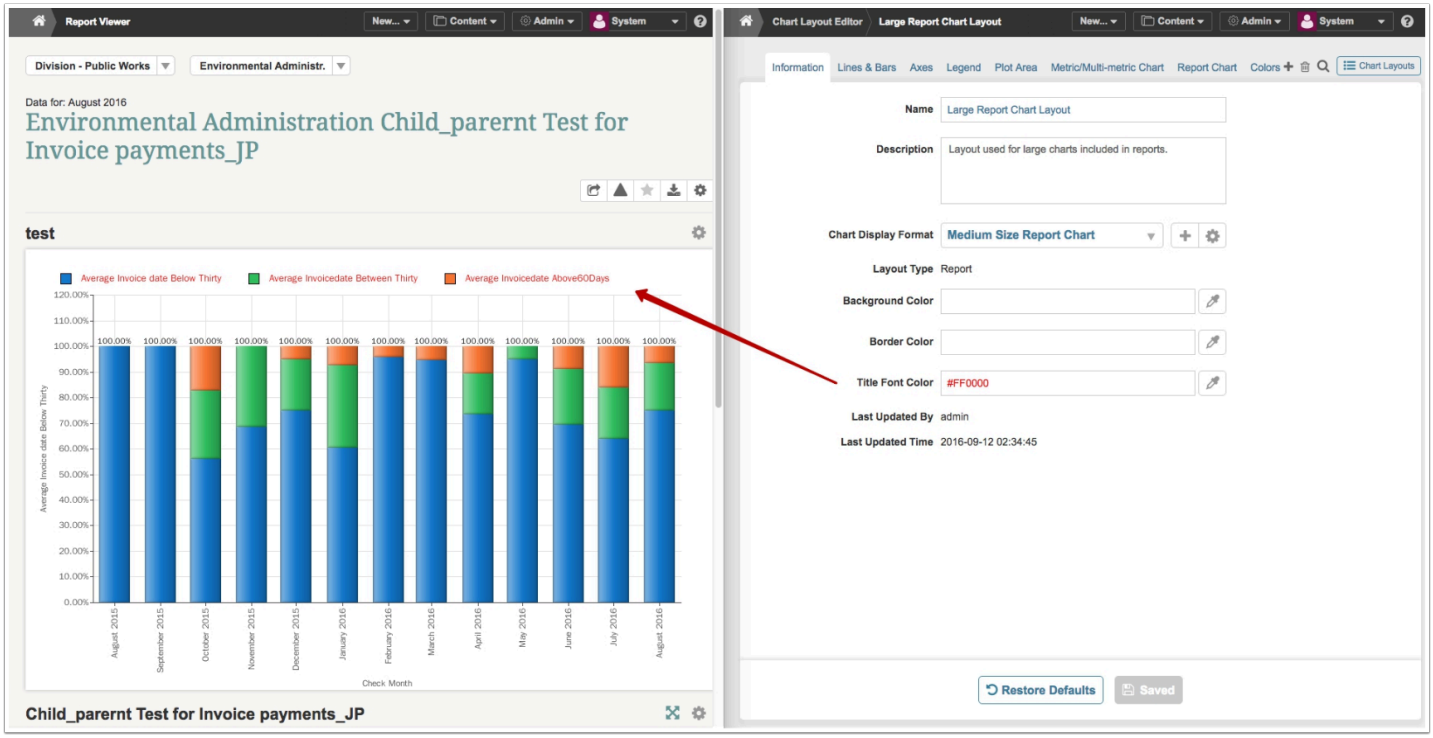
2. Change color in field "Title Font Color"



3. Save

4. Go to any Report editor and Update live Report

Example of change:


Report Viewer
New...
Content
Admin
System
Division - Public Works
Environmental Administr.
Data for: August 2016
Environmental Administration Child_parernt Test for Invoice payments_JP
test
Average Invoice date Below Thirty
Average Invoicedate Between Thirty
Average Invoicedate Above60Days
Average Invoice date Below Thirty
Check Month
Child_parernt Test for Invoice payments_JP
Chart Layout Editor
Large Report Chart Layout
Information
Lines & Bars
Axes
Legend
Plot Area
Metric/Multi-metric Chart
Report Chart
Colors
Chart Layouts
Name
Large Report Chart Layout
Description
Layout used for large charts included in reports.
Chart Display Format
Medium Size Report Chart
Layout Type
Report
Background Color
Border Color
Title Font Color
#FF0000
Last Updated By
admin
Last Updated Time
2016-09-12 02:34:45
Restore Defaults
Saved

How to set your JavaScript chart so it will adapt its size to the container size.

The Metric Insights viewer is not adaptive, meaning it does not allow chart sizes to be set dynamically.

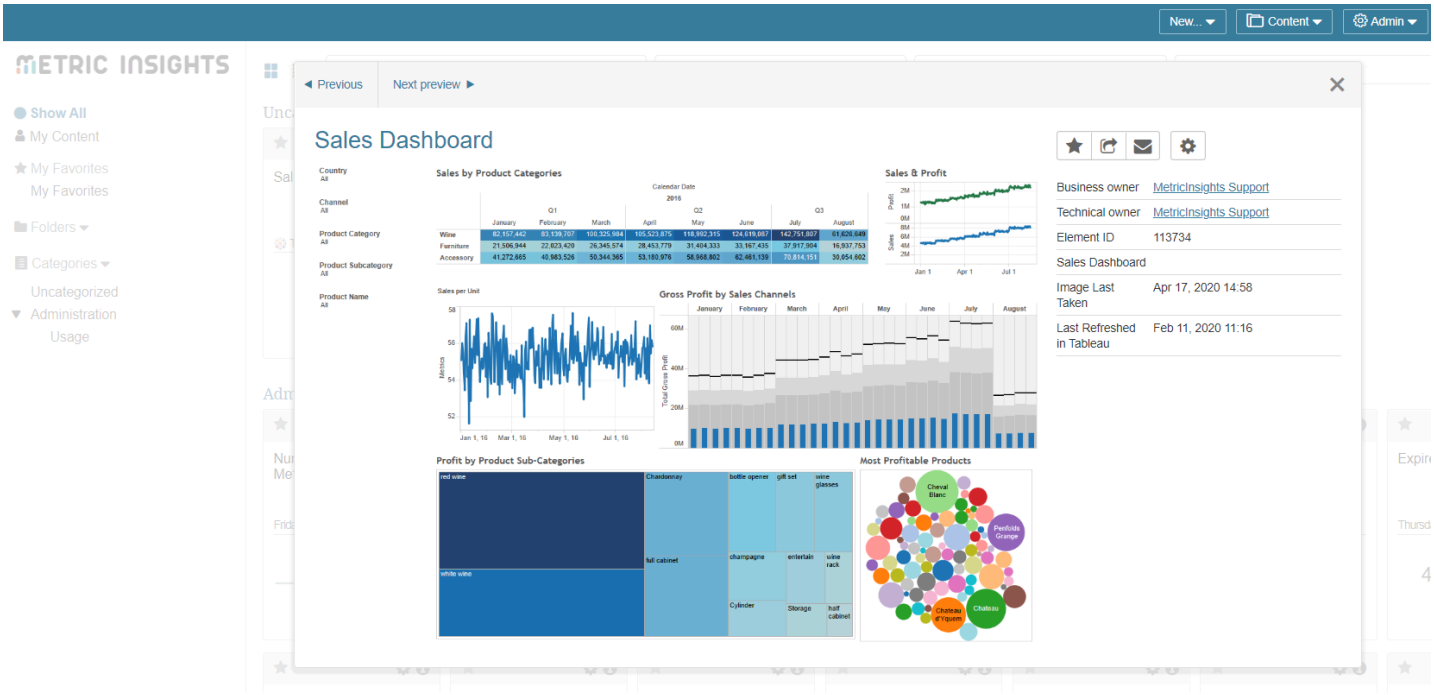
However, if you are displaying a chart in your own web application, you can adapt the chart size to the container size by setting the following parameters:

```
cvWidth = $('body').width();
cvHeight = $('body').height();
```

