DISTECH CONTROLS™

EC-Net System Database and System Indexing Guide

User Guide

Connecting People with
Intelligent Building Solutions

EC-Net System Database and System Indexing Guide

Distech Controls, Inc.
Brossard, Quebec,
Canada

Legal Notice

Contents

About this guide

This topic contains important information about the purpose, content, context, and intended audience for this document.

Product documentation

This document is part of the EC-Net™ technical documentation library. Released versions of EC-Net software include a complete collection of technical information that is provided in both online help and PDF format. The information in this document is written primarily for Systems Integrators. In order to make the most of the information in this book, readers should have some training or previous experience with EC-Net™ 4 or EC-NetAX™ software.

Document content

This document provides an overview of the System Database and System Indexing features in EC-Net 4, as well as details specific to setting up the System Database for indexing both local and remote stations.

Document change log

Changes to this document are listed in this topic.

February 21, 2024

As part of the EC-Net 4.14 OrientDb upgrade, updated “Minimum system requirements” section in the “Requirements” chapter

September 15, 2023

Added to ”systemIndex-SystemIndexService” component re-indexing information resulting from the System Database (OrientDb) upgrade in EC-Net 4.14.

March 2, 2023

Added to ”niagaraDriver-ReachableStationInfo” component the following properties (as of EC-Net 4.13): “Station Version” and “Min Version Along Route”.

October 1, 2022

Added “Multi-tier SystemDb indexing” chapter and related components. Included information about the Orient SystemDb spy.

September 1, 2022

Added note for namespace and query search details in “Query OrdList” topic.

November 10, 2020

Updated “Query OrdList” in the “Requirements” topic.

June 24, 2020

Updated properties in the systemDb-SystemDbService topic to include details on the Orient v3 database. Replaced references to “NiagaraEdgeLiteStation” with “NiagaraEdgeStation”.

May 20, 2020

- Added Note in “Requirements” topic under graphSystemDb module.
- Edited “Requirements” topic to provide example and syntax for **`MaxDirectMemorySize`** in the nre. properties

December 6, 2018

Added Orient Index topics to the Components section.

June 15, 2018

Edited “Requirements” providing additional information on the minimum system requirements for System Database and System Indexing operations. Also, many minor updates throughout the document.

May 17, 2018

Initial publication.

Related documentation

These documents provide additional information.

- *Hierarchies Guide*
- *Drivers Guide*
- *EC-Net Tagging Guide*

Chapter 1 Overview

Topics covered in this chapter

♦ ORD schemes that support querying the SystemDb
♦ Additional core changes to support querying the SystemDb
♦ How information is added to the Database
♦ Requirements

There is support for the System Database and System Indexing features. The purpose of the System Database is to allow you to run queries against your entire EC-Net system (Supervisor and those stations connected to it). This allows your searches and hierarchies to span the entire system.

The System Database (SystemDb) is where a Supervisor station stores indexed entities. It is responsible for adding, removing, and querying entities to/from a backend database. The SystemDb only supports NEQL queries run against it.

Entities in EC-Net are basically composed of an ORD (Object Resolution Descriptor) identifier, tags and relations. For details on tags and relations, see the *EC-Net Tagging Guide* and *Relations Guide*.All components (including networks, devices, and points) satisfy this definition of an entity, while histories, alarms and other types of files do not. Additionally, entities are the basis of NEQL queries which are used by the Hierarchy and Search services.

EC-BOS stations will not have a System Database, but their information may be indexed and placed in the Supervisor's System Database. The Supervisor station is the only station that can access the SystemDb. Once entities are stored in the SystemDb, they can be queried by services, for example HierarchyService and SearchService, and other processes.

NOTE: The SystemDb is currently intended to run only on Supervisor platforms, not on EC-BOS platforms. The System Indexing operations may only affect EC-BOSs if exporting the index data to the Supervisor instead of using the more common import or global import of index data, both of which are configured on the Supervisor side.

When the Supervisor queries the SystemDb for information about subordinate stations (via NEQL), the resulting entities are resolved in the form of EC-Net virtuals, since they already allow for configuring user permissions on virtual components in the Supervisor.

ORD schemes that support querying the SystemDb

The following ORD schemes facilitate the resolution of indexed entities.

CAUTION: Frequent querying can slow down the System Database. If you execute a SystemDb query frequently, the SystemDb performance degrades.

- "nspace" ORD scheme

  Provides global resolution of objects. This common ORD format is used for all entities indexed into the systemDb and is flexible enough to accommodate different use cases.

  – Local station example:

    `nspace:SupervisorName|slot:/a/b/c`

  – Remote station example:

    `nspace:DeviceName|slot:/a/b/c`

    Note that this ORD resolves on a Supervisor station to the following NiagaraVirtual:
    `station:|slot:/Drivers/NiagaraNetwork/DeviceName/virtual|virtual:/a/b/c`

- "sys" ORD scheme

Allows queries against the SystemDb (supports only NEQL, not BQL).

– Query the entire SystemDb for all devices in the system:

```
ip:<SupervisorIP>|foxs:|sys:|neql:n:device
```

– Scoped query against the SystemDb for ModbusTCP devices in subordinate station:

```
ip:<SupervisorIP>|foxs:|nspace:DeviceName|slot:/Drivers/ModbusTCPNetwork|
sys:|neql:n:device
```

Additional core changes to support querying the SystemDb

The following are recent core changes implemented to support querying the SystemDb.

- The `n:station` implied tag in the NiagaraTagDictionary

  The `n:station` tag is added to all indexed entities identifying the entity's source station.

  – Query the entire SystemDb for all devices from a particular subordinate station:

  `sys:|neql:n:device and n:station = 'DeviceName'`

  NOTE: The above example would be slow to process because it must scan the entire SystemDb. A scoped query (using an ORD base that is the root of the indexed device entities) is a more efficient way to perform the same query (e.g., `ip:<SupervisorIP>|foxs:|nspace:DeviceName|slot:/|sys:|neql:n:device`).

- There is added support for NEQL over fox connections (supporting BQL queries piped onto the results of NEQL queries), which displays as a collection table in EC-Net 4 Proor the browser.

  – Displays a collection table with a single ORD column:

  `ip:<SupervisorIP>|foxs:|sys:|neql:n:device`

  – Displays a collection table with two columns (Status and NavOrd):

  `ip:<SupervisorIP>|foxs:|sys:|neql:n:device|bql:select status, navOrd`

- There is added support for translated px graphics on virtual components loaded on demand.

  – This is equivalent to having px graphics imported to the Supervisor station via export tags, but without the additional setup. For details, see "Viewing Virtual Px graphics on demand" in the section on "EC-Net virtuals cache (virtual Policies)" in the *Drivers Guide*.

How information is added to the Database

The System Database is populated by System Indexing. This is the process of crawling the stations (both local and remote) by querying for entities that match certain criteria (for example, networks, devices, points) and storing those indexed entities in the Supervisor station's System Database.

The scope of the system is a Supervisor station and all of the subordinate stations it is connected to via its NiagaraNetwork. To resolve subsequent queries to the System Database, you must enable NiagaraVirtuals for any participating NiagaraStation in the Supervisor's NiagaraNetwork.

IMPORTANT: You must set up user permissions for EC-Net virtuals manually in the Supervisor via roles and permissions. For details, see "Authorization management" in the *Station Security Guide*.

Indexing methods

There are three ways to index, listed here in order of precedence. In each of these cases, the Supervisor station must have the SystemDbService and SystemIndexService installed and be running. Also, in all three cases there is a list of configurable index queries that are run to perform the index operation.

- Individual push (export) to the Supervisor

  The export method is done individually, but may only be useful for certain network topologies. You must configure the Supervisor to accept the index export from the remote station.

- Individual pull (import) from the remote station

  The import method allows you to specify a different set of index queries for one particular remote station. During this pull, the remote station is no longer able to respond to a global index query.

- Global pull (import) from the station(s)

  Global pull is the most common index method, where it imports index entities from all reachable subordinate EC-BOSs. The same list of queries is run on all stations in the NiagaraNetwork, which pulls data from the those stations.

Requirements

The following license features, software modules, and minimum system settings are required in order to use the System Database and System Indexing features.

Required licensing

The Supervisor station must be licensed for the **`systemDb`** feature. There is also a check for the presence of “orientDb=true” attribute under the systemDb feature, which allows use of the default OrientDb backend database.

Also required is the **`systemIndex`** license feature with certain attribute limits. These attributes go under the "systemIndex" license feature in the Supervisor's license (license changes are not required on the remote EC-BOSs)

- station.limit

  Defines how many NiagaraStation instances (for example,EC-BOSs) in the Supervisor’s NiagaraNetwork are allowed to participate in system indexing to put their entities into the SystemDb (subject to the entity limit that follows). This attribute is required for any remote entity indexing to occur against remote NiagaraStations. Once this limit is exceeded, additional NiagaraStations will be prevented from putting their index data into the SystemDb . The NiagaraStation's SystemIndexer component will go into fault indicating the license limit exceeded. Once the license limit is exceeded, the Supervisor station must be restarted in order to recount against the license limits.

- station.entity.limit

  Defines the maximum number of entities that are allowed to be indexed to the Supervisor's SystemDb from an individual remote NiagaraStation source (for example, EC-BOS). This attribute is required for remote entity indexing to occur against remote NiagaraStations. Once the limit is reached, any subsequent entities from a particular NiagaraStation is skipped from indexing.

- edge_station.limit

  Defines how many NiagaraEdgeStation instances in the Supervisor's NiagaraNetwork are allowed to participate in system indexing to put their entities into the Supervisor's SystemDb (subject to the entity limit that follows). This attribute is required for any remote entity indexing to occur against remote NiagaraEdgeStations. Once this limit is exceeded, additional NiagaraEdgeStations are prevented from putting their index data into the SystemDb. The NiagaraEdgeStation's SystemIndexer component will go into fault indicating the license limit exceeded). Once the license limit exceeded, you must restart the Supervisor station to recount against the license limits.

- edge_station.entity.limit

  Defines the maximum number of entities that are allowed to be indexed to the Supervisor's SystemDb from an individual remote NiagaraEdgeStation source (for example, NEL devices). This attribute is required for remote entity indexing to occur against remote NiagaraEdgeStations. Once the limit is reached, any subsequent entities from a particular NiagaraEdgeStation are skipped from indexing.

Required software

You must install the following modules on the Supervisor station.

- **`niagaraDriver`** (-rt)

- **niagaraVirtual** (-rt)
- **niagaraSystemIndex** (-rt, -wb)
- **orientSystemDb** (-se)
- **systemDb** (-rt)
- **systemIndex** (-rt)

If you intend to export index data from the subordinate station, you need to install the following modules on the subordinate station. No additional modules are required.

- **niagaraDriver** (-rt)
- **niagaraVirtual** (-rt)
- **niagaraSystemIndex** (-rt)
- **systemDb** (-rt)
- **systemIndex** (-rt)

Minimum system requirements

The minimum amount of memory needed depends on the number of entities, tags, and relations to be indexed. The following table offers guidelines and does not guarantee that the suggested memory settings suffices in every production setting. These settings are the minimum required (as part of the EC-Net 4.14 OrientDb upgrade) in the performance testing environment that allowed a System Database to function correctly.

CAUTION: Memory settings that are too low could lead to faulty and unpredictable station behavior. When possible, use more than the minimum settings to optimize performance.

Number of Entities	Number of Tags	Number of Relations	-Xmx (MaxHeapSize)
<= 1,579,000	<= 32,500,000	<= 3,210,000	6G

Depending on the number of stations being indexed, you may need to increase the MaxHeapSize beyond the minimum suggested settings. If starting your Supervisor station from the platform (Daemon Home), which is the preferred method, you need to modify these settings in the `nre.properties` file located in the Daemon Home (typically, `C:\ProgramData\Niagara4.x\distech\etc`). If starting the Supervisor station from the command line in EC-Net 4 Pro for test purposes, you need to modify these settings in the `nre.properties` file located in your User Home directory (`C:\Users\userName\Niagara4.x\distech\etc`).

```
station.java.options=-Dfile.encoding=UTF-8 -Xmx6G
```

Other requirements

When the Supervisor queries the System Database for information about subordinate stations (via NEQL), the resulting entities are resolved in the form of EC-Net virtual components. Consequently, you must enable virtuals on the Supervisor station.

IMPORTANT: You must set up user permissions for EC-Net virtuals manually in the Supervisor via roles and permissions. For details, see “Authorization management” in the *Station Security Guide*.

Chapter 2 Using the SystemDb

Topics covered in this chapter

♦ Setting up SystemDb for local station indexing
♦ Setting up SystemDb for global station indexing
♦ Querying your System
♦ Accessing hierarchies scoped against the SystemDb
♦ Setting up individual index exports
♦ Setting up individual index imports
♦ Visualizing System Indexing operations

This section describes how to set up the SystemDb for local and global indexing, as well as how to visualize what happens during indexing operations. Also provided are examples of various methods of running queries against the SystemDb.

Refer to the following procedures.

Setting up SystemDb for local station indexing

Set up the SystemDb to run your index queries against the local Supervisor station. This procedure describes using the preconfigured SystemDbService (contained in the niagaraSystemIndex palette), which is already configured with common settings, to quickly set up the SystemDb to index the local station. This can also be done using the individual unconfigured components, however that method is not described here since the preconfigured service settings handles most use cases.

Prerequisites:

- You have connected locally to an existing Supervisor station that is properly licensed and has the required software installed.
- The **`niagaraSystemIndex`** palette is open.

Step 1 In the **`niagaraSystemIndex`** palette, expand **`Typical configuration`** and drag the **SystemDbService** to the Supervisor's **Services** node, and in the pop-up window click **OK**.

The **SystemDbService** is added to the station.

Step 2 Open the **SystemDbService** property sheet, and in the **`System Database Type Selection`** property, select **orientSystemDb** from the drop-down list and click **Save**.

You have chosen OrientSystemDb as the backend database.

Step 3 Expand **SystemIndexService→LocalSystemIndexer**, and set the **`Enabled`** property to `true` to enable local system indexing.

NOTE: The SystemIndexService actually is a distinct and separate service. However, this service is dependent upon selections made in the SystemDbService for its configuration. So the preconfigured SystemDbService component includes a SystemIndexService, LocalSystemIndexer, as well as other components.

Step 4 To configure a time to periodically re-index the local Supervisor station, under the **LocalSystemIndexer** expand **`Execution Time`**. For example, you can set it to re-index every night at 2:00AM.

NOTE: Typically, entities (components, tags, and relations) are static in nature, so it is not necessary to constantly re-index. However, it is a good practice to set up a schedule to re-index periodically.

CAUTION: Indexing a database is a memory- and CPU-intensive operation that affects the availability of your SystemDb and remote stations. Do not configure a re-index to occur too often so your SystemDb and remote stations can maintain availability throughout the day. If your SystemDb is indexed frequently, query performance degrades.

Step 5 To manually trigger a local index operation, right-click **`Local System Indexer`** and select **Actions→Execute** to launch the local system index operation.

NOTE: By default, the **LocalSystemIndexer** component is configured with a set of default query ORDs which (if not disabled) are used to index the local root component and local (non-NiagaraDriver) networks, device folders, devices, point folders, points, and schedules, and populate your SystemDb. This determines which local entities are indexed. Additionally, you can configure the component with custom query ORDs to run in addition to (or in place of) the default set.

