

# Honeywell

# WEBs-N4 Analytics Framework

OPERATING GUIDE

## Disclaimer

The material in this document is for information purposes only. The content and the product described are subject to change without notice. Honeywell makes no representations or warranties with respect to this document. In no event shall Honeywell be liable for technical or editorial omissions or mistakes in this document, nor shall it be liable for any damages, direct or incidental, arising out of or related to the use of this document. No part of this document may be reproduced in any form or by any means without prior written permission from Honeywell.

Copyright © 2019 HONEYWELL International, Inc. All rights reserved.

## Trademark notice

BACnet and ASHRAE are registered trademarks of American Society of Heating, Refrigerating and Air-Conditioning Engineers. Microsoft, Excel, Internet Explorer, Windows, Windows Vista, Windows Server, and SQL Server are registered trademarks of Microsoft Corporation. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Mozilla and Firefox are trademarks of the Mozilla Foundation. Echelon, LON, LonMark, LonTalk, and LonWorks are registered trademarks of Echelon Corporation.

Niagara Framework<sup>®</sup> is a registered trademark of Tridium Inc.

WEBS-AX, WEBS-N4 and WEBStation are registered trademarks of HONEYWELL International, Inc.

# TABLE OF CONTENTS

Disclaimer .....	2
Trademark notice .....	2
Document change log.....	7
Related documentation.....	7
How the webs-N4 Analytics Framework works.....	8
Features.....	8
Key differences with WEBS-AX.....	9
The framework in the Nav tree .....	9
Acquiring data .....	13
Analytics library.....	13
Best practices.....	14
Configuration overview .....	15
Prerequisites .....	16
Setting up a Supervisor computer .....	17
Installing on a remote host.....	18
Adding the framework to a station.....	18
About licensing .....	19
Configuring the service for licensing.....	19
Determining the number of points used .....	20
Confirming that the AnalyticService component is licensed .....	20
Setting up user authentication.....	20
Tags, hierarchies and relationships.....	22
Adding the tag dictionary .....	22
Setting up a hierarchy .....	23
Relationships.....	25
Associating definitions with tags .....	26
Changing the default behavior for a value tag .....	27

Tagging Niagara Network points with n:history ..... 27

Tagging BACnet Network points with n:history..... 28

Acquiring tags from remote stations ..... 28

Tag inheritance and the a:a tag..... 29

Removing all a:a tags ..... 30

Algorithms and alerts ..... 31

    Creating an algorithm ..... 31

    Defining the data source ..... 33

    Adding logic..... 34

    Filtering algorithm input data ..... 35

    Example: Monitoring temperature and humidity ..... 35

    Creating an alert..... 38

    Editing an algorithm ..... 40

    Using algorithm results in standard logic..... 41

    Algorithm for removing unwanted data ..... 41

    Demand Range Filter Algorithm ..... 42

    Wire sheet view..... 42

    Demand Replace Bad Data ..... 43

    Wire sheet view..... 43

    Viewing an alert in the alarm console..... 44

Data visualization ..... 46

    Rollup and aggregation ..... 46

    Bindings ..... 48

    AnalyticWebTable ..... 48

    Changing rendering limitations ..... 52

    Automatic conversion of metric values in tables..... 53

    Pre-defined charts ..... 54

    Configuring a pre-defined chart..... 55

Creating a new Px view..... 57

Creating a new Ux chart..... 59

Observing patterns using the Spectrum chart..... 61

Changing the aggregation function..... 62

Setting up an analytic table binding..... 63

Exporting a Px chart..... 64

Reports..... 66

Creating Ux reports..... 67

Managing Ux reports..... 67

Creating a dashboard..... 69

Configuring a report or a dashboard..... 70

Normalizing energy consumption values based on floor area..... 71

Normalizing energy consumption values based on degree-day temperature..... 73

Printing a report..... 75

Missing data strategy..... 76

    Linear interpolation..... 76

    K-nearest neighbor (KNN)..... 77

    Aggregation strategies..... 79

    Missing data configuration..... 80

    Missing data indication..... 80

Troubleshooting..... 83

    Point status..... 83

    Enabling error logging..... 84

    Scenarios..... 85

Glossary..... 88

## About this guide

This guide contains important information about how to install and configure the WEBS-N4 Analytics Framework running on a Supervisor or remote host station.

## Audience

The information in this guide is for Systems Integrators and Facility Managers who are responsible for configuring the tools used to manage complex building systems.

## Document Content

This guide provides procedures for configuring each aspect of the WEBS-N4 Analytics Framework, and concludes with a troubleshooting chapter for resolving common problems and answering questions. An index is provided to help you find the specific information you are looking for.

## Product Documentation

This document is part of the Niagara technical documentation library. Released versions of Niagara software include a complete collection of technical information that is provided in both online help and PDF formats.

## Document change log

The following list describes significant documentation changes.

June 7, 2019

Initial release

## Related documentation

Several documents provide additional information about WEBs-N4 Analytics Framework.

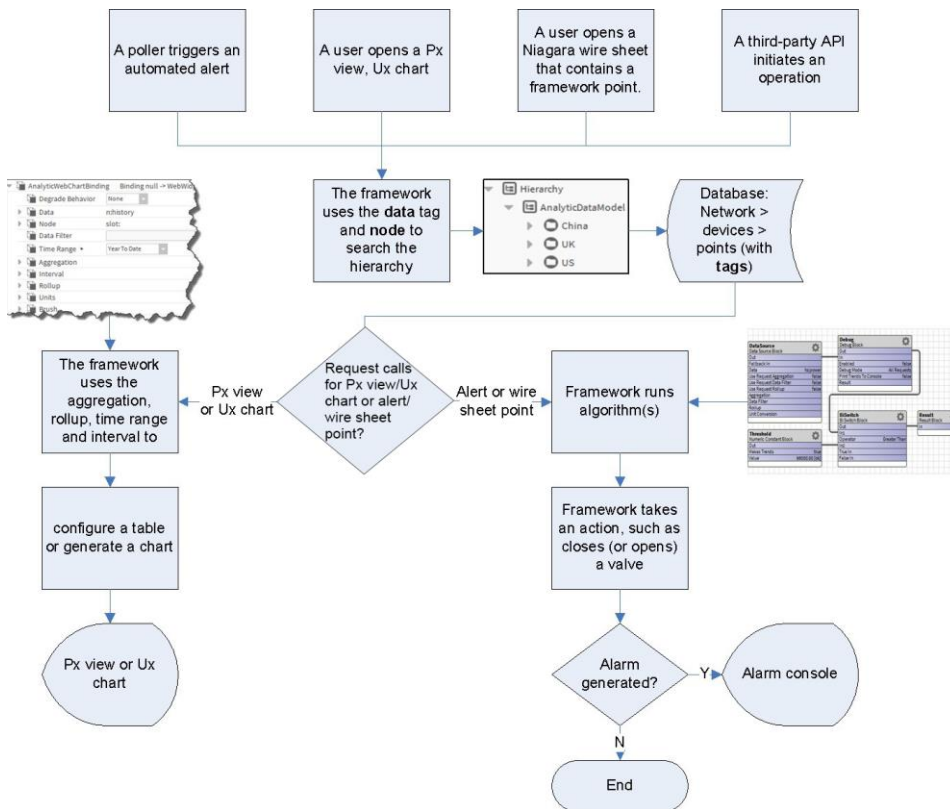
- Honeywell WEBs-N4 Analytics Framework Reference Guide (31-00279).
- Honeywell WEBs-N4 Analytics Framework 2.1 Brochure (31-00285).
- Honeywell WEBs-N4 Analytics Framework 2.1 Datasheet (31-00286).
- WEBs-N4 Analytics Framework Web API Guide documents the code you can use to extend this product.
- Niagara Hierarchies Guide provides information about setting up logical hierarchies.
- Niagara Tagging Guide provides information about adding metadata to objects.
- Niagara Relations Guide explains how to configure relationships in a hierarchy.
- Niagara Graphics Guide provides general information about how to create Px graphics.

# How the webs-N4 Analytics Framework works

The structure of WEBS-N4 makes it possible for the framework to render charts, generate alerts and automate device management with minimal configuration effort.

At the risk of oversimplifying framework workflow, the following flowchart shows how a request for data initiates processing, which results in human-readable analysis and action.

Figure 1 Four events initiate an action



## Related Links

- Features
- Key differences with WEBS-AX
- The framework in the Nav tree
- Acquiring data
- Analytics library
- Best practices

## Features

The heart of the framework is an advanced high-performance calculation engine. With this engine, real-time data can be combined with historical data using a set of wire and property sheets. This visual programming interface defines algorithms (formulas) that analyze the real-time and trend data collected from components, devices, and points. The output from this analysis can be visualized in charts and used as input to standard Niagara logic.



When applied to historical and real-time data, framework algorithms (formulas) can help you gain insight to better manage your operations. The product includes these features:

- An open and extensible analytical environment that you can customize to meet your needs
- Analytic tools that apply to any industry, including manufacturing, as well as building management
- The ability to set up complex analysis without custom programming
- Support for third-party API visualization and other complementary applications

### Related Links

- [How the webs-N4 Analytics Framework works \(Parent Topic\)](#)

## Key differences with WEBS-AX

There are some significant differences between the WEBS-N4 Analytics Framework for WEBS-N4 and the prior version for WEBS-AX.

- Alerts (diagnostics in the previous version) no longer have to be routed as alarms, they can be visualized.
- Alerts and point calculations can now be triggered and chained together for more job-like operation.
- The separate construction of a tree of custom objects is no longer required to model your data. Instead, standard Niagara hierarchies provide the benefit of multiple tree structures.
- Proprietary tags have been replaced with the WEBS-N4 standardized tagging infrastructure.
- Tags replace ORDs when defining data sources.
- Time ranges no longer have to align to the beginning of a period. For example, a business can now visualize quarterly data aligned to its fiscal year, rather than to the calendar year.
- Aggregations and rollups now offer the statistical functions: median, mode and range.
- The Java API has been completely refactored with a public interface enabling third-party software to access data as well as to create custom logic blocks.
- Units are automatically converted into English or metric depending on the user's language and time zone.
- English and metric versions of definitions and algorithms are provided.
- The ability to link the result of an algorithm to a logic block on a standard Niagara wire sheet makes it easier to get data out of the framework than was possible in earlier versions.

### Related Links

- [How the webs-N4 Analytics Framework works \(Parent Topic\)](#)

## The framework in the Nav tree

To begin your understanding of how the WEBS-N4 Analytics Framework data model works symbiotically with Niagara, this topic points out where in a typical Nav tree framework configuration appears.

These are the areas you use most frequently as you set up the framework.

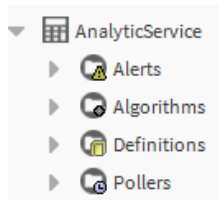
**TIP:** Consider opening each container in a separate tab so you can quickly navigate among them without having to scroll the Nav tree.

## Drivers container

Your Drivers folder models your network and devices. This is where all your points reside. When tagging points, you can use this space or the Hierarchy space (once you create one or more hierarchies).

## The AnalyticService

Figure 2 AnalyticService under the Services folder

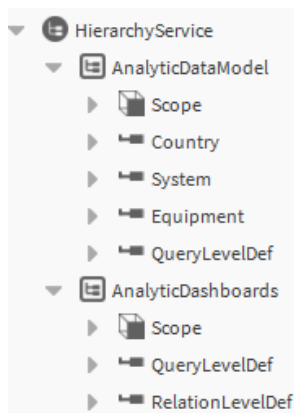


You use the **AnalyticService** in the **Services** container to configure framework properties. This folder also contains four sub-folders for configuring framework features:

- The **Alerts** folder contains alerts. These components use algorithms to evaluate conditions and may or may not generate an alarm.
- The **Algorithms** folder contains the formulas used by alerts, Px View and Web Charts.
- The **Definitions** folder contains properties to associate with tags.
- The **Pollers** folder contains components to manage data sampling frequency.

## HierarchyService

Figure 3 An example of a HierarchyService

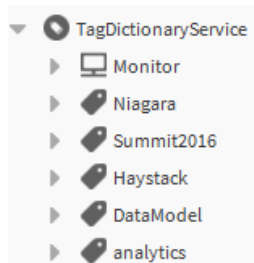


You use the **HierarchyService** to define a logical data model that is independent of your drivers-network-device model.

While this service is a feature of WEBS-N4, and not specifically a feature of the framework, it is indispensable to the framework. The **HierarchyService** provides the foundation for the alternative, meaningful data model that appears in the **Hierarchy** space.

## TagDictionaryService

Figure 4 An example of a TagDictionaryService

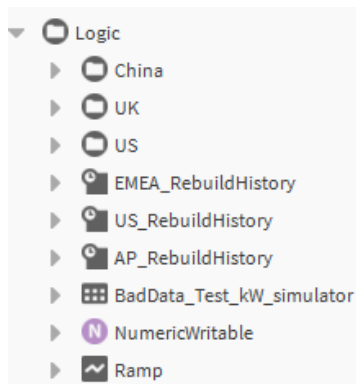


You use the **TagDictionaryService** to set up tags. When assigned to individual points within a hierarchy, the framework uses tags as data sources, which identify the points to analyze.

While the tag dictionary is a feature of WEBS-N4, and not specifically a feature of the framework, the WEBS-N4 Analytics Framework provides one of the most compelling reasons to use tags. This is where you create your own tag dictionary.

## Folder to contain framework logic

Figure 5 An example of a logic folder

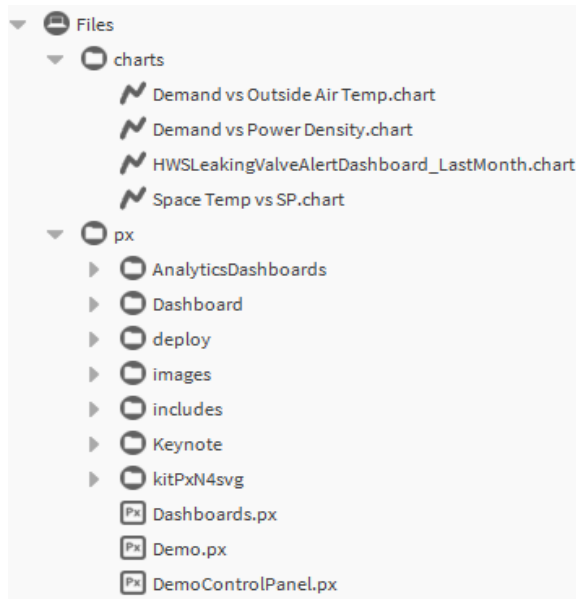


At the same level as the **Services** and **Drivers** containers, you can create a folder to contain framework-specific components, such as Px Views, schedules, special points, time triggers and proxy extensions.

The example screen capture is from a demonstration station. Your logic folder may be very different.

## Files folder

Figure 6 Example of chart files in the Files folder

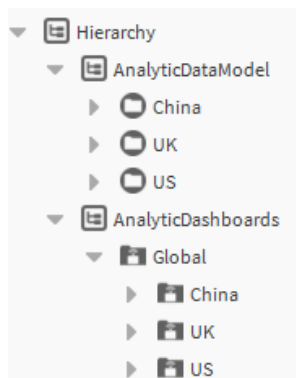


The Files folder contains framework chart files.

In the example, two folders separate the UxCharts (Web Charts) from the Px Views.

## Hierarchy folder

Figure 7 Example of a Hierarchy node



When you save a hierarchy, the **HierarchyService** automatically creates a **Hierarchy** structure at the same level as the **Config** container. You can use this hierarchy space to navigate from point to point.

This example defines two hierarchies. The **AnalyticDataModel** organizes building equipment (AHUs, etc.) by geographic location. The **AnalyticDashboards** hierarchy contains all dashboard representations for the geographical locations.

## Related Links

- How the webs-N4 Analytics Framework works (Parent Topic)

## Acquiring data

The data analyzed by the framework come from the proxy points that reside in the **Points** container of each driver network and device.

Prerequisites: All devices have been physically connected to the appropriate network and host. All networks and devices have been discovered and added to the host station (controller or Supervisor).

Perform the following steps:

1. To open the Points Manager, double-click the **Points** folder under a device in the Nav tree. The Points Manager opens. This view contains a row for each proxy point in the station database.
2. Click the **Discover** button.  
The screen splits into two windows. The upper window, Discovered, represents the newly-discovered data points.
3. Select one or more newly-discovered point(s) and click **Add**.
4. As needed, modify point properties and click **OK**.  
The system adds the point to the database.
5. Double-click the point in the Nav tree.

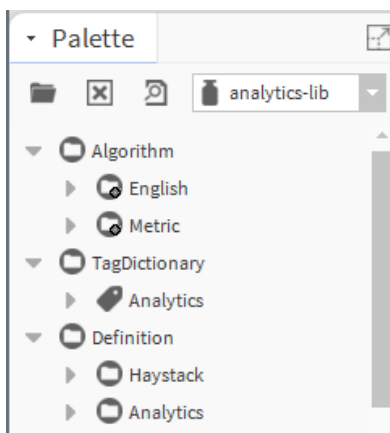
## Related Links

- How the webs-N4 Analytics Framework works (Parent Topic)

## Analytics library

The analytics-lib palette provides the pre-defined algorithms for a variety of common calculations. You can modify them or use them as examples when creating your own algorithms. To use an algorithm, drag it to a Wire Sheet and supply at least a data source. The Algorithm Library chapter of the WEBS-N4 Analytics Framework Reference documents these pre-defined formulas.

Figure 8 Analytics-lib palette



The analytics-lib palette consists of three folders:

- The **Algorithm** folder contains two sets of pre-defined algorithms: one set that supports English and the other that supports Metric units of measure. Each set contains instances of all algorithms including those that do not require specific units.
- The **TagDictionary** folder contains custom tags you can assign to frameworkpoints.
- The **Definition** folder contains English and Metric configurations. A definition is a set of properties that associates specific processing information with each tag, and consequently with each object (point) to which the tag is assigned. You can remove, modify, and add new definitions. Often the best course of action is to copy an existing definition, rename it with a unique name, then modify it to meet your needs.

### Related Links

- How the webs-N4 Analytics Framework works (Parent Topic)

## Best practices

The framework provides powerful tools for managing large and small buildings with increasingly sophisticated control features.

### Running in a remote controller

When running the framework in a remote controller, such as the WEB-8000, configure your PX graphics bindings and number of points carefully. While a remote controller can handle hundreds of points and multiple PX bindings, memory has its limits. During a heavy processing period, the system may slow and report server session time out errors.

### Refreshing cache memory

The framework requires memory based on the number of hierarchies, tags and points you configure for real-time and trend analysis. As you create hierarchies, tag points, build algorithms, and create alerts you should refresh cache memory frequently. To do so you right-click **Config > Services > AnalyticService** and click **Actions > Refresh Cache** or **Actions > Refresh Cache Full**. The difference between the two menu options is that **Refresh Cache Full** includes hierarchies. A simple **Refresh Cache** refreshes tags, algorithms, and alerts.

Once your framework data model is configured and stable, you refresh cache only after changing the configuration in some way. On a system with hundreds of points and complex algorithms, a refresh can take a substantial amount of time. Plan any framework changes so that your station will have time to refresh cache.

The third option, **Stop Caching** interrupts a refresh. You would use this action if a full refresh is taking a very long time.

## Algorithms

As you expand the framework 's potential, keep these algorithm best practices in mind:

- Define data sources carefully. When you refresh cache, the system scans every node in the hierarchy to determine where to run algorithms. The data sources, as well as some configuration properties, determine which nodes to run against.
- Once an algorithm has executed, set up a poll timer/poll queue to run it again at a specified time.

### Related Links

- How the webs-N4 Analytics Framework works (Parent Topic)

# Configuration overview

Configuring the framework for the first time should begin with a planning phase in which you decide on the information to analyze.

The Niagara Framework® maintains a hierarchy of components, devices, and points that reflect the physical network to which each object belongs. While information from each device and point is useful for tracking real-time values, and raising alarms, WEBS-N4 provides separate hierarchies, tags and relations with which to set up more meaningful relationships that may have nothing to do with the physical arrangement of devices on a network. The WEBS-N4 Analytics Framework (referred to as the framework in this documentation) builds on these standard WEBS-N4 features to collect and analyze real-time and historical data in a variety of ways.

For example, your campus may include many buildings. Each building has its own AHU unit for which the data model monitors three points: Cool Setpoint, Heat Setpoint and Supply Temp.

The following list summarizes the tasks involved in configuring the framework in your environment:

- If you are using a Supervisor computer (recommended), you need to add the framework components to the station. A Supervisor computer provides the necessary resources to process large quantities of data.
- If any aspect of the framework will run on a remote host, you need to install the modules and add the framework components to the station.
- Once your Supervisor computer and remote host(s) are ready to go, you add the tag to each device point that will be part of the framework.
- Each tag must be accompanied by a data definition, which identifies the type of data (cooling capacity, temperature, voltage, etc.) the tag represents. You can view the associated data definitions at **AnalyticService > Definitions**.
- After tagging each point, you set up a hierarchy by geographical location or, perhaps, by the person responsible for maintaining the AHU unit(s).
- Next, you use a pre-defined algorithm or create your own algorithm to perform calculations. These calculations define how to combine historical trend data, and maximum and minimum acceptable values.
- Each algorithm includes a DataSourceBlock. The Data property for this component contains the same tag as that used on the device points. Data collection happens by virtue of the assigned tags and hierarchy without requiring complicated programming.
- To visualize the collected data, you bind an algorithm to a Px or Web chart that can take the form of a dashboard or report. To run an algorithm, you open a chart that references the algorithm or set up a poller to run the algorithm at regular intervals.
- Alerts use algorithms that yield a binary result (true or false). A true result can generate an alarm, which appears on the standard alarm console.
- The results of an algorithm can serve as input to a standard Niagara Framework® Wire Sheet for the purpose of controlling device performance based on complex logic.

**NOTE:** You can start by working with a few tags, hierarchies, and algorithms, learning how to visualize and manage the results a little at a time.

## Related Links

- Prerequisites
- Setting up a Supervisor computer
- Installing on a remote host
- Adding the framework to a station
- About licensing
- Setting up user authentication

## Prerequisites

To use WEBs-N4 Analytics Framework, several conditions must be met.

### Niagara certification

As the systems integrator or data model designer, you have completed Niagara certification training and are familiar with the WEBStation workbench interface.

### Supported hosts

WEBs-N4 Analytics Framework runs on a WEBs-N4 Supervisor or WEB-8000 host.

### Memory requirement

The number of points, algorithms and history records a host (PC or remote host) can process is limited by available memory. Compared to running in a remote host, more memory resources are available when running a Supervisor station on a PC.

A lack of adequate memory (heap memory) to run the framework can cause a Supervisor station to fail. The amount of available heap memory on a PC is determined by the Xmx property as configured in the nre.properties (Niagara Runtime Environment) file. This text file is located in the WEBStation user home. Your PC must have enough physical memory resources to accommodate any change you make to the Xmx property.

### License requirement and limitations

WEBs-N4 Analytics Framework must be licensed for your host (Supervisor PC or remote host). You add the feature to an engineering license using the standard Niagara licensing model.

The license limits the number of points the framework can use. An a:a tag on a point marks it as being used by the framework. The **AutoTagAnalyticPoint** property on the AnalyticService controls the automatic tagging of points used by the framework. When this property is set to true, the framework applies the a:a tag to any point referenced by an algorithm or Px view.

