Prometheus Driver for Tridium Niagara

User Guide

baudrate.io

niagara4

June 4, 2026

Contents

1 Introduction **2**

2 Requirements **2**

3 Quick Start **2**

4 Network **3**

5 Authentication **4**

6 Selecting Points to Expose **7**

7 Point Value Freshness **8**

8 Metrics Format **9**

9 Prometheus Scrape Configuration **10**

10 Verification **11**

11 Importing Data From Prometheus Into Niagara **12**

12 FAQ **14**

1 Introduction

Prometheus[1] is the de-facto standard open-source time-series database for monitoring and alerting, widely used together with Grafana for dashboards. It collects metrics by **scraping** HTTP endpoints exposed by the systems it monitors.

The Prometheus driver for Tridium Niagara exposes the station's live point data on a Prometheus-compatible HTTP endpoint, so any Prometheus-compatible database (Prometheus, VictoriaMetrics, Grafana Mimir, Thanos) can scrape the station and store the data for long-term analysis, dashboards, and alerting.

In addition to the scrape endpoint, the driver can also pull historical metrics from a Prometheus database into Niagara histories.

2 Requirements

- Niagara-powered device with software v4.8 or later, such as JACE, Supervisor, or their OEM versions [2]
- Prometheus driver license
- A Niagara user account configured with **HTTP Basic** authentication scheme – this is the account Prometheus will use to scrape the station
- A Prometheus, VictoriaMetrics, Grafana Mimir, or Thanos instance with network access to the Niagara station

3 Quick Start

1. Copy `prometheus-rt.jar` into the Niagara `/modules` folder and restart the station.
    a. If the driver is hosted on a PC: restart Supervisor station.
    b. If the driver is hosted on a JACE: import the module with all its dependencies into JACE and restart it.
2. Add **Prometheus Network** to the station's **Drivers** folder.
3. Enter the license code in the **License** property of the network.
4. Add the **HTTPBasicScheme** to **Services > AuthenticationService > Authentication Schemes** by dragging it from the **baja** palette (it is not installed by default).

[1] All trademarks or registered trademarks are the property of their respective owners
[2] If support for older Niagara versions is required, please contact the vendor

5. Create a **dedicated** Niagara user for Prometheus – e.g. `prometheus` – with the **HTTPBasicScheme** authentication scheme. Do **not** change the scheme on an existing user account: the HTTP Basic scheme is incompatible with normal Workbench/web UI login, so any user switched to it will lose interactive access. See Authentication for the full procedure.
6. Expand the network in the Nav tree, double-click **Export Points** to open the **Prometheus Point Manager**, click **Discover**, scope the BQL query if desired, select the points you want to expose and click **Add**. Added points are exposed immediately and kept live. See Selecting Points to Expose.
7. Note the **Metrics Url** property on the network – this is the path that Prometheus will scrape.
8. Configure your Prometheus instance to scrape this endpoint using the dedicated user's credentials.

4 Network

The **Prometheus Network** is the top-level component that controls the metrics endpoint. Add it under **Drivers** in your station.

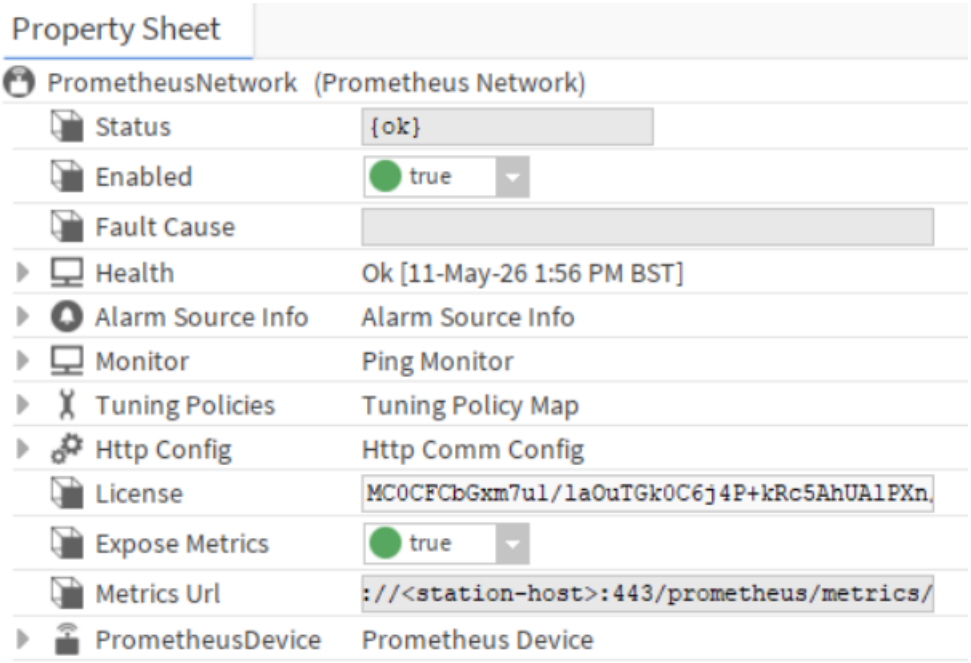
Network properties:

- **License** – the code that licenses the driver on this host.
- **Metrics Url** – read-only display of the URL Prometheus should scrape. Populated when the station starts. The hostname is shown as `<station-host>` because the station cannot reliably know its own externally-reachable address; substitute the actual hostname or IP that Prometheus will use to reach the station.

The metrics endpoint is **always enabled** – there is no on/off toggle. It serves whatever points you have added to the network's **Export Points** table (see Selecting Points to Expose); if the table is empty, only `niagara_build_info` is returned. The endpoint returns HTTP 404 only when no Prometheus Network exists, or the network is in **fault** (typically an invalid or expired license).

Frozen under the network is the **Export Points** table – this is where you discover and add the points to expose, and it holds one child per exposed point. It also carries the **Expose Status** property (see The niagara_point_up status series).

The metrics endpoint is hosted by Niagara's main web server (the same one that serves Workbench Web UI). The scheme (HTTP vs HTTPS) and port follow the station's **Web Service** configuration – there is no separate Prometheus port. If your station is HTTPS-only, Prometheus must scrape via HTTPS.



5 Authentication

The metrics endpoint is protected by HTTP Basic authentication using Niagara's standard user store. Prometheus must present valid credentials of a Niagara user that:

1. Exists in the station's **User Service**.
2. Has the **HTTP Basic** authentication scheme assigned (not the default cookie-based scheme).
3. Is enabled.

When Prometheus scrapes the endpoint:

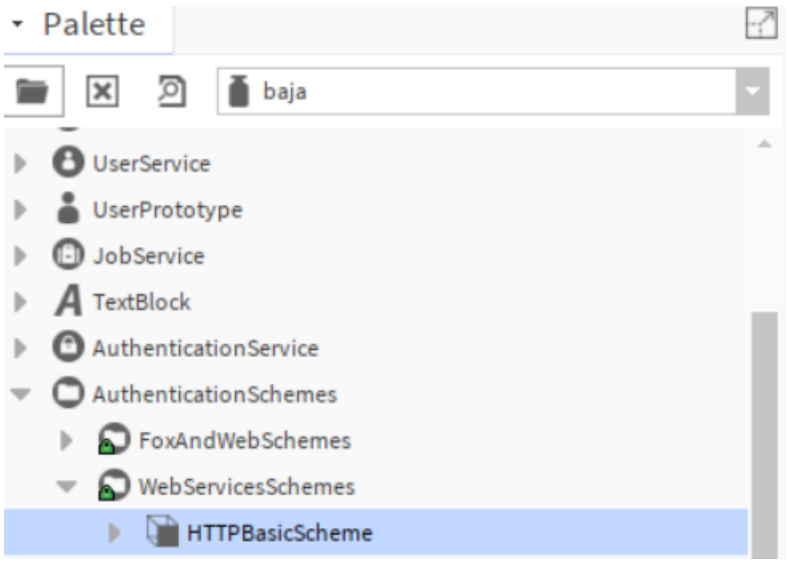
- If credentials are missing or invalid – the request is rejected with HTTP 302 (redirect to login) or HTTP 401, depending on the configured scheme. Prometheus marks the target as **down**.
- If credentials are valid – the request is allowed, and the driver applies per-user permission filtering before returning data (see next section).

Important: create a **new, dedicated user** for Prometheus – do not reuse your operator or admin account. The **HTTPBasicScheme** is not compatible with Niagara's normal Workbench, web UI, or Fox login flows, so any user switched to this scheme will be unable to log in interactively. Keep your regular users on their default authentication scheme and add a separate user just for the scrape endpoint.