External Reports Tile Previews loading insights

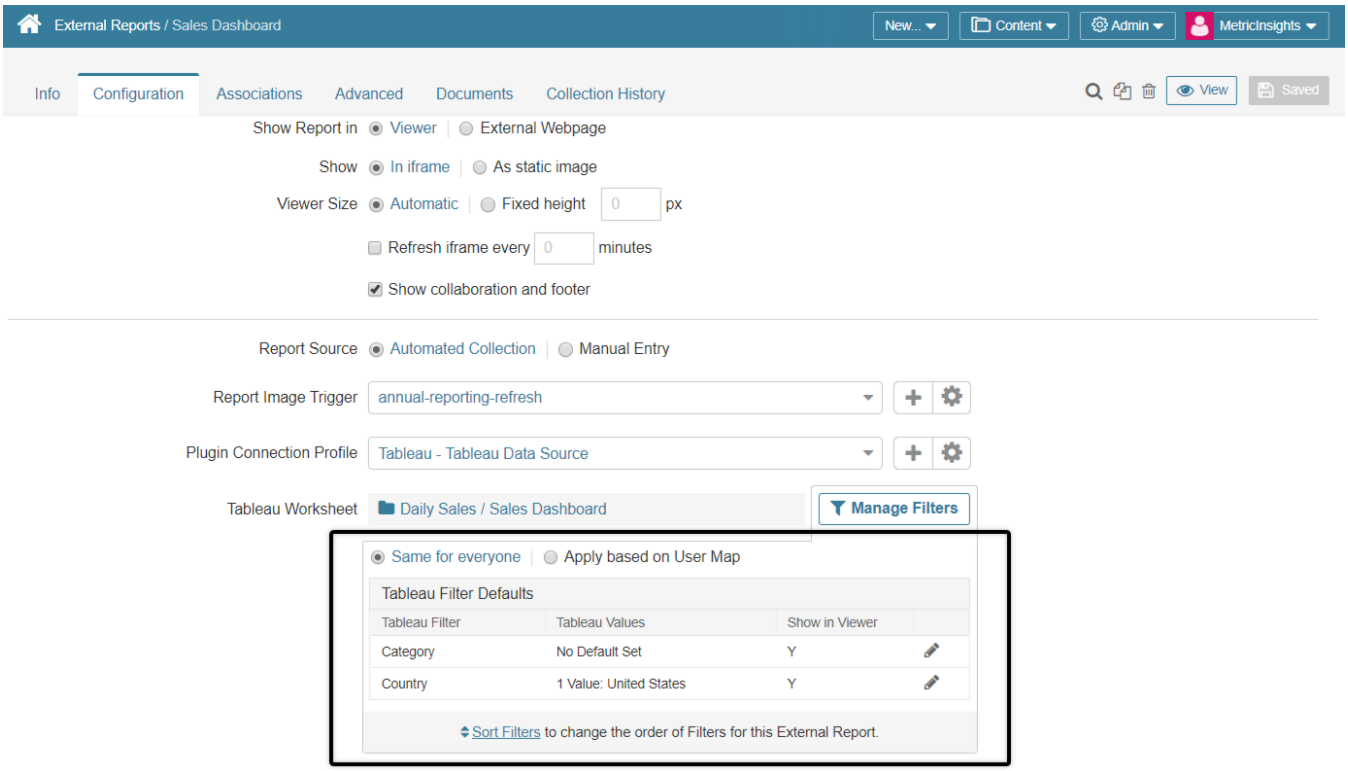
Issue

Why some Tile Previews for External Reports load takes time (from 10 sec to 1 min) even though the Previews show pre-captured images (not live visualizations)?