### Modules required

These modules run under all WEBs-N4 versions.

- analytics-lib-ux.jar
- analytics-rt.jar is required by both stations and engineering platforms running tools (WEBStation workbench)
- analytics-ux.jar
- analytics-wb.jar is the user interface. This module is required to run the engineering tool (WEBStation workbench).

### Core software and modules required



The framework requires WEBS-N4.2 or later. The specific modules the framework requires include:

- alarm.jar
- baja.jar
- bajai.jar
- bql.jar
- control.jar
- driver.jar
- email.jar
- file.jar
- fox.jar
- gx.jar
- history.jar
- niagaraDriver.jar
- platform.jar
- schedule.jar
- wbutil.jar
- web.jar
- workbench.jar

These modules reside in the modules folder.

### Browser requirement

The framework's visualization tools include web charts (Ux charts) with scalable, vector graphics. These graphics require a browser that supports HTML5.

### Station configuration

The procedures in this guide assume that you have configured the network with at least one remote host and station whose device drivers and points have been set up and configured for your application. It also assumes that all proxy points have been discovered and configured in any remote host station, and in your Supervisor station.

### Related Links

- Configuration overview (Parent Topic)

## Setting up a Supervisor computer

To set up the framework on a Supervisor computer you drag the AnalyticService from the analytics palette to the **Services** folder in the Nav tree.

1. Open WEBStation workbench and connect to your Supervisor station.
2. Open the analytics palette.
3. Drag the AnalyticService component from the palette to the **Services** folder in the station's Nav tree.
4. Open the analytics-lib palette and expand the Tag\_Dictionary folder.
5. Drag (or copy and paste) the Analytics container from the palette to the **TagDictionaryService**

in the Nav tree.

6. As a best practice, save the station (right-click the station in the Nav tree and click **Save Station** from the drop-down menu).

### Related Links

- Configuration overview (Parent Topic)

## Installing on a remote host

This is the preferred installation method for installing the framework on a remote host.

1. Open WEBStation on your Supervisor computer.
2. Open a platform connection to the remote controller, and connect to the station.
3. Open the Software Manager view in the remote station and scroll down to the analytics module in the list of software installed on the Supervisor computer.
4. Select analytics and click **Install** (at the bottom of the view).

The installation software checks the versions of all other installed modules on the host, displays a list of any that are out-of-date (compared to the modules installed locally), and pre-selects an install option.

5. De-select the install option for any modules other than the framework modules.  
**CAUTION:** Do not bring any modules other than the framework modules up to date.
6. To install the module files on the remote host, click **Commit**.  
The Software Manager may need to automatically stop any running station. It displays a confirmation window.
7. To confirm the station stop, click **OK**.  
The Software Manager stops the station and continues with the module installation process. When finished, the Software Manager displays that module files are Up to Date.

### Related Links

- Configuration overview (Parent Topic)

## Adding the framework to a station

The last installation step is to add the **AnalyticService** component from the Analytics palette to each station (localhost and remote host). You do this using WEBStation workbench on your Supervisor computer.

Prerequisites: The framework jars are in the modules folder and WEBStation is connected to your local Supervisor station.

1. Open the analytics palette and locate the AnalyticService component.
2. Drag the AnalyticService component to the **Services** container in the station's Nav tree.  
**NOTE:** The **AnalyticService** component is not a driver and does not belong under the **Drivers** container. While you could locate it anywhere in the station Nav tree, as a best practice, locate it directly under the **Services** container.
3. Open the analytics-lib palette and expand the Tag\_Dictionary folder.
4. Drag (or copy and paste) the Analytics container to the **TagDictionaryService** in the Nav tree.

### Related Links

- Configuration overview (Parent Topic)

## About licensing

For a point to be used by the WEBS-N4 Analytics Framework, it must be tagged with the a:a (analytics) tag. The system automatically compares the total number of points thus tagged with the allowed points on your license.

Be aware of these licensing-related factors:

- When set to true, the **AutoTagAnalyticPoint** property on the AnalyticService causes the framework to automatically tag each new point with the a:a tag as required for use with the service. During initial station configuration, setting this property to true saves time. However, once the framework is configured and running in your Supervisor station, you should set this property to false.
- If the **analytics** container and a:a tag are missing from the **TagDictionaryService**, the AnalyticService tries to add them to the **TagDictionaryService**.
- You can use a BQL scaler function in a platform’s Program service to query points with a specific tag (such as the a:a tag) and bulk edit them.
- If you remove the a:a tag from a point, you must refresh cache to decrement the a:a tag counter. To refresh cache, right-click the **AnalyticService** and click Actions > Refresh Cache Full.

### Related Links

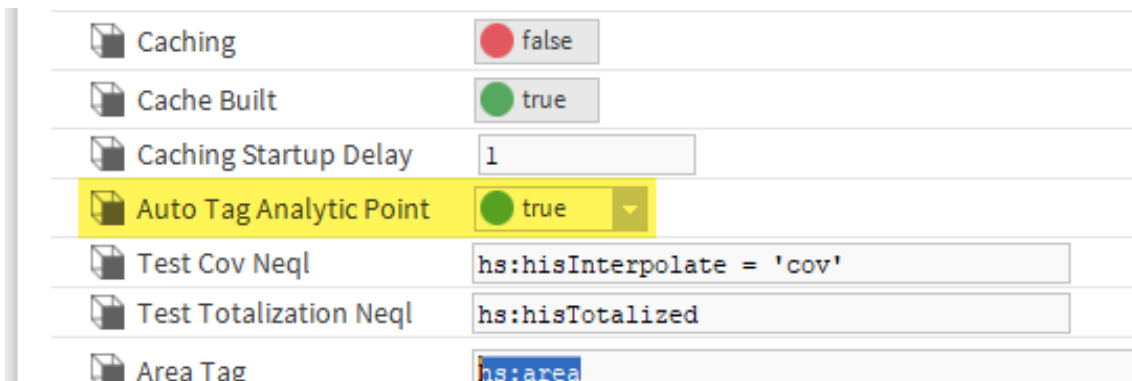
- Configuring the service for licensing
- Determining the number of points used
- Confirming that the AnalyticService component is licensed
- Configuration overview (Parent Topic)

## Configuring the service for licensing

A WEBS-N4 Analytics Framework license is based on the number of points used in algorithms. As you identify the points to use in algorithms and alerts, the system automatically tags each point with the a:a tag. You can view the number of points configured. An AnalyticService property turns the automatic tagging on and off. This procedure explains how to configure this property.

1. Right-click the AnalyticService and click **View > Property Sheet**.

The AnalyticService property sheet opens.



2. To enable automatic tagging with the a:a tag, set Auto Tag Analytic Point to true and click **Save**.

You enable this property before adding algorithms and alerts. Once your framework is configured you should disable this feature.

**CAUTION:** If you leave this feature enabled, and, at some future time, exceed the number of licensed points, the framework will stop working.

### Related Links

- About licensing (Parent Topic)

## Determining the number of points used

This procedure explains how to determine the number of points used. This number must not exceed the number of points allowed by the license. If it does, the framework stops working.

1. Right-click the AnalyticService in the Nav tree and click **View > Property Sheet**.

The AnalyticService property sheet opens.

2. Check the Point Count property.

The system counts each point that has the a:a tag associated with it. The system automatically adds this tag as you configure algorithms and alerts. If you change your mind about using a particular point in a calculation, you can remove the a:a tag from the point.

If you remove the a:a tag, you must rebuild memory cache for the deletion to take effect.

3. To rebuild cache, right-click AnalyticService and click **Actions > Rebuild Cache**.

Rebuilding cache re-calculates the Point Count.

### Related Links

- About licensing (Parent Topic)

## Confirming that the AnalyticService component is licensed

A component is licensed if its Status property reads {Ok}.

1. Open WEBStation workbench and connect to the station (Supervisor or remote).
2. Double-click the AnalyticService component in the Nav tree. The property sheet opens.
3. Confirm that the Status property reads {Ok}.

You are ready to begin analyzing data and spotting trends.

### Related Links

- About licensing (Parent Topic)

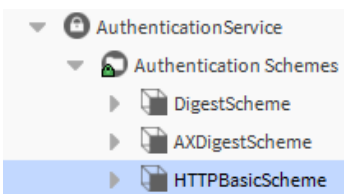
## Setting up user authentication

Access to a station database requires user authentication, which is managed by the station's **AuthenticationService**. As with physical access, programmatic access requires authentication using the HTTPBasicScheme (HTTP Basic Authentication Scheme). Consider using a separate user for each type of access (physical access, programmatic access, etc.). This practice provides additional security as each user requires only the minimum number of access rights necessary to accomplish a specific task. Using roles and tagged categories allows for highly-configurable permissions for accessing various station components.

Prerequisites: You have administrative rights. The station is open in WEBStation workbench.

Perform the following steps:

1. Open the baja palette.



2. Add the **HTTPBasicScheme** under the **Services > Authentication Service > Authentication Schemes** node in the nav tree.

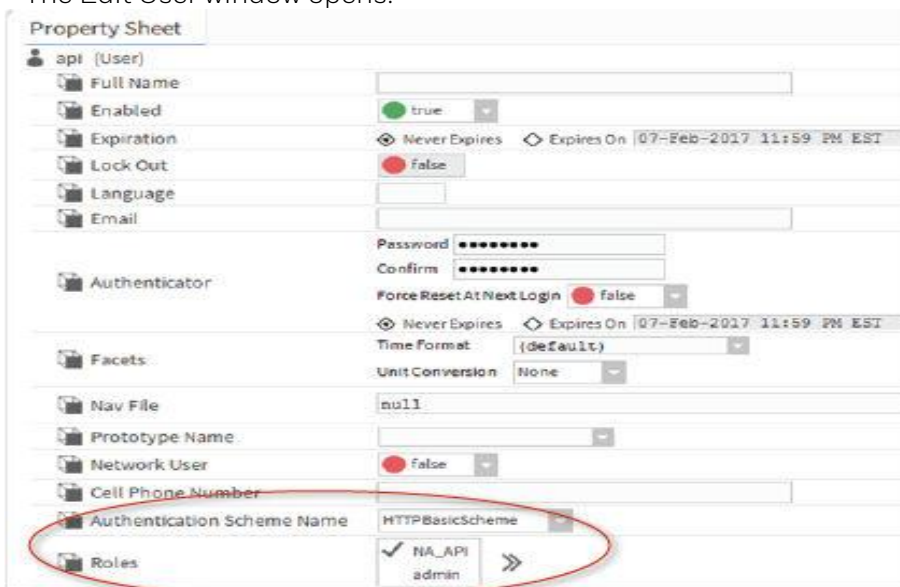
3. Create a role to assign to users based on the type of access.

For example, users who are permitted to create and view web charts may be assigned the “NA\_charts” role. Permissions for this role might allow a user to read from the database but not invoke actions or write records to it.

Users who are permitted to query the station database with API calls may be assigned a “NA\_API” role. Permissions for this role might allow a user to read from the database and invoke actions, but not write records to it.

4. Expand **Services > UserService** in the nav tree and double-click the user name you intend to use to access the station database.

The Edit User window opens.



5. Select HTTPBasicScheme from the Authentication Scheme Name drop-down list, assign the role you created, and click **OK**.

### Related Links

- Configuration overview (Parent Topic)

# Tags, hierarchies and relationships

A station's **Config > Drivers** tree structure organizes a physical group of devices by network protocol, device, and point. This structure does not, for example, identify which building, region or tenant a point belongs to. Using the powerful hierarchy, tag, and relations features of WEBS-N4, the WEBS-N4 Analytics Framework allows you to set up structures for analysis purposes without requiring you to completely configure separate data models.

Hierarchies, tags, and relations are features of WEBS-N4. More information about working with them can be found in the Niagara Hierarchies Guide, Niagara Tagging Guide and Niagara Relations Guide.

## Related Links

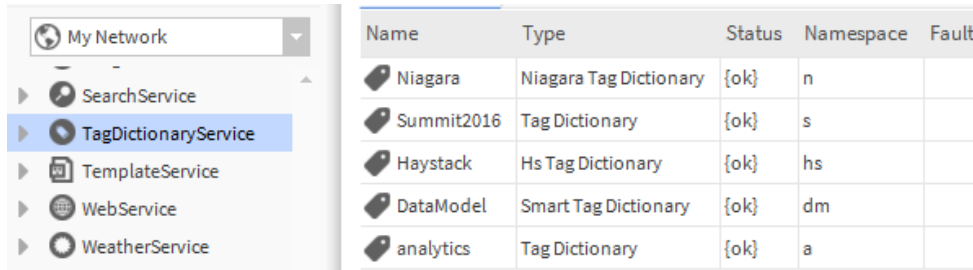
- Adding the tag dictionary
- Setting up a hierarchy
- Relationships
- Associating definitions with tags
- Changing the default behavior for a value tag
- Tagging Niagara Network points with n:history
- Tagging BACnet Network points with n:history
- Acquiring tags from remote stations
- Tag inheritance and the a:a tag
- Removing all a:a tags

## Adding the tag dictionary

By default, two dictionaries are part of every Niagara build: 1) The Niagara dictionary contains commonly-used tags. 2) The Haystack dictionary is an open source dictionary created by Project Haystack to “streamline working with data from the Internet of Things.” (project-haystack.org). The WEBS-N4 Analytics Framework comes with a third pre-configured Analytics Tag Dictionary. Before you create your own dictionary, view the tags provided by these tag dictionaries. They may meet your needs.

Prerequisites: The station is connected.

1. To confirm the presence of tag dictionaries, double-click the **TagDictionaryService** in the Nav tree. The Tag Dictionary Manager opens.



Name	Type	Status	Namespace	Fault
Niagara	Niagara Tag Dictionary	{ok}	n	
Summit2016	Tag Dictionary	{ok}	s	
Haystack	Hs Tag Dictionary	{ok}	hs	
DataModel	Smart Tag Dictionary	{ok}	dm	
analytics	Tag Dictionary	{ok}	a	

- To add the Analytics tag dictionary to a station, open the analytics-lib palette, expand the Tag\_Dictionary folder in the palette, and drag (or copy and paste) the Analytics tag dictionary to the **Config > Services > TagDictionaryService** in the Nav tree.
- To view the tags contained in any tag dictionary, expand the **TagDictionaryService**, expand the specific tag dictionary name, and expand or double-click the **Tag Definitions** node.
- To view the Property Sheet for an individual tag, double-click the tag in the list.

## Related Links

- Tags, hierarchies and relationships (Parent Topic)

## Setting up a hierarchy

A hierarchy is a tree of level definitions, which identify the tags and NEQL queries to use when searching for data. Hierarchies provide meaningful relationships among data points. The same data accessed in different hierarchies can yield different analytical results.

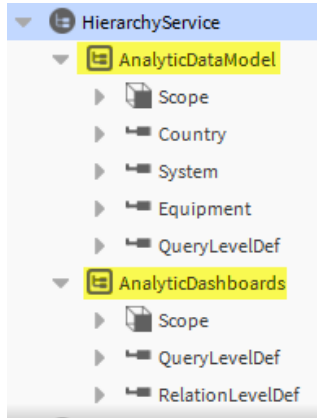
Prerequisites: All devices, proxy points and other components are tagged. The hierarchy palette is open.

- Drag the Hierarchy node from the palette to the **HierarchyService**, giving it a descriptive name. The **Scope** node that comes with the new hierarchy defines the limitations (scope) of the search. The default scope is the station.
- Double-click the **Station node** under the **Scope node**.  
The Property Sheet opens.
- To limit the extent of the hierarchy query that searches for data, you can update the Scope Ord to a specific slot.  
This is an optional step. For example, station:|slog:/Model/westerre/W1 limits the search to a slot in the station logic.
- Drag one GroupLevelDef from the hierarchy palette to the **Hierarchy** node for each collection of related points.  
Group level definitions gather information into groups.  
For example, to group by Country, System and Equipment type you need three GroupLevelDefs under the Hierarchy node.
- For each GroupLevelDef, set the **Group By** property to the value tag associated with the points for each group and enter the value.  
For example, for Country the tag might be n:geoCountry and the value China. For system the tag might be s:system and the value, and for equipment, s:equipment.  
The first example tag is from the Niagara tag dictionary. The other two are ad hoc tags created by the model designer.

6. Add other level definitions as needed.

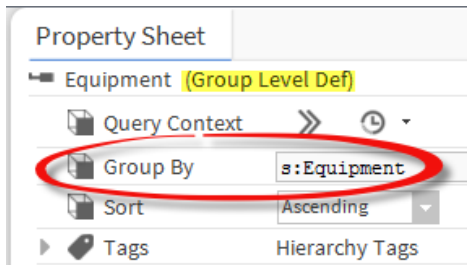
- A ListLevelDef requires one or more NamedGroupDefs and at least one QueryLevelDef below it. Each named group definition serves as a placeholder folder under the ListLevelDef.
- A QueryLevelDef sets up a NEQL query the system uses to search for data.
- A RelationLevelDef defines a relationship with a parent element. The system returns data for all objects that are related to the level immediately above it. The relationship is usually a child relationship (n:child)

The following is an example of two hierarchies.



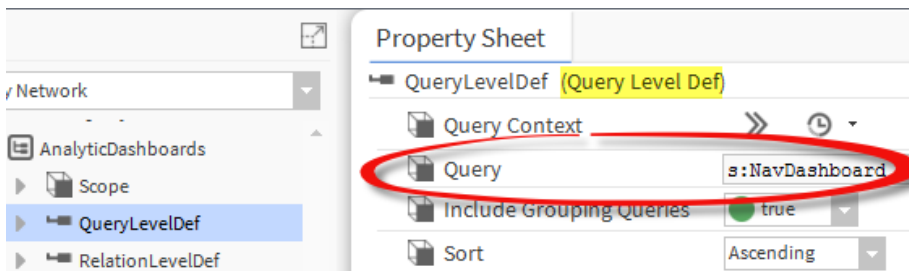
The first hierarchy in the screen capture, **AnalyticDataModel**, uses three GroupLevelDefs (**Country**, **System** and **Equipment**). A level definition (LevelDef) is an element used to structure the hierarchy.

Property sheet for Group Level Def Equipment



Each level definition defines the tag the system uses to collect information. In this case the tags are: n:geoCountry, s:System, and s:Equipment.

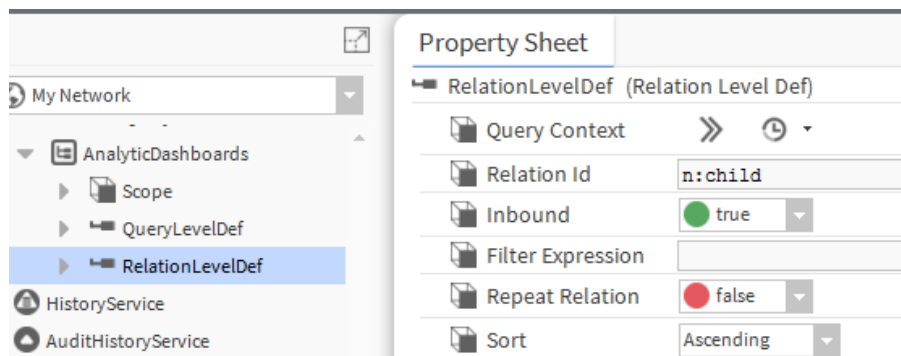
Property sheet for a Query Level Def





The second hierarchy, **AnalyticDashboards**, collects some of the same information, but for a different purpose. Its Query Level Def combines data from all points tagged with s:NavDashboard.

The RelationLevelDef sets up a relationship with the level element's parent.



This level element in the hierarchy tells the system to search the parent node of the station logic for data.

### Related Links

- Tags, hierarchies and relationships (Parent Topic)

## Relationships

The relations feature provides the mechanism for structuring relationships between points in a hierarchy.

A relationship exists between two components. A parent component has one or more child relationships. The collection of data flows from one point to another based on the relationship between points. There are two types of relations:

- Direct relations are those that you apply (using relation markers) directly to a point. These relations are limited to relations that are defined in a tag dictionary. The query configured by a relation definition on a hierarchy causes the system to return data
- Implied relations are defined by tag rules in a Smart Tag Dictionary and applied automatically by the system. The query configured by a relation definition on a hierarchy causes the Smart Tag Dictionary to interpret the tag rules against the given point and return a list of implied relations.

You establish relationships in two places:

- By placing relations markers on points. These identify parent and child points.
- By adding relation level definitions to hierarchies set up where the system searches for data beyond the individual point.

Relations themselves may be tagged. The Relations Guide describes more fully how to set up relationships.

## Related Links

- Tags, hierarchies and relationships (Parent Topic)

## Associating definitions with tags

The **Definitions** container under the **AnalyticService** contains information types that identify the characteristics of the data to be analyzed. A set of pre-defined definitions are in the analytics-lib palette. You can create your own definitions. This topic explains how to modify a definition, how to copy another, and customize it to create a new definition. This work is done in WEBStation running on a Supervisor platform.

Prerequisites: WEBStation is connected to the station, the AnalyticService component is in the Config container, and the TagDictionaryService includes the Haystack set of tags.

1. To access the definitions, double-click **AnalyticService > Definitions**.  
The Analytic Data Manager opens.
2. Do one of the following:
  - To create a new definition, click **New** at the bottom of the window.
  - To **edit** an existing definition, double-click it in the Nav tree or Analytic Data Manager, or select it in the manager and click Edit.

The definition Property Sheet or Edit window opens.

The screenshot shows a 'Property Sheet' window for an 'Analytic Data Definition' named 'coolingCapacity'. The properties are as follows:

Property	Value
Status	{ok}
Fault Cause	
Facets	units=BTU/hr,precision=2 BTU/hr,min=-inf BTU/hr,ma... >> ⌚
Aggregation	First
Rollup	First
Id	hs:coolingCapacity

The example shows definition properties as they appear in the Property Sheet.

3. Configure the properties as needed.  
The Id property associates the definition with a tag.

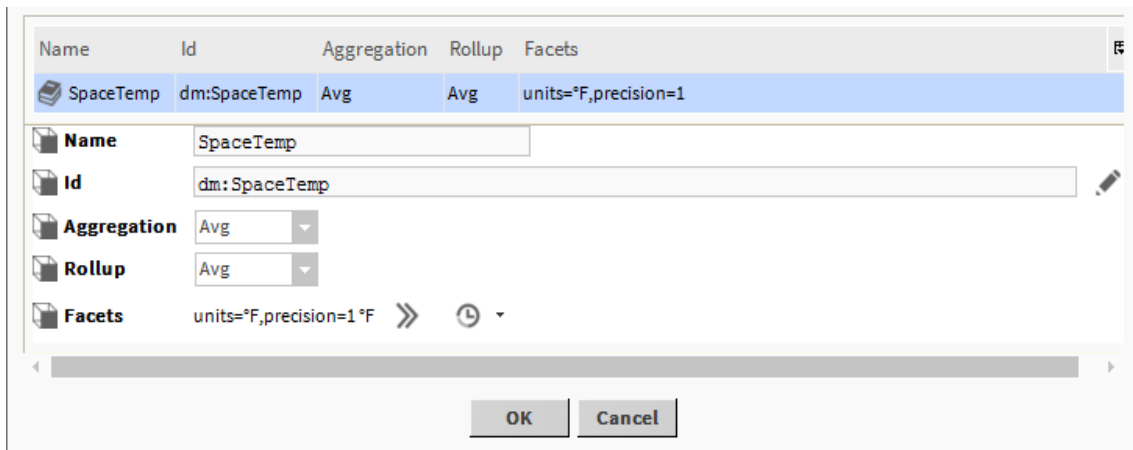
## Related Links

- Tags, hierarchies and relationships (Parent Topic)

## Changing the default behavior for a value tag

Each value tag has a definition associated with it. This definition sets up its default value and controls other aspects of its behavior.