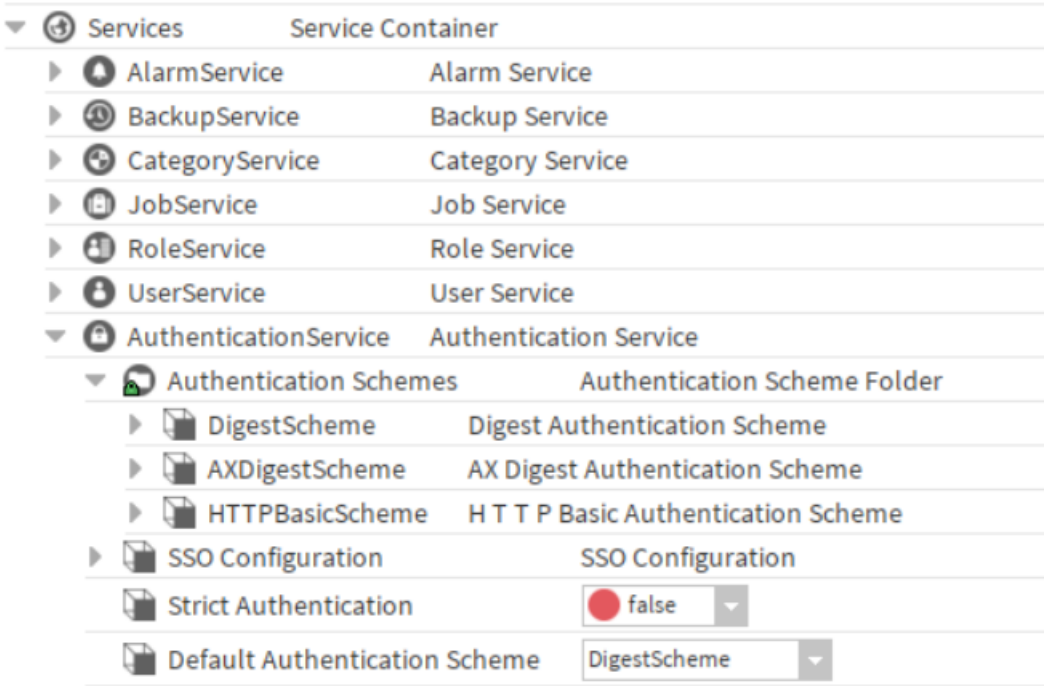
The **HTTPBasicScheme** is not present in a station by default – it must be added to the **AuthenticationService** from the **baja** palette before it can be assigned to any user.

To add the HTTP Basic scheme to the station:

1. In Workbench, open the **baja** palette (**Window > Side Bars > Palette**, then open the `baja` module).
2. Expand **AuthenticationSchemes** in the palette.

