TRIDIUM

NS2022

ACCELERATING INNOVATION

Building a Niagara Archive History Provider

Scott Hoye

Software Engineer – Tridium

TRIDIUM

Overview

What is an Archive History Provider?

Why is it useful?

When can it be implemented?

How can I build my own?

Demo

TRIDIUM

What is an Archive History Provider?

- Introduced in Niagara 4.11, Archive History Providers allow local history data to be supplemented by archived history data from another (remote) source at query time.
- Supports on-demand read access to a large store of archived history data during traditional history use cases.
- Rdb Archive History Provider: 1st implementation of this new public Niagara API, enabling supervisor stations to query archived history data from relational databases on the fly.

Why is it useful?

- Allows reducing the capacity of local histories, shrinking the resource burden of the local station.

- Plugs in at a low level of Niagara, benefiting all applications and views that query Niagara histories (existing & future).

- Multiple providers can be installed, allowing histories to read archived data from different remote sources
  - Note: An *individual* history cannot be split across multiple sources, but *different* histories can be split across multiple sources

When can it be implemented?

- Eligible to develop when you have local Niagara histories that were exported or imported to/from another source.
- Local histories are still required at a reduced capacity, leaving older (less frequently used) data in the archive source.
  - Needed to enforce user authorization, supply metadata (units), and improve performance for recent time queries.
  - Archived history data is only tapped when a history query's time range exceeds what is available in local data.
- The archive source should have fast query response times

How can I build my own?

- Requires the **`Tridium.historyArchive`** license feature
- Subclass: **`javax.baja.history.db.BArchiveHistoryProvider`**
- Only two abstract methods need to be implemented!

```
public abstract boolean isLikelyToContainArchivedHistory(BHistoryConfig config,
                                                         Context cx);

protected abstract Optional<Cursor<BHistoryRecord>> doTimeQuery(BHistoryConfig config,
                                                                BAbsTime startTime,
                                                                BAbsTime endTime,
                                                                boolean descending,
                                                                int limit,
                                                                Context cx);
```

`isLikelyToContainArchivedHistory(BHistoryConfig config, Context cx)`

- A quick, unreliable existence check to indicate whether the archive source is likely to contain data for the given history.
- The result gives Niagara an indication of which provider is most likely able to process a pending history query.
  - Even if false is returned, Niagara may query providers until one that actually contains archived data for the requested history is found.
- **Implementation Tip**: Find a matching history import/export descriptor in the station to decide the most likely source.

```
doTimeQuery(BHistoryConfig config,
            BAbsTime startTime, BAbsTime endTime,
            boolean descending, int limit, Context cx)
```

- Returns `Optional<Cursor<BHistoryRecord>>` for efficiency.
  - Return `Optional.empty()` if history data doesn't exist in the archive.
  - Return a wrapped `Cursor` with no records if the history exists in the archive, but no matching records are found for the time range.
  - Otherwise return a wrapped `Cursor` with history records streamed from the archive that fall within the given time range (inclusive), sorted in the requested timestamp order, and capped at the specified record limit (favoring most recent records).

- Not so bad, right? There's one more important part…

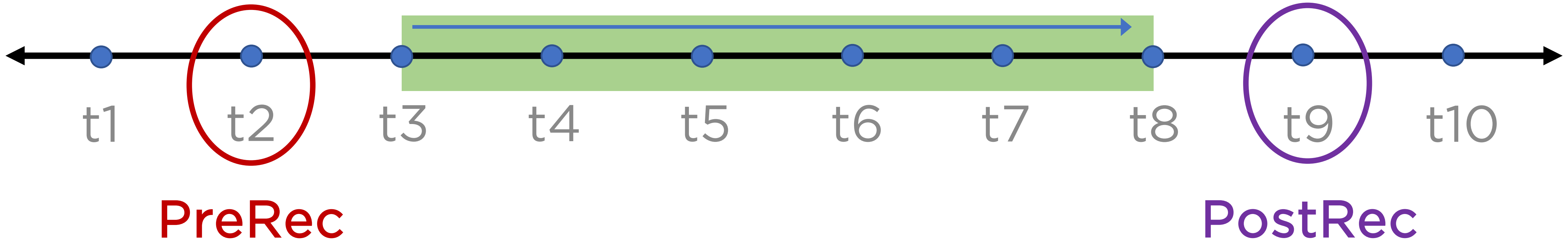
The Returned Cursor’s Context Needs Facets

