Prometheus Driver for Tridium Niagara

User Guide

baudrate.io

niagara4

May 11, 2026

Contents

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. Tick the **Expose Metrics** property to enable the scrape endpoint. See Metrics Format for the exact shape of the data that will be served, and Per-User Access Control for how visibility is filtered.

[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. Note the **Metrics Url** property – this is the path that Prometheus will scrape.
7. Configure your Prometheus instance to scrape this endpoint using the dedicated user's credentials.

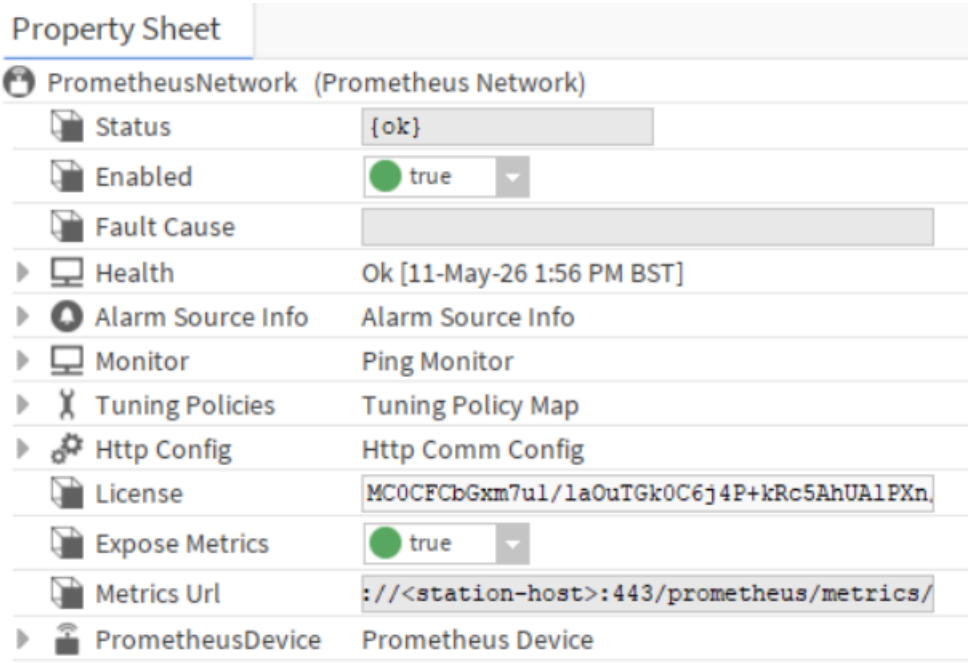
4 Network

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.
- **Expose Metrics** – master kill-switch for the metrics endpoint. When unticked the endpoint returns HTTP 404 to all scrape requests. Default: **false** (off). See Metrics Format for what is served when ticked, and Per-User Access Control for how visibility is filtered.
- **Metrics Url** – read-only display of the URL Prometheus should scrape. Populated when the station starts and refreshed whenever **Expose Metrics** is toggled. 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 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.