Explanation

Prior to v6.1.2. The Tile Previews loading may take time if they have *External Filters*.



For External Reports with Filters Tile Preview images are collected and cached only when you open the Preview directly. The image that was collected during Trigger run is not cached therefore makes no impact on the picture load for Tile Preview.
The cache is cleaned after every trigger run, so for daily triggers the first Tile Preview opening will take time every day. If you open the same Tile Preview several times per day, subsequent openings load quickly.

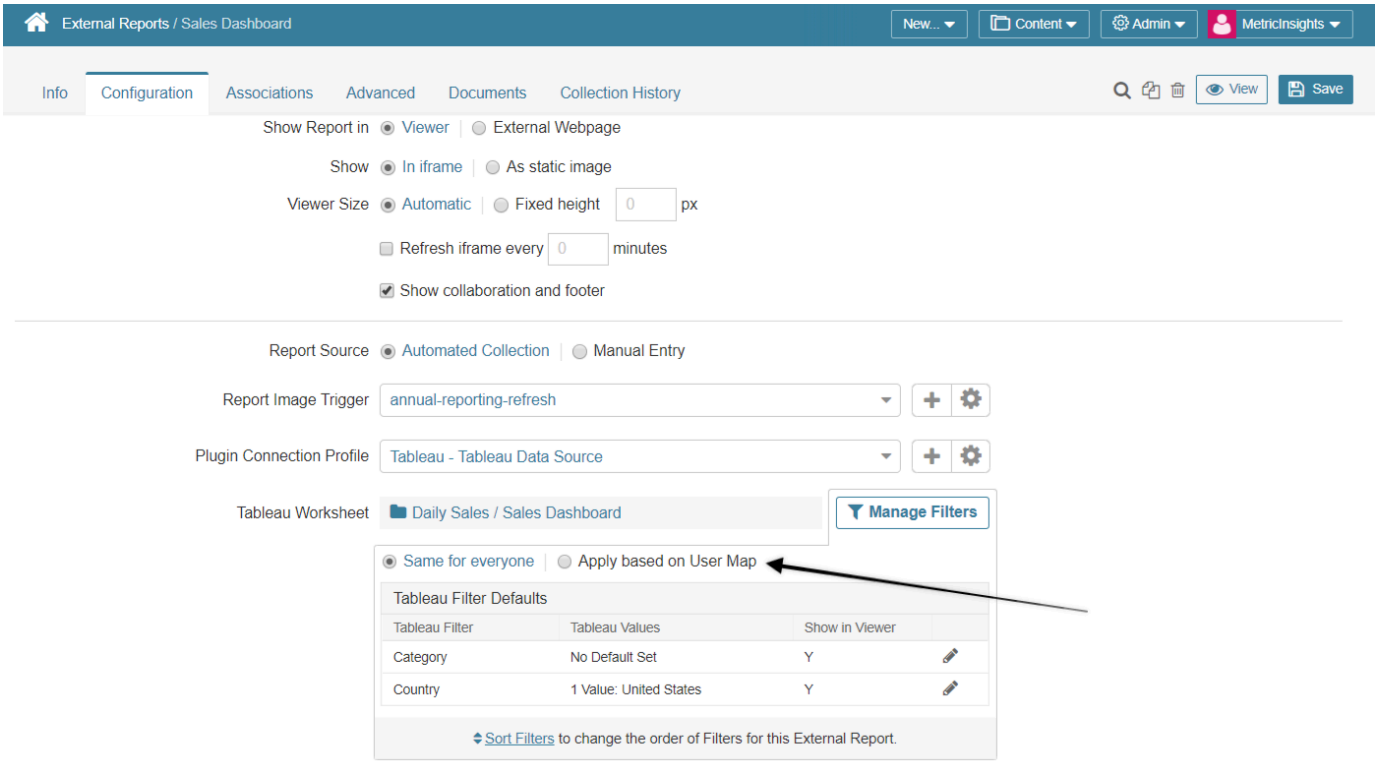
As for External Reports *without filters*, Tile Preview loads the image that was collected during Trigger run as it is cached.