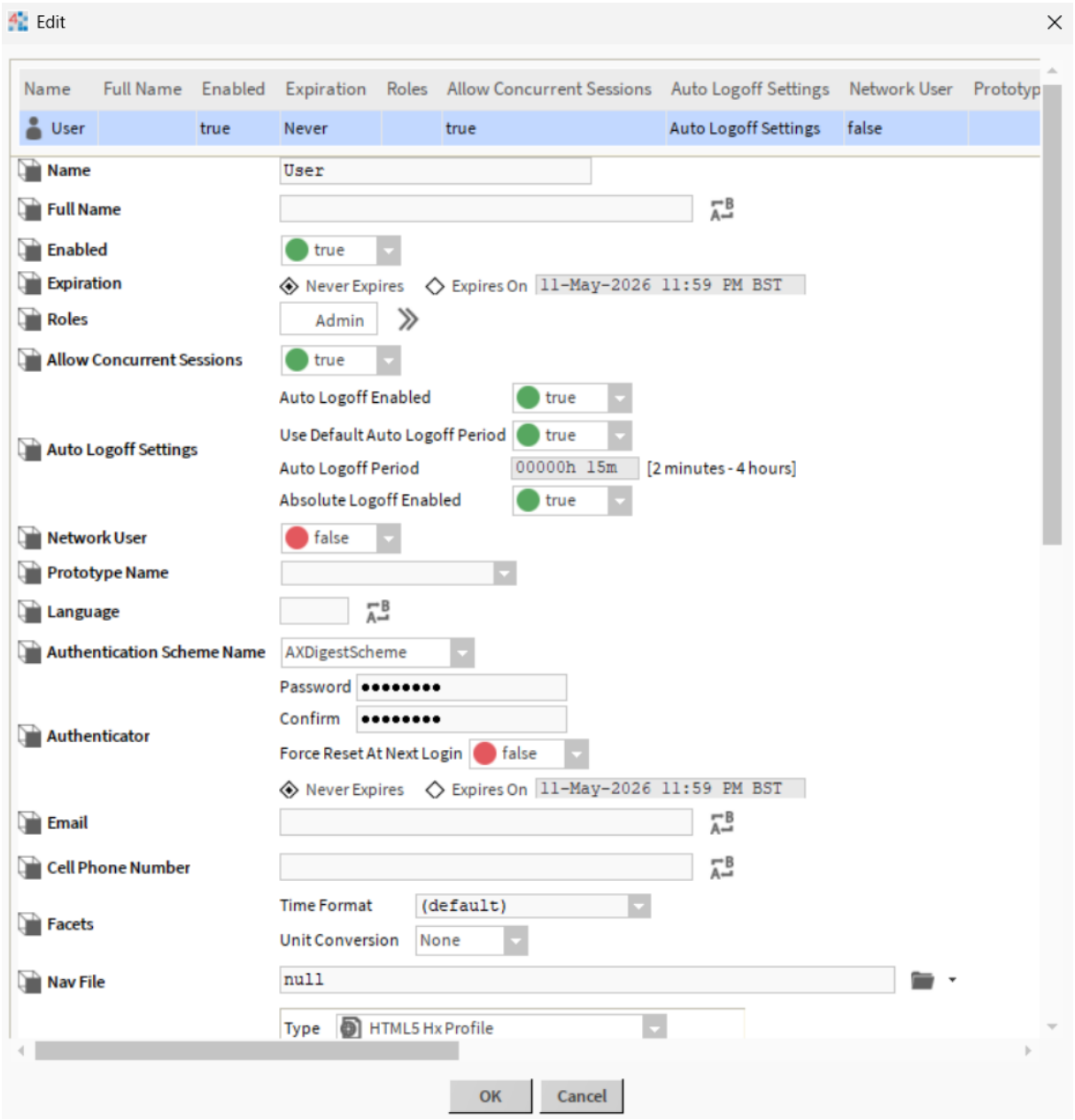


3. Drag **HTTPBasicScheme** from the palette and drop it onto **Station > Config > Services > AuthenticationService > Authentication Schemes** in the Nav tree. Accept the default name when prompted.



To create the dedicated scrape user:

1. In Workbench, open **Services > UserService**.
2. Click **New** and create a user, e.g. `prometheus`.
3. In the user's **Authentication Scheme Name** property, select **HTTPBasicScheme** (the scheme you added in the previous step).
4. Assign the **Categories** that determine which parts of the station the user can read. These categories drive the per-user filtering described in the next section.
5. Save.



6 Selecting Points to Expose

You choose exactly which station points are exposed – the driver does **not** dump the whole station. Exposed points live in the network's **Export Points** table, which you populate with the **Prometheus Point Manager**.

To add points:

1. In the Nav tree, expand the **Prometheus Network** and double-click **Export Points** to open the **Prometheus Point Manager** (a two-pane discover/database view, like the BACnet or other driver point managers).
2. Click **Discover**. A **BQL query builder** dialog opens, pre-filled with `select * from control:ControlPoint`. Accept it to find every control point in the station, or refine it (add a `where` clause, restrict to a subtree) to narrow the results.
3. The discovered points appear in the top pane, each with its **Path**, current **Status Value**, and point icon.
4. Select the points you want and click **Add**. A row is created in the bottom (database) pane for each – these are the points now exposed to Prometheus. Re-running Discover marks points that are already added.

Each exposed point (a row in the database pane) has:

- **Name** – the component name in the table.
- **Metric Name** – *editable*. This becomes the `path` label in the exposed metric. It defaults to the source point's decoded slot path; change it to give the series a cleaner label.
- **Status Value** – read-only, live display of the source point's current value and status.

Removing a point is just deleting its row. Added points are exposed and kept live immediately, with no further configuration – see Point Value Freshness.

The source point is referenced by a Niagara **handle ord**, so an exposed point keeps working if the source is renamed or moved. If the source point is *deleted*, its row shows the error in its (hidden) **Fault Cause** and the point reports as not-up.