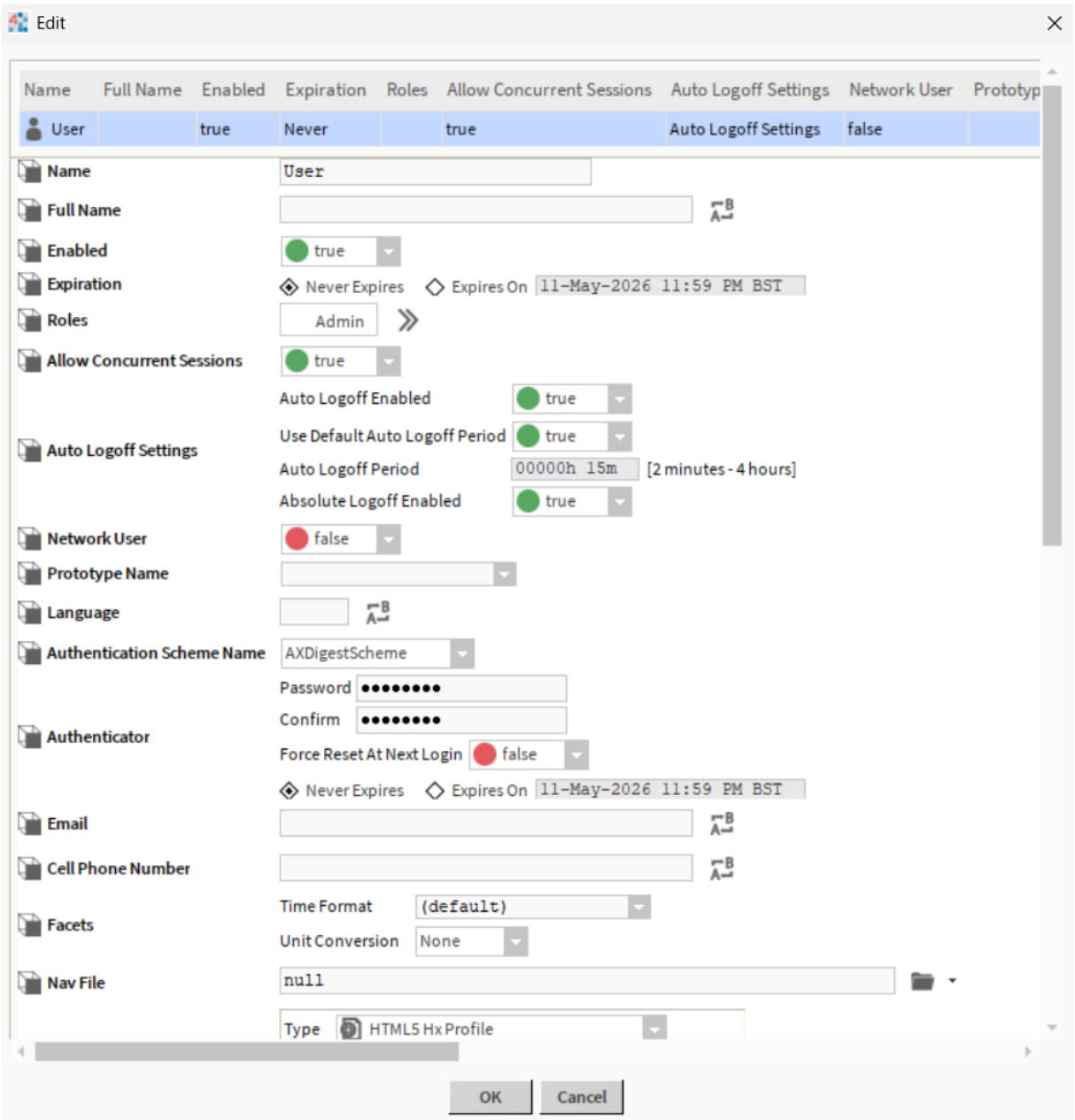
When Prometheus scrapes the endpoint:

- 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.

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**.
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 Per-User Access Control

The driver does **not** maintain its own allowlist or denylist. Instead, every scrape filters the station tree by the **Niagara categories** assigned to the authenticated user. Any component the user has Operator Read permission on is included; everything else is silently omitted.

This means:

- Different scrape users can see different subsets of the station data.
- To expose a specific subset of points to Prometheus, assign those points to a category that the scrape user can read and exclude the rest.
- You can have multiple scrape users (e.g. one per Prometheus job or per consumer) each seeing a different slice.

If a parent component is hidden from the user by category permissions, all components below it in the tree are hidden too – the walk does not descend into unreadable subtrees.

7 Metrics Format

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

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

For every readable point in the station with a numeric, boolean, or enum value, the driver emits two metrics:

```
niagara_point_value{path="...",unit="..."} <number>
niagara_point_up{path="..."} <0|1>
```

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

In addition, one metadata metric is emitted per scrape:

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

Labels:

- `path` – the slot path of the point, with Niagara's `$XX` hex escapes decoded back to 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`, `m³/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 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**.

9 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 the scrape user can read
- `niagara_build_info` should return one series with the station's host ID and Niagara version

9.1 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**.

9.2 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.