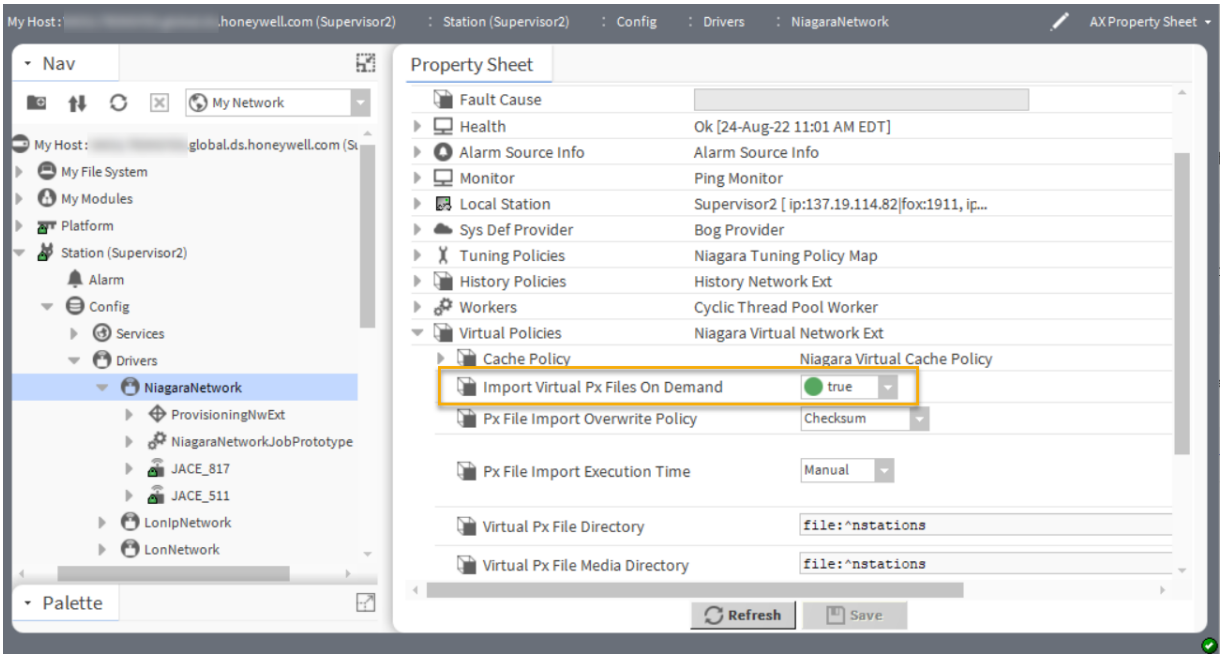
The system index operation populates the SystemDb with entities, tags and relations detected in the local station. At this point you can run NEQL queries against the SystemDb, and the Search results show all of the indexed entities. For information on how to run a scoped search against the SystemDb,

Setting up SystemDb for global station indexing

Set up your single-tier or multi-tier SystemDb to run your index queries against all of the subordinate stations in the Supervisor's NiagaraNetwork. This procedure describes using the preconfigured SystemDbService, which is already configured with common settings, to quickly set up the SystemDb for indexing the subordinate stations.

Prerequisites:

- You have a Supervisor station that is properly licensed and has the required software already installed.
- The Supervisor station has a few configuration settings (such as those listed here) and you have established connections to one or more subordinate stations in the NiagaraNetwork.
  - You may wish to have history import descriptors already set up to import from the subordinate stations to the Supervisor. However, this standard history import functionality is unrelated to system indexing.
  - You may also wish to enable the **`Import Virtual Px Files On Demand`** property under **`Virtual Policies`** in the NiagaraNetwork. This is to allow EC-Net virtuals to attempt to pull px graphics from subordinate EC-BOSs to the Supervisor on demand and caching them locally as you navigate in the virtual space. Since these virtual px files are created on demand and cleaned up after the view changes, the impact on the memory of the embedded platforms is eased.



As of EC-Net 4.13, EC-Net virtuals can pull Px graphics from subordinate stations that are not direct children of the Supervisor station in a multi-tier System Db.

- The **`niagaraSystemIndex`** palette is open.

- You need to enable virtuals on all reachable subordinate NiagaraStations in the Supervisor's NiagaraNetwork and set up user permissions in the category browser.

  IMPORTANT: You must set up user permissions for EC-Net virtuals manually in the Supervisor via roles and permissions. For details, see “Authorization management” in the *Station Security Guide*.

NOTE: You can skip steps 1–2 if you have already performed them in procedure “Setting up SystemDb for local indexing”.

Step 1 In the **`niagaraSystemIndex`** palette, expand the **`Typical configuration`** folder , and drag the **SystemDbService** to the Supervisor’s **Services** node, and in the pop-up window click **OK**.

The **SystemDbService** is added to the station.

Step 2 Open a **Property Sheet** view of the **SystemDbService**, and in the **`System Database Type Selection`** property, select **orientSystemDb** from the drop-down list and click **Save**.

You have chosen OrientDb as the backend database.

Step 3 To navigate to the **Global Niagara Network Indexer**, expand **SystemIndexService→NiagaraNetworkSystemIndexSource**.

NOTE: As of EC-Net 4.13 you have the option to enable **Include Reachable Stations** on the **Niagara Network System Indexer** property sheet if you wish to expand the reach of global system indexing by also including any reachable stations.

Step 4 To manually trigger a global indexing operation, right-click **Global Niagara Network Indexer**, select **Actions** and click **Execute** to launch the system index operation.

On completion, the **`Global Index Last Result`** field shows the number of EC-Net stations that successfully responded to the system index operation.

NOTE: Any subordinate station whose status is “down”, “disabled”, or “fatal fault” is skipped by the indexing operation.

Step 5 To configure a time to periodically re-index the system, under the **Global Niagara Network Indexer** expand **Execution Time** to configure a time to periodically re-index your system. For example, you can configure to re-index every night at 2:00AM.

NOTE: Typically, entities (components, tags, and relations) are static in nature, so it is not necessary to constantly re-index. However, it is a good practice to set up a schedule to re-index periodically.

CAUTION: Indexing a database is a memory- and CPU-intensive operation that affects the availability of your SystemDb and remote stations. Do not configure a re-index to occur too often so your SystemDb and remote stations can maintain availability throughout the day. If your SystemDb is indexed frequently, query performance degrades.

NOTE: By default, the **GlobalNiagaraNetworkIndexer** component is configured with a set of default query ORDs, which (if not disabled) are used to index remote root components and (non-EC-Net driver) networks, device folders, devices, point folders, points, and schedules, and populate your SystemDb. This determines which entities are indexed. Additionally, you can configure the component with custom query ORDs to run in addition to (or in place of) the default queries.

The system index operation populates the SystemDb with entities, tags and relations detected in the subordinate stations. At this point you can run NEQL queries against the SystemDb, and the Search results will show all of the indexed entities.

Querying your System

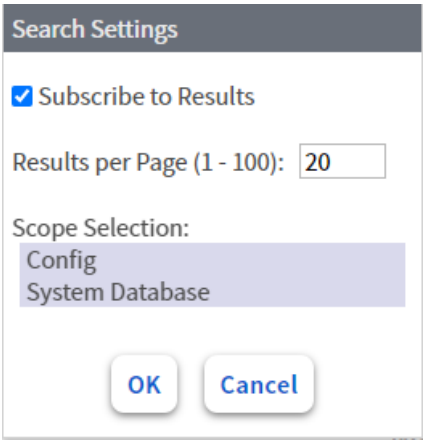
Once the system has been indexed, you can search the System Database. The search results can show all of the indexed entities. You can run the search against either the local station or the SystemDb.

Prerequisites:

- You have an open connection to a Supervisor station (via EC-Net 4 Pro or browser).
- You have already set up the SystemDb and System Index functionality on the Supervisor.

- You have successfully indexed subordinate stations in your **NiagaraNetwork**, and EC-Net virtuals must be enabled and setup properly for user permissions.

Step 1 Open the **Search** sidebar and click **Settings** ( ) to open the **Search Settings** window.

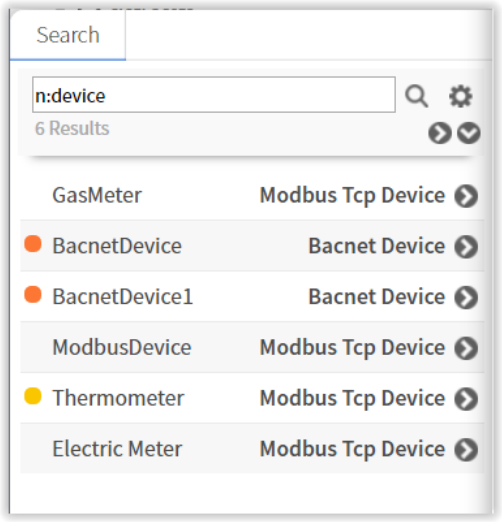


Step 2 Under **Scope Selection**, select **System Database** and click **OK**.

This configures search to run your NEQL queries against the SystemDb.

NOTE: If you want to search against the local station, select **Config**, or you can optionally select both. The query runs against both scopes. Also note that if you use the **LocalSystemIndexer** to index local entities into the System Database, you only need to use the **System Database** search scope (and not **Config**).

Step 3 In the **Search** entry field, enter the NEQL query: `n:device`.



The search results display a list of all devices that were pulled into the SystemDb. Note that any remote devices are modeled as virtual components.

Accessing hierarchies scoped against the SystemDb

You can access hierarchies scoped against the SystemDb, which allows the Supervisor station to navigate hierarchies with data based on entities from the remote EC-BOS stations. The purpose of the System Database is

to allow you to run queries against your entire EC-Net system (the Supervisor and those EC-Net stations connected to it), which allows your searches and hierarchies to span the entire system.

Prerequisites:

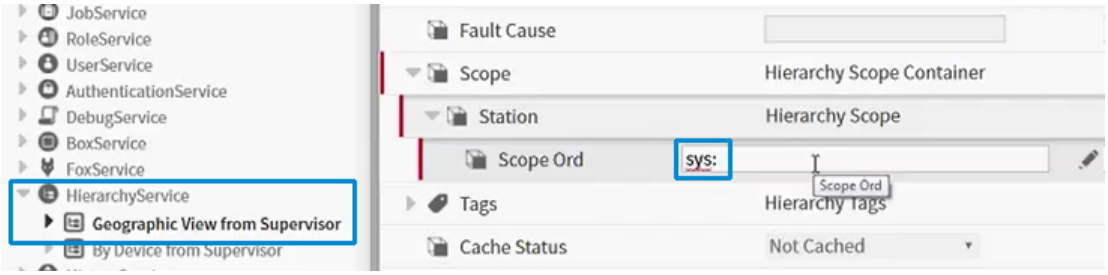
- You have an open connection to a Supervisor station (via EC-Net 4 Pro or browser).
- The SystemDb is installed and configured on the Supervisor.
- In order for any tagged entities to resolve correctly in virtual hierarchies, you must have installed the same tag dictionaries on the Supervisor station as are installed in the subordinate stations.
- Entities in the subordinate stations are already tagged as needed.
- You have the **`hierarchy`** palette open.
- You have successfully indexed subordinate stations in your NiagaraNetwork, and EC-Net virtuals must be enabled and set up properly for user permissions.

Although not a prerequisite, you may want to copy existing hierarchy definitions configured on the subordinate stations and paste those into to the HierarchyService on the Supervisor station. Also, you need to modify those definitions to point the hierarchy scope to the System Database. Otherwise, you need to create a properly scoped hierarchy definition as described here.

Step 1 In the Nav Tree of the Supervisor station, expand the **Services** node.

Step 2 In the **`hierarchy`** palette, drag the **Hierarchy** component to the station's **HierarchyService** and in the **Name** window, enter the desired name. Alternatively, there is a preconfigured SystemDbHierarchy component in the **`hierarchy`** palette that already has the SystemDb ("sys:") scope configured.

Step 3 Open a **Property Sheet** view of the **HierarchyService**, expand **Hierarchy→Scope→Station** and in the **ScopeOrd** field replace the default value by entering: "`sys:`".



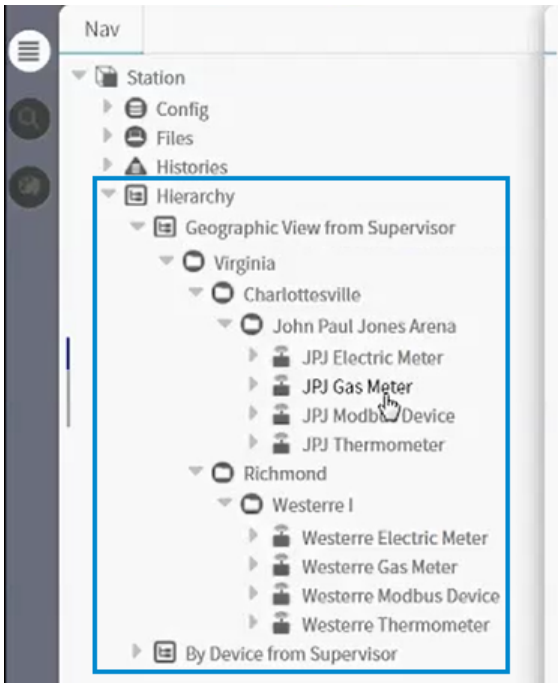
You have changed the Hierarchy scope to run against the SystemDb.

Step 4 Continue to add level definitions as needed (based on objects and tags as used in the subordinate stations and how you wish to navigate those stations). For more details, see procedure "Setting up a Hierarchy Definition".

Step 5 Right-click in the **Nav Tree** and select **Refresh Tree Node**.

Step 6 Expand the station's **Hierarchy** space and navigate throughout the entities (including points and schedules) located on the subordinate stations as organized by the hierarchies.

Results vary based on how your hierarchy is defined, but here is an example of how one such hierarchy might look.



NOTE: If you add a new EC-BOS to the NiagaraNetwork (one that has EC-Net virtuals set up properly for user permissions) and the station has been indexed, the new subordinate station will automatically show up in the Supervisor's Hierarchy space, provided the entities on the subordinate station are properly tagged.

For more details on SystemDb, see the *EC-Net System Database and System Indexing Guide*.

Setting up individual index exports

This procedure describes how to export (push) entity data from a remote EC-Net station to the Supervisor, putting the data into the Supervisor's SystemDb.

Prerequisites:

- You have already set up the SystemDb and System Index functionality on the Supervisor.
- The EC-BOS station must have a configured (active) NiagaraNetwork connection to the Supervisor to push its data there.
- You have the **`niagaraSystemIndex`** palette open.

For cases where you have a particular EC-BOS on which you want to run a different indexing operation, possibly using a select set of index queries, the SystemIndexer device extension can be a container for a NiagaraSystemIndexExport. The added (and enabled) NiagaraSystemIndexExport takes precedence over the **`Global Niagara Network Indexer`** because only one system indexer is allowed to run at any given time.

Step 1 To set up the export descriptor on the EC-BOS station side, in the **Nav Tree** expand the station's NiagaraNetwork.