- More is needed to support history queries from Niagara (applies when a Cursor is returned from the provider, even for no matching records). These Cursor Facets are needed:

Context Facet Key	Context Facet Value
historyRecordTypeSpec	The history record type in Type Spec String form.
historyCursorPreRec	The history record just prior to the time range, subject to the asc/desc order and limit. Excluded only if no prior record exists in the archive.
historyCursorPostRec	The history record just after the time range, subject to the asc/desc order and limit. Excluded only if no trailing record exists in the archive.
archiveHistoryLimitExceeded	Indicates that more archived records match the requested time range than fit into the requested limit. Excluded only if the limit was not exceeded.

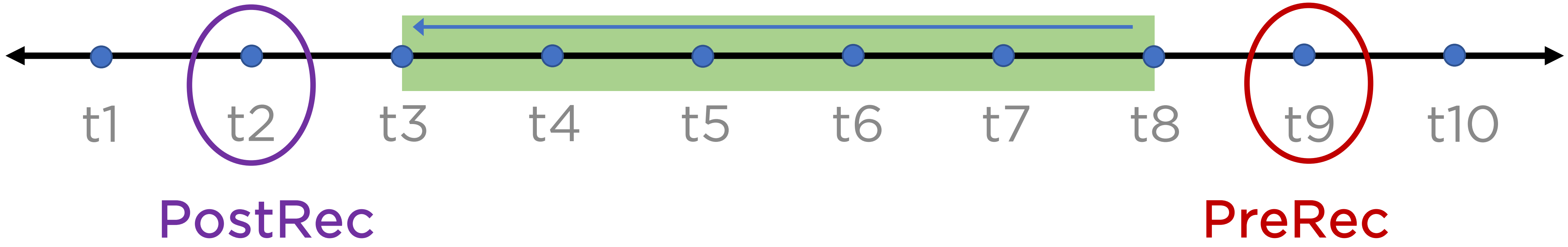
Pre/Post History Record Computation

Start: t3
End: t8
Descending: False
Limit: 100



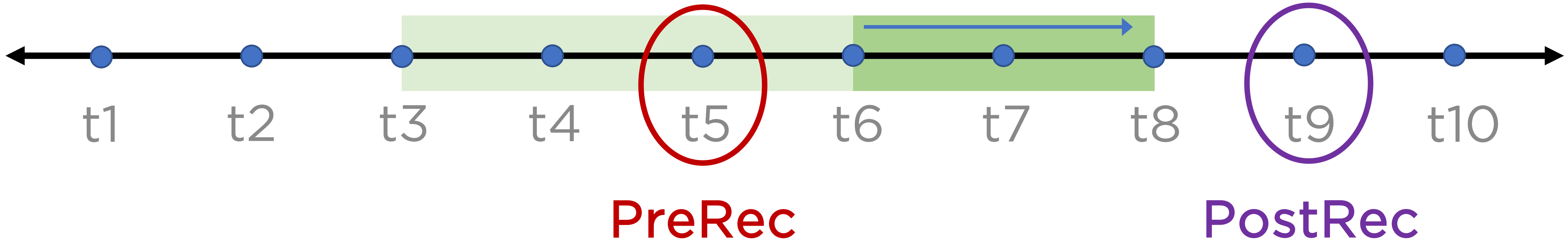
Pre/Post History Record Computation

Start: t3
End: t8
Descending: **True**
Limit: 100



Pre/Post History Record Computation

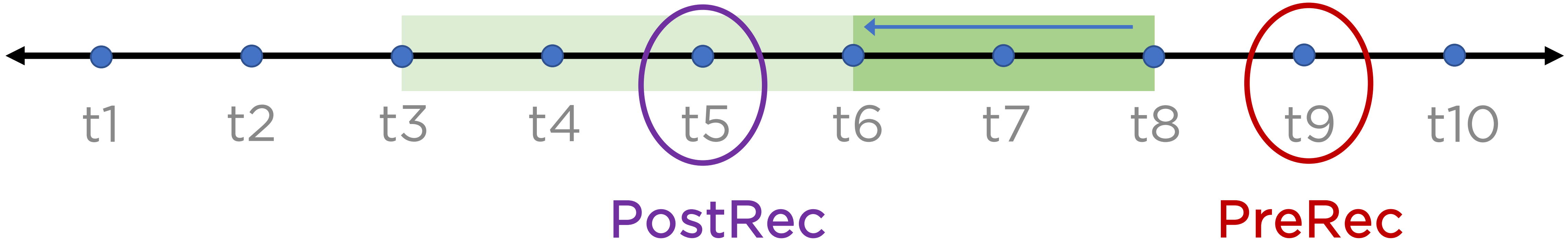
Start: t3
End: t8
Descending: False
Limit: **3**



archiveHistoryLimitExceeded = true

Pre/Post History Record Computation

Start: t3
End: t8
Descending: True
Limit: 3



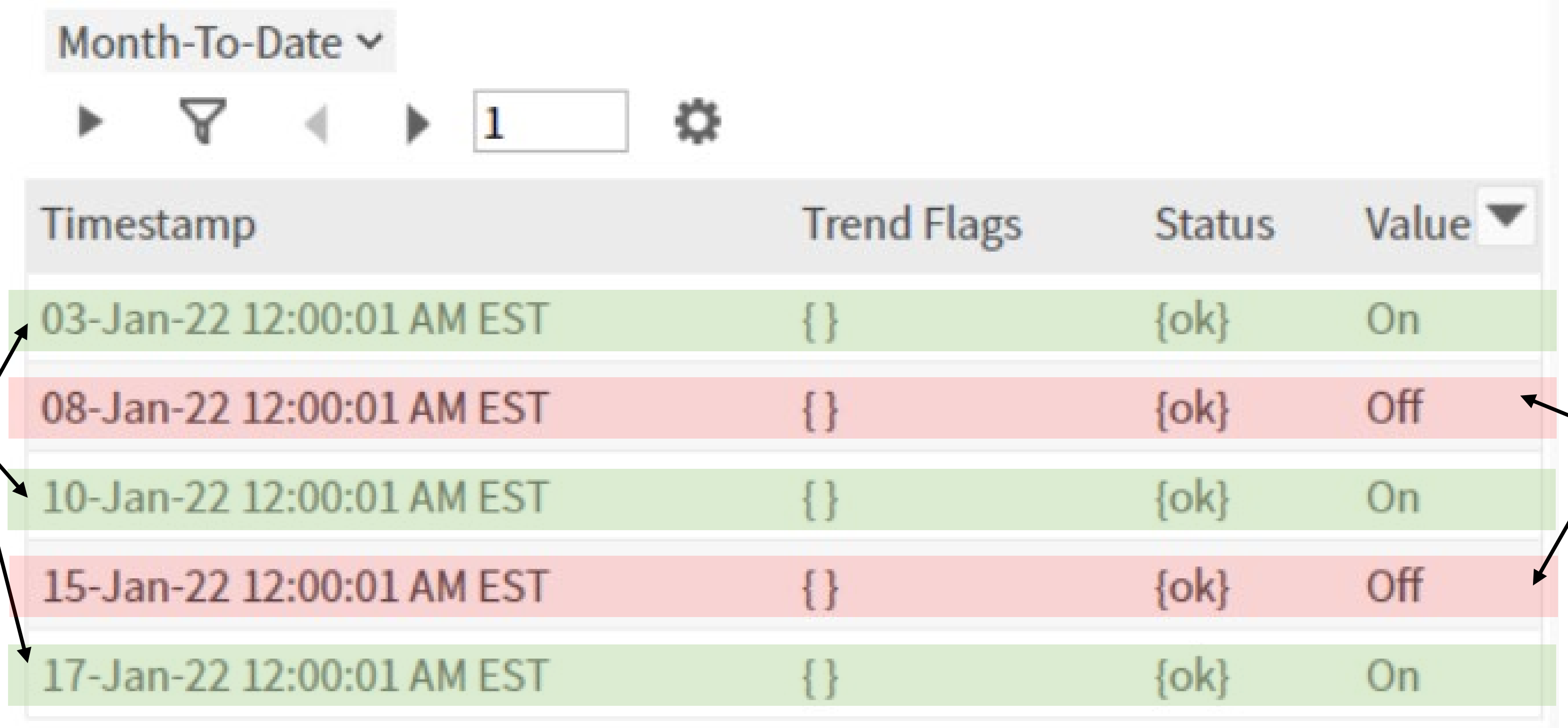
archiveHistoryLimitExceeded = true

Why are the Cursor Context Facets so important?