6.1 Per-User Access Control

On top of the explicit export table, every scrape also applies the **Niagara categories** of the authenticated scrape user: a point is only included if that user has Operator Read permission on it. This lets different scrape users (e.g. one per Prometheus job) see different subsets of the same export table – a point you've added is served only to users allowed to read it.

7 Point Value Freshness

Niagara drivers only poll a proxy point while something is actively **subscribed** to it; a point that nothing is watching is not polled, and simply reading its value does not trigger a poll.

The driver therefore keeps **each point you add to the Export Points table continuously subscribed** for as long as it is in the table. That subscription is what makes the owning driver (BACnet, Modbus, the EcoStruxure driver, etc.) keep the point in its poll cycle, so the value is always current.

As a result:

- The **scrape itself does no polling and never blocks** – it reads the already-fresh value and returns immediately, well within Prometheus's `scrape_timeout`.
- Values are live whether or not anyone has the point open in Workbench.

The trade-off is that every exposed point is polled continuously while it is in the table, so the field-bus load scales with how many points you expose – select the points you need to monitor rather than exposing the whole station.

8 Metrics Format

The endpoint serves data in standard Prometheus exposition format (text/plain) at the path:

```
<scheme>://<station-host>:<port>/prometheus/metrics/
```

For every exposed point with a numeric, boolean, or enum value, the driver emits the value metric:

```
niagara_point_value{path="...",unit="..."} <number>
```

- `niagara_point_value` – the current value of the point. Numeric points emit their raw value, boolean points emit 0 or 1, enum points emit the ordinal.

In addition, one metadata metric is emitted per scrape:

```
niagara_build_info{station="...",version="...",host_id="..."} 1
```

Labels:

- `path` – the point's **Metric Name** (editable per point; defaults to the decoded slot path, with Niagara's `$XX` hex escapes turned back into spaces and special characters – e.g. `Analog Value` rather than `Analog$20Value`).
- `unit` – the engineering unit from the point's `units` facet, when set (e.g. `°C`, m^3/s). UTF-8 is passed through unchanged. The label is omitted entirely when the point has no unit.
- `station`, `version`, `host_id` – on `niagara_build_info` only; identify the station.

8.1 The niagara_point_up status series

By default the driver emits **only** `niagara_point_value`. If you also want a per-point communication-status series, set **Expose Status** to `true` on the **Export Points** table. The scrape then additionally emits, for each point:

```
niagara_point_up{path="..."} <0|1>
```

- `niagara_point_up` – 1 if the point's status is OK (live data), 0 if it is in fault/down/stale/null state.

When **Expose Status** is `false` (the default), no `niagara_point_up` series – nor its `# HELP`/`# TYPE` lines – are produced at all.

9 Prometheus Scrape Configuration

Add a scrape job to your `prometheus.yml`:

```yaml
scrape_configs:
  - job_name: niagara
    scheme: https
    metrics_path: /prometheus/metrics/
    tls_config:
      insecure_skip_verify: true
    basic_auth:
      username: prometheus
      password: <password>
    static_configs:
      - targets:
          - 192.168.1.100:443
```

Notes:

- The `metrics_path` value must include the trailing slash and the `/prometheus/` module prefix.
- `tls_config.insecure_skip_verify: true` is needed if your station uses Niagara's default self-signed certificate. For production deployments install a proper certificate and remove this line.
- Replace `192.168.1.100:443` with the actual reachable host and port of your station.
- Replace `<password>` with the scrape user's password.

Reload Prometheus by running this command in a terminal on the Prometheus host (Command Prompt or PowerShell on Windows, Terminal on macOS/Linux):

```
curl -X POST http://prometheus-host:9090/-/reload
```

Then check **Status > Targets** in the Prometheus web UI. The `niagara` target should show **up**.

10 Verification

From a terminal on any machine that can reach the Niagara station (Command Prompt or PowerShell on Windows, Terminal on macOS/Linux):

```
curl -k -u prometheus:<password> https://<station-host>/prometheus/metrics/
```

Expected response:

- HTTP 200
- `Content-Type: text/plain; charset=UTF-8`
- Lines of the form `niagara_point_value{path="..."} <number>`

In the Prometheus web UI:

- `up{job="niagara"}` should equal `1`
- `count(niagara_point_value)` should equal the number of points you added to the **Export Points** table (that the scrape user is allowed to read)
- `niagara_build_info` should return one series with the station's host ID and Niagara version

10.1 Troubleshooting: returns HTTP 302

Niagara redirected to login. Either no credentials were sent, the user does not exist, or the user is not configured with the HTTP Basic authentication scheme. Verify the user's **Authentication Scheme Name** is **HTTPBasicScheme**, and that the scheme itself has been added to **Services > AuthenticationService > Authentication Schemes** from the **baja** palette.