Step 2 From the palette, expand the **Individual components** folder, and drag the **`SystemIndexer`** (NiagaraSystemIndexDeviceExt) component onto the EC-BOS station's NiagaraNetwork for the NiagaraStation that represents the Supervisor.

Step 3 From the same palette folder, drag the **`NiagaraSystemIndexExport`** component onto the EC-BOS station's SystemIndexer.

Step 4 Double-click the SystemIndexer to open a **Property Sheet** view, expand the NiagaraSystemIndexExport, click on the **Enabled** property and set it to **true**.

This enables the index export. Once enabled, the NiagaraSystemIndexExport takes precedence over the Supervisor's Global Niagara Network Indexer.

Step 5 To configure the Supervisor station to allow the export (push) to be accepted, navigate to the Supervisor station's NiagaraNetwork, for the NiagaraStation that is pushing the data, expand the (NiagaraSystemIndexDeviceExt) and for the **`Receive Exported Index From Remote Station`** property set **Enabled** to **true**.

This extra step is needed to prevent unwanted pushes of index data from being accepted on the Supervisor for security reasons.

NOTE: If using the SystemDbService from the **Typical configuration** folder in the **`niagaraSystemIndex`** palette, it is already configured with the NiagaraNetworkSystemIndexSource and so, the SystemIndexer device extension will already be added on any subordinate stations.

The remaining optional steps are performed on the EC-BOS station.

Step 6 To schedule when the indexer is to run, on the **NiagaraSystemIndexExport** expand **`Execution Time`** and configure the properties as desired.

Step 7 To configure the **NiagaraSystemIndexExport** to use only the custom index queries that you enter:

a. Click the **`Use Default Index Queries`** drop-down list and select **false**.

b. In the **`Custom Index Queries`** field, enter the desired queries.

Step 8 To manually run the indexer, right-click **`NiagaraSystemIndexExport`** and select **Actions→Execute**.

The indexer executes the index queries.

On completion of the indexing operation, the **NiagaraSystemIndexExport** component pushes the indexed entity data from the remote NiagaraStation to the Supervisor, putting the data into the Supervisor's SystemDb.

CAUTION: Indexing a database is a memory- and CPU-intensive operation that affects the availability of your SystemDb and remote stations. Do not configure a re-index to occur too often so your SystemDb and remote stations can maintain availability throughout the day. If your SystemDb is indexed frequently, query performance degrades.

Setting up individual index imports

This procedure describes how to import (pull) entity data from a remote EC-Net station, putting the data into the Supervisor's SystemDb.

Prerequisites:

- SystemDbService and SystemIndexService are already installed and running on the Supervisor station
- Supervisor station has a connection to one or more EC-BOS stations
- You have the **`niagaraSystemIndex`** palette open.

Adding the **NiagaraNetworkSystemIndexSource** to the **SystemIndexService** on the Supervisor triggers a new **NiagaraSystemIndexDeviceExt**, named **System Indexer**, to be added to all EC-Net stations in the NiagaraNetwork of the Supervisor station.

As of EC-Net 4.13, you can also add and configure individual index imports against reachable stations. Similar to the **NiagaraSystemIndexDeviceExt** mixin, a **ReachableStationSystemIndexDeviceExt**, also named **System Indexer**, is automatically added as a mixin under any reachable station components. Therefore, you can follow similar steps to add a **NiagaraSystemIndexImport** to that location for setting up an individual index import from a particular reachable station.

NOTE: If using the SystemDbService from the Typical configuration folder in the niagaraSystemIndex palette, it is already configured with the NiagaraNetworkSystemIndexSource and so, the SystemIndexer device extension will already be added on any subordinate stations.

For cases where you have a particular EC-BOS on which you want to run a different indexing operation, possibly using a select set of index queries, the SystemIndexer device extension can be a container for a NiagaraSystemIndexImport. The added (and enabled) **NiagaraSystemIndexImport** takes precedence over the **Global Niagara Network Indexer** because only one system indexer is allowed to run at any given time.

Step 1 In the **Nav Tree**, expand the Supervisor station's NiagaraNetwork and a subordinate station.

Step 2 Under the subordinate station, open a **Property Sheet** view of the **SystemIndexer** device extension.

Step 3 From the palette, expand the **Individual components** folder, and drag the **NiagaraSystemIndexImport** component onto the **SystemIndexer**.

The **NiagaraSystemIndexImport** is added to the **SystemIndexer** device extension.

Step 4 Expand the **NiagaraSystemIndexImport**, set the **Enabled** property to **true**.

This enables the index import. Once enabled, the **NiagaraSystemIndexImport** takes precedence over the **Global Niagara Network Indexer**.

Step 5 If desired, expand **Execution Time** to schedule when the indexer is to run.

Step 6 Set the **Use Default Index Queries** to `false`.

This configures the SystemIndexer to use only the custom index queries that you enter.

Step 7 In the **Custom Index Queries** field, enter the desired queries.

Step 8 If desired, to manually run the indexer. right-click **NiagaraSystemIndexImport** and select **Actions→Execute**

The indexer executes the custom queries.

On completion of the indexing operation, **NiagaraSystemIndexImport** component pulls the indexed entity data from the remote EC-Net station to the Supervisor, putting the data into the Supervisor's SystemDb.

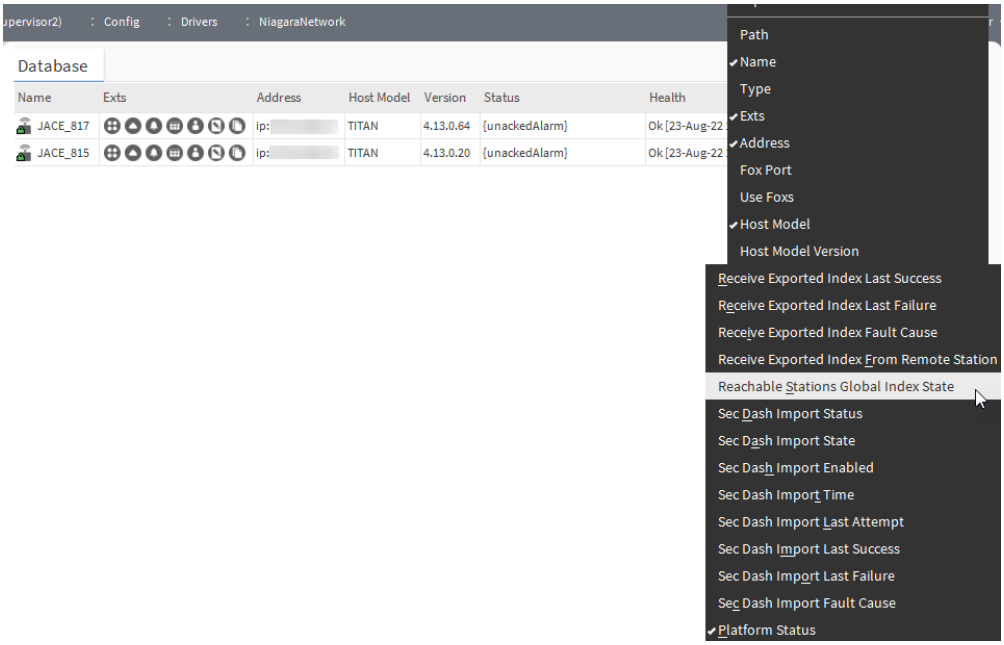
CAUTION: Indexing a database is a memory- and CPU-intensive operation that affects the availability of your SystemDb and remote stations. Do not configure a re-index to occur too often so your SystemDb and remote stations can maintain availability throughout the day. If your SystemDb is indexed frequently, query performance degrades.

Visualizing System Indexing operations

On the Supervisor station, you can use the **StationManager** view and **Spy Remote** view to see results and various details of System Indexing operations. Similarly, you can configure the JobService to display a job for every System Index operation.

Columns for viewing system indexing results are optionally available in the **Station Manager** view on the NiagaraNetwork.

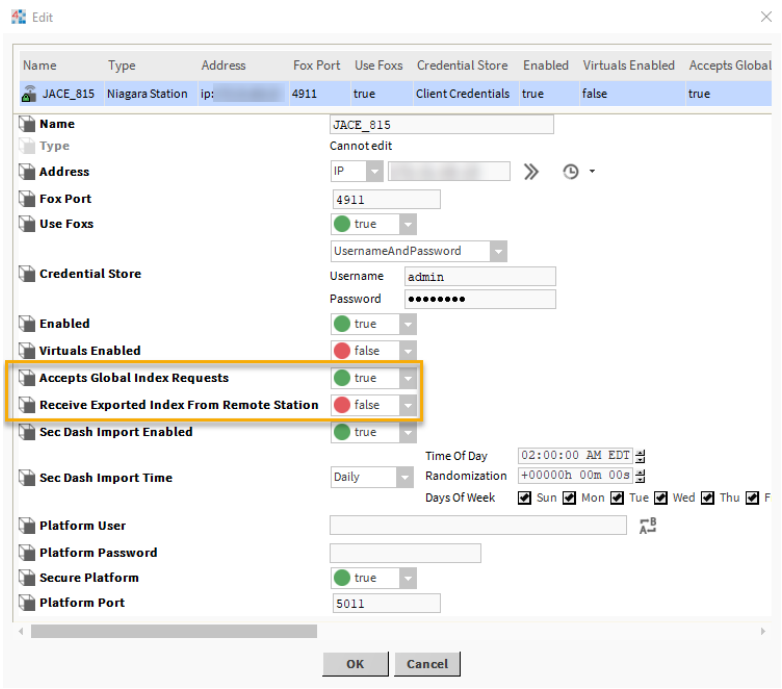
Figure 1 Station Manager view optional columns for viewing system indexing results



As of EC-Net 4.13, to view the indexing progress of the downstream reachable stations, you can select **Reachable Stations Global Index State** from the list of optional columns.

Additionally, the **Station Manager** view allows for batch editing of the **`Accepts Global Index Requests`** and **`Receive Exported Index From Remote Station`** properties.

Figure 2 Station Manager view batch editing properties



The following topics describe several methods of visualizing what is happening during system indexing.

Viewing System Indexing operations via JobService

Use the JobService to visualize what happens during system indexing by displaying every system index operation in the Job log.

Prerequisites:

- You have already set up the SystemDb and System Index functionality on the Supervisor.
- You have successfully indexed subordinate stations in your NiagaraNetwork, and EC-Net virtuals must be enabled and setup properly for user permissions.

Step 1 In theSupervisor station, expand **Services** and double-click **DebugService**.

The **LoggerConfiguration** view opens.

Step 2 In the **`Log Category`** field, enter "`systemIndex.job`", and select `FINE`from the drop-down list.

This sets the log level, which causes a job for every system index operation to be displayed in the **Job Service Manager**.

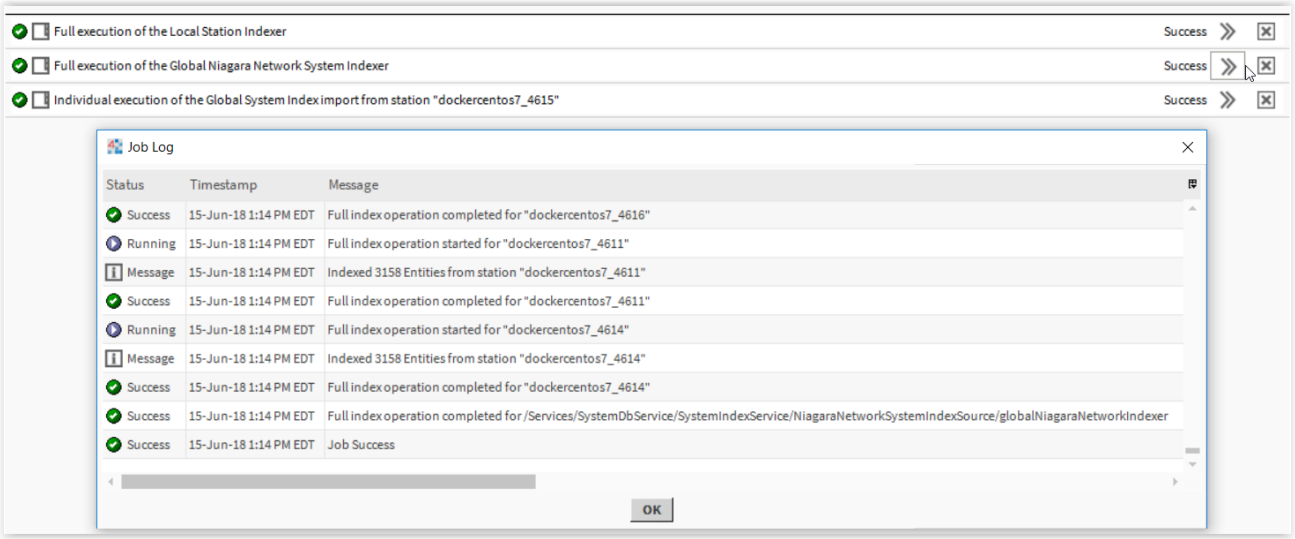
Step 3 Click **Save**.

NOTE: A debug job such as this should only be turned on for troubleshooting purposes since it may affect performance. Also, in addition to reducing performance, it can increase memory use. For those reasons, this log should only be set to FINE for debugging purposes, and then turned back off again.

Step 4 Expand the Supervisor station's **SystemIndexService→NiagaraNetworkSystemIndexSource**, right-click **`Global Niagara Network Indexer`** and select **Actions→Execute**to manually trigger the indexing operation.

Step 5 To open the **Job Service Manager** view, double-click **JobService**, select the system index job of interest, and click the browse icon (>>) to open the report in the **Job Log** window.

This job log report shows which EC-BOSs were actually indexed and shows what entities were indexed.



The **JobService** limits system indexing job logs to just three, and at this point any subsequent system indexing job replaces the oldest one in the list.

Viewing System Indexing details via Spy Remote

The **Spy** view available on various system indexing components can be useful to superusers to see system indexing diagnostic information, providing statistics at every level. For example, you can see global indexing statistics such as the total number of attempts, average execution time, and which operation took the longest

amount of time to complete. For statistics at a more granular level, you can look at the Spy statistics for an individual indexer component to see that last time it attempted and how indexing occurred.