1. Expand the **Config > Services > AnalyticService** container and double-click **Definitions**.  
The Analytic Data Manager opens.
2. To create a definition, click **New** at the bottom of the window or double-click a definition row to edit an existing definition.  
The Edit window opens.



3. Change the properties associated with the tag identified by the Id property.  
You can set facets in the definition. This is important to ensure that all aggregated data use the same units and precision.  
As with other such Edit windows in Niagara, you can change the properties for more than one tag at the same time.

### Related Links

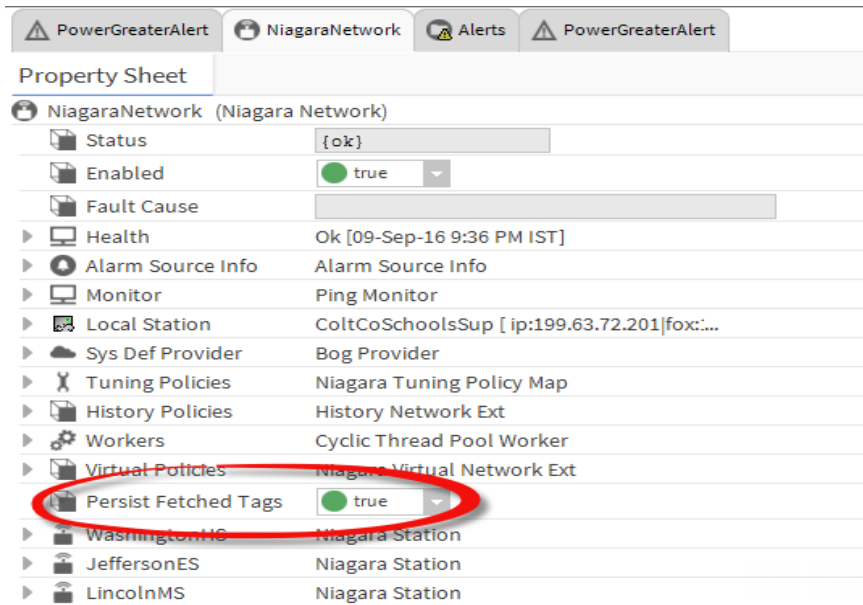
- [Tags, hierarchies and relationships \(Parent Topic\)](#)

## Tagging Niagara Network points with n:history

To render data on charts and in tables, each point requires an n:history tag. As newly-discovered Niagara Network points do not have history extensions with an implied n:history tag, use this procedure to automatically add the n:history tag to each newly-discovered point.

Prerequisites: Points have been discovered.

1. Right-click the **Config > Drivers > NiagaraNetwork node** in the Nav tree and click **Views > Property Sheet**.  
The Niagara Network Property Sheet opens.



2. Enable Persist Fetched Tags (change its value from false to true).
3. To finish preparing the points for analysis by the framework, right-click again and click **Actions >Force Update Niagara Proxy Points**.

This adds the points.

**NOTE:** It may take a few seconds to add the n:history tag to all newly-discovered points. Be patient and wait for the procedure to complete.

### Related Links

- [Tags, hierarchies and relationships \(Parent Topic\)](#)

## Tagging BACnet Network points with n:history

To render data on charts and in tables, each point requires an n:history tag. Newly-discovered BACnet points do not have history extensions with an implied n:history tag and no action exists in the BACnet Driver for automatically tagging these newly-discovered points. Instead, users need to write a Program Object that will add the n:history tag to the discovered points. Sample code is available. The n:history tag supports BFormat, which allows the linking of points to histories in a flexible manner.

### Related Links

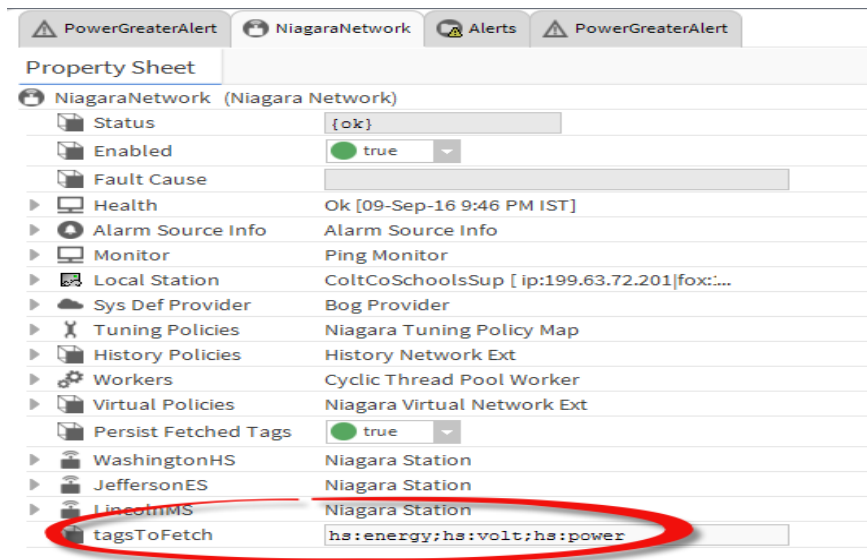
- [Tags, hierarchies and relationships \(Parent Topic\)](#)

## Acquiring tags from remote stations

The tags in a Supervisor (or other) station may not include the tags that identify Niagara Network points in a remote station. To analyze data collected from remote proxy points, the tags must be added to the central station.

Prerequisites: The tags used by Niagara Network device points in the remote station are different from those used in the Supervisor or other central station.

1. Right-click the **Config > Drivers > NiagaraNetwork node** in the Nav tree and click **Views > Slot Sheet**.  
The Niagara Network Slot Sheet opens.
2. Right-click the slot sheet, click **Add Slot**, name the slot something like tagsToFetch, and click **OK**.
3. Right-click **NiagaraNetwork** again and click **Views > Property Sheet**.



4. Type the tags from the remote station into the tagsToFetch slot property using a semi- colon (;) to separate multiple tags.

At the next poll, the framework will pull the tags to the requesting station along with the point data.

## Related Links

- [Tags, hierarchies and relationships \(Parent Topic\)](#)

## Tag inheritance and the a:a tag

The special tag ( a:a) identifies each point used by the framework. It is an origin entity. Entities that declare a tag are an instance of that tag. Think of this entity as having an “is a” relationship with the tag, while descendants of this entity have an “in a” relationship. This is a useful concept for algorithms.

When you add the a:a tag to a device or point, all descendant devices and points inherit the tag. In a building example, entities representing floors, rooms and devices under a building also receive the building tag, even though you did not explicitly add it to them. Tagging a device or point with several tags automatically, based on inheritance, can save configuration time.

Hierarchies group tags hierarchically. Any device or point to which you assign a tag as part of a hierarchy also inherits the tag’s ancestor tags.

## Related Links

- Tags, hierarchies and relationships (Parent Topic)

## Removing all a:a tags

You can use the ProgramService's Batch Editor to remove the a:a tag from all points used in an alert or algorithm.

1. Open the BQL query builder and copy this command:  
analytics:BqlLib.hasTag('a:a')
2. Open the ProgramService and click **Find Objects**.  
The Bql Query Builder opens.
3. Paste the command in the left-hand **Match** property.  
The operator should be equals.
4. In the right-hand field type true and click **OK**.
5. Click **Remove Slot** (button along the bottom of the window), select the a:a property, and click **OK**.
6. Click **Find Objects**.  
The Bql Query Builder opens.
7. In the Find section, click the search icon.  
The Choose Root window opens.
8. Select **Config** and click **OK**.  
The system performs the request.
9. Right-click AnalyticService and click **Actions > Rebuild Cache**.

## Related Links

- Tags, hierarchies and relationships (Parent Topic)

# Algorithms and alerts

An algorithm performs a calculation on real-time or historical (trend) data to generate a result. The result can trigger an alert, be displayed on a chart, or can become an input to another calculation (another algorithm or regular Niagara wire sheet).

A single algorithm can run against data collected from an entire building. For example, assume your building has 100 air handling units and an algorithm to monitor their performance. When you add 50 more units, and tag each unit appropriately, without any additional effort on your part the original algorithm applies to all 150 units.

Using a poller, an alert runs an algorithm to monitor point performance. Alerts may trigger alarms, which appear on the normal alarm console. If the alert that triggered an alarm continues to exist, the alarm persists on the alarm console even after it has been acknowledged or force cleared.

The framework provides two uses for algorithms. You can use them to:

- Test for an individual condition. The alert running the algorithm can then automatically sound an alarm or trigger a remedy.
- Look back at the history of a point and report any events that meet defined criteria. For example, an algorithm can answer the question: Has there been any time in the past when a hot water valve was open more than 90% with a room temperature of three degrees below the setpoint for an hour? Depending on the amount of data (that is, how far back in the past the stored data go), you may be able to identify a consistent pattern (a trend) that points to a condition requiring attention.

**NOTE:** Although algorithms run on any platform, if you intend to process large volumes of historical data, consider running algorithms on a Supervisor platform.

## Related Links

- [Creating an algorithm](#)
- [Creating an alert](#)
- [Editing an algorithm](#)
- [Using algorithm results in standard logic](#)
- [Algorithm for removing unwanted data](#)
- [Viewing an alert in the alarm console](#)

## Creating an algorithm

Algorithms are formulas created on a wire sheet that enable calculations, where the framework can combine real-time and historical data to produce values and analyze trends (time series).

Prerequisites: All objects are tagged, hierarchies created, and relationships established.

1. In the station's Nav tree, expand **Config > Services > AnalyticService**, and double-click **Algorithms**.

The Algorithm Manager opens to the Database view.

2. To group algorithms, create a folder by clicking **New Folder** at the bottom of the manager and supplying a descriptive name.
3. Do one of the following
  - a. To create a new algorithm, click **New**, select the number of algorithms to add, and click **OK**.
  - b. To edit an existing algorithm, select it in the Database table and click **Edit** (or double-click the row in the table).

The New or Edit window opens.

Name	Enabled	Makes Trends	Aggregation	Rollup	Min Interval	Max Interval	Facets
Algorithm	true	true	First	First	None	None	

**Name**: Algorithm  
**Enabled**: true  
**Makes Trends**: true  
**Aggregation**: First  
**Rollup**: First  
**Min Interval**: None  
**Max Interval**: None  
**Facets**: >> ↻

OK Cancel

4. Give the algorithm a name, make any changes to the default properties and click **OK**.

The framework requires data policies for these algorithm properties:

- Aggregation – defines how to combine current values from multiple data sources.
- Rollup – defines how to combine the multiple values contained in a single interval of a trend.
- Min and Max interval – some data may have limitations that establish the acceptable minimum and maximum period to use for rollups. For example, heat and cooling degree days should not have an interval of less than one day.

The algorithm and properties display as a row in the Database table.

## Related Links

- Defining the data source
- Adding logic
- Filtering algorithm input data
- Example: Monitoring temperature and humidity
- Algorithms and alerts (Parent Topic)



## Defining the data source

The data source defines the tag that supplies the data for processing and charting.

Prerequisites: The Algorithm Manager is open.

1. If necessary, expand the AnalyticService and double-click **Algorithms**.

The Algorithm Manager opens.

2. Right-click the new algorithm in the Algorithm Manager view and click **Views > Wire Sheet**.

A new algorithm contains a single result block.

3. Do one of the following:
  - a. To start with a pre-configured algorithm from the library, open the analytics-lib palette, expand the Algorithm node and version (English or Metric), and drag an algorithm to the Wire Sheet.
  - b. To build an algorithm from scratch, open the analytics palette, drag a DataSourceBlock to the Wire Sheet, and click **OK**.

You do not need to name a data source block.

4. To open the data source property sheet, double-click the block and, if necessary, expand the window to see all properties.
5. Enter a tag name for the **Data** property.

Tags and tag groups identify the data sources that feed an algorithm's calculations. Tags and tag groups replace the ORDs that in previous versions of the framework pointed to specific data sources. The client of the framework does not distinguish data from a point and a result calculated on the fly by an algorithm.

A single algorithm can run against data collected from an entire building. For example, assume your building has 100 air handling units and an algorithm to monitor their performance. When you add 50 more units, and tag each unit appropriately, without any additional effort on your part the original algorithm applies to all 150 units.

Algorithms can also receive tags (you can put a tag on an algorithm).

6. Expand the algorithm facets and assign the appropriate unit of measure to use for output values.

No unit conversion from a data source block occurs unless the algorithm defines its unit of measure. If algorithms are chained together, the next algorithm in the chain defaults to the previous algorithm's unit. If no unit is defined in the previous algorithm, the next algorithm defaults to the unit defined in its data source block.

7. Configure any other properties and click **Save**.

### Related Links

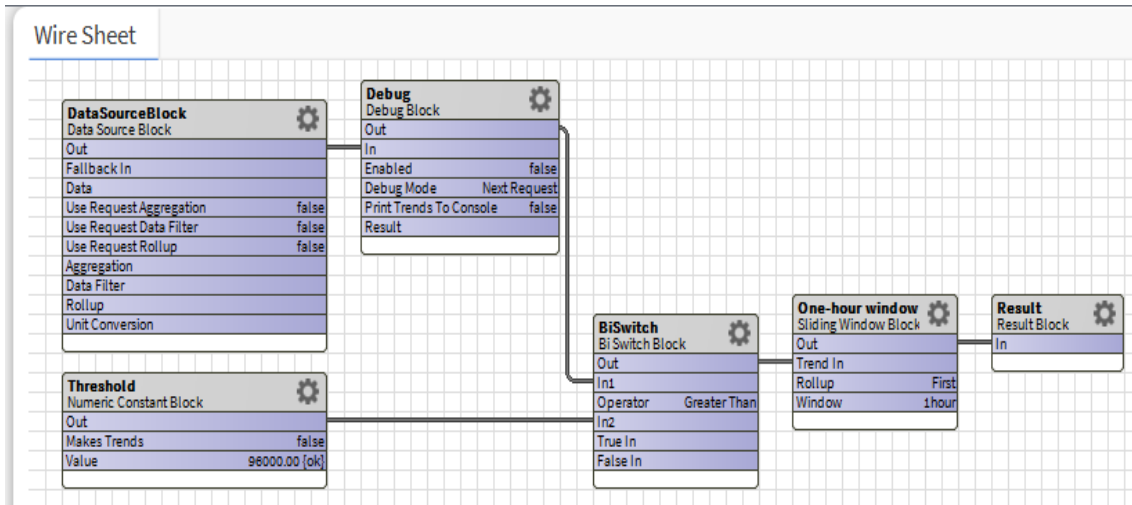
- [Creating an algorithm \(Parent Topic\)](#)

## Adding logic

Logic includes constants, function blocks, such as Math, and switch blocks.

Prerequisites: The Wire Sheet view of the algorithm is open.

1. To add logic, drag a constant or function from the analytics palette and name it appropriately.



2. If you added a constant, double-click the object and set its Value property.

**NOTE:** Instead of double-clicking the object to open the property sheet, you can right-click the object, click Actions > Value and enter a value or select an option. For example, when creating a constant, this action assumes you want to set the value. This right-click shortcut works with all algorithm components to speed the configuration process.

3. If you added a math block, double-click it and set its Operator (or right-click and use the action).

The UniMath and BiMath objects indicate the number of inputs. Something similar is true for the switches.

**NOTE:** You must connect all inputs required for each math object. For example, do not use a bi object if you have only one input.

4. Link the outputs of the DataSourceBlock object(s) to the logic block input slots.

**CAUTION:** It is easy to misconnect the output link from the input points to the logic objects. Make sure you connect from the Out slot, not the Value slot.

5. To add comparison logic, drag a BiSwitch from the palette Switches section and name it based on the condition you are testing for.

For example, if you are testing for a temperature that is too high you might name it OverSetpoint.

6. Continuing with comparison logic, double-click the switch (or right-click and use the action) and define the Operator (Greater Than, Less Than, etc.), then connect the appropriate Out slot(s) to it.

**NOTE:** If you edit a property, click Save. Sometimes the connector lines disappear after updating a property. If you view a Wire Sheet and only its objects are visible, click the refresh button.

7. To add a time constraint to the formula, drag the SlidingWindow object from the palette and name it for the period of time.

For example, OneHourWindow.

8. Continuing with the time constraint, drag a BiSwitch object from the palette and configure its Operator to And.
9. Complete the connections and double-check all of them.
10. Refresh cache by right-clicking **AnalyticService > Actions > Refresh Cache Full**, and save the station. Refreshing cache locates all nodes and sets up a pre-defined path to each node.

## Related Links

- [Creating an algorithm \(Parent Topic\)](#)

## Filtering algorithm input data

You may control algorithm results by configuring which data enter the algorithm for processing. There are three ways to do this: assign a tag other than the tag called for by the DataSourceBlock to a device point, change the Data source property, or configure a Data Filter. An algorithm may include or exclude data tagged with a specific tag. This topic provides steps for controlling algorithm input by configuring DataSourceBlock properties.

Prerequisites: All objects are appropriately tagged. Algorithms exist.

1. Expand the AnalyticService folder.
2. To open the algorithm wire sheet, double-click the algorithm.
3. To open the properties for the data source, double-click a DataSourceBlock logic block.
4. To filter source input, configure one (or more) of the following DataSourceBlock properties:
  - a. Edit the Dataproperty to identify a different source tag.
  - b. Edit the Data Filterproperty.

**NOTE:** Data filtering applies at the time an algorithm begins processing. The ability to filter is also available in Px bindings, however, the framework may ignore any filtering configured in the binding used to view algorithm output.

## Related Links

- [Creating an algorithm \(Parent Topic\)](#)

## Example: Monitoring temperature and humidity

This task creates an algorithm to monitor temperature and humidity, including defining appropriate units, Metric or English.

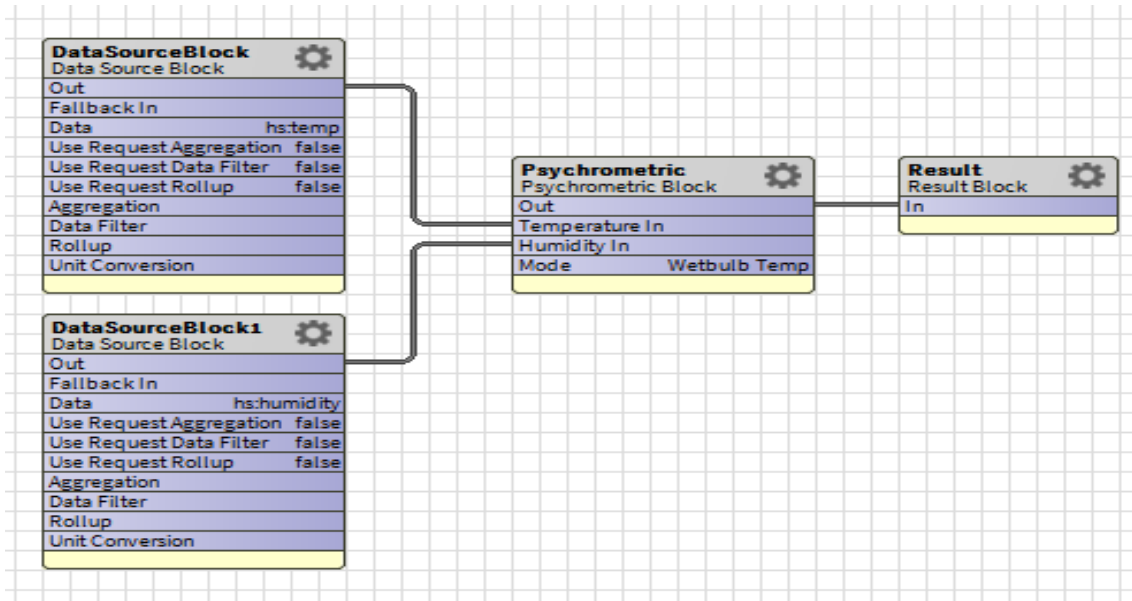
Prerequisites:

Perform the following steps:

1. Create the algorithm, give it a name, and configure its properties.

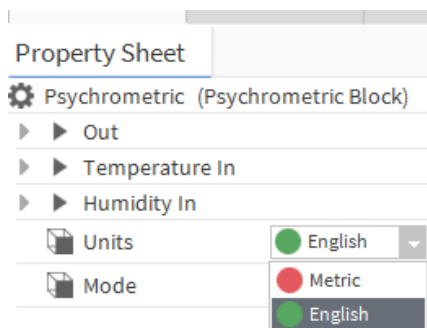
- With the algorithm in the Algorithm Manager, select the Wire Sheet view, open the analytics palette, and add logic, including two DataSourceBlocks and a Psychrometric block.

The algorithm should look something like this:



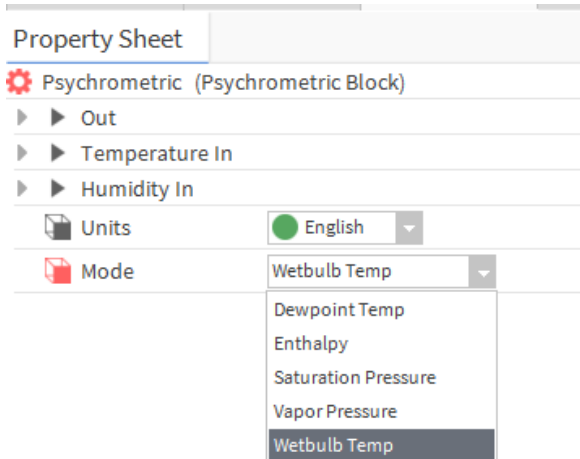
Each new algorithm contains a Result block by default.

- Define the **Data** name for the first DataSourceBlock as hs:temp, and the **Data** name for the second DataSourceBlock as hs:humidity.
- Add a Psychrometric block (under General Blocks) and select the units to use.

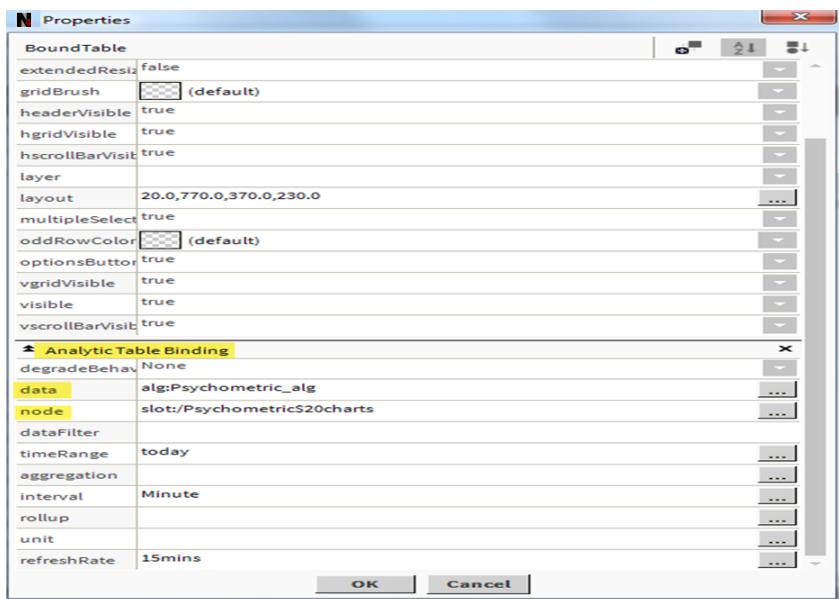


The system converts the raw data to the units you require. For information about Metric and English temperature values see the Niagara Analytics Reference manual.