Starting from v6.1.2 the way it works for External Reports *with filters* was changed: the image collected during Trigger run is also cached with default set of filters. Tile Previews use that image so the load is faster.

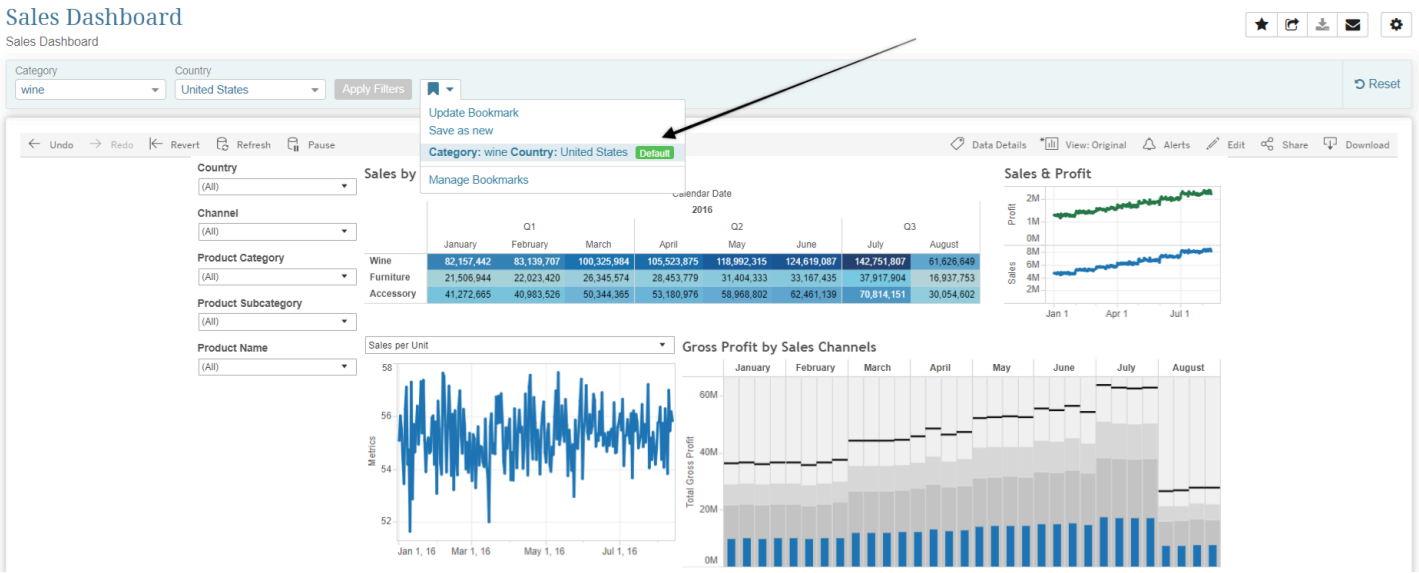
Exceptions

The Tile Preview loading will still take time if there is a personal set of filters for a certain User:

If Apply based on User Map option is set to define External Filters



If User has bookmarked certain Filter Values in the Element Viewer



In last two cases the Tile Preview is to be loaded personally for certain User with a personal set of filters.