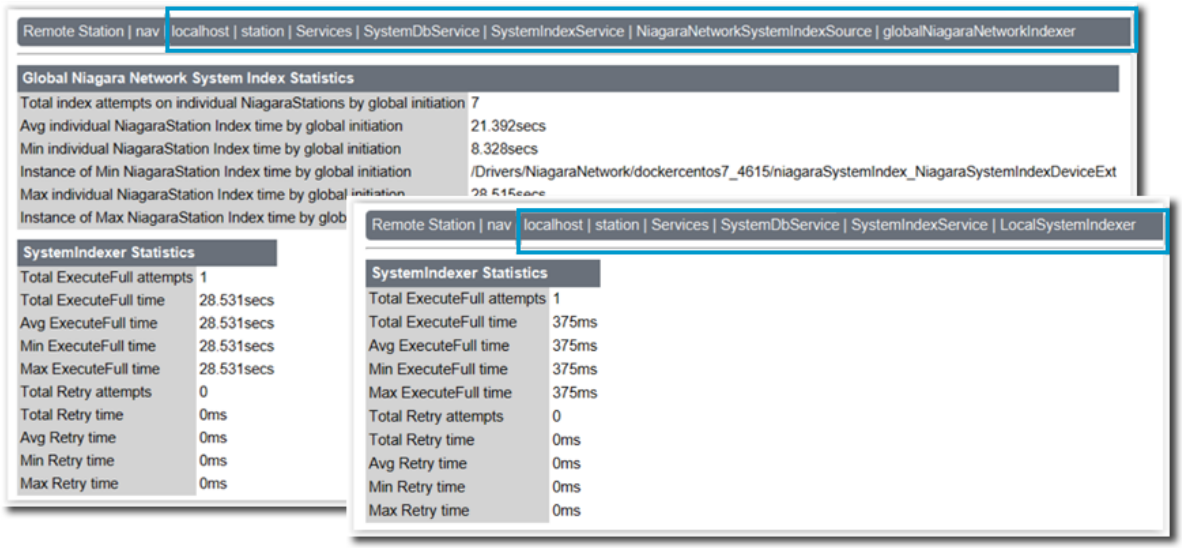
Prerequisites:

- Connection to your Supervisor station and its subordinate stations
- SystemDbService and SystemIndexingService are already installed and configured
- You have successfully executed system indexing operations on the Supervisor and its subordinate stations.

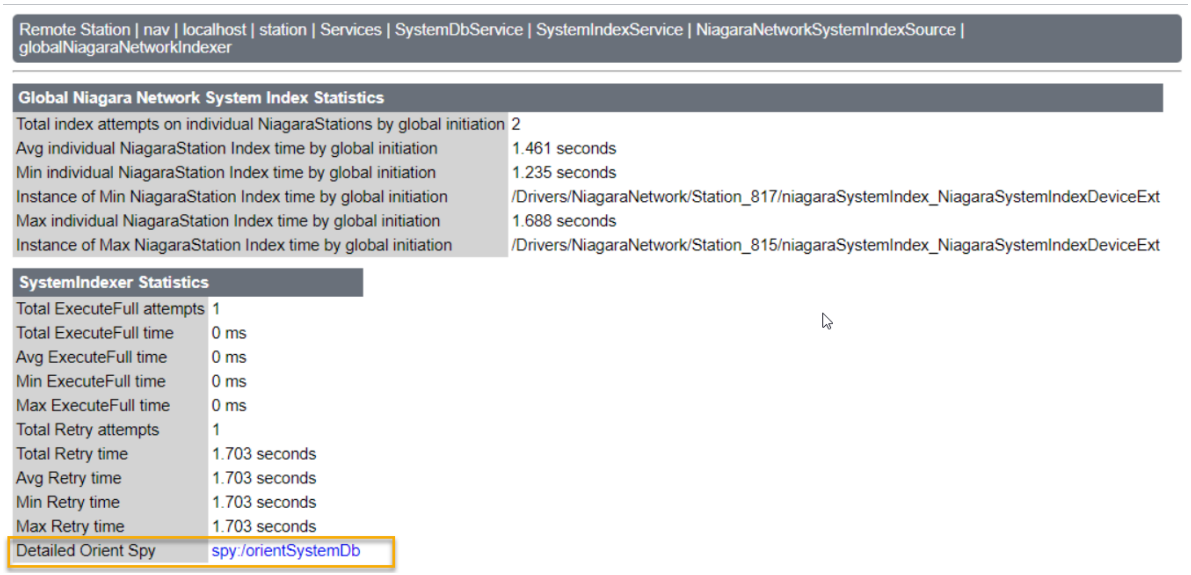
Step 1 In the EC-Net 4 Pro **Nav tree** or **Property Sheet** view, in the Supervisor station, expand **Services→SystemDbService→SystemIndexService** and perform either of the following actions:

- Right-click **LocalSystemIndexer** and click **Views→Spy Remote** to view system indexing statistics on the local station.
- Right-click on **GlobalNiagaraSystemIndexer** and click **Views→Spy Remote** to view system indexing statistics on the subordinate stations.

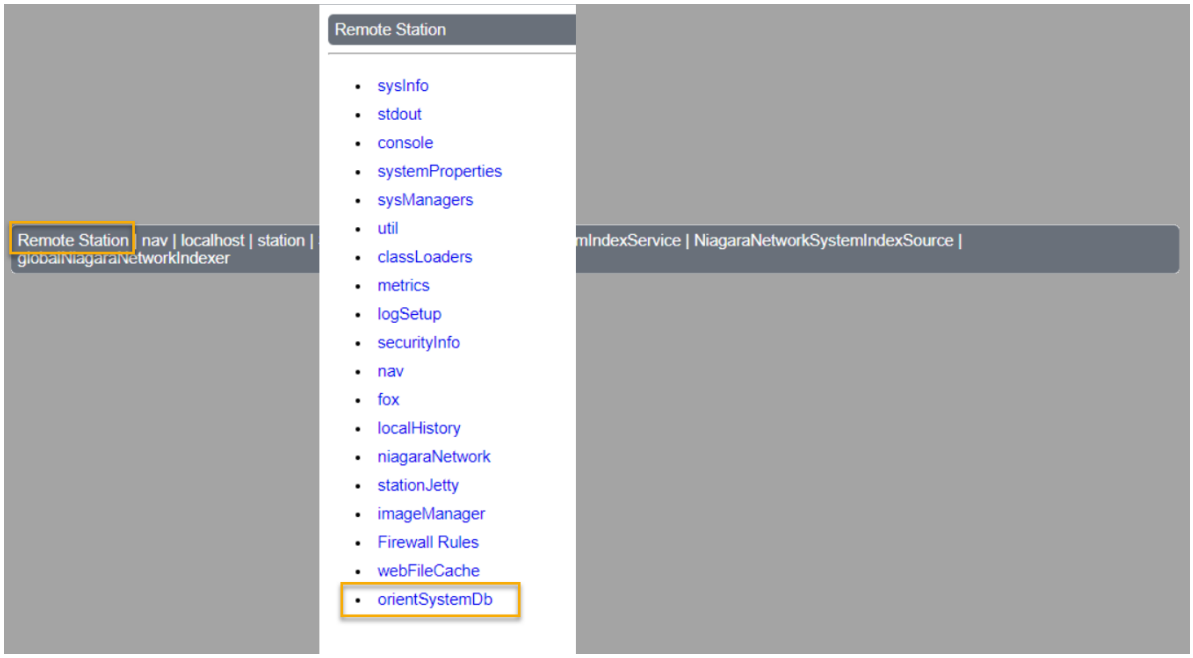
Step 2 Scroll down to see available statistics on the indexing operations, similar to those shown here.



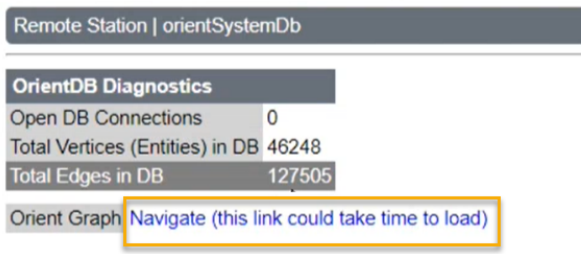
Step 3 As of EC-Net 4.13, you can click the **Detailed Orient Spy** link to drill down even further into what is contained in the SystemDb. The spy gives insight into the entities (ex: tags, relations) currently indexed in the SystemDb.



Alternatively, you can access the **Detailed Orient Spy** in the root of the Remote Spy.

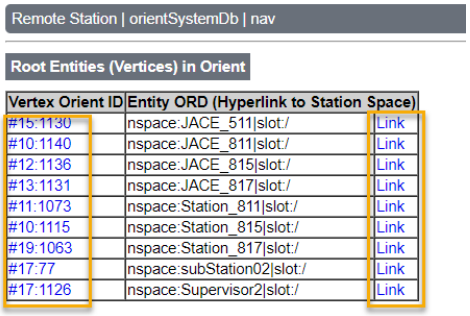


The **OrientDB Diagnostics** page opens and displays an overview of the number of open Db connections, total number of vertices and Edges in the Db.



Step 4 To navigate through the **Orient Graph** to see what the Db contains, click **Navigate (this link could take time to load)**.

A list of discovered root entities (vertices) in the Orient Db is displayed. From here, you can start diving into the root entity of each listed station. The assigned IDs in the **Vertex Orient ID** column are hyperlinks that allow you to continue navigating down in the Spy to downstream entities.. The links in the **Entity ORD (Hyperlink to Station Space)** column navigate you back to the actual component in the virtual space or the station space.

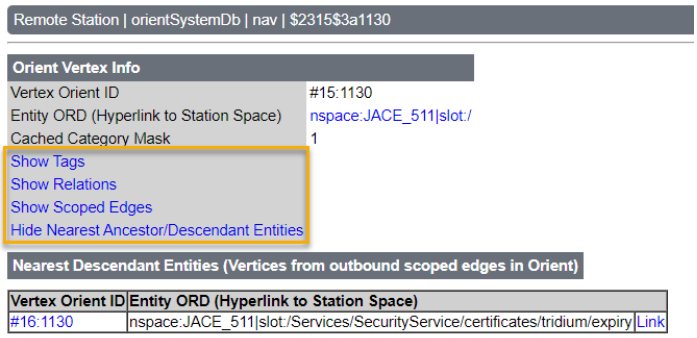


Remote Station | orientSystemDb | nav

Root Entities (Vertices) in Orient

Vertex Orient ID	Entity ORD (Hyperlink to Station Space)	
#15:1130	nspace:JACE_511|slot:/	Link
#10:1140	nspace:JACE_811|slot:/	Link
#12:1136	nspace:JACE_815|slot:/	Link
#13:1131	nspace:JACE_817|slot:/	Link
#11:1073	nspace:Station_811|slot:/	Link
#10:1115	nspace:Station_815|slot:/	Link
#19:1063	nspace:Station_817|slot:/	Link
#17:77	nspace:subStation02|slot:/	Link
#17:1126	nspace:Supervisor2|slot:/	Link

Step 5 To display additional information about the tags, relations, or scoped edges of a particular entity , or if you want to show and hide the nearest ancestor or descendant entities, click the appropriate link as highlighted in the screenshot below.



Remote Station | orientSystemDb | nav | $2315$3a1130

Orient Vertex Info

Vertex Orient ID	#15:1130
Entity ORD (Hyperlink to Station Space)	nspace:JACE_511|slot:/
Cached Category Mask	1

Show Tags
Show Relations
Show Scoped Edges
Hide Nearest Ancestor/Descendant Entities

Nearest Descendant Entities (Vertices from outbound scoped edges in Orient)

Vertex Orient ID	Entity ORD (Hyperlink to Station Space)	
#16:1130	nspace:JACE_511|slot:/Services/SecurityService/certificates/tridium/expiry	Link

Chapter 3 Multi-tier SystemDb indexing

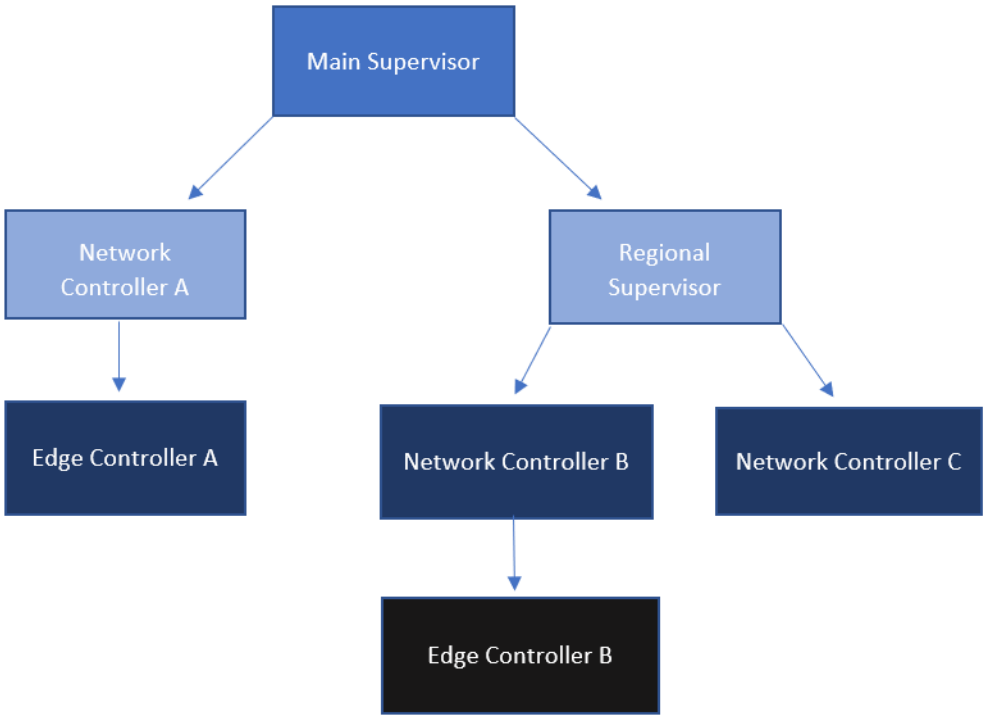
Topics covered in this chapter

♦ Configuring global system indexing to include reachable stations
♦ Viewing reachable stations
♦ Setting up individual system index imports against reachable stations
♦ Viewing reachable stations using Spy Remote
♦ Disabling route to a reachable station
♦ Virtual px view support in multi-tier system

As of EC-Net 4.13, you can use a multi-tier systemDb that allows you to crawl and index entities from stations that are not directly subordinate to your Supervisor station.

As a result, any downstream connection to other stations, which can be routed through a direct subordinate EC-Net station, allows those stations to be reached by the Supervisor. For the Supervisor to see the mid-tier stations, it is not necessary to push information directly to the Supervisor via proxy points. However, EC-Net virtuals need to be enabled and properly configured for access control along the entire route to each reachable station.

You can then configure the system indexing operation on the Supervisor to crawl all reachable stations in addition to the direct subordinate stations by querying for entities that match a certain criteria. It will then store those indexed entities (components, tags, relations) in the Supervisor's system database.



Requirements: Any stations that store the SystemDb, such as the top-most station, must be a Supervisor station. The top-most and mid-tier stations must be upgraded to EC-Net 4.13 or later. You can index EC-Net 4 v4.4 stations and later as long as they have an edge-most position within the network.