- Change the Psychrometric Block Mode property to Wetbulb Temp, and click **Save**.



6. Link the output and input slots, refresh cache, and save the station.
7. To see the data, add a table binding, configure it with data, configure the node and other properties and click **OK**.



The data display in a list.

Timestamp	Value	Status	
30-Aug-17 11:03:00 AM IST	16.29	{ok}	▲
30-Aug-17 11:04:00 AM IST	16.29	{ok}	
30-Aug-17 11:05:00 AM IST	16.29	{ok}	
30-Aug-17 11:06:00 AM IST	16.29	{ok}	
30-Aug-17 11:07:00 AM IST	16.29	{ok}	
30-Aug-17 11:08:00 AM IST	16.29	{ok}	
30-Aug-17 11:09:00 AM IST	16.29	{ok}	
30-Aug-17 11:10:00 AM IST	16.29	{ok}	
30-Aug-17 11:11:00 AM IST	16.29	{ok}	▼

The values populate, but the column heading does not identify the unit of measure. Refer back to the Units and Mode you configured above.

## Related Links

- [Creating an algorithm \(Parent Topic\)](#)

## Creating an alert

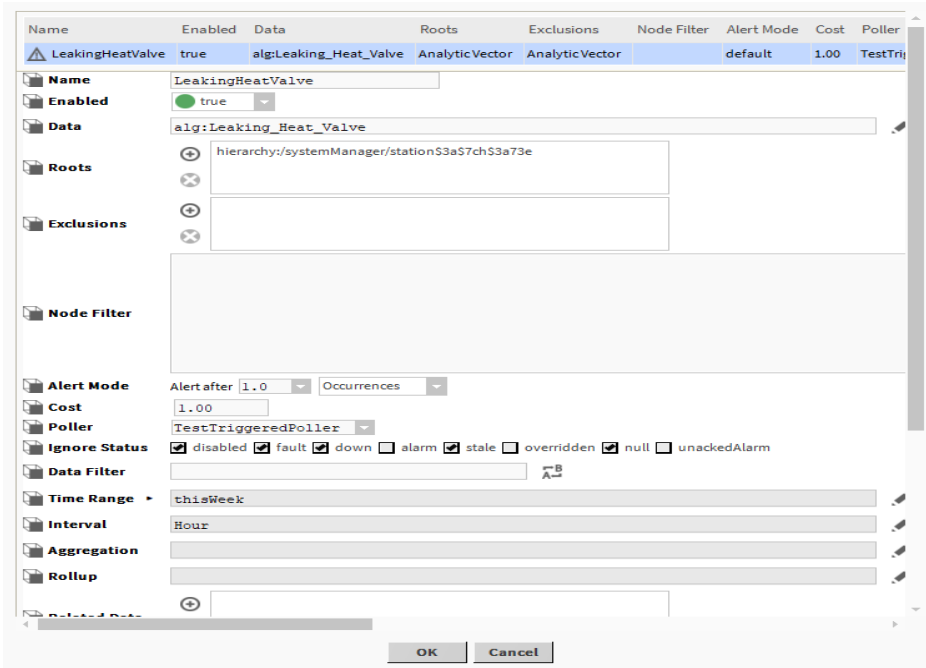
An algorithm needs an alert or the opening of a Px View or Web Chart to trigger its calculations. Alerts monitor Boolean data sources, which, when true, indicate there is an issue that requires attention. Alerts can monitor tags and tag groups like any other data, but typically alerts have framework algorithms as data sources. A poller provides the timer that runs the algorithm associated with each alert. When an alert algorithm returns true, an alert condition exists. An alert can be configured to generate an alarm. Alarms leverage the existing Niagara infrastructure for routing.

1. Double-click AnalyticService > Alerts.

The Analytic Alert Manager opens.

2. Do one of the following:
  - To create a new alert, click **New**.
  - To modify an existing alert, select the alert and click **Edit** or double-click the alert in the row.

The New or Edit window opens.



Pay special attention to these properties:

Data	Defines the algorithm that returns true or false. When true, the alert exists; when false, no alert exists.
Roots	Defines where to start running the alert. In the example, this alert runs against all devices under the Drivers/BacnetNetwork/ Global folder. This means that this alert applies to the entire station because all devices are under this folder.
Node Filter	Reduces the number of nodes to search. n:devicelimits alerts to devices. This is an appropriate place to use a NEQL query.
Alert Mode	The first drop-down list, Alert after, configures when to generate the alert. The second drop-down list identifies how many of what to use to generate the alert. Occurrences generates an alert after the selected number of the same event occurs. Seconds, Minutes and Hours define how much time must pass for an alert to be generated. This value is part of the total cost calculation.
Cost	Associates a currency value with each alert occurrence.
Poller	Defines how frequently to run the alert.
Alarm	Controls alarm generation. The default is false (no alarm generation). If you are testing, consider creating an Analytics Smart Alarm Class and send all framework alarms to this class as an easy way to differentiate between alarms coming from a standard alarms extension and the alarms coming from the framework.

The rest of the properties are described in the WEBs-N4 Analytics Framework Reference.

3. Configure properties and click **OK**.
4. After creating the alert, double-click it in the Nav tree.

The Alert Nodes View opens.

This view displays the current state of all points monitored by the alert.

5. If the view is empty, right-click the **AnalyticService** and click **Actions > Refresh Cache Full**  
If you still do not see the points in the table, click **Refresh** at the bottom of the window.

## Related Links

- Algorithms and alerts (Parent Topic)

## Editing an algorithm

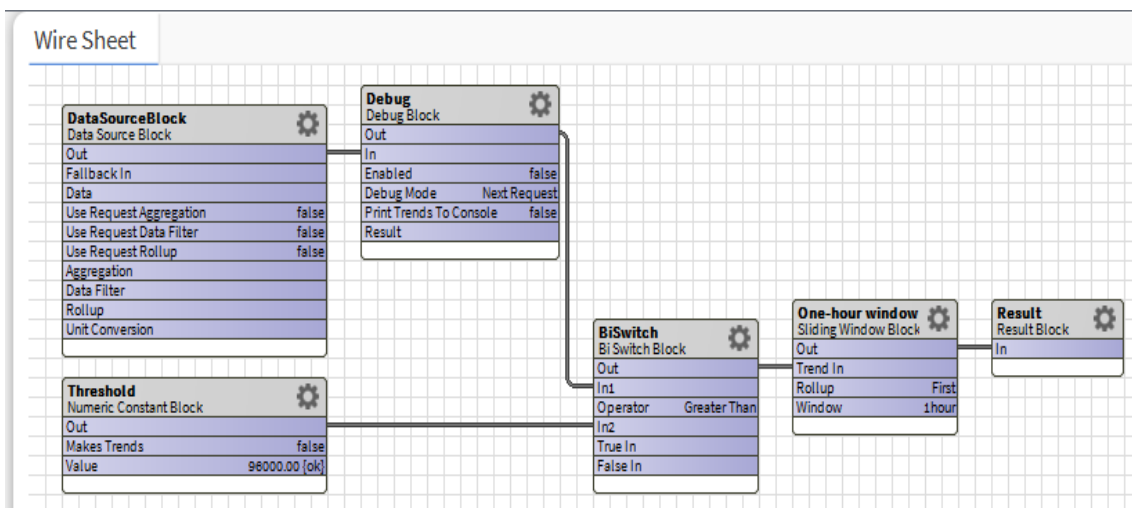
The pre-defined algorithms may be edited and changed to suit your needs.

Prerequisites: The algorithm exists.

Perform the following steps:

1. Expand the **Config > Services > AnalyticService > Algorithms** folder in the Nav tree.
2. Double-click the algorithm to edit.

The algorithm Wire Sheet opens.



3. To configure block properties, double-click the block.

Tags that begin with a: come from a specialized analytics tag dictionary. This dictionary is a standard feature of the framework. Tags that begin with hs: are from the Haystack tag dictionary. This dictionary is a standard feature of WEBs-N4.



## Related Links

- Algorithms and alerts (Parent Topic)

## Using algorithm results in standard logic

The output from an algorithm can feed back into a station to optimize energy, reduce faults, or perform any other task based on historical and current conditions. For example, if an algorithm determines that a hot water valve is open while room temperature is high for an hour or more, standard wire sheet logic can close the hot water valve. This type of closed-loop analytics uses an analytic point from the analytics palette to add an analytics proxy extension anywhere in a station.

Prerequisites: Standard logic already exists.

1. Open the analytics palette and expand the Points folder.
2. Expand the **Drivers > Network node**, right-click a Points node, and click **Views > Wire Sheet**.
3. Drag an analytics NumericPoint to the Wire Sheet.
4. This action drags an object from the analytics palette to a non-analytics wire sheet, a powerful feature of the software.
5. Expand the proxy extension.

By default, a new proxy extension is in fault. This is because it does not know what data to use and what node to run on. It has only the default poller.

6. Set Data to an algorithm, configure the Node property to tell the software where in the station to begin running, and define the Poller.
7. Continue setting up your standard logic blocks.

## Related Links

- Algorithms and alerts (Parent Topic)

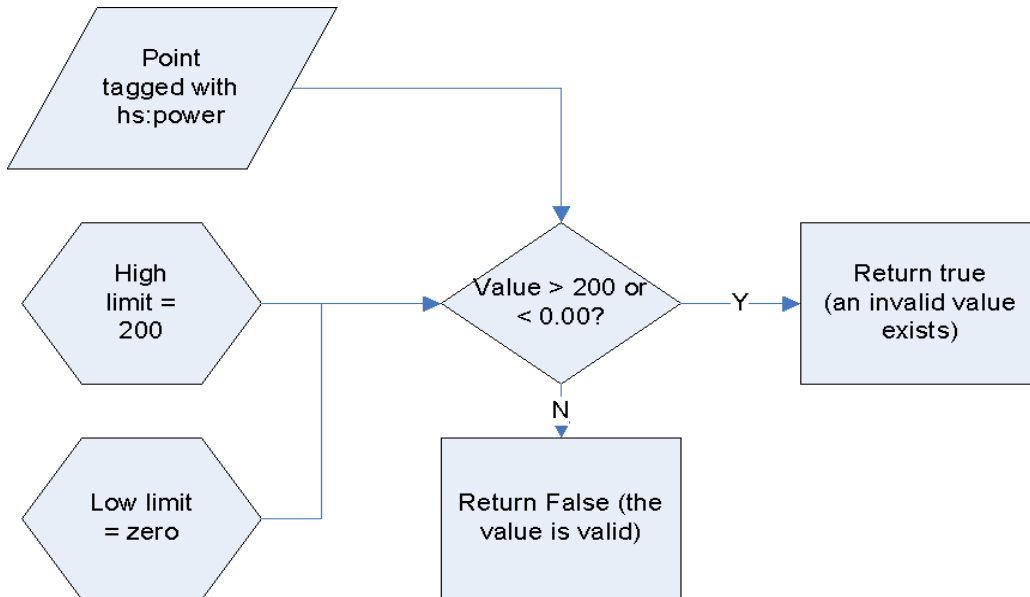
## Algorithm for removing unwanted data

From time to time, a point does not report a plausible realistic value. For example, a value may get corrupted coming over the network. This erroneous value could be a very high or a negative number. Such data can render Px views and charts meaningless, especially if you are using aggregation or rollup, or if you are passing the data to another function. Typically, an organization may have to send the data outside of the system to cleanse it or manually modify the histories. This topic documents two algorithms. The first filters out corrupt data. The second not only filters the data out, but also changes the data to a valid value.

## Demand Range Filter Algorithm

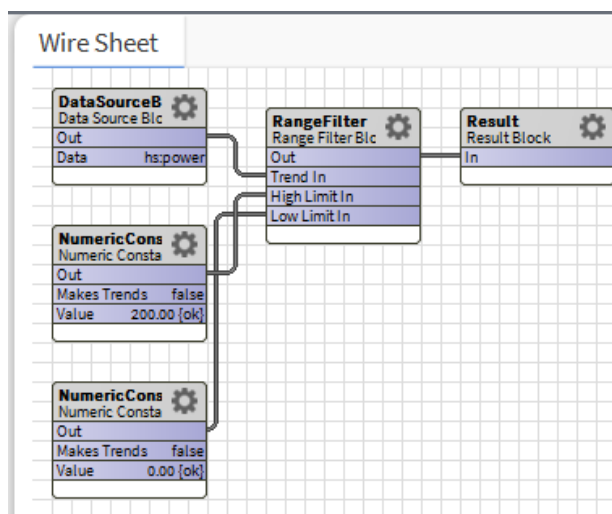
This algorithm determines if a value falls within a valid range.

Figure 12 Demand range filter algorithm flowchart



## Wire sheet view

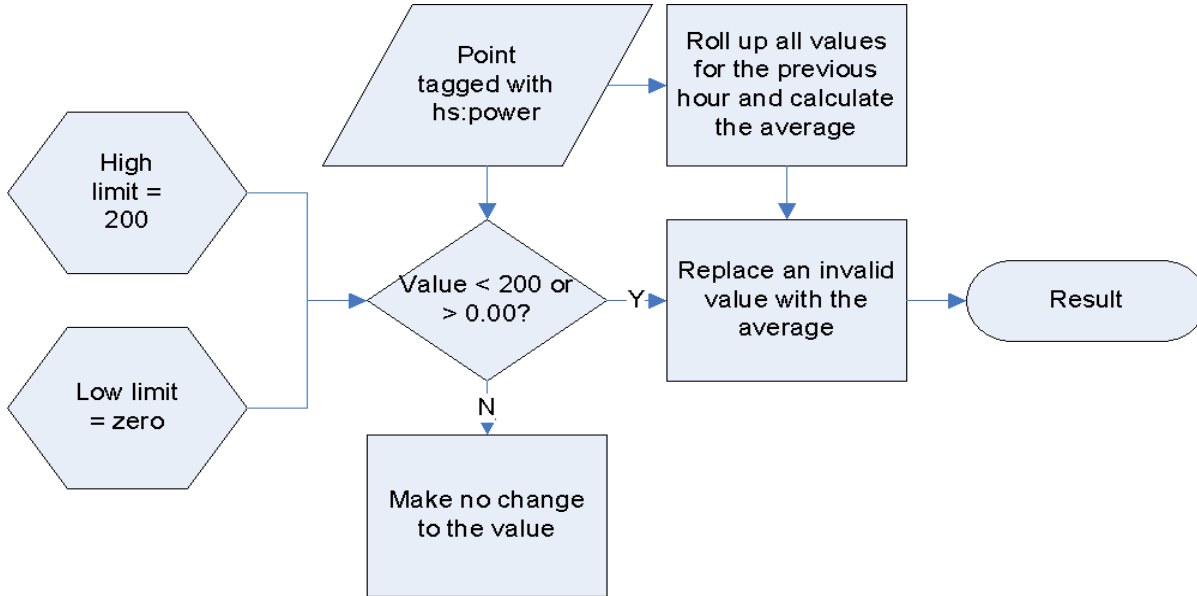
Figure 9 Demand range filter used to cleanse erroneous data



### Demand Replace Bad Data

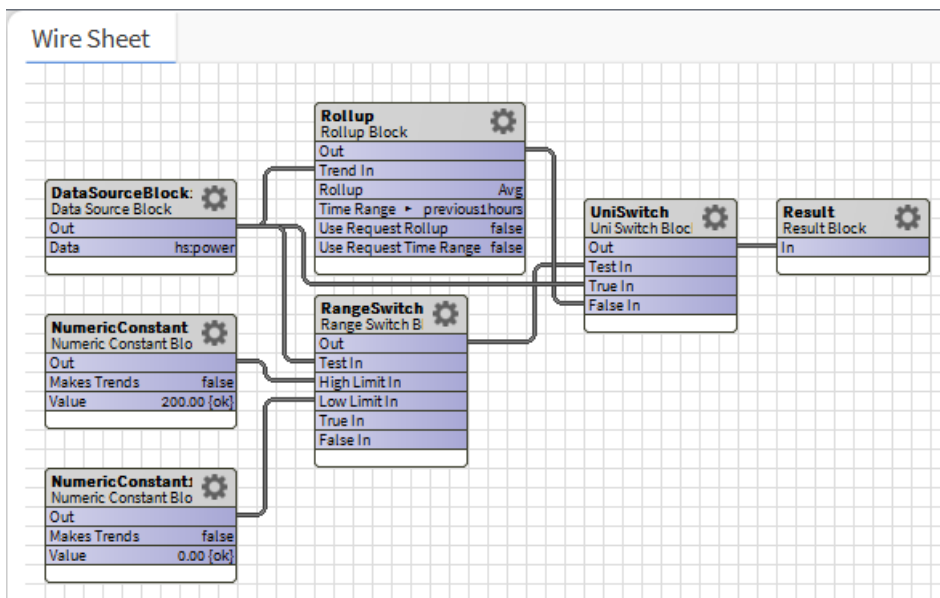
This algorithm replaces invalid data with the average of data values over the previous one hour.

Figure 10 Demand Replace Bad Data algorithm flowchart



### Wire sheet view

Figure 11 Demand Replace Bad Data algorithm



### Related Links

- Algorithms and alerts (Parent Topic)

## Viewing an alert in the alarm console

Alerts can generate alarms, which appear in the alarm console.

Prerequisites: An alert has been configured to display the source node.

Perform the following steps:

1. Access the alarm console.

The alarm console opens.

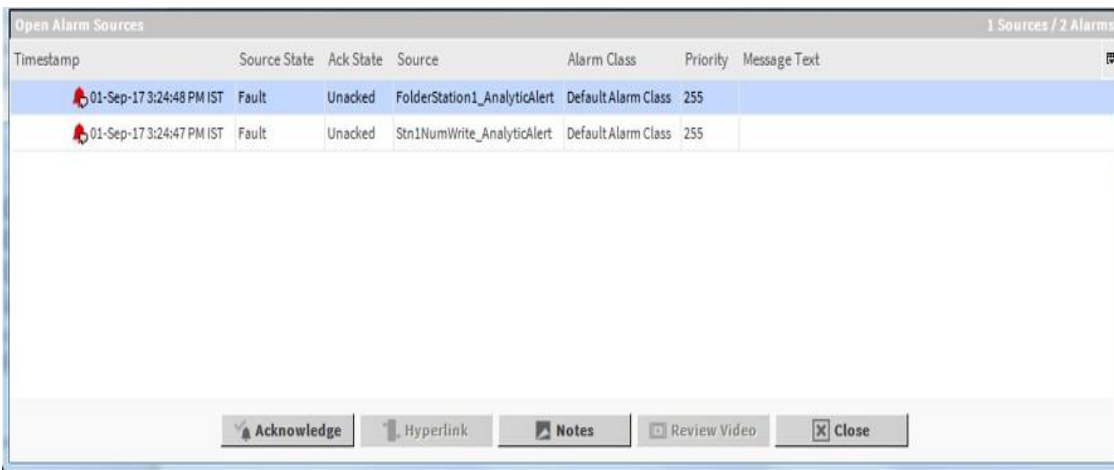


The screen capture shows an alarm that was generated by an alert. The alarm console’s Source column identifies the source of the alert: FolderStation1 as a node in the Nav tree and AnalyticAlert describes what happened. The Message Text column displays any notes associated with the alert.

If the alert was defined incorrectly, the default BFormat (%node.navName%\_%alert.name%) displays in the Source column instead of the alert source information.

2. Double-click the alarm record generated by the alert.

The Open Alarm Sources view opens.



This view filters the alarms to display only alarms generated by alerts.

3. To display additional information or to add your own comments, select the alarm row and click the **Notes** button.

### Related Links

- [Algorithms and alerts \(Parent Topic\)](#)

# Data visualization

Charts take large volumes of detailed data and make them usable to detect trends and solve problems. A set of pre-defined charts can work together to accomplish these goals. You can also create your own charts.

## Related Links

- Rollup and aggregation
- Bindings
- AnalyticWebTable
- Changing rendering limitations
- Automatic conversion of metric values in tables
- Pre-defined charts
- Reports

## Rollup and aggregation

Rollup and aggregation are features the framework uses to combine data for meaningful analysis. A rollup combines multiple adjacent rows of historical data into a single row. An aggregation combines current data values from multiple disparate data sources. Rollups and aggregation control how an algorithm queries the database for input data.

A rollup uses a function (sum, average, etc.) to combine historical data. It allows you to view dissimilar histories at common intervals. For example, if one point samples data at five-minute intervals, and another samples at 10-minute intervals, you can use rollup to calculate and compare their values at, say, 15-minute intervals.

An aggregation searches the data model tree for all points tagged with a specific tag, and uses a function (sum, average, etc.) to combine their current values into a single value. The default method for combining is to display the first value that the framework finds.

### Both features share the same set of functions

Function	Description
And	Logical “and”. Only for Boolean values.
Avg	Calculates the sum divided by the count. Use it only with numerics.
Count	Returns the number of values.
First	Returns the initial value in the set.
Last	Returns the final value in the set.
Load Factor	Returns the average value divided by peak (Max) value.
Max	For numerics, this is the greatest value. For Booleans, false = 0 and true = 1. For enums, this returns the greatest ordinal.

Function	Description
Median	Returns the value in the middle of a sorted combination—the number that separates the higher half from the lower half.
Mean	Returns the arithmetic mean (average) of the values in the data source(s).
Min	For numerics, this is the smallest value. For Booleans, false = 0 and true = 1. For enums, this returns the smallest ordinal.
Mode	Returns the statistically most frequently occurring number in the combination.
Or	Logical “or”. Only for Boolean values.
Std Dev	Returns the standard deviation of the values in the combination.
Sum	Adds all values together. Use only with numerics.

### When the difference between two values matters

Some of the functions used by aggregation and rollup may not make sense for certain values. For example, the sum of KWh for a group of points yields an ever-increasing meaningless number. In another example, summing air temperature readings does not yield a useful number. You may be more interested in the delta (change) that occurs between the current, real-time value and the most recently-saved historical value. To have the system calculate this value, make sure that Totalized is set to true in each point’s Property Sheet.

### Best practice

As you configure the visualization of values and trends, experiment with the rollup and aggregation properties on the binding. If you get a result you do not expect, consider the settings for these properties.

### Related Links

- [Data visualization \(Parent Topic\)](#)

## Bindings

The bindings display individual values, combine (roll up) values from the same point into meaningful intervals, and aggregate separate current values.