Statistical queries (SQL statements)

Query for report that lists all elements with alert subscriptions for every user

Use the following sample query (SQL statement) if you need report that lists all elements with alert subscriptions for every user (Data Source = Dashboard DB (SQL))

```
SELECT `t`.* FROM
  (SELECT
     u.display_name AS `User`,
     IF(arei.visualization_type = 'other', arei.visualization_element_id,arei.
element_id) AS `element_id`,
     ldesv.element_dashboard_name AS `Element Name`,
     de.type AS `Type`,
     mi.name `Frequency`
    FROM `user_alert_rule` AS `uar`
    JOIN `alert_rule` AS `ar` ON ar.alert_rule_id=uar.alert_rule_id
    JOIN `alert_rule_element_info` AS `arei`
      ON arei.alert_rule_id=uar.alert_rule_id AND arei.element_id = uar.element_id
    JOIN `dashboard_element` AS `de`
      ON de.element_id=IF(arei.visualization_type = 'other',arei.
visualization_element_id,arei.element_id)
    JOIN `user_dashboard_element_instance` AS `udei`
      ON udei.user_id=uar.user_id AND udei.element_id=de.element_id
         AND udei.segment_value_id=
             IF(uar.segment_value_id=-1,udei.segment_value_id,
                IF( (SELECT segment_id FROM dashboard_element
                        WHERE element_id = IF(arei.visualization_type = "other",arei.
visualization_element_id,arei.element_id)) = 0, 0, uar.segment_value_id))
         AND udei.`favorite_id`=0
    LEFT JOIN `segment_value` AS `sv` ON sv.segment_value_id=udei.segment_value_id
    LEFT JOIN `last_dashboard_element_segment_value` AS `ldesv`
      ON (ldesv.element_id,ldesv.segment_value_id)=(udei.element_id,udei.
segment_value_id)
    JOIN user AS u ON u.user_id = uar.user_id
    JOIN measurement_interval AS mi ON mi.measurement_interval_id = de.
measurement_interval_id
  WHERE (IF(sv.total_ind IS NULL,1,sv.total_ind = "N" OR (de.show_segment_total_ind="Y"
AND sv.total_ind = "Y")))
  GROUP BY element_id, uar.user_id
  HAVING (element_id IS NOT NULL)) AS `t`
ORDER BY 1,2 ASC;
```

Query for the report that lists the number of alert notifications for each day (delivered not more than a week ago)

Metric Insights stores information about delivery within the week.
Use the following sample query (SQL statement) if you need report that lists the number of alert notifications (distinct emails / SMS messages / smartphone alerts) for each day (Data Source = Dashboard DB (SQL)).
But it does not give results on Alerts which were delivered more than week ago.

```
SELECT
  CAST(IFNULL(eq.created, eqdl.created) AS DATE) AS date,
  IF(aeen.is_alert_phone_ind IS NULL,NULL,
      IF(aeen.is_alert_phone_ind = 'Y', 'phone tray',
         IF(aeen.is_sms_ind = 'Y', 'sms',
            IF(aeen.is_immediate_ind = 'Y', 'immediate email', 'email digest')))) AS
`delivery_method`,
  COUNT(*) AS amount
FROM alert_event_email_notification AS aeen
  LEFT JOIN `email_queue` AS `eq` ON eq.email_queue_id = aeen.email_queue_id
  LEFT JOIN `email_queue_delete_log` AS `eqdl` ON eqdl.email_queue_id = aeen.
email_queue_id
  WHERE eq.created IS NOT NULL OR eqdl.created IS NOT NULL AND aeen.
user_email_notification_enabled_ind='Y'
GROUP BY 1, 2;
```

Query for report that lists all expired/active elements according to categories

Use the following sample query (SQL statement) if you need report that lists all **expired** elements for each category.

```
SELECT DISTINCT dc.category, de.name, de.element_id,
IF(de.type = "multi-metric chart",ldesv_mm.expiration_time,ldesv.expiration_time) AS
`Expiration Time`
FROM dashboard_element AS de
JOIN dashboard_category AS dc ON (dc.category_id=de.category_id)
JOIN last_dashboard_element_segment_value AS ldesv ON (ldesv.element_id=de.element_id)
LEFT JOIN dashboard_element de_mm_stoplight ON de_mm_stoplight.element_id = de.
multi_chart_stoplight_metric_element_id
LEFT JOIN segment_value_info AS sv_mm_stoplight ON sv_mm_stoplight.segment_id =
de_mm_stoplight.segment_id AND sv_mm_stoplight.segment_value_id IN (0, ldesv.
segment_value_id)
LEFT JOIN last_dashboard_element_segment_value ldesv_mm ON (ldesv_mm.element_id,
ldesv_mm.segment_value_id) = (de_mm_stoplight.element_id, sv_mm_stoplight.
segment_value_id) AND ldesv_mm.expiration_time IS NOT NULL
WHERE
(de.type = "multi-metric chart" AND NOW() > ldesv_mm.expiration_time)
OR (de.type != "multi-metric chart" AND NOW() > ldesv.expiration_time)
ORDER BY 1,2
;
```