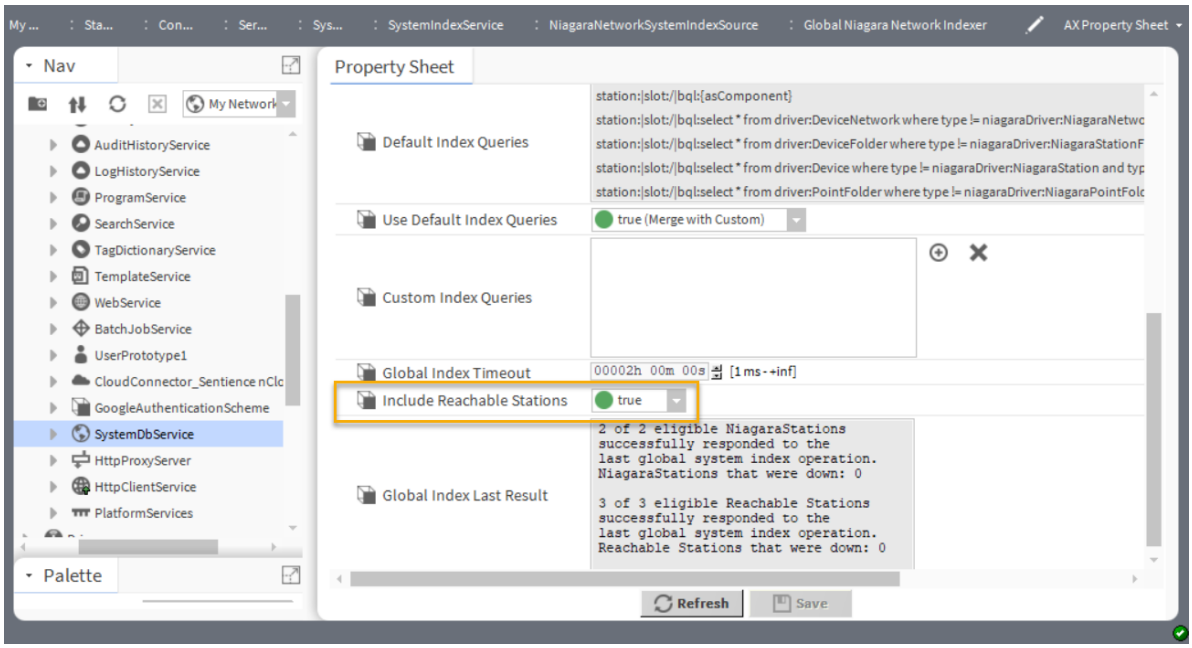
Configuring global system indexing to include reachable stations

You can expand the reach of global system indexing by setting up your multi-tier SystemDb to run your index queries against subordinate stations including all reachable stations in the Supervisor's NiagaraNetwork. The setup procedure is similar to the procedure of the SystemDb setup for the global indexing. For more details, see Setting up SystemDb for global station indexing, page 12.

Prerequisites: You have configured your multi-tier SystemDb according to the settings of global system indexing.

Optional: You have also enabled the **`Import Virtual Px Files On Demand`** property under **`Virtual Policies`** in the NiagaraNetwork to allow virtuals to pull px graphics from reachable stations to the Supervisor

Step 1 To include all reachable stations into global system indexing, expand **SystemIndexService→NiagaraNetworkSystemIndexSource→GlobalNiagaraNetworkIndexer** and enable **Include Reachable Stations** property.



Step 2 On the next execution of the **GlobalNiagaraNetworkIndexer**, either manually invoked or via the trigger, reachable stations are included in the results.

Viewing reachable stations

For a Supervisor station to successfully index all reachable stations in its NiagaraNetwork, it needs to know about their existence. The Supervisor station creates a routing table of stations to target them for indexing and resolve virtuals to them by using nspace ORDs.

Prerequisites:

- You are connected to a Supervisor station that is properly licensed.
- You have upgraded any direct (mid-tier) subordinate stations and other downstream mid-tier stations that you wish to participate to EC-Net 4.13.
- You have active downstream connections (via the NiagaraNetwork) with EC-Net virtuals enabled with access control properly configured along all routes.
- The edge-most stations can run EC-Net versions

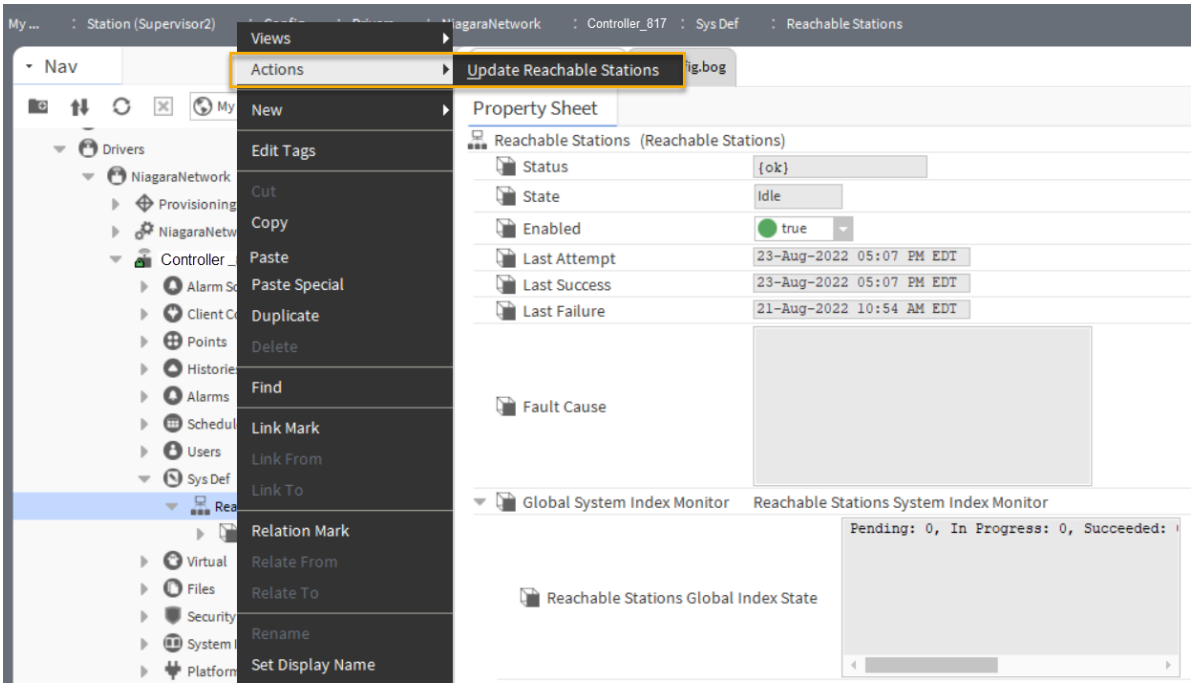
Reachable stations do not appear in the **Reachable Stations** container under the **Sys Def** device ext of the NiagaraStations of the Supervisor's NiagaraNetwork unless one or more of the following conditions apply.

NOTE: The second two conditions outlined below are not a common practice and are likely to only be executed when you diagnose reachable station problems. The first condition is the most likely practice, which would cause reachable stations to appear.

- The **Global Niagara Network Indexer** was executed with **`Include Reachable Stations`** set to `true`, which forces an update of all reachable stations under the covers (see Configuring global system indexing to include reachable stations, page 26). This is the most common way reachable stations get populated in their containers, which may result in you not having to navigate to the individual reachable station containers.
- You have run the remote spy for finding reachable stations (see Viewing reachable stations using Spy Remote, page 29)
- You have manually invoked the **Update Reachable Stations** action on this container previously.

The **Reachable Stations** container component is available on the level of each NiagaraStation for security reasons. For example, if you have already set up permissions or roles per NiagaraStation, the stations' **Reachable Stations** container inherits those settings.

Step 1 To identify all reachable stations in your Supervisor's NiagaraNetwork, expand **Config→Drivers→NiagaraNetwork→NiagaraStation→SysDef**.



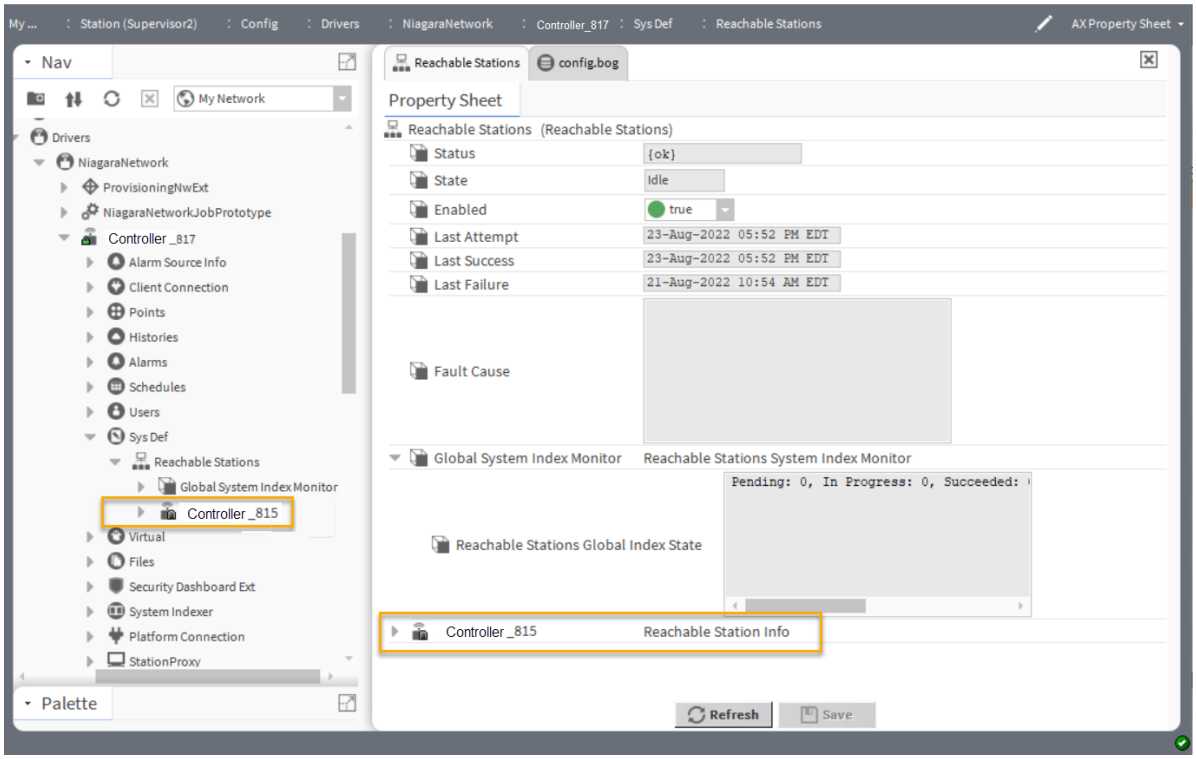
Step 2 Right-click **Reachable Stations** and select **Actions→Update Reachable Stations**.

Step 3 On the **Update Reachable Stations** window, click **Yes**.

Step 4 From the drop-down menu, you can choose one of the two following options and click **OK**:

- If you have already found stations and you want to add only the newly discovered ones, select **Only append new reachable stations**.
- If you want to remove the stations that are already found and rebuild all reachable stations, select **Clear and Rebuild All Reachable Stations**.

In this example, as long as the connections along the route are operational at the time of invocation, the Supervisor finds reachable stations accessible through "Controller_817", which is "Controller_815" and adds it to the **Reachable Stations** property sheet.



Setting up individual system index imports against reachable stations

You can add and configure individual index imports against reachable stations.

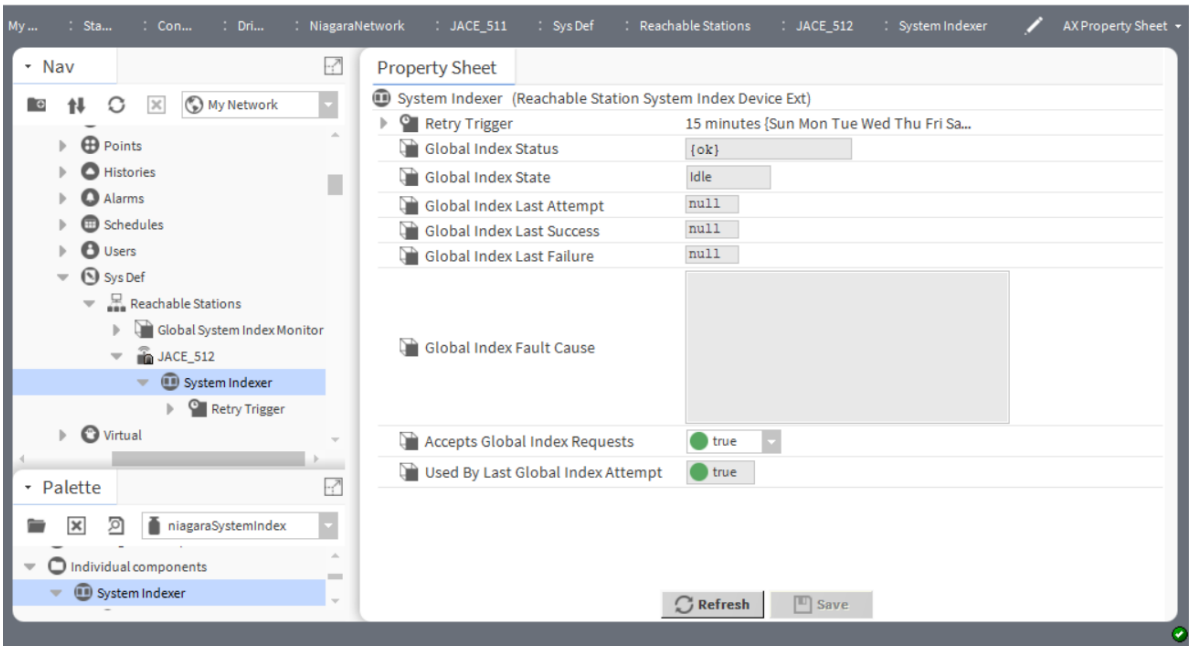
Prerequisites:

- SystemDbService and SystemIndexService are already installed and running on the Supervisor station
- Supervisor station has a connection to one or more EC-BOS stations. In addition, reachable stations have been discovered.
- You have the **`niagaraSystemIndex`** palette open.

Similar to the NiagaraSystemIndexDeviceExt mixin, a **ReachableStationSystemIndexDeviceExt** (also named **System Indexer**) is automatically added as a mixin under any reachable station components.

Step 1 To set up individual system index imports against reachable stations, follow the steps provided in Setting up individual index imports, page 17.

NOTE: For the reachable station case, the individual imports are placed under the **ReachableStationSystemIndexDeviceExt**.



Viewing reachable stations using Spy Remote

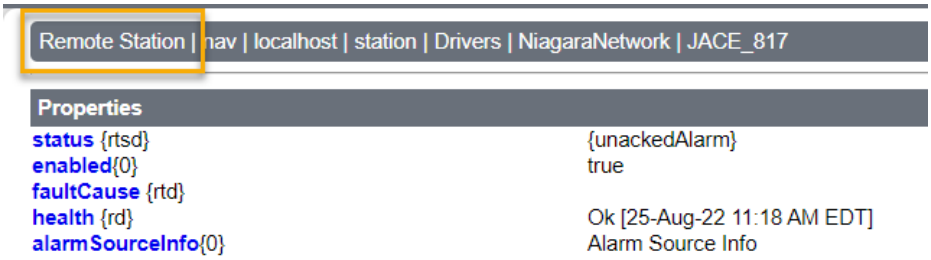
As of EC-Net 4.13, by using the **Spy Remote** view, you can use programmatic calls to display and update reachable stations for your entire **NiagaraNetwork** and details about system indexing operations.