The bindings include:

- **Analytics Value Binding:** This binding displays a single value for a point, which is either the real-time, current value or the most recent value stored in the history file. It can also aggregate the current value of disparate data sources of the same type under the selected node. This binding does not use a time range.
- **Analytics Rollup Binding:** This binding returns a single, historical value based on a time range. It uses functions (min, max, avg, sum, etc.) to roll up the historical data.
- **Analytics Web Chart Binding:** This binding, programmed in HTML5 is designed to visualize historical data on a chart running in a browser.
- **Analytics Table Binding:** This binding displays historical information in tabular form. It extends the standard bound table widget.
- **Analytics Web Rollup Binding:** This binding is added automatically to the Ranking Chart. It handles the roll-up of data based on how you configure the chart's **Interval**, **Time Range** and **Rollup** properties.

### Related Links

- [Data visualization \(Parent Topic\)](#)

## AnalyticWebTable

Tables configure any analytic binding to view data in a tabular format. The framework dynamically forms the table columns based on the return values from the ord or binding. You drag a table from the analytics palette to a Px view.

Figure 12 Sample table configured with an analytic web chart binding

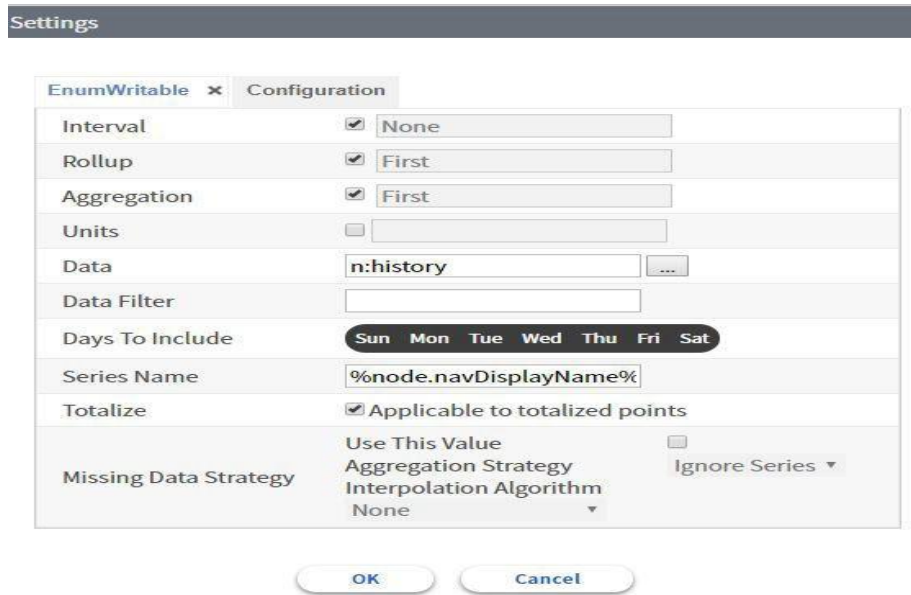
tamp	AHU-1-history Value	AHU-1-history Status
18 12:00:00.000 AM UTC+05:30	21.600000381469727	{ok}
18 12:15:00.000 AM UTC+05:30	24.299999237060547	{ok}
18 12:30:00.000 AM UTC+05:30	24.299999237060547	{ok}
18 12:45:00.000 AM UTC+05:30	23.299999237060547	{ok}
18 1:00:00.000 AM UTC+05:30	23.299999237060547	{ok}
18 1:15:00.000 AM UTC+05:30	20.799999237060547	{ok}
18 1:30:00.000 AM UTC+05:30	22.5	{ok}
18 1:45:00.000 AM UTC+05:30	19.700000762939453	{ok}
18 2:00:00.000 AM UTC+05:30	18.700000762939453	{ok}
18 2:15:00.000 AM UTC+05:30	15.199999809265137	{ok}
18 2:30:00.000 AM UTC+05:30	15.899999618530273	{ok}
18 2:45:00.000 AM UTC+05:30	17.200000762939453	{ok}
18 3:00:00.000 AM UTC+05:30	16.899999618530273	{ok}
18 3:15:00.000 AM UTC+05:30	16.5	{ok}
18 3:30:00.000 AM UTC+05:30	16.5	{ok}
18 3:45:00.000 AM UTC+05:30	112	{ok}
18 4:00:00.000 AM UTC+05:30	90.30000305175781	{ok}

### Settings window



Clicking the wrench icon opens the Settings window with which to configure table properties.

Figure 13 Settings window



For history data, the table shows the timestamp history value and status. Pagination and search features are available for the table.

Property	Value	Description
Interval	drop-down list	Defines the frequency at which the framework saves a history record.
Rollup (property) or rollup (ORD parameter)	check box (if optional, and) drop-down list or ORD parameter (rollup=option) (when configured in the data definition, defaults to First, when configured elsewhere, defaults to the value as defined in the data definition)	<p>Defines the mathematical function to be used to combine data from a single source.</p> <p>If rollup is not enabled in the binding/settings window, the rollup value configured in the data definition applies to all chart bindings, reports and tables.</p> <p>And returns the logical “and” of Boolean values.</p> <p>Avg returns the statistical mean, which is determined by calculating the sum of all values and dividing by the number of values.</p> <p>Count returns the total number or quantity of values in a combination. If you request this value on a binding in a PX view, the system counts the number of values based on the properties defined by the data source block and the algorithm’s property sheet.</p> <p>First returns the first value in the combination. This generates the fastest result.</p> <p>Last returns the last value in the combination.</p> <p>Max returns the highest value in the combination.</p>

Property	Value	Description
		<p>Median returns the value in the middle of a sorted combination—the number that separates the higher half from the lower half.</p> <p>Min returns the lowest value in the combination.</p> <p>Mode returns the statistically most frequently occurring number in the combination.</p> <p>Or returns the logical “or” of Boolean values. Range returns the statistical difference between the largest and smallest values in the combination.</p> <p>Sum adds together all values in the combination resulting in a single value.</p> <p>Std Dev calculates the standard deviation of the values in the combination.</p> <p>Load Factor calculates the average divided by peak (Max) value.</p>
<p>Aggregation (property) or aggregation (ORD parameter)</p>	<p>check box (if optional, and) drop-down list (defaults to First) or ORD parameter (aggregation=option)</p>	<p>Defines the mathematical function to use to combine data from multiple data sources.</p> <p>If aggregation is not enabled in the binding/settings window, the aggregation value defined in the data definition applies to all chart bindings, reports and tables.</p> <p>And returns the logical “and” of Boolean values.</p> <p>Avg returns the statistical mean, which is determined by calculating the sum of all values and dividing by the number of values.</p> <p>Count returns the total number or quantity of values in a combination. If you request this value on a binding in a PX view, the system counts the number of values based on the properties defined by the data source block and the algorithm’s property sheet.</p> <p>First returns the first value in the combination. This generates the fastest result.</p> <p>Last returns the last value in the combination.</p> <p>Max returns the highest value in the combination.</p> <p>Median returns the value in the middle of a sorted combination—the number that separates the higher half from the lower half.</p> <p>Min returns the lowest value in the combination.</p> <p>Mode returns the statistically most frequently occurring number in the combination.</p> <p>Or returns the logical “or” of Boolean values. Range returns the statistical difference between the largest and smallest values in the combination.</p> <p>Sum adds together all values in the combination resulting in a single value.</p> <p>Load Factor returns the average value divided by peak (Max) value.</p>

Property	Value	Description
		Std Dev returns the standard deviation of the values in the combination.
Units	drop-down list of units of measure	Defines the unit of measure for the data gathered from each point.
Data (property) or data (ORD parameter)	tag or algorithm name	Specifies the tag used to retrieve data. This tag can be from the Haystack dictionary, Niagara dictionary or any other custom tag dictionary. Instead of a tag, this value can specify an algorithm. This is how output from one algorithm becomes the input data source to another algorithm. The prefix for algorithms is alg:.
Data Filter (property) or dataFilter (ORD parameter)	optional NEQL query (property) or ORD parameter (dataFilter=query)	Identifies data sources in the subtree of the request node. When a predicate accepts a node that contains the desired data, the system does not search the node's subtree for additional values (including the root node).
Days to Include	days-of-the-week selector	Chooses the days of the week to include in the table.
Series Name (property) or seriesName (ORD scheme)	BFormat (property) or text string (ORD parameter) with the following syntax: %node.navDisplayName-data.name%	<p>Defines either a display name (text string) and a BFormat that identifies the node responsible for the charted data. As a BFormat, it can be configured, for example, to search for an ord, etc.</p> <p>Node identifies a name in the nav tree. navDisplayName is a text string. It lets you to add a descriptive name associated with this binding. Data.name is a tag that identifies the points to use. If you enter an invalid value, the system displays, "Problem with seriesName BFormat," and logs a message in the station console.</p> <p>The name you enter here displays as the legend and tool tip in the Web chart. If left blank, no legend displays for the tool tip.</p>
Missing Data Strategy or Algorithm, Use This Value	check box	Enables and disables missing data interpolation for the current value.
Missing Data Strategy, Aggregation Strategy	drop-down list	<p>Selects the missing data aggregation strategy, which defines how to handle data in a series when even a single record for an interval is missing.</p> <p>Ignore Point tells the system to ignore any missing records and aggregate the values in the existing records.</p> <p>Ignore Series tells the system to ignore the entire series if the record for even one interval in the series is missing.</p>

Property	Value	Description
Missing Data Strategy or Algorithm, Interpolation Algorithm	drop-down list	Selects the missing data interpolation algorithm, which defines the value to replace a missing value. Linear Interpolation replaces a missing value by linearly interpolating the missing value. K-Nearest Neighbor is for numeric, enum and Boolean records. This strategy replaces a missing value by calculating the majority value recorded for the item's nearest neighbors.
Missing Data Strategy, K Value	number	Indicates the number of neighbors to a missing data item that the interpolation algorithm should include in its calculation.

### Related Links

- [Data visualization \(Parent Topic\)](#)

## Changing rendering limitations

The number of records included in a chart or report relates directly to system speed and performance. By default, the framework limits this number. The default values for each chart and report were set based upon extensive testing in the laboratory. The configuration of these values is hidden so that they cannot be changed unintentionally.

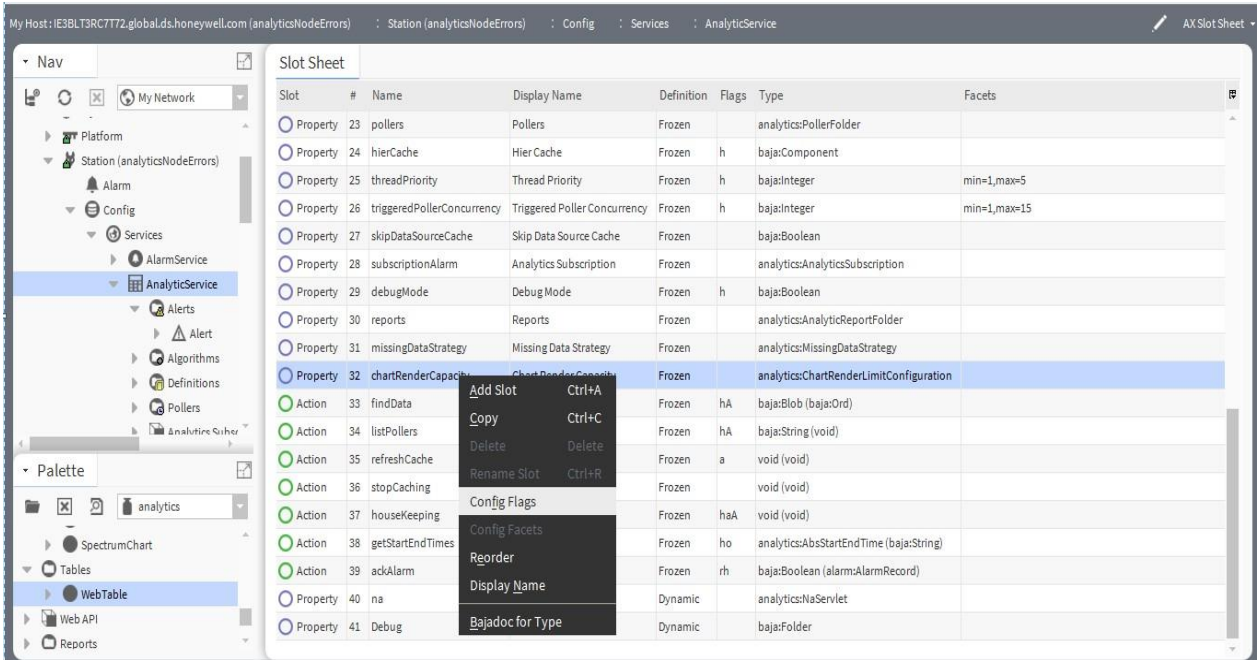
Prerequisites: You understand the impact on framework performance of increasing the number of records rendered on charts and reports. You are working in WEBStation.

**CAUTION:** Increasing the number of records rendered can slow system performance to an unacceptable level.

Perform the following steps:

1. Expand Config > Services, click the AnalyticService, and select AX Slot Sheet from the drop-down list in the upper right corner of the view.

The slot sheet opens.



2. Scroll down, right-click the chartRenderCapacity property, and click to remove the hidden check mark.

**Related Links**

- Data visualization (Parent Topic)

**Automatic conversion of metric values in tables**

When the system converts one unit, such as KW, to another, such as W, it automatically shortens the unit’s metric value and adds a suffix. This improves the readability of large data and data with multiple decimal places in report tables.

For example, a value of 16100 converts in the table as 16.10k.

Figure 14 A table with converted values

Time Of Day	NumericWritable (W per degree day)
00:00:00	16.10k
06:00:00	62.45k
12:00:00	76.53k
18:00:00	26.53k

## Number conversion

These abbreviations are for values that are greater than 1,000.

Suffix symbol	Name	Positive orders of 10
T	trillion	1,000,000,000,000
G	billion	1,000,000,000
M	million	1,000,000
k	thousand	1,000

## Decimal number conversion

These abbreviations are for values that are less than 1.

Suffix symbol	Name	Negative orders of 10
m	thousandth	0.0001
μ	millionth	0.000 001
n	billionth	0.000 000 0001
p	trillionth	0.000 000 000 001

## Related Links

- [Data visualization \(Parent Topic\)](#)

## Pre-defined charts

The pre-defined charts work both in WEBStation and a browser.

Here are some needs a Facility Manager, Data Center Manager, or Utility Manager may have and which chart to use to meet each need:

The Need (what you might want to do)	The Recommended Chart to use
<ul style="list-style-type: none"> <li>• Combine data from separate nodes into single figures.</li> <li>• View spikes in power usage at a point in time.</li> </ul>	AggregationChart combines values across a selected time range showing the average for each value. For example, if you select This Week, the chart reports the average for each day of the week.
<ul style="list-style-type: none"> <li>• View power spikes grouped by day.</li> <li>• Evaluate the average monthly temperature in a storage area for a period of a year.</li> </ul>	AverageProfileChart plots a graph that shows the average of each value for all bindings grouped by time.

The Need (what you might want to do)	The Recommended Chart to use
View equipment status (when was the power on and when was it off?)	EquipmentOperationChart shows when a piece of equipment is powered on and off, providing insight into equipment operating patterns.
<ul style="list-style-type: none"> <li>• Compare the same value across multiple locations.</li> <li>• Determine the energy required when adding additional equipment.</li> </ul>	RankingChart uses vertical bars to compare binding values from lowest to highest (left to right).
Determine for how long a value was at a specific level (high/ low), such as to view the number of hours that a generator generated specific kilowatts of power per hour.	LoadDurationChart plots load versus duration.
<ul style="list-style-type: none"> <li>• Drill down to the specific time when something occurred.</li> <li>• Observe temperature or pressure over a period of time.</li> </ul>	SpectrumChart uses pattern recognition techniques and color coding to illustrate multiple aggregated values obtained from the same data source. The colors on the chart quickly identify trouble spots for further investigation.

Details about each chart and a screen capture of each are in the WEBs-N4 Analytics Framework Reference

### Related Links

- [Configuring a pre-defined chart](#)
- [Creating a new Px view](#)
- [Creating a new Ux chart](#)
- [Observing patterns using the Spectrum chart](#)
- [Changing the aggregation function](#)
- [Setting up an analytic table binding](#)
- [Exporting a Px chart](#)
- [Data visualization \(Parent Topic\)](#)

## Configuring a pre-defined chart

The pre-defined charts work both in WEBStation Px Views and as well as in a browser (Web Charts). This topic provides basic instructions using framework examples.

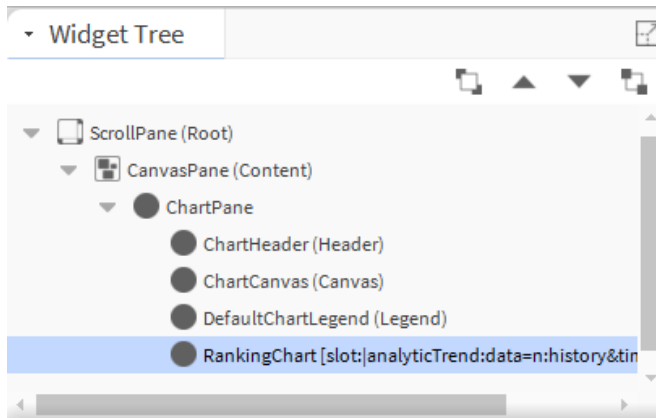
Prerequisites: The analytics palette is open.

1. Right-click your logic folder in the Nav tree and click **Views > New View**.  
The Px Editor opens.
2. Select the background canvas, and change the viewSize property to 640 x 480 pixels.

The minimum height for an Aggregation chart is 560 pixels. When set to 550 or less, the Time Range is not available for this chart.

3. Scroll down to the bottom of the analytics palette, expand the Charts folder, and drag a chart to the Px Editor.
4. Drag the chart to fill the canvas.

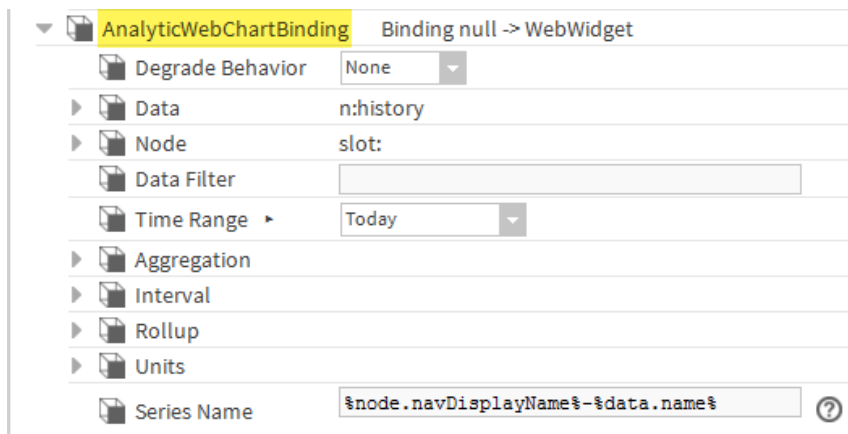
The system populates the tabs to the right of the window, one of which is the Widget Tree.



The screen capture is for a Ranking Chart.

5. Double-click the widget.

The Properties window for the chart opens.



The properties related to the framework are in the AnalyticChartBinding or AnalyticWebChartBinding section of the window.

6. For an Aggregation chart, confirm that Time Range is available.
7. Configure at least the Data property by clicking the chooser button (...) and selecting a data source tag from the drop-down list.

The remaining properties default to current values.



## Related Links


- Pre-defined charts (Parent Topic)

## Creating a new Px view


In standard Niagara, a widget (label or chart) is associated with a data source (object) using a binding. This binding defines an ORD property that identifies the location of the object that updates and animates the widget. The framework replaces the ORD with a tag, which causes the binding to collect data from all points tagged with the same tag. You set up a Px View in WEBStation to visualize framework data the same way you set up a regular Niagara Px View. This topic provides basic instructions using framework examples.

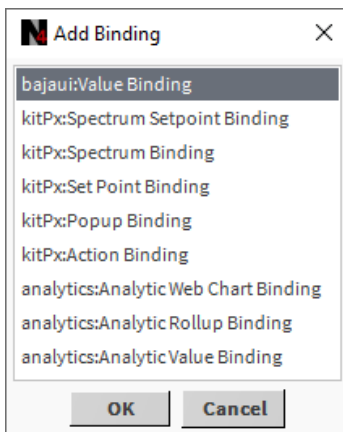
1. Right-click your equivalent of the Logic folder, click Views > New View, and assign a view name. To edit an existing view select it instead of the New View menu option.

The Px Editor opens.

2. To edit an existing view, click the Toggle View/Edit Mode button (  ) at the top of the window.
3. Right-click the canvas pane and click **New > Label** (or duplicate a similar label you already created).

The system creates an unbound label and populates the Properties tab at the bottom right corner of the Px Editor view.

4. Expand the size of the label and double-click it. The Properties window for the Label opens.
5. Click the add binding button (  ) at the top of the window. The Add Binding window opens.



6. To an unbound label, select one of the analytics: bindings and click OK.
  - The analytics:Analytic Rollup Binding combines (rolls up) multiple instances of a value into a single value.
  - The analytics:Analytic Value Binding reports the current, real-time value of the point. The system associates the binding with the label and displays the binding properties.
7. Scroll down in the Properties window until you see the binding properties for the analytics: binding you selected.

Property	Value	Control
degradeBehavior	None	Dropdown
hyperlink	null	File Finder (...)
summary	%displayName?typeDisplayName% = %.%	File Finder (...)
popupEnabled	false	Dropdown
data	n:history	File Finder (...)
node	slot:	File Finder (...)
dataFilter		
timeRange	today	File Finder (...)
aggregation		File Finder (...)
rollup		File Finder (...)
unit		File Finder (...)

The example shows the Analytic Rollup Binding properties.

The first four properties: degradeBehavior, hyperlink, summary, and popupEnabled are familiar Niagara properties. The rest are unique to the framework.

Any request (query) from the database requires you to configure at least the data source. The framework pulls data from one or more points with this tag.

The other values are optional depending on the binding request. A value binding always deals with current values. The system ignores any setting of the rollup property for a value binding. An aggregation requires the starting node when aggregating multiple point values. This node is usually a container that identifies a building or geographic location.

- Use the file finder button (  ) and component chooser to populate the data and node properties.

**NOTE:** Unlike building a Px View in standard Niagara, you do not select a point to establish an ORD. Your tag and node selections determine the point(s) to use. These properties take the place of a traditional ORD.

- Do one or both of the following:
  - If you are configuring a rollup value based on historical data, click the rollup property, enable Use This Value, select (from the drop-down list) the function (count, first, last, avg, etc.) to use to roll up the data and click **OK**.
  - If you are aggregating multiple current values, click the aggregation property, enable Use This Value, select (from the drop-down list) the function (count, first, last, avg, etc.) to use to aggregate all values into a single resulting value, and click **OK**.

Both rollup and aggregation default to their preferred settings in the data definition that is associated with each tag.

- If you started this procedure from an unbound label, scroll up in the Properties tab, right-click text and click **Animate**.

The Animate window opens with the default format set to %. In regular Niagara, you would configure this property to read %out.value%. The equivalent in the WEBs-N4 Analytics Framework is %value%.

11. Change Format to %value% and click OK.

The framework returns the value and not the point status.

If the chart includes a large number of bindings, and some bindings yield small quantities of data (very near each other on the chart), the labels may overlap and become unreadable in a PDF. To fix this problem, increase the size of the chart to allow space for label placement without overlaps.