Use the following sample query (SQL statement) if you need report that lists all **active** elements for each category.

```
SELECT DISTINCT dc.category, de.name, de.element_id,
IF(de.type = "multi-metric chart",ldesv_mm.expiration_time,ldesv.expiration_time) AS
`Expiration Time`
FROM dashboard_element AS de
JOIN dashboard_category AS dc ON (dc.category_id=de.category_id)
JOIN last_dashboard_element_segment_value AS ldesv ON (ldesv.element_id=de.element_id)
LEFT JOIN dashboard_element de_mm_stoplight ON de_mm_stoplight.element_id = de.
multi_chart_stoplight_metric_element_id
LEFT JOIN segment_value_info AS sv_mm_stoplight ON sv_mm_stoplight.segment_id =
de_mm_stoplight.segment_id AND sv_mm_stoplight.segment_value_id IN (0, ldesv.
segment_value_id)
LEFT JOIN last_dashboard_element_segment_value ldesv_mm ON (ldesv_mm.element_id,
ldesv_mm.segment_value_id) = (de_mm_stoplight.element_id, sv_mm_stoplight.
segment_value_id)
AND ldesv_mm.expiration_time IS NULL
AND ldesv.expiration_time IS NULL
WHERE
```

```
(de.type = "multi-metric chart" AND NOW() < ldesv_mm.expiration_time)
OR (de.type != "multi-metric chart" AND NOW() < ldesv.expiration_time)
ORDER BY 1,2
;
```

Why is there an anomalous number of element views being logged in the dashboard database?

ISSUE

There's an unusually high number of view times being logged in an engagement report that queries the *dashboard_element_view_log_detail* table. Why is this happening? Nobody is viewing the element that many times!

RESOLUTION

The issue is that the system generates a "view" each time an image is generated from a report dataset. These "views" are currently being logged to the *dashboard_element_view_log_detail* table. All of these "fake views" are assigned to an administrator user with the lowest *user_id*. Please use following sql statement to find this Administrator user:
select user_id, username, first_name, last_name from user where is_administrator_ind = "Y" order by user_id asc limit 1;

Please exclude view logs for this user from your engagement report (in the SQL statement) to avoid displaying unusually high number of element views that are not really "views."

Known Issues

Known Issues 2020

Each section covers a topic relating to an object or function within Metric Insights.

1. Security Model

1.1. Portal Page Extended Security is associated with the wrong parent

When adding Portal Page Privileges to a PU that has none,

1. Choose Extended Security: "Allow Power Users to grant access to Portal Pages, Templates and/or Layouts and as well as to a Portal Page or Asset Folder to any User or Group"
    1. Actual Behavior: Manage Portal Page Assets is selected as the "parent"
    2. Expected Behavior: Create/Edit Portal Pages is selected automatically as the parent

1.2. Inconsistent Behavior on List Pages

There are three types of behavior:

- Dataset List: (Security Model) When a PU is given Edit Access to a Dataset from this page, the "Create Datasets" Privilege is automatically granted in error.
- Elements List:
    - PU's elements with View Access do not appear but other page shown objects without Active Name list
    - PU's with Edit Access will be directed from the Active Name link to an Error Page if all Privileges and/or Permission have not been granted
- Other Objects List Pages
    - Active Name links do not appear until all Privileges and/or Permission have been granted