Prerequisites:

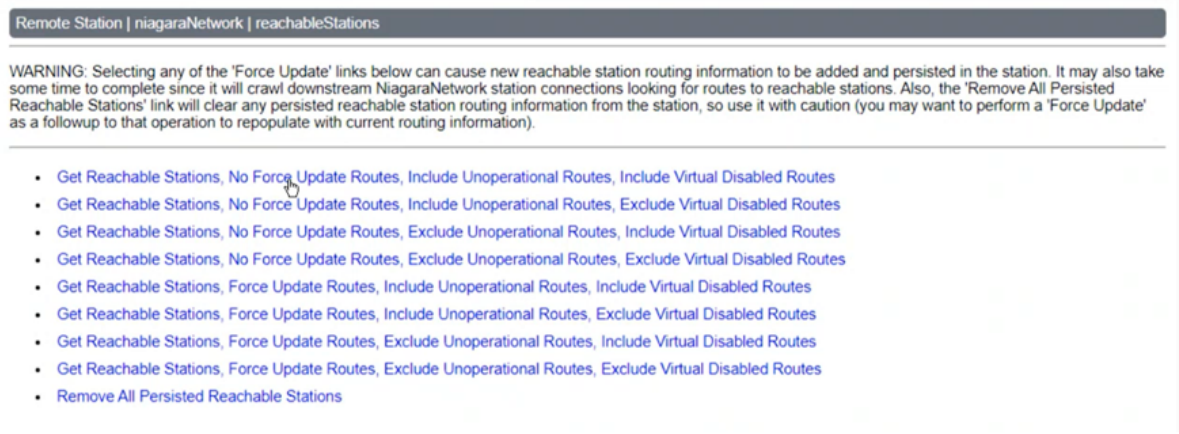
- You have an open connection to a Supervisor station (via EC-Net 4 Pro or browser)
- The Supervisor station has established connections to subordinate stations in the **NiagaraNetwork**.
- You are connected to the Supervisor as a superuser to access any remote spy page.
- The Supervisor and mid-tier stations are upgraded to EC-Net 4.13 or later and have enabled virtuals along reachable station routes.

The **Spy Remote** view identifies the stations that are reachable through subordinate stations.

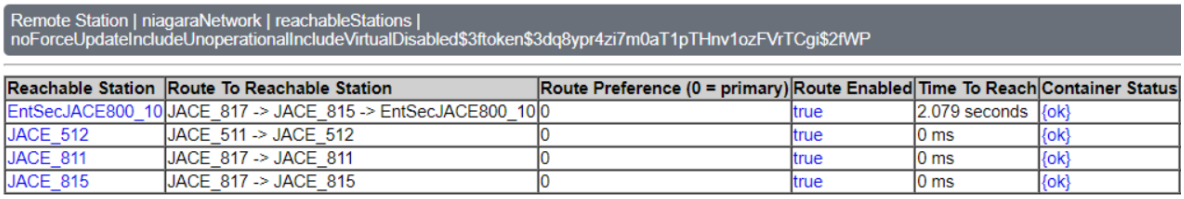
Step 1 To find reachable stations, right-click any component in the station from EC-Net 4 Pro and select **Views→Spy Remote**.



Step 2 On the Web browser view that opens, select the **Remote Station** tab and click **niagaraNetwork→reachableStations**.



Step 3 You can choose if you want to view reachable stations with or without forcing an update of routes.



The resulting routing table displays the stations that are found and the routes to reach them.

NOTE: If there are multiple routes to a station, the smallest number of hops to reach the station takes preference. If the number of hops are identical, the fastest route is favored.

Invoking the call to find reachable stations and to force update routes may create reachable station entries on demand in the corresponding **Reachable Stations** container components back in the station (see Viewing reachable stations, page 26).

Disabling route to a reachable station

You can disable a particular system indexing route to a reachable station if needed.

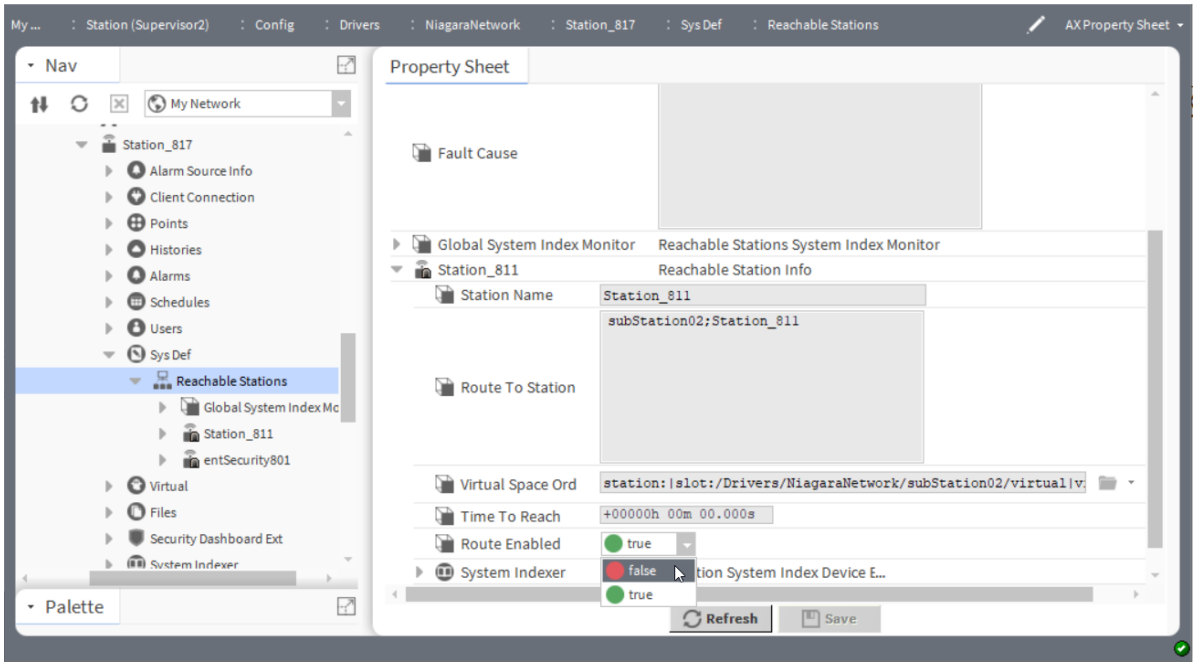
Prerequisites:

- You have an open connection to a properly licensed Supervisor station (via EC-Net 4 Pro or browser)
- The Supervisor station has established connections to subordinate stations in the **NiagaraNetwork** and you have discovered reachable stations (see Viewing reachable stations, page 26).
- The Supervisor and mid-tier stations are upgraded to EC-Net 4.13 or later and have enabled virtuals along the reachable station routes.

Possible use cases can be as follows:

- You have a particular reachable station that you want to exclude from system indexing.
- You have multiple downstream routes to a particular reachable station, and you want to always use a particular route and disable other routes instead of letting EC-Net choose an optimal route on-the-fly.

Step 1 To disable a route to a reachable station, on your Supervisor navigate to the station that starts the route to the target reachable station, which is to be disabled, expand **SysDef** and double-click **Reachable Stations**.



Step 2 Navigate to the property sheet of the reachable station for which you wish to disable the route, set **`Route Enabled`** to `false` and click **Save**.

Virtual px view support in multi-tier system

In a multi-tier EC-Net network with virtuals enabled along the routes, you can also utilize on-demand EC-Net virtual Px views against virtual components from reachable stations. In addition, the SystemDb allows you to easily query to locate and navigate to such virtual components.

As of EC-Net 4.13, multi-tier support of Niagara virtual px views is available, which requires that you enable the **NiagaraNetwork→Virtual Policies→Import Virtual Px Files On Demand** property in the Supervisor.

Virtual Px views are then allowed to load on-demand for both EC-Net virtuals and virtuals to virtuals.

NOTE: The systemDb indexed with entities from reachable stations is only required for EC-Net virtual Px if tag-based bindings are used.

Chapter 4 Components, views, and examples

Topics covered in this chapter

♦ systemDb-SystemDbService
♦ systemIndex-SystemIndexService
♦ systemIndex-LocalSystemIndexer
♦ niagaraSystemIndex-Typical configuration
♦ niagaraSystemIndex-NiagaraNetworkSystemIndexSource
♦ systemIndex-Query OrdList
♦ niagaraSystemIndex-Unconfigured Components
♦ niagaraSystemIndex-Individual components
♦ niagaraSystemIndex-NiagaraSystemIndexDeviceExt
♦ systemIndex-NiagaraSystemIndexExport
♦ niagaraSystemIndex-NiagaraSystemIndexImport
♦ niagaraDriver-ReachableStations
♦ niagaraDriver-ReachableStationInfo
♦ niagaraSystemIndex-ReachableStationsSystemIndexMonitor
♦ niagaraSystemIndex-ReachableStationSystemIndexDeviceExt
♦ Default Index Queries
♦ NEQL query examples

Components include services, folders and other model building blocks associated with a module. You drag them to a property or wire sheet from a palette. Views are plugins that can be accessed by double-clicking a component in the Nav tree or right-clicking a component and selecting its view from the **Views** menu.

The component topics that follow appear as context-sensitive help topics when accessed by:

- Right-clicking on the object and selecting **Views→Guide Help**
- Clicking **Help→Guide On Target**.

Additionally, “NEQL query examples” topic provides grammar and syntax examples to help you construct queries.

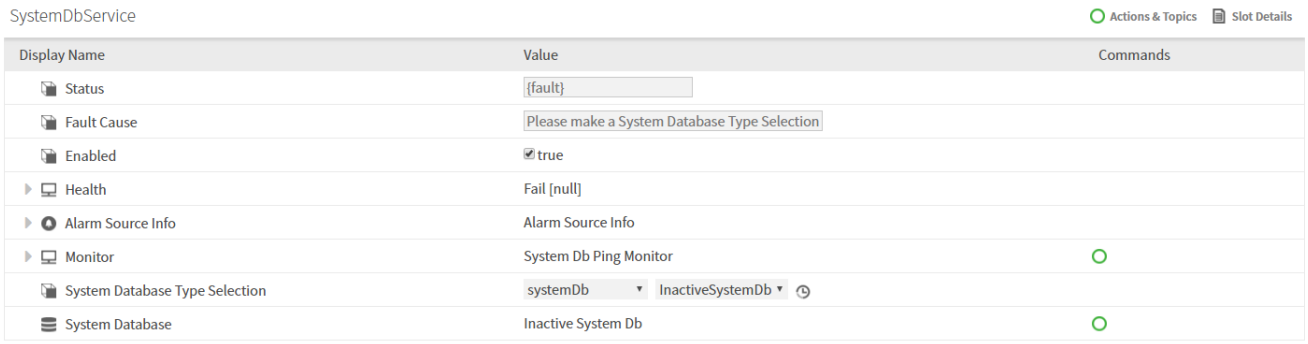
systemDb-SystemDbService

The SystemDbService in the Supervisor station is a generic service used to provide access to the System Database for entity data storage and the ability to query for data persisted in the database.

The database is “pluggable”. The default selection is the InactiveSystemDb. Currently, only the OrientDb selection is available.

OrientIndex configuration components are exposed in the **`orientSystemDb-se`** module palette and allow you to manually drag and drop them under a container under the OrientSystemDb component in a running or offline station. When enabled, the OrientIndex components create orient indexes based on the configured settings and are mainly used to improve query performance for commonly queried tags. When disabled or removed, the corresponding orient index is dropped from the underlying Orient database.

The System Database stores entities, tags, and relations indexed from the Supervisor’s local station, or from subordinate stations in its NiagaraNetwork. NEQL is able to query against the System Database. This allows EC-Net Search and Hierarchies to query against it.



Properties

In addition to standard properties (for example, Status, Enabled, Fault Cause, Alarm Source Info, and Monitor, etc.), this component provides the following configuration properties.

Property	Value	Description
System Database Type Selection	InactiveSystemDb (default), OrientDb	Indicates the type of backend database to be used. When initially installed, the default selection is “InactiveSystemDb”. This default inactive selection is intentional because when OrientDb is selected it creates a lot of files in the station protected home directory. So this default inactive selection prevents that from happening if the SystemDbService is accidentally dropped into a station.
System Database	additional properties	See following section.

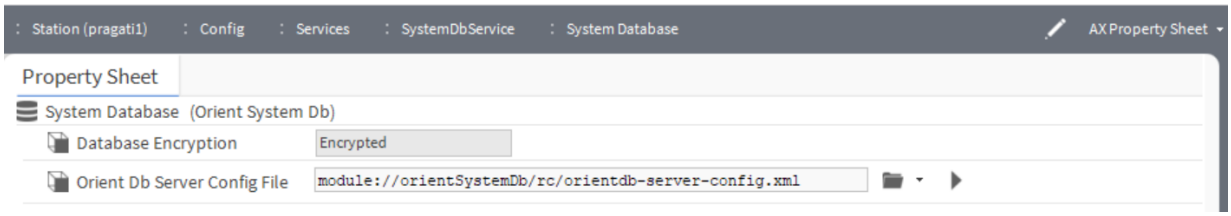
System Database

EC-Net 4 v4.10 and later supports Orient database Encryption. By default database encryption is Enabled. When converting from Orient v2 (4.9 and earlier) to Orient v3 (in 4.10 and later), the Encryption property may degrade query processing performance. This performance degradation may or may not be noticeable. To improve query performance, you can change the Encryption state to Disabled.

To change the encryption state, right click **System Database→Actions→Change Database Encryption**.

NOTE: Changing the encryption state clears the database. To repopulate the database you need to re-index the system.

Figure 3 System Database property sheet



Property	Value	Description
System Database	read-only	Shows selected systemDb.
Orient Db Server Config File	`module://orientSystemDb/rc/orientdb-server-config.xml`	Read-only field, shows the name of the backend server configuration file.
Database Encryption	`Encrypted` (default) or `Unencrypted`	Read-only field, indicates current Encryption state.

Actions list

Five actions are available on **System Database**.

- **Start Database**

  Use Start Database Action Menu to manually start the backend **System Database** process, when the **System Database** is not running. This action automatically pop up when the parent **SystemDbService** Starts. So no need to invoke it manually. It is useful when debugging in conjunction with the Stop Database action.

- **Stop Database**

  Manually stops the backend **System Database** process. This action automatically pop up when the parent **SystemDbService** Starts. So no need to invoke it manually. It is useful when debugging in conjunction with the Start Database action.

- **Delete Station Record**

  Manually deletes entries that are previously indexed into the **System Database** for particular NiagaraStations. Normally system indexing updates the **System Database**, but if a NiagaraStation is manually removed from the NiagaraNetwork or NiagaraStation is renamed then use this action to manually remove old **System Database** entries.

- **Delete All Records**

  Deletes all entries from the **System Database**. It clears and restart the **System Database**. You need to wait for the next scheduled **System Index** operation or manually invoke it to repopulate the **System Database** entries after invoking this action.