10.2 Troubleshooting: returns HTTP 401

The user exists with HTTP Basic auth but the password is wrong. Check the password in Prometheus's `basic_auth` block.

10.3 Troubleshooting: returns HTTP 404

Either:

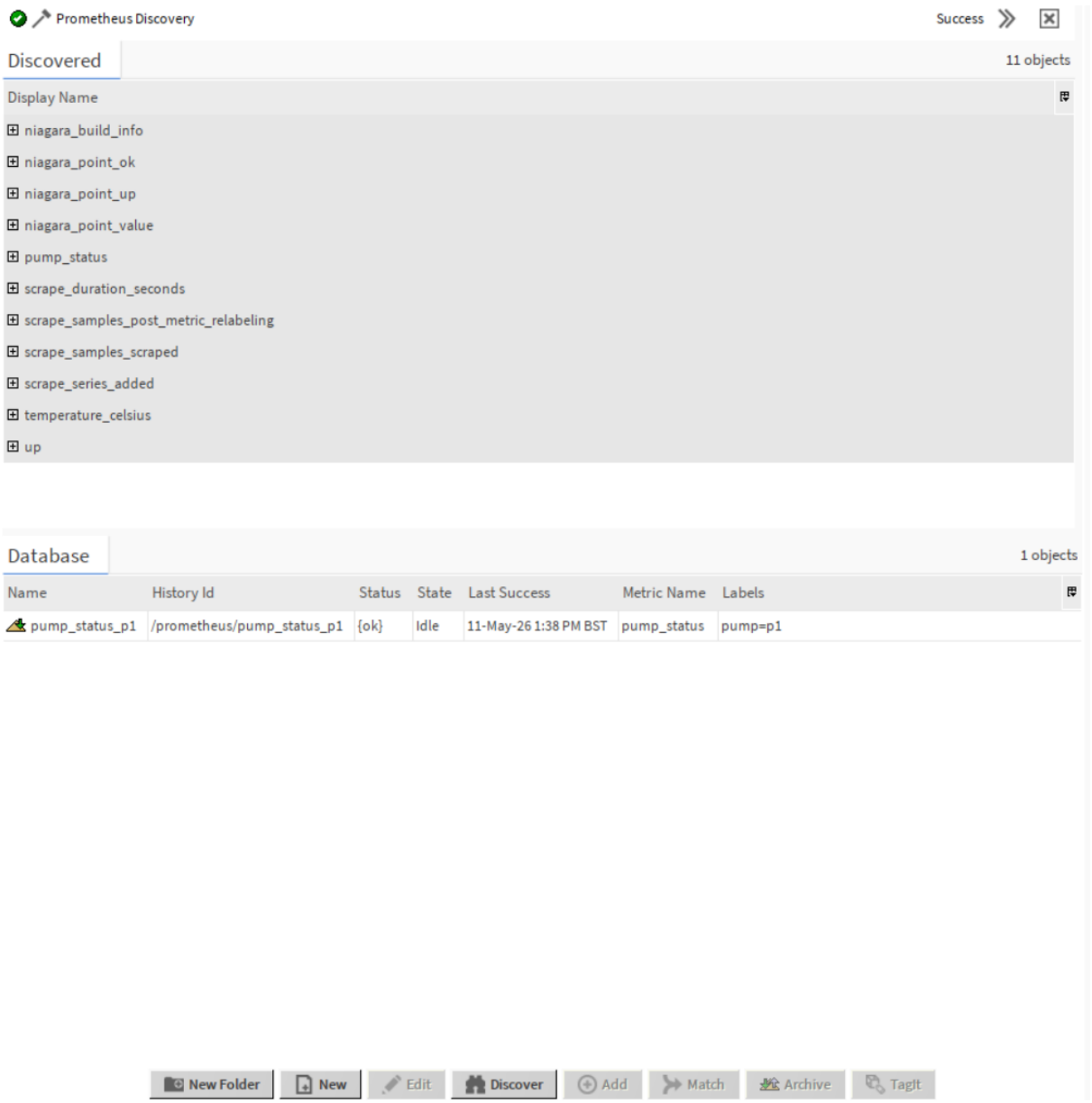
- The Prometheus Network is in **fault** status, typically due to an invalid or expired license. Check the network's **Status** and **Fault Cause**. If the cause is licensing, enter a valid license code.
- No Prometheus Network exists in the station at all. Add one under **Drivers**.