- There are many reasons, most notably:
  - Allow Niagara to warn users if the archive limit was reached and therefore history records are missing from the displayed results.
  - Due to Niagara's flexible history chart features, these Facets are needed for charting arbitrary time queries against arbitrary history types that have arbitrary collection intervals.
    - Huh? What does that mean?

Example: COV History for Equipment running only Mon - Fri

Turns on at the start of Mondays

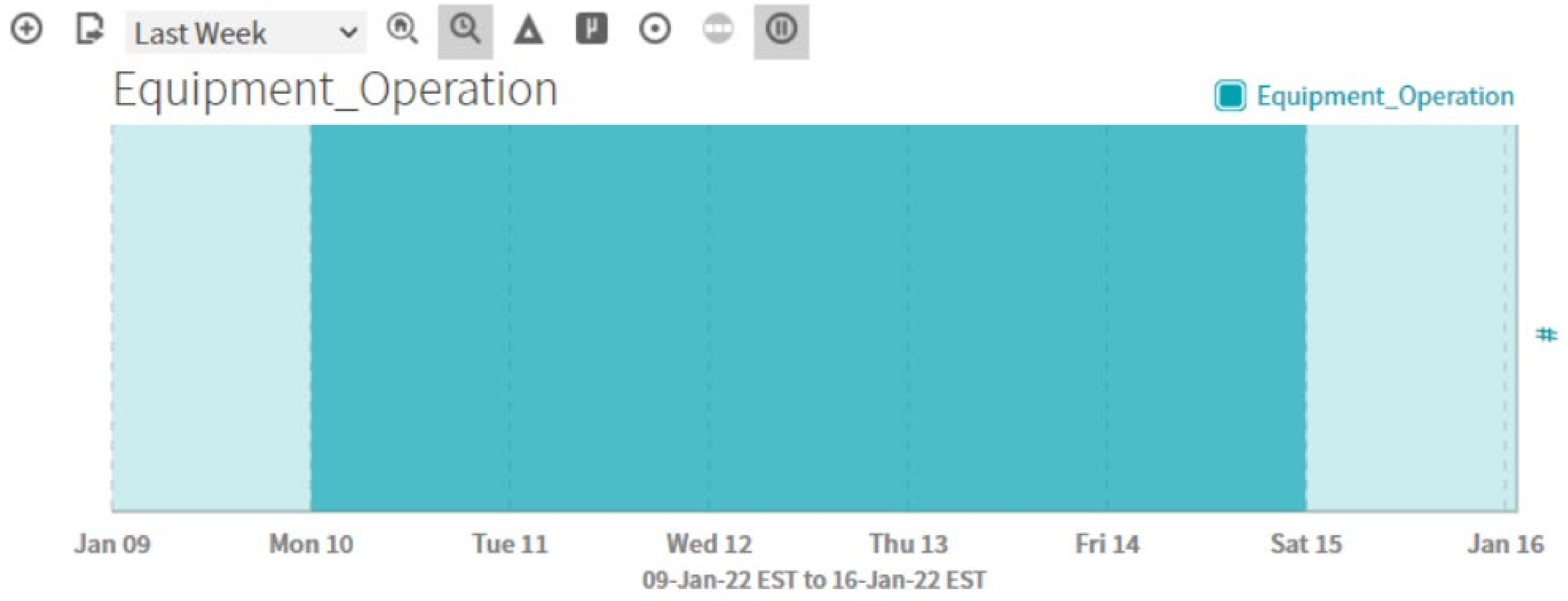


Timestamp	Trend Flags	Status	Value
03-Jan-22 12:00:01 AM EST	{}	{ok}	On
08-Jan-22 12:00:01 AM EST	{}	{ok}	Off
10-Jan-22 12:00:01 AM EST	{}	{ok}	On
15-Jan-22 12:00:01 AM EST	{}	{ok}	Off
17-Jan-22 12:00:01 AM EST	{}	{ok}	On