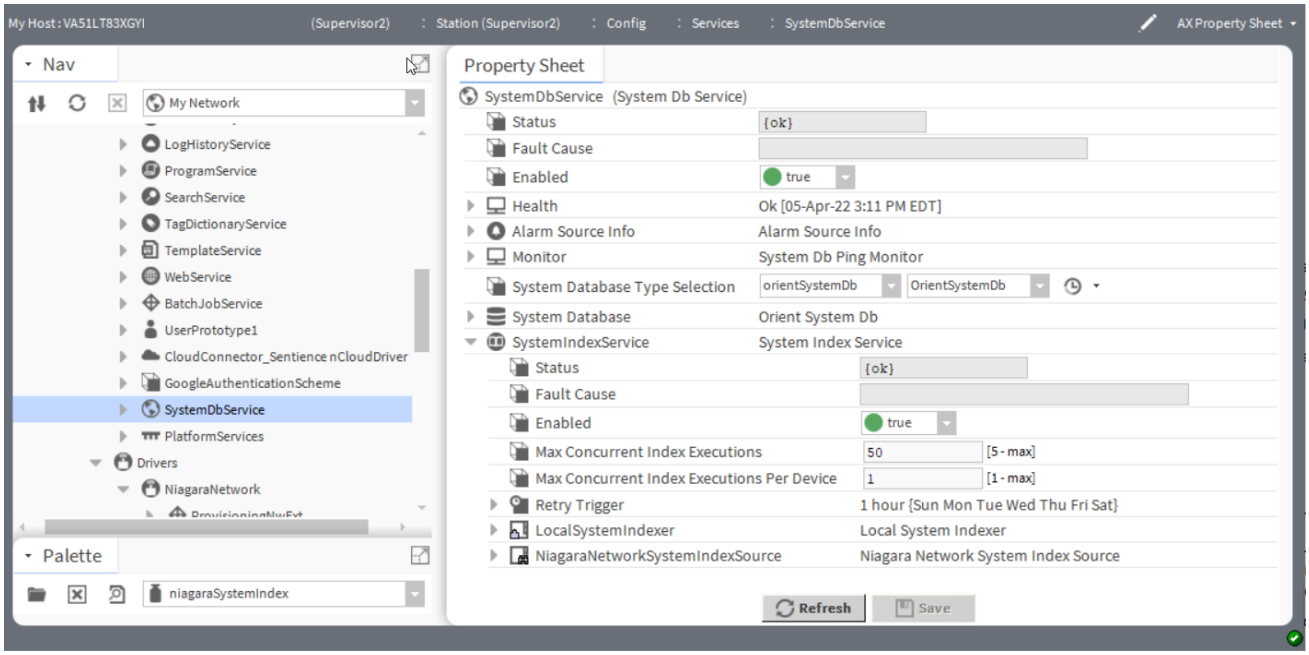
- **Change Database Encryption**

  In EC-Net 4 v4.10 and later, in cases where query performance may be noticeably degraded you can change the encryption state to `Disabled` to improve performance.

systemIndex-SystemIndexService

The purpose of System Indexing is to schedule indexing (and re-indexing) of tagged data (entities) from the local station or any subordinate stations under the NiagaraNetwork. The **SystemIndexService** can be configured to crawl the entire system (local and remote stations) by querying for entities that match certain criteria (networks, devices, points) and storing those indexed entities in the Supervisor station's system database.

IMPORTANT: In EC-Net 4.14, the back-end graph database used by the System Database (OrientDb) has been upgraded from v3.0.X to v3.2.X. As a result, if you have an existing N4Supervisor station already using the System Database that you upgrade to EC-Net 4.14, the upgrade erases the contents of the Supervisor station's existing System Database. Until the next scheduled full re-index of your EC-Net system, this will temporarily cause queries against the System Database and return no results. A subsequent full system re-index repopulates the data in the System Database. Re-indexing typically occurs on a triggered basis, but to repopulate the data quicker, you can manually invoke a full system index immediately after upgrading by invoking the Execute action on any System Indexer in use.



NOTE: The **`tagdictionary`** palette includes two smart tag dictionaries that support the System Database and System Indexing features. The dictionaries allow you to easily exclude portions of a NiagaraStation from indexing operations, which you might want to do for licensing reasons. The SystemIndex dictionary has a smart tag, the scoped tag. If you add this dictionary to your station, you can drop a **systemIndex:excluded** marker tag on a component and any descendants will have this same tag implied on them. To prevent implying this tag on all descendants you can apply a **systemIndex:included** marker tag where necessary. The SystemIndex dictionary has a scoped tag rule that can be configured to imply the **systemIndex:excluded** tag on portions of the station as specified in one or more Ord scopes.

The SystemIndexService is typically installed under the SystemDbService (although it can be placed anywhere under the Services Container). It must be enabled for any child indexers to function. All child indexers have their own configurable execution triggers, but the service also provides a retry trigger that can re-execute any failed indexers at a faster rate.

Properties

In addition to standard properties (for example, Status, Enabled, Fault Cause), this component provides the following configuration properties.

Property	Value	Description
Max Concurrent Index Executions	5–max (defaults 50)	Defines the maximum number of concurrent threads that can be processing system index requests from different sources in parallel.
Max Concurrent Index Executions Per Device	1–max (defaults 1)	Defines the maximum number of concurrent index executions per device (EC-Net station). This includes index executions for reachable stations that route through a particular EC-Net station. This can be important to throttle back concurrent index requests for downstream reachable stations that all route through the same starting EC-Net station device. **NOTE:** This value should never be set equal or larger than the **`Max Concurrent Index Executions`** value.
Retry Trigger	additional properties	See following section.

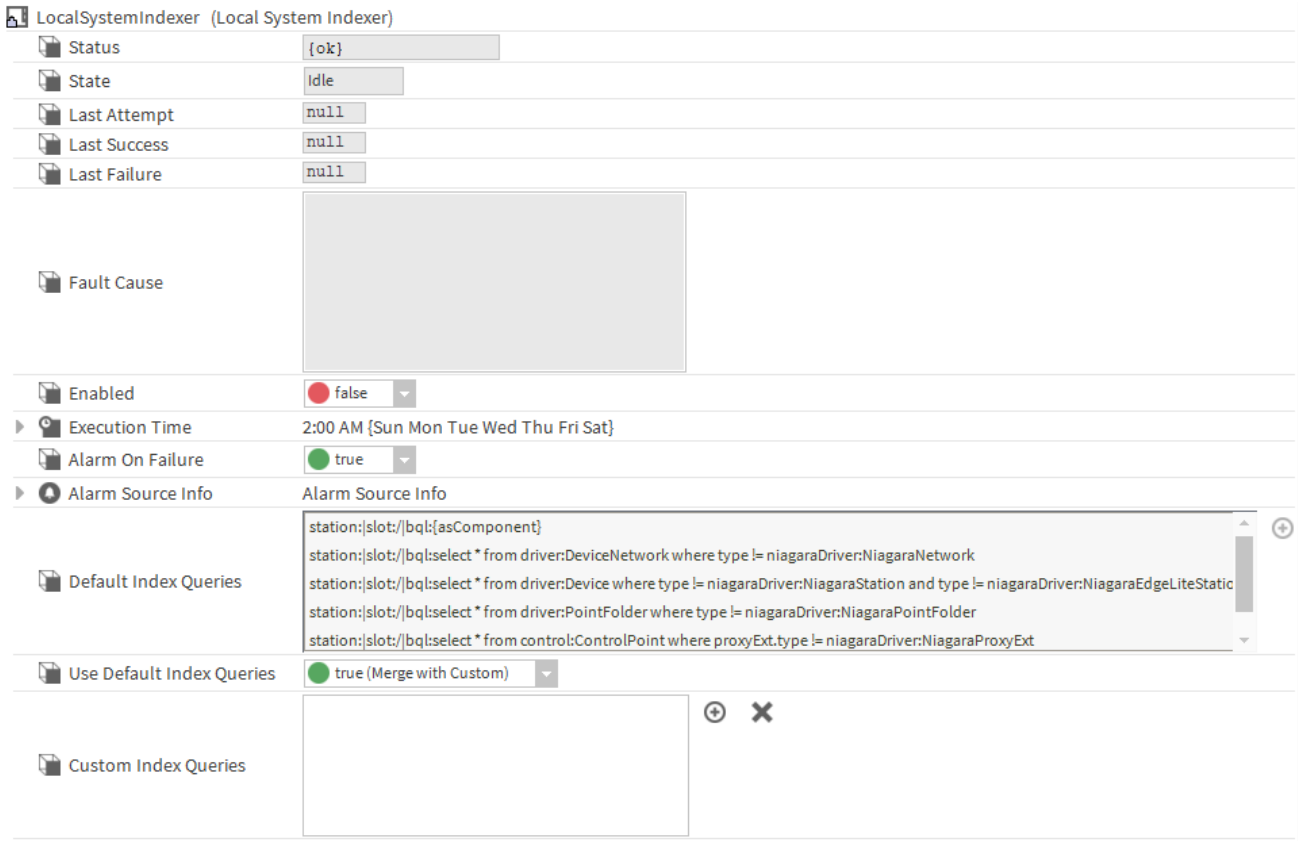
Retry Trigger properties

Property	Value	Description
Retry Trigger	1 hour (all days of the week) (default)	Retries any System Indexer components that had a failure on their last execution (so that they will be retried at a quicker interval than the normal execution time).
Trigger Mode	interval, daily (default), manual	Determines when a TimeTrigger fires. `Daily` fires the trigger at a specific time on selected days of the week, and includes a randomized interval so that the trigger does not fire at exactly the same time every day. Randomized firing is useful when handling large volumes of data. `Interval` fires a trigger each time the specified interval elapses. You would use it to fire the trigger several times per day (for example, every 5 minutes). `Manual` requires a human to fire a trigger. When this trigger fires for a station database, a system indexer (**`LocalSystemIndexer`** or **`GlobalNiagaraNetworkIndexer`**) executes its index operation. **CAUTION:** Indexing a database is a memory- and CPU-intensive operation that affects the availability of your SystemDb and remote stations. Do not configure a re-index to occur too often so your SystemDb and remote stations can maintain availability throughout the day. If your SystemDb is indexed frequently, query performance degrades.
Last Trigger	read-only	Records the timestamp of when last retry occurred.
Next Trigger	read-only	Reports when the trigger is scheduled to fire again.

systemIndex-LocalSystemIndexer

This component is used to enable and configure indexing of the Supervisor station into its own SystemDb. It is optional, only added to the station if you intend to index the Supervisor station. It is included with the SystemIndexService in the **`niagaraSystemIndex`** palette's Typical configuration folder, and is enabled by default. If using Unconfigured components, this can be optionally installed directly under the SystemIndexService on the Supervisor.

Figure 4 Local System Indexer (unconfigured) properties



NOTE: The LocalSystemIndexer component is valid only on a Supervisor station running the SystemDbService. Attempting to enable this component on a remote controller station causes it to go into fault. This is because it will detect that no SystemDbService is installed and running there.

This component is found in the **`systemIndex`** and **`niagaraSystemIndex`** palettes.

Properties

In addition to standard properties (e.g. Status, Enabled, Fault Cause, Alarm Source Info, and etc.), this component provides the following configuration properties.

Property	Value	Description

State	read-only	A frozen property on a component, indicates the current state of operations.
		`Idle` (default)
		`Pending`
		`In Progress`

Last Attempt	read-only (defaults to null)	Reports the date and time of the last attempted execution.
Last Success	read-only	Reports the last time the station successfully performed this function.
Last Failure	read-only	Reports the last time the system failed to perform this function. Refer to **`Fault Cause`** for details.

Property	Value	Description
Execution Time	additional properties	See following sections.
Alarm on Failure	true (default), false	Enables/disables the generation of an alarm on indexing failure.
Alarm Source Info	additional properties	Contains a set of properties for configuring and routing alarms when this component is the alarm source. For property descriptions, refer to the *Alarms Guide*
Default Index Queries	BQL queries (default)	Provides pre-configured default query ORDs. It is a read-only OrdList (frozen property) on several indexing components: LocalSystemIndexer, NiagaraNetworkSystemIndexSource, NiagaraSystemIndexExport, and NiagaraSystemIndexImport). For more details, refer the separate topic in this guide titled “Default Index Queries”.
Use Default Index Queries	`true` (Merge with Custom) (default), `false` (Custom Only)	If true, indexing uses the **`Default Index Queries`** and merges those with any Custom Index Queries that have been entered. If false, indexing uses only the **`Custom Index Queries`** that have been entered.
Custom Index Queries	empty (default), text	Provides an admin-writable OrdList, a frozen property with which to specify additional query ORDs that the framework merges with the **`Default Index Queries`** (unless **`Use Default Index Queries`** is set to `false`). This property is empty unless you have entered custom queries. Click (+) to enter a new custom query.

Execution Time properties

Property	Value	Description
Trigger Mode	interval, daily (default), manual	Determines when a TimeTrigger fires. `Daily` fires the trigger at a specific time on selected days of the week, and includes a randomized interval so that the trigger does not fire at exactly the same time every day. Randomized firing is useful when handling large volumes of data. `Interval` fires a trigger each time the specified interval elapses. You would use it to fire the trigger several times per day (for example, every 5 minutes). `Manual` requires a human to fire a trigger. When this trigger fires for a station database, a system indexer (**`LocalSystemIndexer`** or **`GlobalNiagaraNetworkIndexer`**) executes its index operation. **CAUTION:** Indexing a database is a memory- and CPU-intensive operation that affects the availability of your SystemDb and remote stations. Do not configure a re-index to occur too often so your SystemDb and remote stations can maintain availability throughout the day. If your SystemDb is indexed frequently, query performance degrades.
Last Trigger	read-only	Reports when (by displaying a timestamp) the last trigger fired.
Next Trigger	read-only	Reports when the trigger is scheduled to fire again.

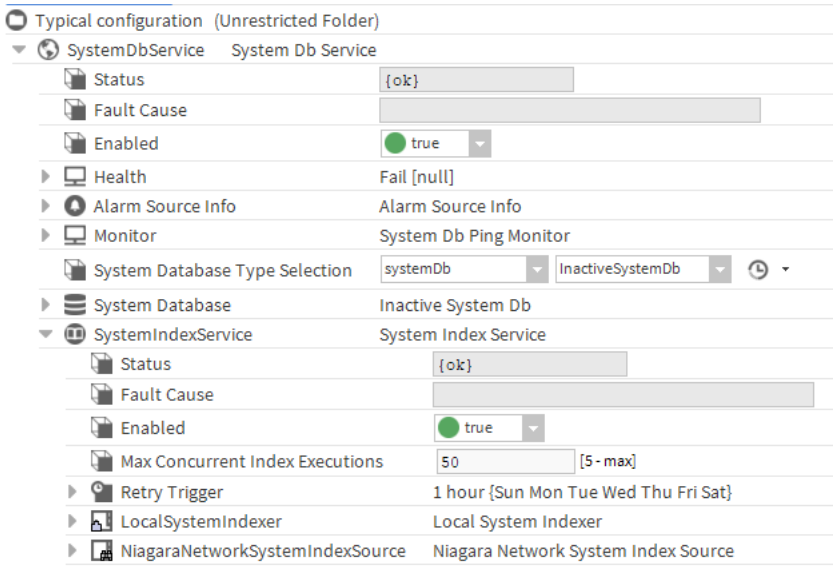
Actions

- **Execute** — Manually trigger the indexing operation
- **Retry failed indexes** — Manually trigger a subsequent attempt on failed indexing operations

niagaraSystemIndex-Typical configuration

This folder contains a set of components required for SystemDatabase operations that are preconfigured with common SystemDatabase settings. These are provided to make it easier to set up the SystemDatabase on your system.