9.3 Troubleshooting: returns HTTP 404

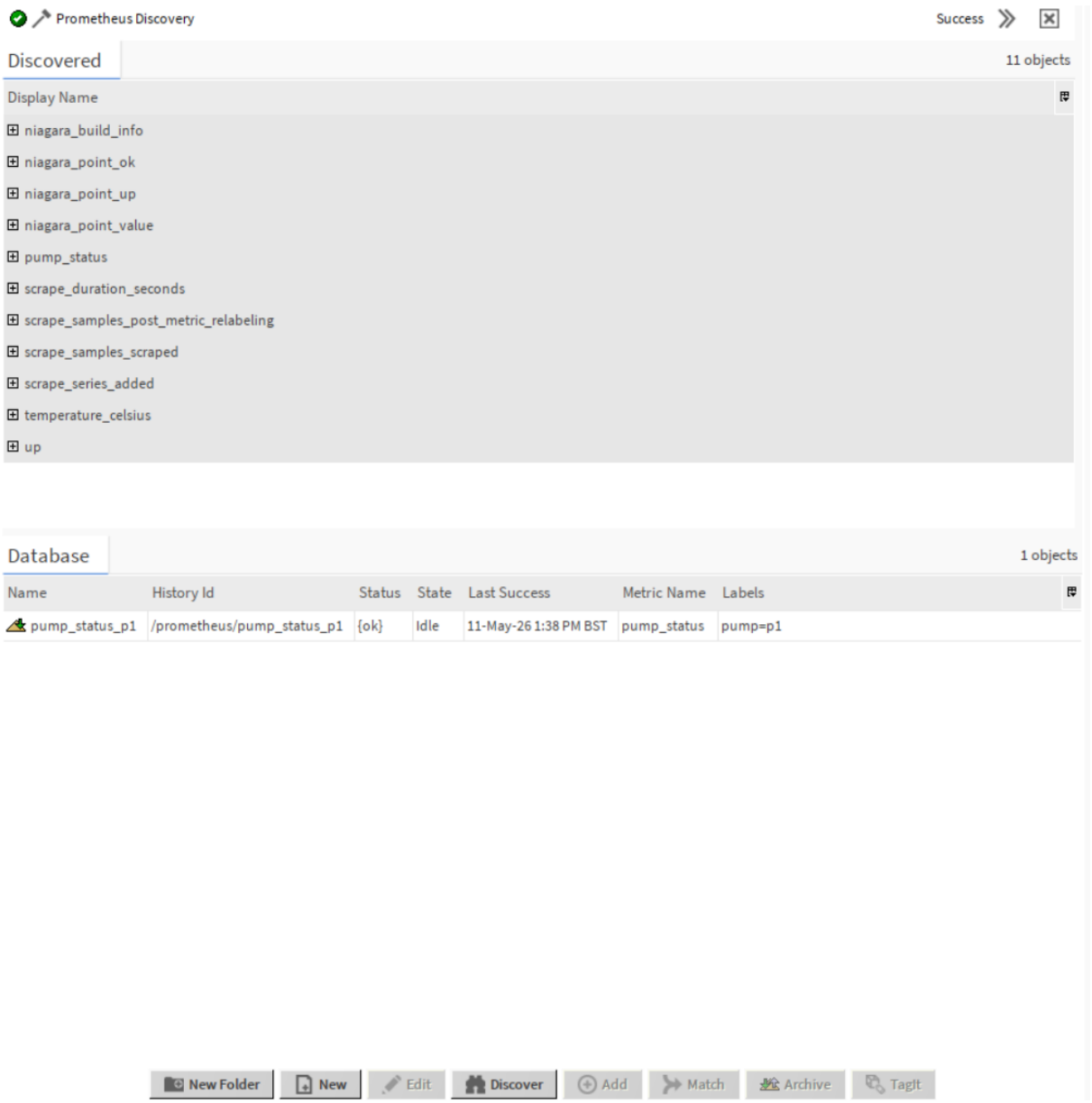
Either:

- The **Expose Metrics** property on the Prometheus Network is unticked. Tick it and save.
- 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**.

10 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.

11 FAQ

11.1 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.

11.2 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.

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

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 stale values in alerts.

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

Most likely the scrape user does not have read permission on the missing points' categories. The driver filters by Niagara's category-based permission model on every scrape. To expose additional points, either grant the scrape user broader categories or assign the points to a category the user already has.

11.5 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.

11.6 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.