(An empty result – only `niagara_build_info` – is HTTP 200, not 404; it just means no points have been added to the **Export Points** table yet.)

11 Importing Data From Prometheus Into Niagara

In addition to exposing data to Prometheus, the driver can also import metrics from a Prometheus database into Niagara histories.

1. Add a **Prometheus Device** under the network.
2. Configure the device with the Prometheus database's address, port, credentials, and scheme.
3. Right-click the device's **Histories** extension and open the **Prometheus Import Manager**.
4. Click **Discover** – the driver queries `/api/v1/label/__name__/values` to list available metrics, then `/api/v1/series` to list label combinations per metric.
5. Pick the specific metric series you want to import and **Add** them.
6. Each import is a `BPrometheusHistoryImport` component with a `metricName`, optional `labels`, and a target `historyId`. Action **Execute** to pull data; subsequent executions incrementally fetch new samples using the stored `lastTimestamp`.



If a metric matches multiple label-combination series and no `labels` filter is set, the import fails explicitly with a “specify labels to disambiguate” message rather than silently picking an arbitrary series.

12 FAQ

12.1 Why does the metrics URL show `<station-host>` instead of an actual hostname?

A Niagara station cannot reliably determine its own externally-reachable address – it may have multiple network interfaces, sit behind NAT, or be accessed via different DNS names by different consumers. The displayed URL shows the scheme, port, and path that the station knows are correct, and uses `<station-host>` as a placeholder for the host. Substitute the actual hostname or IP that Prometheus will use to reach the station.

12.2 How do I switch between HTTP and HTTPS?

The metrics endpoint shares Niagara's main web server, so the scheme is governed by the station's **Web Service** settings under **Services > WebService**:

- `httpsOnly = true` – HTTPS only; Prometheus must scrape via HTTPS.
- `httpsEnabled = true, httpsOnly = false` – both work; Prometheus can scrape via either.
- `httpsEnabled = false` – HTTP only.

There is no per-driver toggle.

12.3 I don't see `niagara_point_up` at all. Why?

The status series is opt-in. Set **Expose Status** to `true` on the network's **Export Points** table. By default only `niagara_point_value` is emitted. See The niagara_point_up status series.

12.4 A point shows `niagara_point_up = 0`. What does that mean?

(Only relevant when **Expose Status** is enabled.) The point's Niagara status is not OK – it may be in fault, down, stale, or null. This typically means the underlying device or driver communication is broken. The `niagara_point_value` for that path may still be present (showing the last known value) but should not be trusted while `up = 0`. Use the PromQL pattern `niagara_point_value unless on(path) (niagara_point_up == 0)` if you want to ignore those values in alerts.

12.5 Are the values I scrape live, or could they be stale?

They are live. Each point you add to the **Export Points** table is kept continuously subscribed, so its owning driver polls it on its normal cycle and the scrape always reads a current value. See Point Value Freshness.

12.6 Could exposing many points slow the station down?

Each exposed point is polled continuously while it is in the table, so the field-bus / engine load scales with how many points you expose. This is why you select points explicitly (via Discover) rather than exposing the whole station. Expose the points you actually need to monitor; the scrape itself is cheap regardless.

12.7 Why does my Prometheus query show fewer points than I expected?

Two possible reasons: (1) the point was never **Added** to the Export Points table – only added points are exposed; or (2) the scrape user does not have Operator-Read permission on the point's category. Add the missing points via Discover, and/or grant the scrape user the right categories.

12.8 Can I customize the metric names?

Not in this version. All numeric/boolean/enum points share a single metric name `niagara_point_value` (and corresponding `niagara_point_up`), differentiated by the `path` label. This matches Prometheus best practice ("one metric per concept, distinguish by labels") and avoids the need to derive valid Prometheus identifiers from arbitrary Niagara slot paths.

12.9 What about points that have engineering units?

When a point has a `units` facet set in Niagara, the driver attaches it as a `unit="..."` label on `niagara_point_value`. UTF-8 characters (°, ³, etc.) are passed through unchanged. Points without units do not get a `unit` label.