Turns off at the end of Fridays

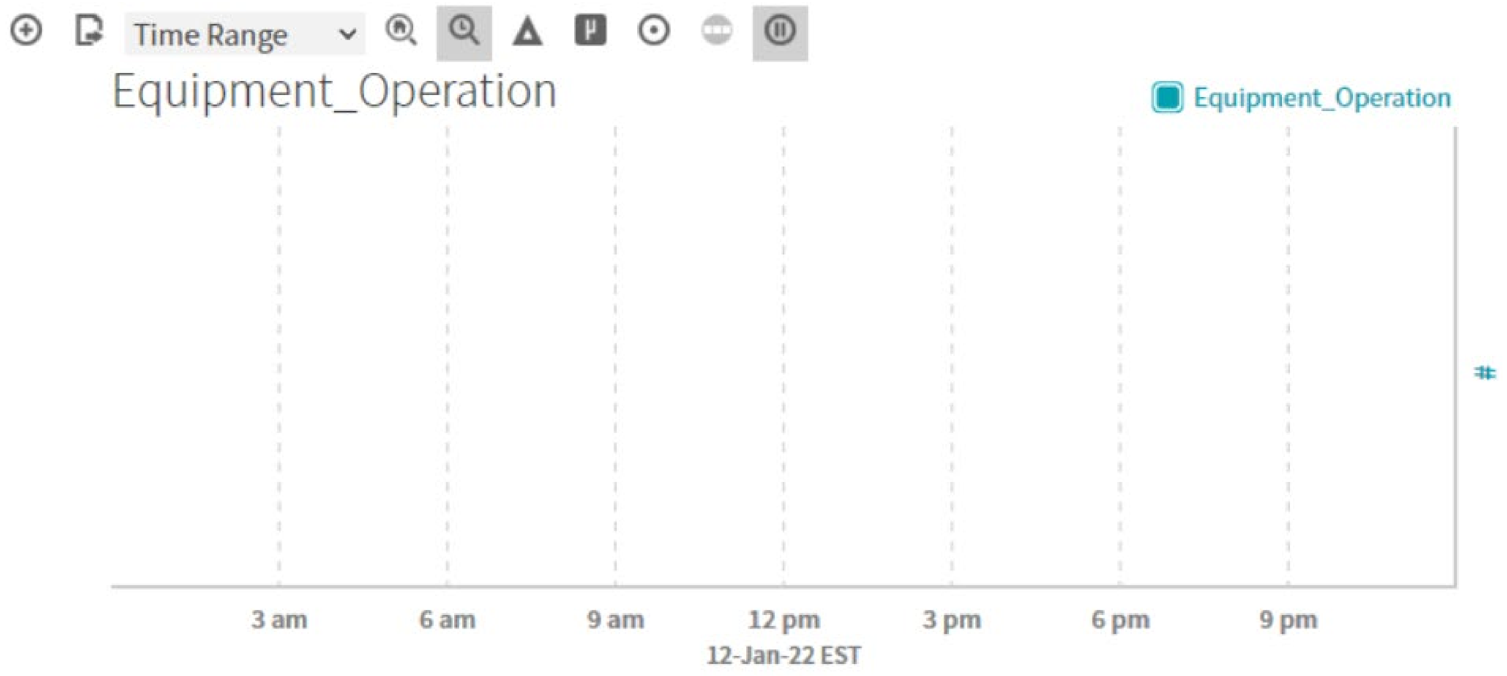
niagara SUMMIT

Example: COV History for Equipment running only Mon - Fri



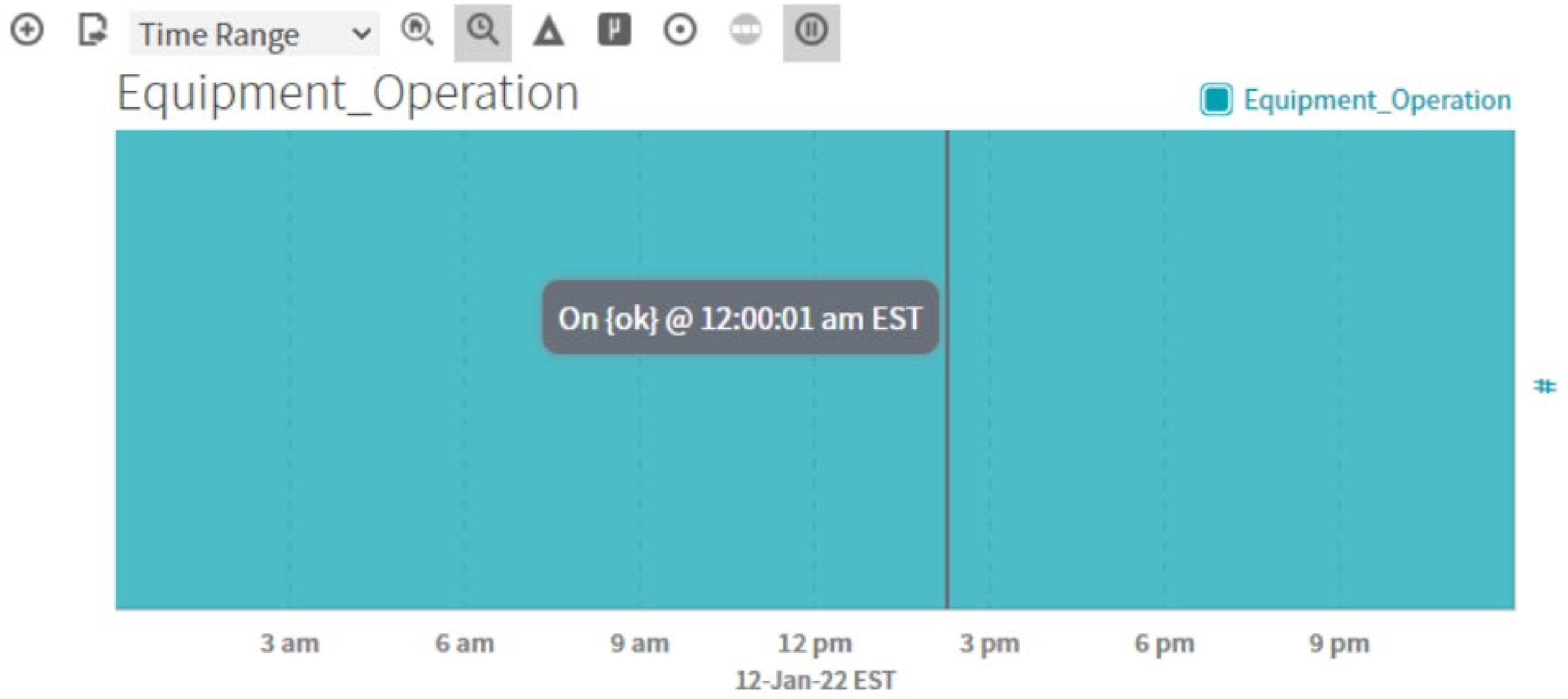
Example: COV History for Equipment running only Mon - Fri

If the query time range is changed to Wednesday only (no history records recorded in that window of time), without the Cursor Context Facets, the chart would incorrectly appear as:



Example: COV History for Equipment running only Mon - Fri

But with the proper Cursor Context Facets, the chart will correctly appear as:



On {ok} @ 12:00:01 am EST

Adding the Cursor Context Facets

- The additional information must be supplied as `BFacets` on the result of the Cursor's `getContext()` method.
  - Must be ready as soon as `doTimeQuery()` returns the Cursor.
- **Implementation Tip**: The following convenience methods on **`javax.baja.history.HistoryCursor`** can be used:

```
public static BFacets makeBoundaryRecordFacets(BHistoryRecord preRec,
                                               BHistoryRecord postRec)

public static Context makeArchiveLimitExceededContext(Context cx)
```

Now you're ready!

Let's look at a demo

https://github.com/tridium/summit2022-file-archive-history-provider

TRIDIUM

Summary

Archive History Providers allow on-demand access to archived history data, reducing the resource burden on the local Niagara station

Want more information?

Recorded Tridium Talk:

https://www.tridium.com/us/en/services-support/events/2021/10/2021-10-21-archive-history-provider

Visit Bajadoc for

`javax.baja.history.db.BArchiveHistoryProvider`

TRIDIUM

Questions?

TRIDIUM