## Related Links

- Pre-defined charts (Parent Topic)

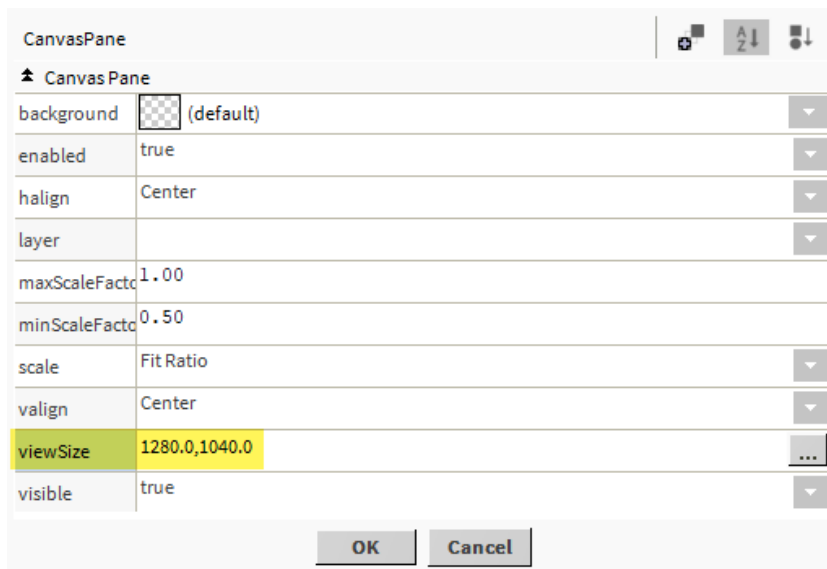
## Creating a new Ux chart

Ux charts are web charts. Programmed using HTML5 these charts are designed to work best when viewed in a browser. They also work as Px views.

1. Right-click your equivalent of the **Logic** folder and **click Views > New View** and assign a view name.

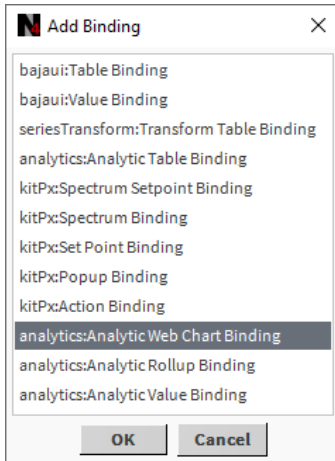
The Px Editor opens.

2. Expand the canvas size by right-clicking the wire sheet, clicking **Edit Properties** and changing viewSize to 1280 x 1040.



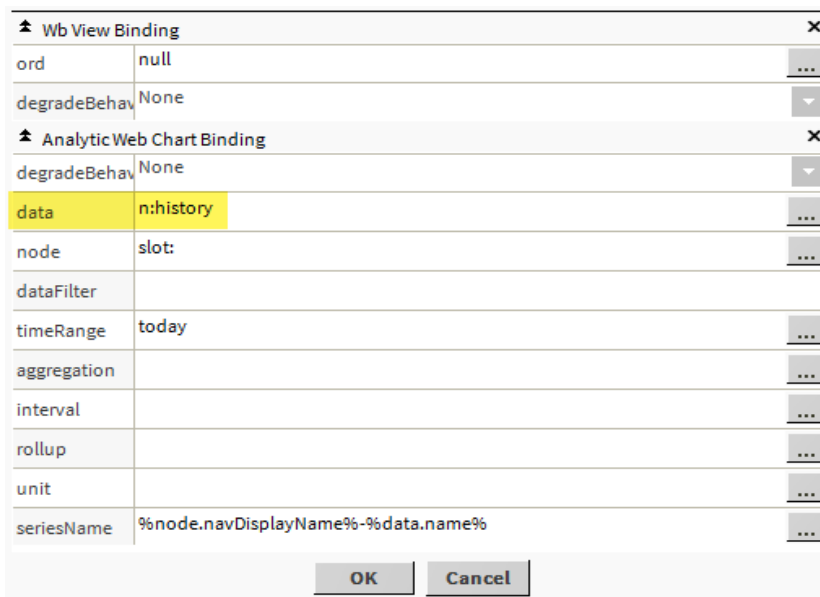
3. Open the webChart palette, drag a Chart component to the Wire Sheet, and position it for easy viewing.
4. Click the add binding button ( ) next to WebWidget in the Properties pane.

The Add Binding window opens.



5. Select analyticsAnalytic Web Chart Binding and click **OK**.
6. Double-click the chart component.

The Properties window opens.



7. Define the data source (data property). Leave other properties set to their default values. The table displays all values plotted throughout the day (today is the default interval).
8. Using the drop-down list at the top of the chart, change the interval to Last Week, Last Month, etc.
9. Go back to the binding and change the node to a specific schedule, then add another Web Chart binding and configure its node for a second schedule.

Two bindings demonstrate how to configure multiple plots on the same graph. For example, you could plot values for two different networks or two different buildings on the same graph. This would work for comparing energy usage or possibly space temperature in a couple of different rooms.

Using the Settings icon (⚙️) at the top of the chart you can configure the same properties that appear on the binding property sheet.

### Related Links

- Pre-defined charts (Parent Topic)

## Observing patterns using the Spectrum chart

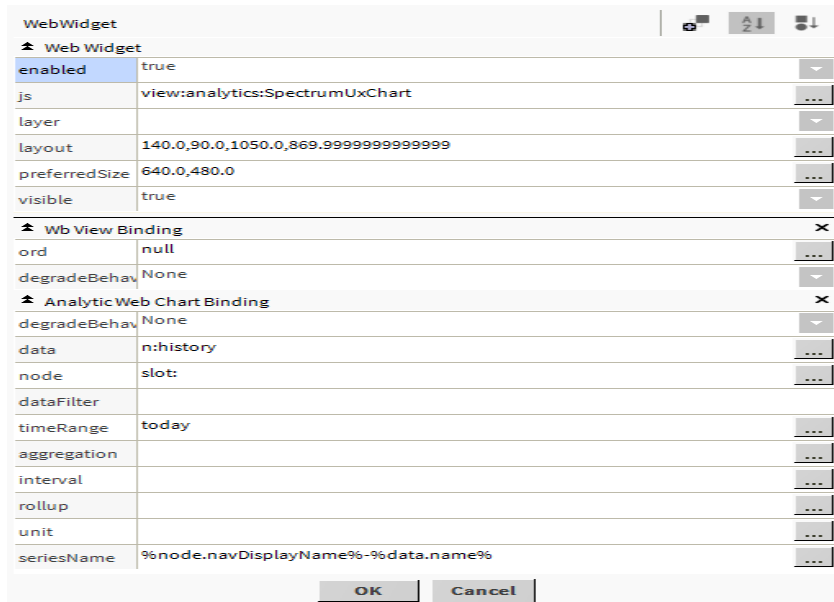
This Ux chart provides a powerful tool that converts raw data into visual patterns, which you can use to identify subtle problems before they develop into major system issues. This chart works with historical data. The procedure for configuring this chart is similar to those for configuring the other charts.

Prerequisites: The system has collected enough historical data to produce a meaningful chart. WEBStation or your browser is open and connected to a station, most likely a Supervisor station.

1. Open the analytics palette and expand the Charts folder.
2. Open a new Px view.
3. Drag the Spectrum Chart from the palette to the Px view.
4. Double-click the chart or right-click the chart and click Edit Properties.

You can change the values plotted by manipulating the chart’s wire sheet properties, or by right-clicking the chart and using the properties window.

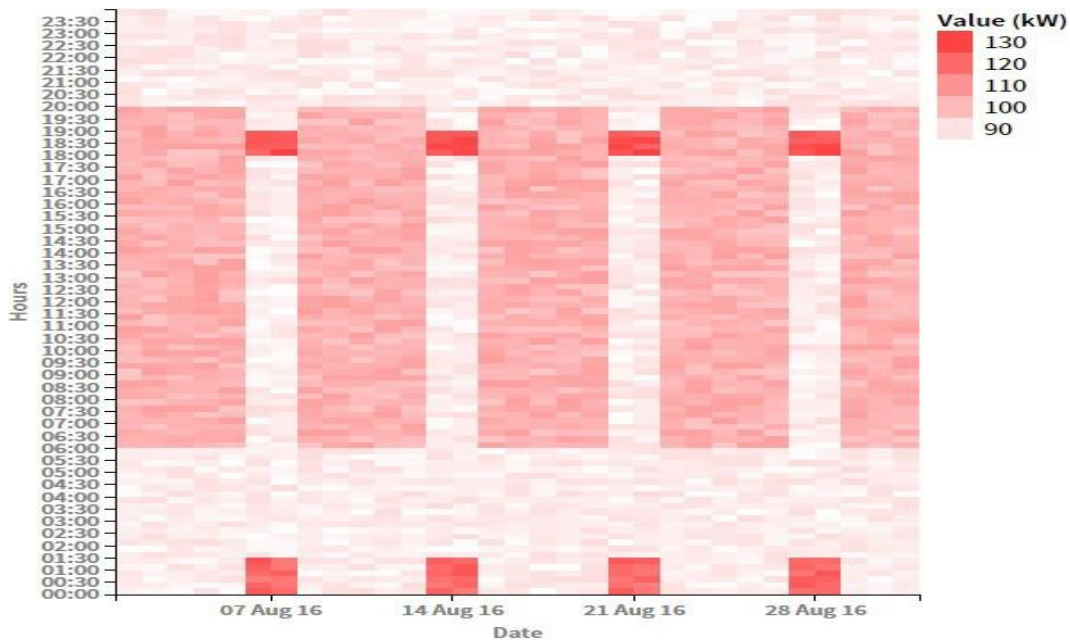
The Properties window for the chart opens.



5. Configure properties and click OK.

**NOTE:** For best results, configure this chart with at least a height of 480 Abs or greater.

The framework renders the chart.



This chart shows last month's demand data. Notice that between midnight and 1 am as well as from 18:00 to 19:00 (6–7 p.m.), power consumption was higher than expected for each of the four days reported. This could be an anomaly or it could indicate a condition that requires further investigation. Such a spectrum chart allows an energy analyst to quickly spot a potential problem.

- To plot a pattern for another point, drag the point from the Nav tree to the chart.

## Related Links

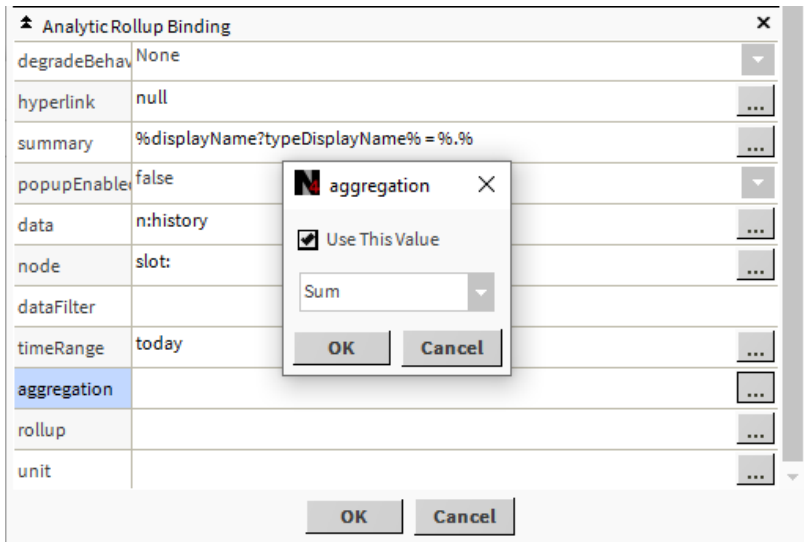
- Pre-defined charts (Parent Topic)

## Changing the aggregation function

The default aggregation function displays the first value that the framework finds.

Prerequisites: A Px view exists that uses aggregation to combine real-time values.

- Open an existing Px View.
- Double-click the chart widget.  
The Properties window opens.
- If necessary, scroll down in the window to the Analytic binding properties.
- Click the chooser button (**...**) to the right of the aggregation row. The aggregation window opens.



5. Enable Use This Value, choose a value other than First from the drop-down list, click **OK**, and click **OK** again to close the Properties window.


## Related Links

- Pre-defined charts (Parent Topic)

## Setting up an analytic table binding

Tables by their nature are historical. Setting up an analytic table binding assumes that the points you are using have histories with them. Using a bound table is a good way to troubleshoot problems with bound labels because in a table you see the historical data that the framework is processing.

Prerequisites: The points you are using have history extensions.

1. Open the bajoui palette, expand the Widgets folder and drag a BoundTable widget to the Wire Sheet.
2. Double-click the widget, click the Px binding button (  ) select an analytics Table Binding, and click **OK**.
3. Scroll down to the Analytic Table Binding section of the window and enter the tag for data that identifies the data source.
4. Change any other properties and click **OK**. The framework renders the table.

A word about intervals. The default interval is 15 minutes for histories. If your histories are captured at different intervals, change the interval property to either your largest interval or to a value that is possibly greater. For example, if the history extension for one point captured data at 10 minute intervals and another history extension captured at 15 minute intervals, you would configure your table binding at a 30-minute interval value, or perhaps at an hour. It is important to make sure that you are comparing apples with apples.

A word about rollup and aggregation. Both default to First.

For a given point, a rollup reports the sum of all historical values logged in the database during the defined interval. A rollup of First reports the rolled up value for the first interval in the series. A rollup of First does not actually come into play until the database contains more than one interval of historical data.

For a given set of points, an aggregation reports the sum all the current values for each point in the set. An aggregation of First means that the framework reports the value for the first point in the set.

You see the result of the change in the table. Rollup does not actually impact the results until the interval changes.

## Related Links


- Pre-defined charts (Parent Topic)

## Exporting a Px chart

This procedure explains how to export a Px chart with a Wb View Binding.

Prerequisites: The chart is open in a Px view.

Perform the following steps:

1. At the top left of the view, click the Export button (  ) The Export Wizard opens.
2. As a minimum, define the **File Name** for the chart and click **OK**.

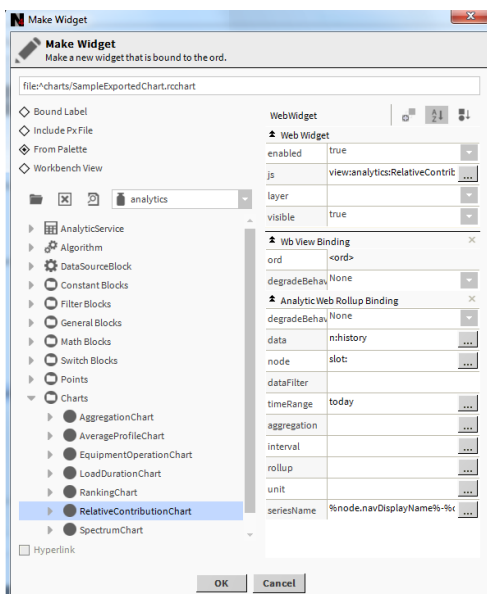
The system saves the chart either in the station (default) or another location.

3. Create a new view.

See the topic titled Creating a New Px View.

4. Add the chart you exported to the new view by locating the chart under the **Station > Files > charts folder**, and dragging it to the newly-created view.

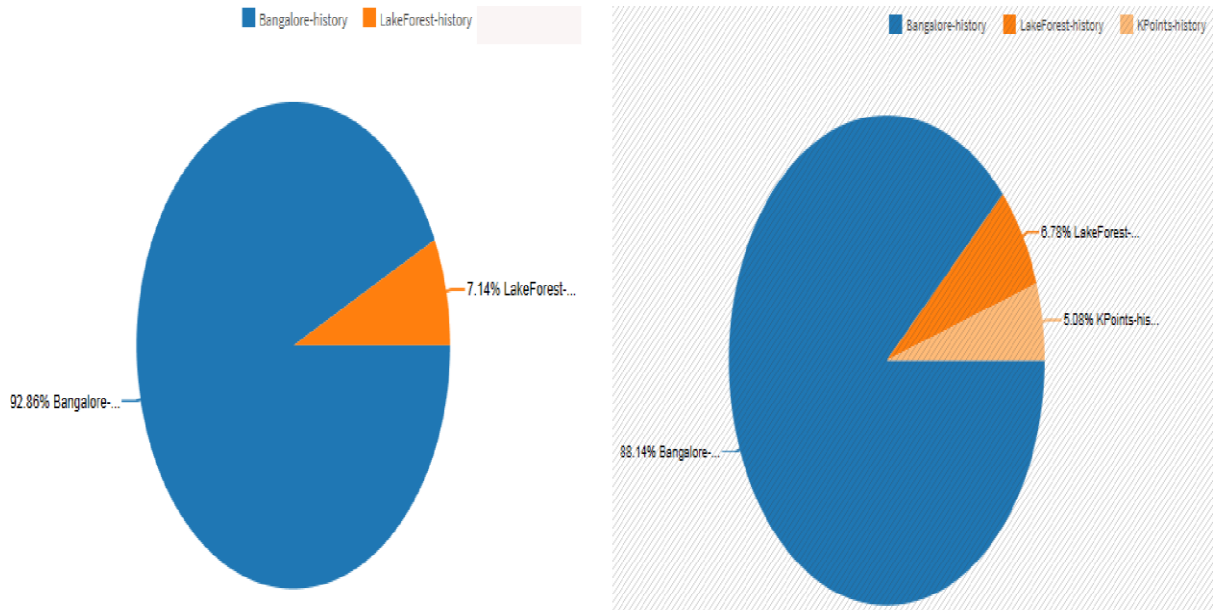
The Make Widget wizard opens.





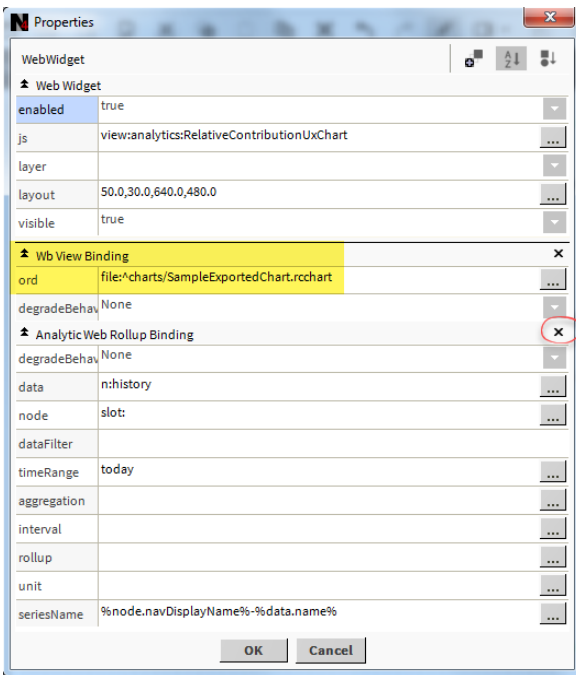
- 5. Select the chart type, and click **OK**.

The view opens with the chart, which includes a history binding.



This example uses a relative contributions chart. The version on the left shows how the chart appeared before exporting it at the beginning of this procedure. The version on the right is how the chart appears after dragging it to the new Px view. The extra history wedge (binding) is not needed.

- 6. To remove the extra binding, double-click on the chart. The edit mode for the chart opens.



- 7. Since we are using a Wb View Binding for this chart, delete all other bindings by clicking the X to the right of the extra binding(s).

8. To return to normal view, click **OK**.

The chart displays.

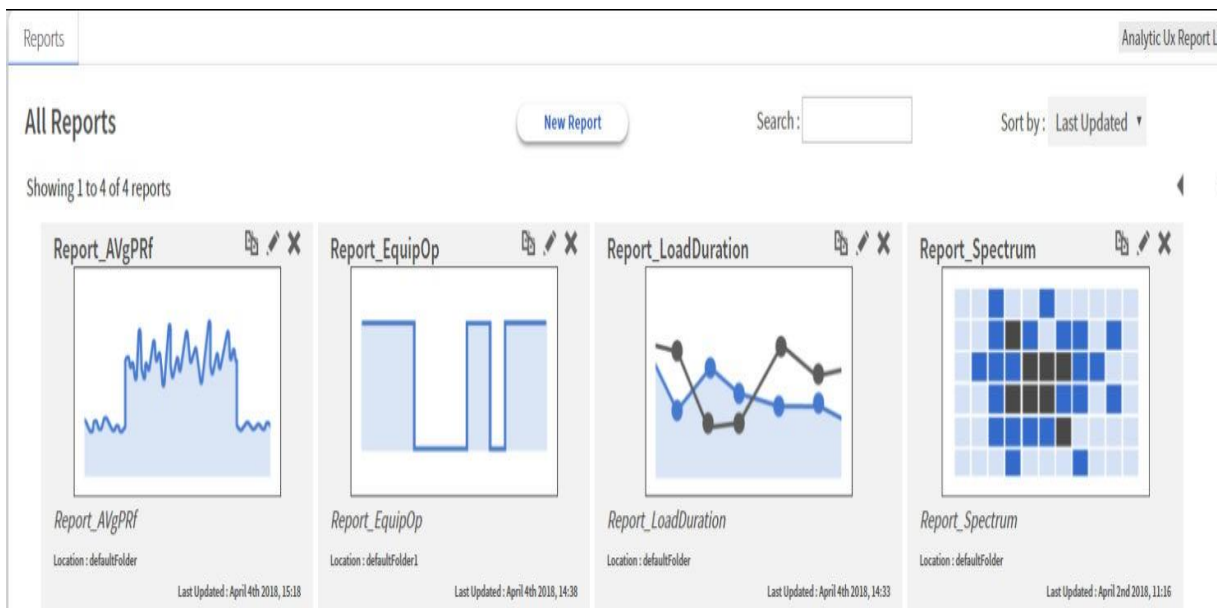
## Related Links

- Pre-defined charts (Parent Topic)

## Reports

Reports make sense of the data collected from the building automation equipment. To view each report click its thumbnail.

Figure 14 All Reports thumbnails



The framework intelligently selects the interval to use for each report based on the optimal number of records for each report. You can change the interval by clicking the **Advanced** button in the report editor.

## Related Links

- Creating Ux reports
- Managing Ux reports
- Creating a dashboard
- Configuring a report or a dashboard
- Normalizing energy consumption values based on floor area
- Normalizing energy consumption values based on degree-day temperature
- Printing a report
- Data visualization (Parent Topic)

## Creating Ux reports

Ux (User Experience) reports are designed to be human-friendly and easy to use. Where Px reports can seem complicated to configure, Ux reports share a centralized home page with large visual tiles and identifiable icons. From the single centralized view you can create, edit, delete and clone reports as well as search and sort report lists. Only reports created using this view are visible in this view. Px reports are not visible here.

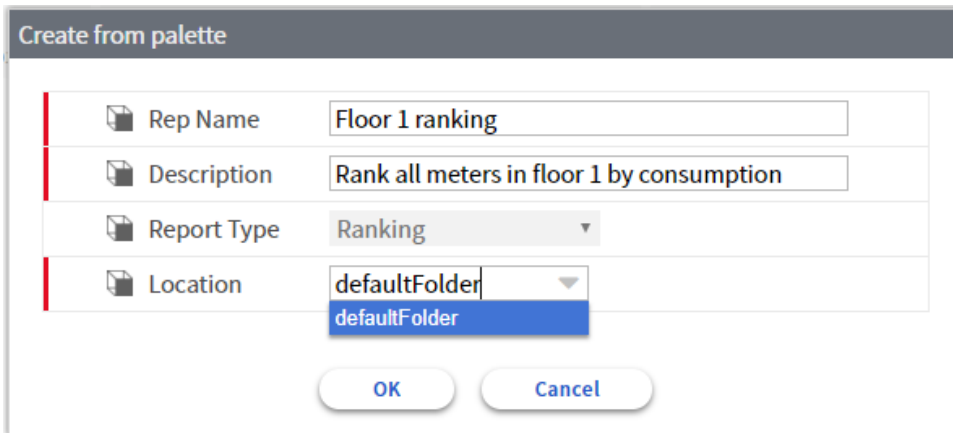
Prerequisites: You are working in WEBStation. The analytics palette is open.

1. Right-click **AnalyticService > Reports** and click **Views > Analytic Ux Report List View**.

The All Reports view opens.

2. Click the **New Report** button.

The Create from palette window opens.



The screenshot shows a dialog box titled "Create from palette". It has four rows of input fields, each with a folder icon on the left:

- Rep Name:** Floor 1 ranking
- Description:** Rank all meters in floor 1 by consumption
- Report Type:** Ranking (dropdown menu)
- Location:** defaultFolder (dropdown menu with "defaultFolder" selected)

At the bottom of the dialog are two buttons: "OK" and "Cancel".

3. Give the report a name and description, select the type of report from the drop-down list, and select where to store the report file and click **OK**.

### Related Links

- Reports (Parent Topic)

## Managing Ux reports

Once a Ux report exists, you may edit its properties, clone it and delete it.

Prerequisites: The report exists.

1. Right-click **AnalyticService > Reports** and click **Views > Analytic Ux Report List View**.

The All Reports view opens.

2. Scroll to the report or search for the report by name.

3. To edit the report, click the edit icon (✎).

The Edit report window opens.


**Edit report**

Rep Name

Description

OK Cancel

You can change the name and description.

4. To make a new report by using an existing report (clone), find the report and click the clone icon ().
- The Clone report window opens.


**Clone report 'Report\_AVgPRf'**

Rep Name

Description

Location

OK Cancel

5. To continue cloning the report, enter a name, description, and location for the new report.
6. To delete a report, locate it and click the delete icon (). The framework asks you to confirm the deletion.

Are you sure you want to delete the report?

OK Cancel

7. To continue with the deletion, click **OK**.

## Related Links

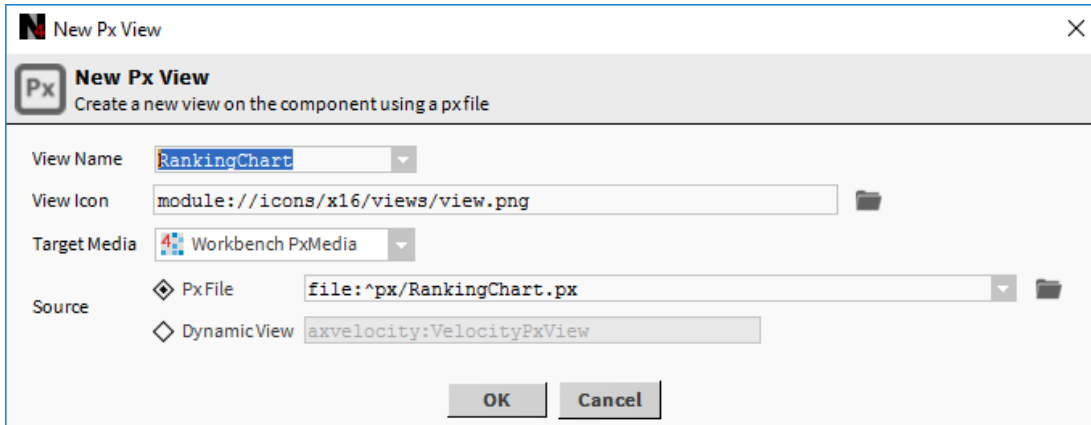
- Reports (Parent Topic)

## Creating a dashboard

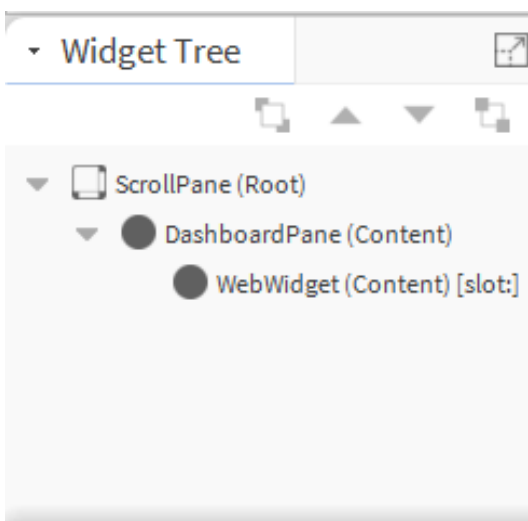
A dashboard is a specialized report.

Prerequisites: The dashboard palette is open.

1. Make a folder in your station to contain dashboards.
2. Do one of the following:
  - a. To create a new view, right-click the folder, click **Views > New View**, and assign a view name.
  - b. To edit an existing view, double-click the view name. The New Px View window opens.



3. Fill in the **View Name** for the dashboard and click **OK**.
4. Expand the wire sheet **viewSize** to 1024 x 768 (this property is in the Properties side pane).
5. Drag the Dashboard pane from the palette to the Widget Tree (side panel in the report editor view).
6. Open the analytics palette, expand the reports node, and drag a report to the wire sheet. The system adds the report to the Widget Tree panel. For example:



## Related Links

- Reports (Parent Topic)

## Configuring a report or a dashboard

Configuration involves selecting nodes, picking colors selecting the date range and other options.

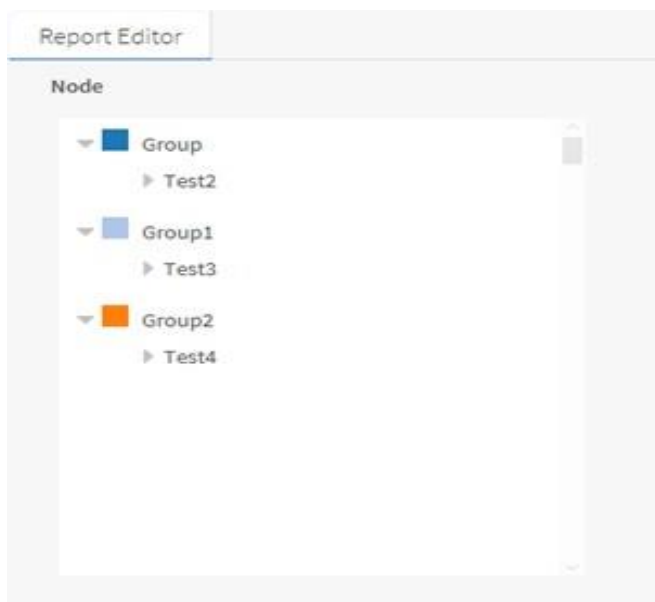
Prerequisites: You are connected to the station and the report or dashboard to configure exists.

Perform the following steps:

1. Expand the Nav tree to locate the points to include in the report, then expand the report or dashboard folder in the station, and double-click the report or dashboard name.
2. Drag the nodes to analyze from the Nav tree to the Node pane.

To group nodes in the Node pane, drop the node you are dragging onto the node parent group. To configure a parallel group for analysis, drop the node you are dragging outside the structure in the Node pane.

**NOTE:** The Aggregation and Spectrum reports support only one group.



The screen capture is an example of the Node pane with three groups.

3. To change the color associated with the group, select the group and select the color in the Color Picker.
4. To rename a group, right-click it, click Rename, enter a new name and click **OK**.
5. Click the Tag Chooser and define the data type.

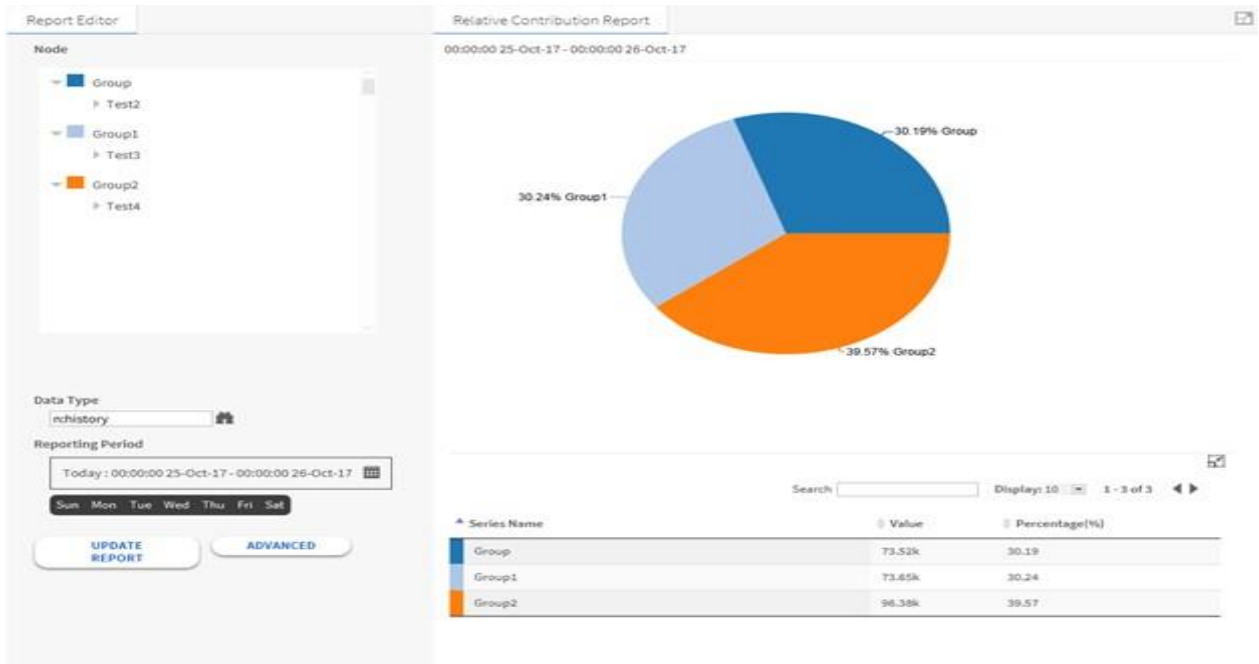
Click the calendar chooser and select the day or date range.




The Time Range window selects a general time period or allows you to define a specific time and date to start and end data collection.

6. Click **Run Report**.

The report opens in the display pane. The screen capture is an example of the chart pane for a Relative

Contribution report.



7. Do one of the following:
  - a. To expand the view to see the chart only, click the expand icon (  ) in the upper right corner of the chart area.
  - b. To expand the view to see the table only, click the expand icon (  ) in the upper right corner of the table area.
8. To return to the overall view, click the contract icon (  ).
9. To apply the changes, you make, click the **Update Report** button.
10. To configure additional properties, click the **Advanced** button.

The additional properties vary depending on the report. See the Reference manual for a description of each property.

**Related Links**

- Reports (Parent Topic)

**Normalizing energy consumption values based on floor area**

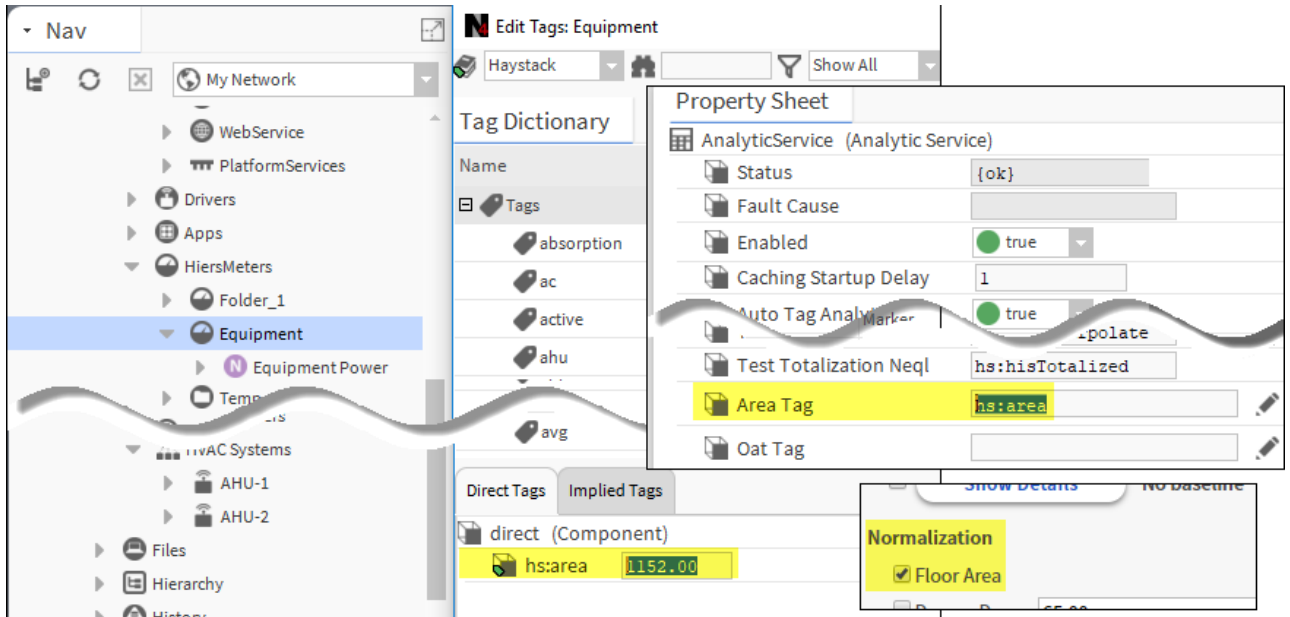
An energy procurement strategy can reduce consumption volatility. It also sets up a load profile that is more attractive to energy providers and can reduce your energy costs. To create an energy procurement strategy, you need to understand current energy consumption patterns exhibited by your equipment. Normalizing the floor area of a facility evens out the energy consumption or demand differences caused by large and small spaces. This results in more useful comparisons. This topic documents how to configure the framework’s Average Profile report to plot and report these patterns for a single piece of equipment.

Prerequisites: You are working in WEBStation with any tag dictionary in the TagDictionary service. You have historical energy consumption data collected for the piece of equipment in the station database. In this procedure’s example the facility uses Hiers meters to monitor equipment power usage.

Perform the following steps:

1. Right-click the equipment and select **Edit Tags** from the popup menu.

The Edit Tags view opens.



2. Select a tag dictionary (the example uses the Haystack dictionary), select a tag (area in the example), and click **Add Tag**.

The **Direct Tags** tab in the lower half of the window displays the hs:Area tag.

3. Enter your facility's area (square feet or square meters) and click **Save**. The area in the example is configured for 1152 square units.
4. To configure **AnalyticService** properties, right-click the service in the Nav tree, click Property Sheet, and edit the Area Tag property with the area tag name.

If you do not configure the Area Tag property in the **AnalyticService**, the framework defaults to hs:area.

5. Expand the AnalyticService in the Nav tree and double-click the **Reports** node. The set of available Analytic Ux Report List View report tiles opens.
6. Do one of the following:
  - To create a new report, click New Report; supply a Rep Name and Description; select Average Profile or Load Duration Report for Report Type, click OK; and click the report tile.
  - To edit an existing report, click its tile. The Report Editor opens.



7. Configure the Reporting Period; click to enable Floor Area under Normalization, and click Run.

The framework produces the report.

If neither the direct tag on the equipment node nor one of its children has an area value configured, or if the value is zero (0), the framework performs no normalization even if Floor Area normalization is enabled in the Report Editor.


## Related Links

- Reports (Parent Topic)







## Normalizing energy consumption values based on degree-day temperature

Degree-days help determine if your building's energy consumption is saving or losing money. You can compare this year's weather with last years at the same time and explain expenditures in general terms, but this comparison does not answer the question, "Have the cost-saving mechanisms and procedures we put in place at all company locations saved money?" A fair comparison should be independent of the weather. The keys to such a comparison involve calculating degree-days and normalizing the weather.

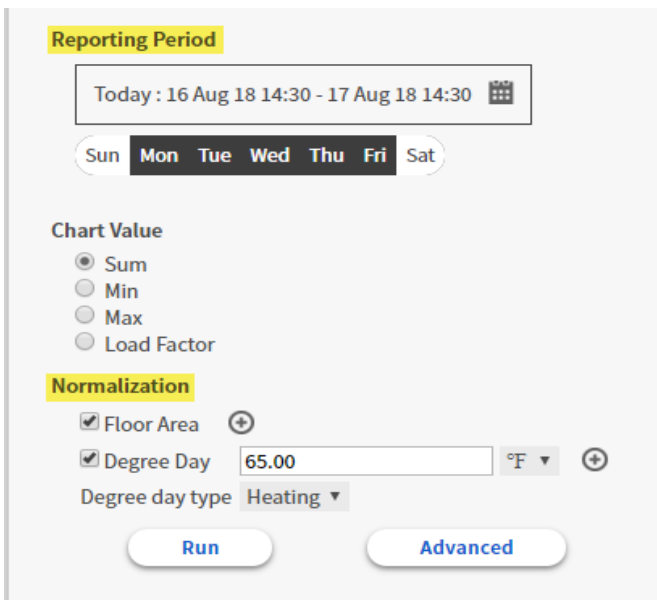
Prerequisites: You are working in the web UI with the AnalyticService. Historical OAT (Outside Air Temperature) data are available in the station database and tagged with an identifiable OAT Tag.

1. To view the Nav tree, click the Navigation Tree button (.
2. Right-click the **Config > Services > AnalyticService** in the Nav tree and click **Views > Property Sheet**.

The **AnalyticService** Property Sheet opens.

 Test Cov Neql	<input type="text" value="hs:hisInterpolate = 'cov'"/>
 Test Totalization Neql	<input type="text" value="hs:hisTotalized"/>
 Area Tag	<input type="text" value="hs:area"/>
 Oat Tag	<input type="text" value="hs:outsideAirTempSensor"/>
 Alerts	Alert Folder
 Algorithms	Algorithm Folder

3. In the Oat Tag property, enter the tag that identifies the OAT values in the database.
4. Expand the AnalyticService in the Nav tree and double-click the **Reports** node. The set of available Analytic Ux Report List View report tiles opens.
5. Do one of the following:
  - To create a new report, click **New Report**; supply a Rep Name and Description; select Average Profile for Report Type, click **OK**; and click the report tile.
  - To edit an existing report, click its tile. The Report Editor opens.



**Reporting Period**

Today: 16 Aug 18 14:30 - 17 Aug 18 14:30

Sun Mon Tue Wed Thu Fri Sat

**Chart Value**

Sum  
 Min  
 Max  
 Load Factor

**Normalization**

Floor Area

Degree Day  °F

Degree day type Heating

Run Advanced

6. Configure the Reporting Period, including the days of the week; under Normalization, click to enable Degree Day; and enter the outdoor temperature at which neither heat nor air conditioning is required to maintain a satisfactory indoor air temperature.

This outside temperature value defaults to 65.00, which is a Fahrenheit temperature.

7. If you are using this default temperature, select °F from the temperature scale drop-down list, otherwise, enter a Celsius or Kelvin temperature value and select °C (Celsius) or K (Kelvin) from the drop-down list.
8. To select the type of temperature supplementation, select Heating or Cooling for Degree day type.
9. To identify the point node in the Nav tree to provide the outside air temperature values, click the plus icon to the right of the Degree Day properties.

The Degree day mapping window lists the temperature equipment groups and nodes that are available.

Group Name	Node
Equipment	slot:/
HVAC	slot:/
Lights	slot:/
Plug Load	slot:/

OK Cancel

10. If necessary, edit the group and node, then click **OK** followed by clicking **Run**.

For the Reporting Period you defined, the framework calculates the degree-day value by taking the average of the differences between the base outside air temperature (default: 65.00) and each actual outside air temperature value. Then it calculates the kWh per degree-day by dividing the energy consumption for each interval by the average degree-day value.

### Related Links

- Reports (Parent Topic)

## Printing a report

There are two ways to create a hard copy of a report: export the report to a PDF and print the PDF from your PC, or print the report from the web UI using either Chrome or Firefox.

Prerequisites: The report exists. You are working in the web UI.

1. Do one of the following:
  - To prepare to print a report in Chrome, click **Ctrl + P**.
  - To prepare to print a report in Firefox, click **Menu > Print**.

For Chrome, the print preview page opens with the Print properties in the left pane.

For Firefox, the print options appear along the top of the page.

2. Click the tiny arrow above the report editor to minimize it so that only the report itself prints.
3. After configuring properties, click **Print**.

### Related Links

- Reports (Parent Topic)

# Missing data strategy

Data collected from meters, sensors and other building automation devices may not be complete. Gaps in the data set occur when devices stop working or incur a fault. In statistics, imputation is the process of replacing missing data with substituted values. Incomplete, incorrect, inaccurate and irrelevant data are replaced, modified, or deleted. This is also known as data cleansing or data cleaning.

Where the data are missing in the series plays an important role in the calculations. Data may be missing at the beginning of the data set, interspersed among the other data, or at the end of the data set.

The framework offers multiple strategies for managing missing data:

- Linear interpolation derives an estimated value of the missing data in the series.
- K-nearest neighbor finds the nearest neighbors to a missing datum, identifies the majority value represented by the neighbors, and fills in that value for the missing datum.
- Aggregation strategies: ignore series and ignore point

## Related Links

- [Linear interpolation](#)
- [K-nearest neighbor \(KNN\)](#)
- [Aggregation strategies](#)
- [Missing data configuration](#)
- [Missing data indication](#)

## Linear interpolation

This interpolation algorithm linearly interpolates the missing values based on the surrounding values in the series.

There are three locations where data can be missing:

- At the beginning of the series
- Interspersed among the series
- At the end of the series

### Data missing at the beginning of the series

The system replaces missing values with the first available value in the series. For example, if M1 is a faulty meter installed in Building 1, which fails to record a daily energy reading for three days, and its replacement meter records 20 on day four, linear interpolation assigns 20 to each of the missing days.

**Data missing interspersed among the series**

The system replaces missing values by calculating the slope between the last and next collected values. The interpolation equation is:

$$\text{slope} = (\text{nextValue} - \text{previousValue}) / (\text{nextTimestamp} - \text{currentTimestamp})$$

For example, meter 1 functioned accurately and logged values for three days after which a fault occurred in the meter. It took three days to identify and fix the fault. On day six the system began to log values again. The missing values in this data set occur between intervals. If the recorded value for each day at the beginning of the series is 20, and the recorded value for the sixth day is 30, the calculation for day four is:

$$20 + ((30-20)/(6-3)) = 23.33$$

And the calculation for day 5 is:

$$20 + ((30-23.33)/(6-5)) = 26.66$$

**Data missing at the end of the series**

The system replaces missing values with the last recorded value in the series. For example, meter 1 takes a reading for each of three days after which it goes into fault for two days, ending the series. The reading for day three is 20. It makes no difference what the readings are for days one and two. The system interpolates the value for days four and five as 20.

**Related Links**

- [Missing data strategy \(Parent Topic\)](#)

**K-nearest neighbor (KNN)**

KNN is for numeric, enum and Boolean records. For intervals, other than none, this strategy replaces a missing value by calculating the majority value recorded for the item’s k nearest neighbors.

The number of neighbors to consider is k. The system selects k previous and next nearest neighbors to calculate the missing value. If a tie occurs between two values, the algorithm selects the lowest timestamp value.

**Boolean data series example**

For example, the system monitors if meter 1 is on or off reporting a value of true (on) or false (off) every 15 minutes (the interval). If a value for 2:15 is missing between time stamps 2 pm and 3 pm, and k = 3, the system finds the three nearest timestamps to 2:15, which are 1:45, 2:00, and 2:30 and takes the majority of these three values. The table records these values:

Timestamp	Meter 1 on state
1:30	true
1:45	false
2:00	true
2:15	false (interpolated value)
2:30	false
2:45	false

In the example, the majority value, considering the three neighbors, is “false.” The system assigns this value to 2:15.

### Tie example

A tie can occur with numeric, Boolean and enum data. The system handles a tie in a particular way:

1. First, it gives preference to the highest frequency of k nearest neighbors.
2. Next, when the same frequency (k) of nearest neighbors exists, the system gives preference to the value associated with the lower timestamp.
3. Finally, when the timestamps are equidistant from the missing data, the system gives preference to the record above the missing data.

In this first table, k=4, its lower (earlier) nearest neighbors’ majority value is zero. Its higher (next) nearest neighbors’ majority value is 1. Using rule 2 above, the system breaks the tie by assigning the missing value to the same value as the lower timestamp (1:45).

### K = 4, Same frequency of nearest neighbors

Timestamp	Enum data
1:30	0
1:45	0
2:00	0 (interpolated value)
2:15	1
2:30	1

In the following table, k=1, its previous nearest neighbor’s value is 1. Its next nearest neighbor’s value is 3. The system recorded both values at the same interval before and after the missing value, so it gives preference to the lower timestamp, and sets the missing data value to: 1.

### K = 1 Timestamps equidistant from the missing data

Timestamp	Enum data
1:30	0
1:45	1
2:00	1 (interpolated value)
2:15	3

### Related Links

- [Missing data strategy \(Parent Topic\)](#)

## Aggregation strategies

These strategies configure the system to ignore either the aggregated sum of a series with missing data or to ignore only the missing values while calculating the aggregated sum of the records with values.

### Ignore series

This strategy ignores any aggregated sum that includes missing data, even if only a single record is missing.

For example, a meter is added to a site, and four days later it starts recording energy consumption data. On a report or chart configured to aggregate the sum of all energy meters, the system ignores the aggregated sum for days 1–3 because the calculation for at least one meter contains missing data.

Day	Meter 1 energy values	Meter 2 energy values	Aggregated values (sum)
1	-	10	-
2	-	10	-
3	-	10	-
4	30	20	50
5	30	30	60

The aggregated sum ignores the fact that meter 2 recorded values of 10 for the first three days.

### Ignore point

This strategy ignores only the values in the interval that are missing and accommodates the recorded values for the overall calculation.

For example, using the same meter as in the example of ignoring the series, the system aggregates the sum of all values ignoring only the missing values themselves.

Day	Meter 1 energy values	Meter 2 energy values	Aggregated values (sum)
1	-	10	10
2	-	10	10
3	-	10	10
4	30	20	50
5	30	30	60

The system counts Meter 2's values for days 1–2.

### Related Links

- [Missing data strategy \(Parent Topic\)](#)

## Missing data configuration

Missing data are configured at a number of places.

Missing data strategies (algorithms) apply to these types of points:

Interpolation algorithm	Numeric Point	Boolean Point	Enum Point
K-nearest neighbor	p	p	p
Linear interpolation	p		

**NOTE:** Linear interpolation works on raw history and also when an interval is selected. Any interval can be applied to get interpolated records. KNN works only when an interval other than none is selected for the series.

The missing data properties are available in these primary locations:

- Charts
- Reports
- Bound tables
- Web tables
- Proxy extensions
- Alerts

If no missing data strategy is defined in these locations, the system defaults to the missing data strategy configured on the data definition for the tagged points.

If no missing data strategy is defined on the data definition for the tagged points, the system defaults to the MDS configuration defined on the AnalyticService.

### Related Links

- [Missing data strategy \(Parent Topic\)](#)

## Missing data indication

The system identifies data that were interpolated on both tabular views as well as report and chart views.

### Tabular views

Across all controls, flags identify interpolated data:

- `Li` indicates that linear interpolation was applied.
- `{knn}` indicates that k-nearest neighbor interpolation was applied.
- `{igp}` indicates that Ignore Point was selected for aggregation strategy.
- `{ }` indicates that no interpolation algorithm was applied.
- `{knn,igp}` indicates that k-nearest neighbor interpolation was applied with Ignore Point as aggregation strategy.



Figure 15 Example of a web table with interpolated data

Timestamp	NumericWritable-history Status	NumericWritable-history Value	NumericWritable-history InterpolationStatus
10-Aug-18 2:45:00.000 PM UTC+05:30	{ok}	66.55207061767578	{L1}
10-Aug-18 3:00:00.000 PM UTC+05:30	{ok}	66.55207061767578	{}

The Interpolation Status column to the right indicates how the data were interpolated.

Figure 16 Example of a bound table with interpolated data

Timestamp	Value	Status	Trend Flags
12-Jul-18 12:00 AM IST	true	{ok}	{knn}
12-Jul-18 12:05 AM IST	true	{ok}	{knn}
12-Jul-18 12:10 AM IST	true	{ok}	{knn}
12-Jul-18 12:15 AM IST	true	{ok}	{knn}
12-Jul-18 12:20 AM IST	true	{ok}	{knn}
12-Jul-18 12:25 AM IST	true	{ok}	{knn}
12-Jul-18 12:30 AM IST	true	{ok}	{knn}

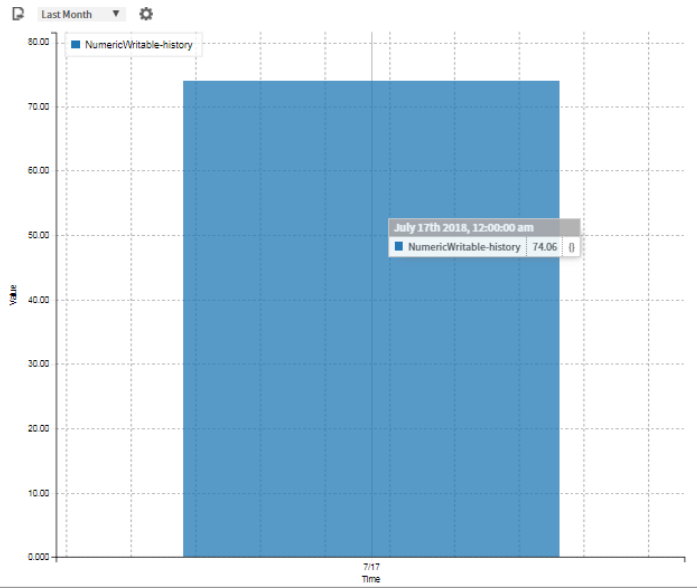
The Trend Flags column contains the interpolated data flags.

**NOTE:** Ignore Series never triggers a flag because this property causes the system to ignore the entire series. Interpolated data do not appear in the table, report or chart. This was the framework’s default behavior when no data are available.

**Charts**

A tool tip on a chart indicates an interpolated or a real record.

Figure 17 Chart with interpolated data

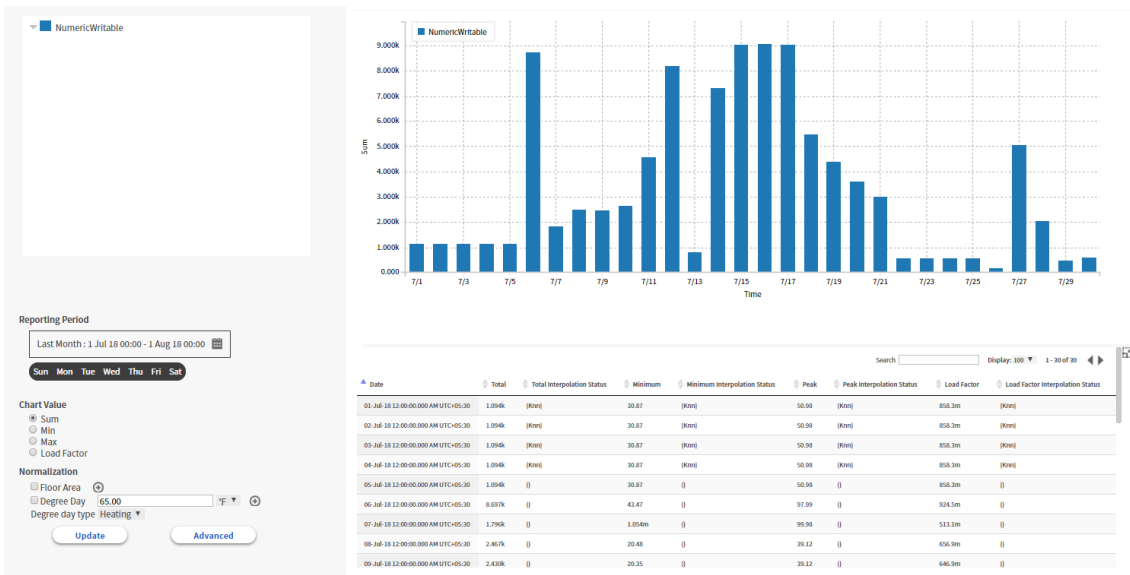


The tool tip indicates that no interpolation algorithm was applied.

### Reports

Reports provide interpolated data in a separate column. To view these data, you must enable Show Interpolation Status in advanced settings.

Figure 18 A report configured to show interpolated data



The interpolated data are visible in the table.

### Related Links

- [Missing data strategy \(Parent Topic\)](#)

# Troubleshooting

Make sure all cables are correctly connected and all equipment turned on. This topic covers general issues that occur after the system is installed and configured.

## Definitions

If you are having difficulty visualizing data that appears to be configured correctly, make sure that you have a definition for each tag. This is especially important if you created your own tag dictionary.

When you create a definition, you associate it with a tag by entering the tag name space and name in the Id property. To set up definitions, expand **Services > AnalyticService > Definitions**.

## Logs

System logs contain information you can use when debugging problems.

If a point matches the criteria required to generate an alert, appears in a Proxy Extension, or is included in a binding, but the point does not have an a:a tag, the system logs an error. The log level for this error is FINER. At the default log level, the system does not print this error, but you can change the log level using the Station spy option to view the FINER log.

## Related Links

- Point status
- Enabling error logging
- Scenarios

## Point status

Point status reports the current status flag(s). More about these flags and point status is in Getting Started with Niagara

The status of points, especially proxy points, may explain what appears or does not appear as expected in a bound table or on a web chart. The following summarizes the meaning of each status and what, if anything, to do about it.

### Point status

Status	Description	Remedy
alarm	The point has a value in an alarm range as defined by a property in its alarm extension.	Acknowledge the alarm, investigate and fix the condition that caused the alarm.
down	Driver communication with the parent device as configured in the extension has been lost. All proxy point children of the device report a status of "down." This status originates from a proxy point only.	Confirm that the parent device is on line and functioning correctly.
disabled	The proxy extension has been disabled. Polling stops for the point. This status originates from a proxy point only.	Enable the proxy extension.

Status	Description	Remedy
fault	Typically, this indicates a configuration or license error. If a fault occurs following normal {ok} status, it could be a condition detected within the device, or perhaps some other fault criterion that was met.	Check the point's proxy extension's <b>Fault Cause</b> text for more information.
null	Indicates no status is available.	Check the point configuration.
ok	Indicates that the point is functioning as expected. No status flag(s) are set.	No action required.
overridden	The functioning of the point has been stopped usually by a hardware override switch.	Make a physical inspection of the device.
stale	Since the last poll update, the system has not updated the point's value within the specified <b>Stale Time</b> of its Tuning Policy. This status originates from a proxy point only.	This status clears upon receipt of the next poll value.

**Related Links**

- Troubleshooting (Parent Topic)

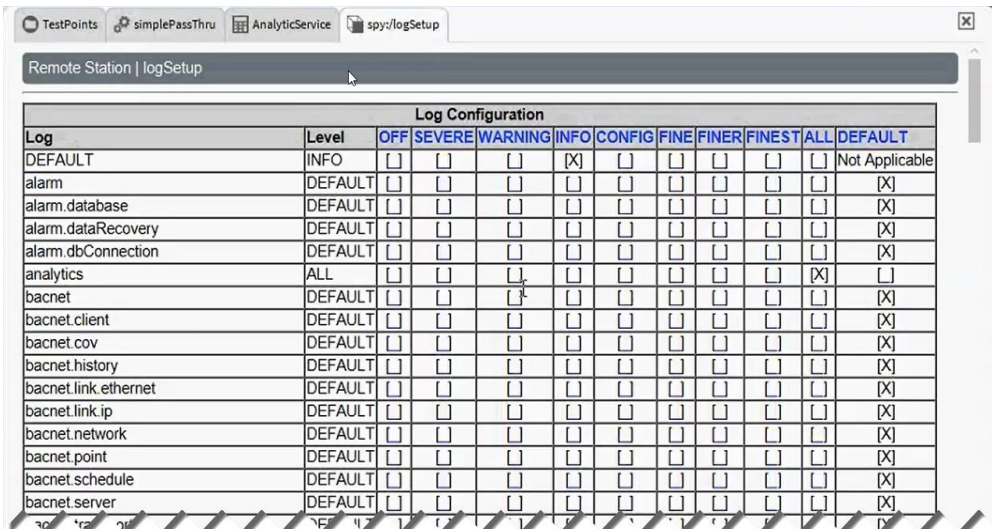
**Enabling error logging**

To debug, configure the module you are working with to output all error messages to the station log. This involves configuring the Spy.

**NOTE:** Configure the Spy to output all messages only while debugging. This feature requires system resources and could slow operations if left on.

Perform the following steps:

1. Right-click the station and click **Spy**. The Log Configuration table opens.



2. Configure the module level to log ALL messages.
3. Debug the feature.
4. Change the log level back to a limited number of messages.

## Related Links

- [Troubleshooting \(Parent Topic\)](#)

## Scenarios

Use this topic to read about a condition others have experienced that you may also experience, and what to do about it.

### **Everything was working just fine when suddenly the framework stopped working.**

You may have exceeded the license point limit allowed for your system. This can happen if the `AutoTagAnalyticPoint` property on the `AnalyticService` property sheet is set to true. This property should be set to false, unless you are configuring the system for the first time.

### **My chart is not displaying data correctly.**

Confirm that the number of bindings is correct. If more bindings are specified than are required, nothing happens. Binding support is as follows:

- Average Profile Chart supports a single binding.
- Ranking Chart supports multiple bindings.
- Equipment Operation Chart supports a single binding.
- Load Duration supports multiple bindings.
- Relative Contribution Chart supports multiple bindings.
- Spectrum Chart supports a single binding.

**I configured the Time Range and Interval, but my native Spectrum Chart is completely blank.**

If the Spectrum Chart binding returns less than three data points (that is 0, 1 or 2 data points), the chart fails and reports an exception. To investigate, calculate the total number of expected data points by dividing Time Range by Interval, and adjust the Time Range or Interval to report three or more data points.

**How do I limit the data source to average for a min/max of one day in an algorithm?**

Enable User Request Rollup on the data source and set Rollup to min or max as required.

**I can add an Analytic Web Chart Binding to a web widget and configure it, but I cannot seem to get it to work.**

After adding the Analytic Web Chart Binding, double-click the widget to open the Properties view, and delete the default WbView Binding.

**I tried to use the Analytic Value Binding, but where do I define the period as average?**

The Analytic Value Binding gives the current value without any rollup.

**I imported histories and points under my NiagaraNetwork from my remote host and tagged them, but I cannot visualize anything.**

Enable the Persist Fetched Tags on the AX Property Sheet of the **NiagaraNetwork** driver, right-click on the Niagara Driver and click **Actions > Force Update Niagara Proxy Points**. This ensures that the histories are associated with the points discovered in the Niagara Network.

**I want my algorithm to return trend results if a value was less than or equal to zero. Algorithm results show in a bound table but not on a web chart.**

For a Niagara Network, right-click the **Config > Drivers > NiagaraNetwork** folder in the Nav tree and click **Actions > Force Update Niagara Proxy Points**. This applies a direct n:history tag to any point that lacks a history extension.

For all other networks, use a Program Object in the V2 bog file to add an n:history tag to each point.

**I'm running in a WEB-8000. My Web chart causes a server session time-out.**

Check to see if you have specified a COV (change of value) point directly in the binding. A COV point that changes frequently can cause CPU spikes. As a best practice, instead of specifying the COV point directly, specify its parent in the binding. For other point types, configure a less frequent Refresh Rate in the binding to minimize CPU spikes.

**I notice that, when I run PX views, my WEB-8000 slows down, and sometimes reports server session time out errors.**

For best performance on the WEB-8000, limit the number of points configured in a PX view to 100–200 with no more than 200 tags, 500 history rollups and five bindings. For more complicated configurations, set up PX graphics in a Supervisor station running on a PC.

**For an analytic request (binding, alert, etc.), the unit of measure output from an algorithm is not being converted correctly or it does not match the unit set in the algorithm.**

Check the algorithm to ensure that the correct unit of measure is defined (by facets on the algorithm's property sheet). Algorithms perform no unit conversion from data source to algorithm output. The unit of measure defined in the algorithm's facets is directly output with the calculated value. This makes it imperative to define the correct unit of measure on the algorithm's property sheet.

For a series of chained algorithms, for example: Algorithm 1 becomes a data source for Algorithm 2, which, in turn, becomes a data source for a Px binding, the system converts the output unit from Algorithm 1 (assuming the Algorithm 1 unit is defined in its facets) to the unit specified for the data source in Algorithm 2. If Algorithm 1 has no unit defined, and Algorithm 2 has the unit defined, Algorithm 2 applies its unit of measure to the input it receives from Algorithm 1.

**I configured a data source for my web chart, but the system says that the data source is not available.**

Check the station log to identify the origin of the request for data.

Figure 24 Station log

```

Administrator: Niagara Command Line - station unitTest -@agentlib:jwp=transport=dt_socket,se...
:97)      at com.tridium.fox.sys.data.BDataChannel.circuitOpened(BDataChannel.java
464)      at com.tridium.fox.sys.BFoxConnection.circuitOpened(BFoxConnection.java:
         at com.tridium.fox.session.SessionCircuits$ServiceThread.run(SessionCirc
         at java.lang.Thread.run(Thread.java:745)
java.lang.IllegalArgumentException: Data Not Available
origin - user request
data - alg:simplePassThru1
node - local:!station:!slot:/TestPoints
user - admin

origin - user request
data - hs:air1
node - local:!station:!slot:/TestPoints
algorithm - simplePassThru

origin - user request
data - alg:simplePassThru
node - local:!station:!slot:/TestPoints
algorithm - simplePassThru1

         at javax.bajax.analytics.BAnalyticService.dataAvailabilityExceptionHandl

```

A request from a graphic is classified as a “user request.” This is followed by which point, node, and user are involved in this request, as well as the name of the algorithm in which the request was made. The multiple origins in the screen capture example represent nested algorithms.

If this does not help you solve the problem, open the Property Sheet for the AnalyticService, and set Skip Data Source Cache to true. The framework engine caches memory to improve response time. Disabling memory cache, by setting this property to true, causes the system to display the current error in the station log.

**NOTE:** When you are finished debugging, make sure you set Skip Data Source Cache to false again so not to impact performance.

**I am trying to figure out why an alert occurred.**

Check the station log. It shows which point generated the user request that triggered the alert along with the alert name.

**I tagged points to be used with analytics and my AnalyticService is now in fault and all analytic requests fail. What happened?**

The number of points you can use for the framework is limited by your license. If you tag more points than you are licensed to use, the service goes into fault. Update your license to add more points or remove the extra tags.

**I set up a Source Name for an alert expecting that the text and BFormat I entered would display in the alarm console. Instead, the alarm console displays the default BFormat (%node.navName%\_%alert.name%) in its Source column.**

You have a syntax error in your BFormat. Clicking the Notes button displays this message, “The BFormat value for sourceName is invalid for alert AnalyticAlert.” where AnalyticAlert is replaced by the name you configured for the alert. Check the Reference manual for examples of BFormat syntax.

## Related Links

- Troubleshooting (Parent Topic)

# Glossary

aggregation	The process of combining multiple pieces of the same type of data into a single value (sum, average, maximum, minimum, median, etc.) Each piece of information comes from a different data source. The request for the specific type of data starts at a specific node and travels down the data model tree aggregating all sources (points) tagged with the search argument tag.
alert	A warning regarding a condition identified by a WEBs-N4 Analytics Framework that can be routed to an alarm or used to visualize real-time and historical data.
algorithm	A formula that uses real-time values, historical trend data, and the results of calculations made by other algorithms to analyze data collected by the system and automatically manage remote devices.
analytics	The discovery and presentation of meaningful patterns in data. Analytics rely on the simultaneous application of statistics and computer programming to quantify performance, communicating the results on graphs and charts, as well as using results to control devices.
COV or Cov	Change-of-Value. Characterizes the option to track data based on when a value changes rather than at a consistent interval, such as every minute, 15 minutes, etc.
data definition, definition	Defines the type of information a request is looking for, such as real energy, zone temperature, air flow quantity, set point, phase A amperage, phase B amperage. Definitions are related to tags in that the definition <b>Id</b> is the tag name space and name used to search the database.
data model	A hierarchical tree structure that organizes points based on the usage or reporting of information rather than on the drivers required to manage physical devices. A typical Niagara station is built around device drivers (lon, bacnet, modbus device, etc.). Data modeling allows you to structure information in potentially more useful ways, such as by geographic location, equipment type or responsible party.
direct tag/relation	A tag or relationship that has been manually associated with an entity.
indirect tag/relation	A tag or relationship tag automatically assigned by the system to an entity.
origin entity	The object in a hierarchy from which a search begins.



request	The query for input data that seeks either a point's current or most recent historical value.
rollup	The process of combining historical data for a single data source into one value (sum, maximum, minimum, average, etc.).
scope	In programming, the range within a program's source code within which an element name is recognized without qualification. Variable definitions are not limited to the beginning of a block of code, however, they must be declared before they can be used. In Niagara, the scope of an action applies to the selected components. The component tree is hierarchical. If you delete or move a component that contains other components, you are deleting or moving all items that are contained in that container component (its scope).
tag	<p>A piece of semantic information (metadata) associated with a device or point (entity) for the purpose of filtering or grouping entities. Tags identify the purpose of the component or point and its relationship to other entities. For example, you may wish to view only data collected from meters located in maintenance buildings as opposed to those located in office buildings or schools. For this grouping to work, the metering device in each maintenance building includes a tag that associates the meter with all the other maintenance buildings in your system.</p> <p>Devices are associated with Supervisors based on tags; searching is done based on tags.</p> <p>Tags are contained in tag dictionaries. Each tag dictionary is referenced by a unique namespace.</p>
trend	The result of analyzing historical data collected by the system. A trend involves rolling up data into meaningful intervals.