Figure 5 Property Sheet view of Typical configuration folder



Specifically, this provides a SystemDbService component that is already configured with a SystemIndexService. This a separate service however, it works together with the SystemDbService and so it is commonly placed underneath it. Additionally, the SystemIndexService includes both a LocalSystemIndexer and a GlobalNiagaraNetworkIndexer which can optionally be enabled.

This folder is found in the **`niagaraSystemIndex`** palette.

niagaraSystemIndex-NiagaraNetworkSystemIndexSource

When installed and enabled under the SystemIndexService, it enables the NiagaraSystemIndexDeviceExt “mixin” on all NiagaraStations in the NiagaraNetwork. This causes a new **`SystemIndexer`** component property to be added dynamically under any NigaraStation or NiagaraEdgeStation in the Supervisor’s NiagaraNetwork.

This component is found in the **`niagaraSystemIndex`** palette.

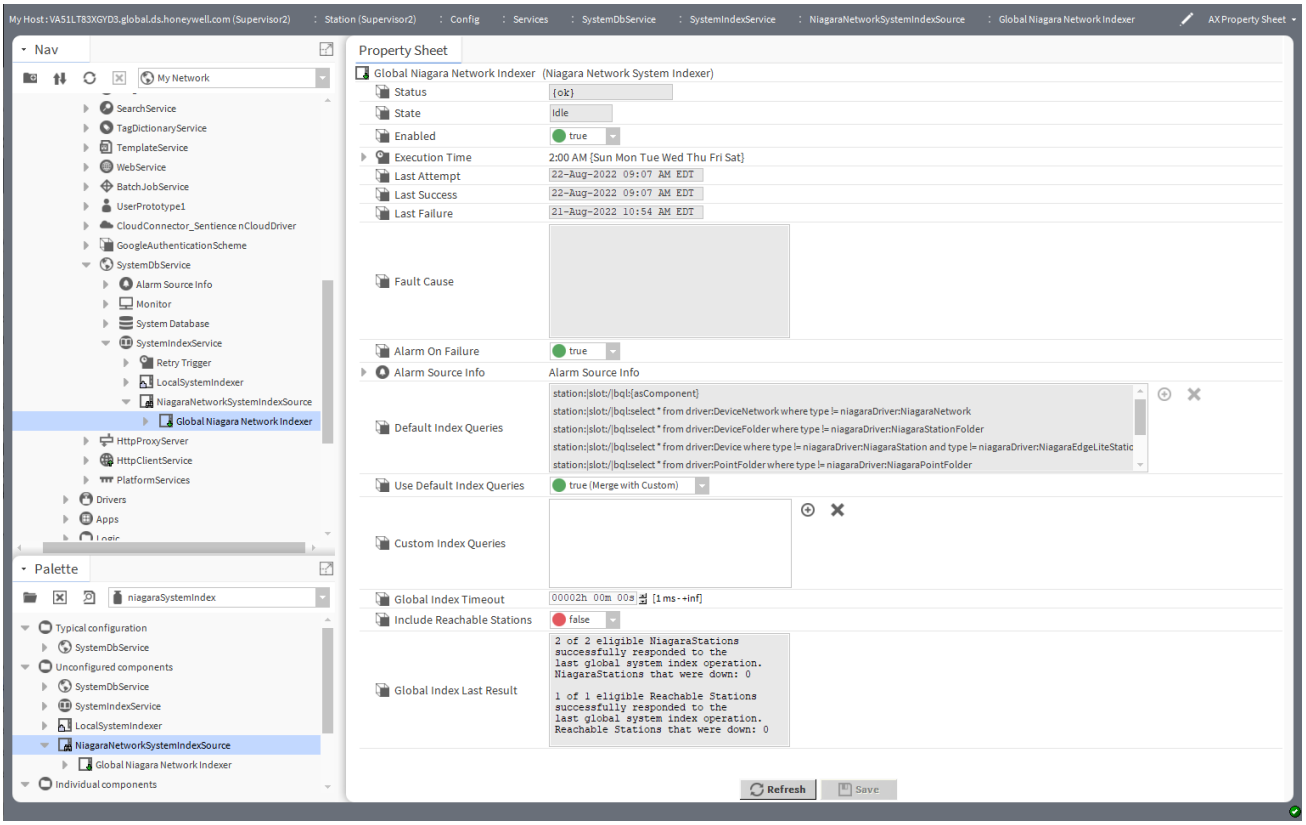
Global Niagara Network Indexer

A frozen property on the NiagaraNetworkSystemIndexSource, which allows for specifying a global index query to periodically run against all active NiagaraStations in the NiagaraNetwork (and optionally their reachable stations), pulling entity data into the Supervisor’s SystemDb.

The global indexing process skips any NiagaraStation that is specifically disabled from global indexing, or that already has an active NiagaraSystemIndexImport in the Supervisor or a corresponding NiagaraSystemIndexExport that is pushing the data from the station. Since there can be only one master index source per NiagaraStation, the order of precedence is as follows.

1. An active NiagaraSystemIndexExport on the remote station
2. An active NiagaraSystemIndexImport on the Supervisor
3. The global NiagaraNetworkSystemIndexer

Figure 6 GlobalNiagaraNetworkIndexer property



Properties

In addition to standard properties (for example, Status, Enabled, Fault Cause, State, Alarm Source Info), the following configuration properties are available.

Property	Value	Description
Last Attempt	read-only, null (default)	Reports the date and time of the last attempted execution.
Last Success	read-only, null (default)	Reports the last time the station successfully performed this function.
Last Failure	read-only, null (default)	Reports the last time the system failed to perform this function. Refer to **`Fault Cause`** for details.
Execution Time	additional properties	See following sections.
Alarm on Failure	true (default), false	Enables/disables the generation of an alarm on indexing failure.
Default Index Queries	BQL queries (default)	Provides pre-configured default query ORDs. It is a read-only OrdList (frozen property) on several indexing components: LocalSystemIndexer, NiagaraNetworkSystemIndexSource, NiagaraSystemIndexExport, and NiagaraSystemIndexImport).

Property	Value	Description
		For more details, refer the separate topic in this guide titled “Default Index Queries”.
Use Default Index Queries	`true` (Merge with Custom) (default), `false` (Custom Only)	If true, indexing uses the **`Default Index Queries`** and merges those with any Custom Index Queries that have been entered. If false, indexing uses only the **`Custom Index Queries`** that have been entered.
Custom Index Queries	empty (default), text	Provides an admin-writable OrdList, a frozen property with which to specify additional query ORDs that the framework merges with the **`Default Index Queries`** (unless **`Use Default Index Queries`** is set to `false`). This property is empty unless you have entered custom queries. Click (+) to enter a new custom query.
Global Index Timeout	00002h 00m 00s (default)	Defines how long the system will wait for a global system index operation to complete. If a global index operation takes this long and is still in progress, it will timeout and assume a failure.
Include Reachable Stations	`false` (default), `true`	Defines whether to include reachable stations in the global system index operations. Reachable stations are those that are not direct NiagaraStations in the local Supervisor’s NiagaraNetwork, but those that are further downstream and accessible via EC-Net virtuals routed through a subordinate NiagaraStation. Setting this property to `true` will cause reachable stations to be discovered (and persisted in the Supervisor). If left at `false`, reachable stations will not be discovered and included, and only direct NiagaraStations in the local Supervisor's NiagaraNetwork will be system-indexed.
Global Index Last Result	read-only	Provides a summary of the results of the last global system index operation. If any eligible NiagaraStations did not index, it is indicated here.

Execution Time properties

Property	Value	Description
Trigger Mode	interval, daily (default), manual	Determines when a TimeTrigger fires. `Daily` fires the trigger at a specific time on selected days of the week, and includes a randomized interval so that the trigger does not fire at exactly the same time every day. Randomized firing is useful when handling large volumes of data. `Interval` fires a trigger each time the specified interval elapses. You would use it to fire the trigger several times per day (for example, every 5 minutes). `Manual` requires a human to fire a trigger. When this trigger fires for a station database, a system indexer (**`LocalSystemIndexer`** or **`GlobalNiagaraNetworkIndexer`**) executes its index operation. **CAUTION:** Indexing a database is a memory- and CPU-intensive operation that affects the availability of your SystemDb and remote stations. Do not configure a re-index to occur too often so your SystemDb and remote stations can maintain availability throughout the day. If your SystemDb is indexed frequently, query performance degrades.
Last Trigger	read-only	Reports when (by displaying a timestamp) the last trigger fired.
Next Trigger	read-only	Reports when the trigger is scheduled to fire again.

Actions

- **Execute** — Manually trigger the indexing operation
- **Retry failed indexes** — Manually trigger a subsequent attempt on failed indexing operations

systemIndex-Query OrdList

The Query OrdList is a frozen property on indexing components (LocalSystemIndexer, NiagaraNetworkSystemIndexer), which contains a list of query ORDs that determine what to be indexed into the Supervisor's SystemDb. These can be NEQL or BQL query ORDs. For more details, see descriptions in systemIndex-LocalSystemIndexer or systemIndex-GlobalNiagaraNetworkIndexer.

For BQL queries, the queries must resolve to entities (or BComponents). This typically means they must be of the form "`station:|slot:/|bql:select * from <typeSpec of a component type>...`.

Only the **`Custom Index Queries`** properties can be modified. The **`Default Index Queries`** properties are read-only. For more details, see the "Default Index Queries" section in the systemIndex-LocalSystemIndexer topic in this chapter.

For security reasons, these ORDs are validated on the server side and only certain ORD schemes are allowed ("`station:`", "`sys:`", "`slot:`", "`h:`", "`neql:`", "`bql:`", and "`namespace:`", "`nspace`" and "`virtual`").

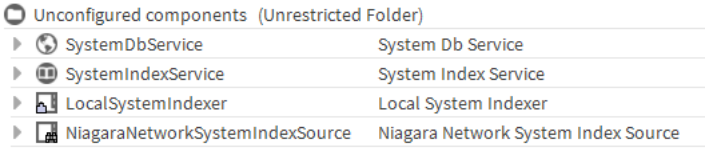
NOTE: Do not create a namespace for a tag dictionary if that namespace is already used for an existing ORD scheme (NEQL or BQL). Since the search service allows for different types of searches (for example, NEQL and BQL), precede the query search with "neql:" to specify that it should run a neql query explicitly, for example: "`neql:h:test`".

niagaraSystemIndex-Unconfigured Components

This folder contains a set of components required for System Database and SystemIndex operations that are not preconfigured.

This folder is found in the **niagaraSystemIndex** palette.

Figure 7 Property Sheet view of Preconfigured Components folder



niagaraSystemIndex-Individual components

The **Individual components** folder contains several components used to index the individual NiagaraStations.

This folder is found in the **niagaraSystemIndex** palette.

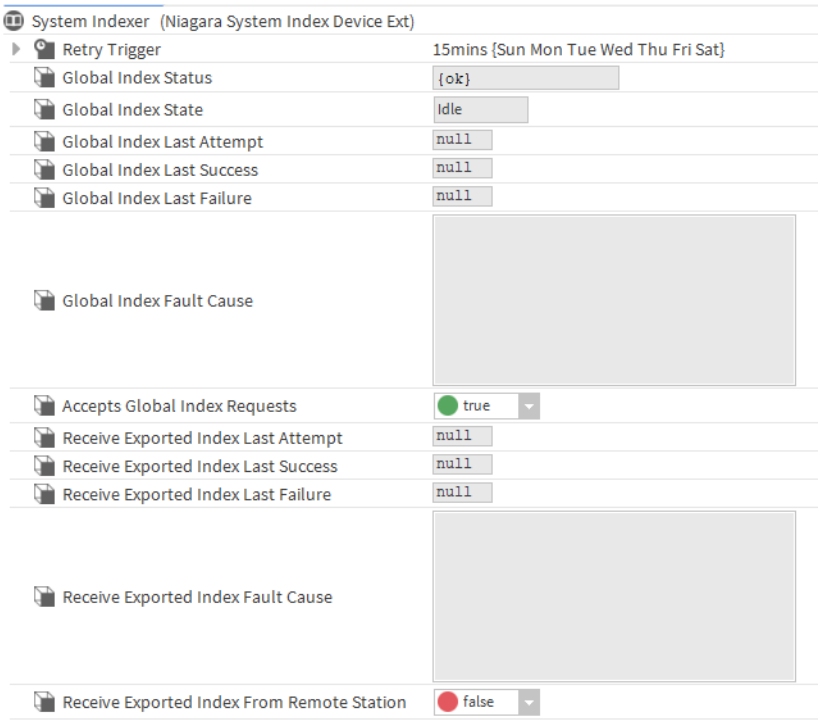
Figure 8 Individual components folder

niagaraSystemIndex-NiagaraSystemIndexDeviceExt

This component is automatically added to NiagaraStations and NiagaraEdgeStations in the Supervisor's NiagaraNetwork when the SystemIndexService is installed. In general, this component manages global index queries sent to it from the global system indexer and executes them for an individual NiagaraStation (or NiagaraEdgeStation), unless an individual import or export is configured.

The only case where you would manually add this component to a NiagaraStation is when you want to set up the EC-BOS to push (export) its index data to the Supervisor. In that case, you would add this component under the NiagaraStation modeling the Supervisor in the EC-BOS's NiagaraNetwork

This component is found in the **niagaraSystemIndex** palette.



Properties

In addition to the standard Retry Trigger, this component provides the following configuration properties.

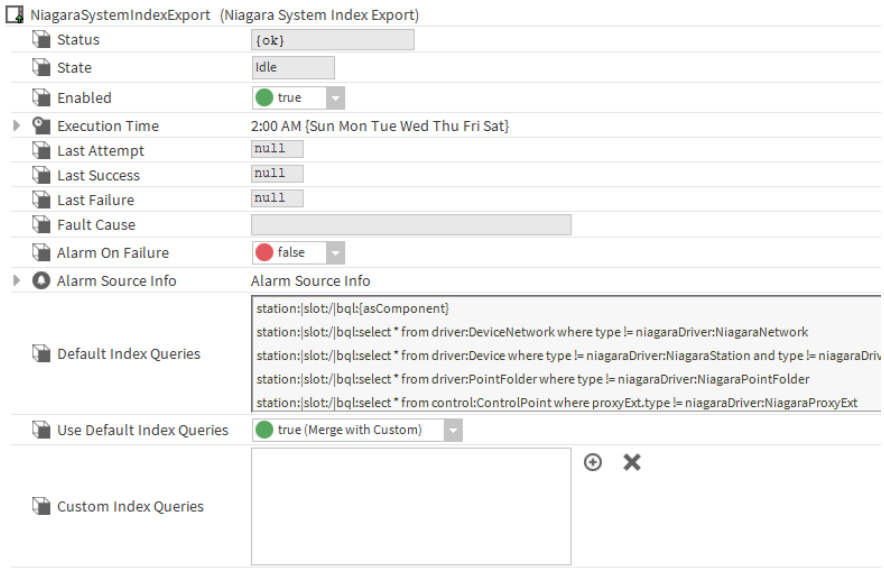
Property	Value	Description
Global Index Status	read-only	Indicates the status of the last global index operation, in the scope of the individual NiagaraStation on which this deviceExt resides.
Global Index State	Idle (default), Pending, In Progress	Indicates the state of the last global index operation, in the scope of the individual NiagaraStation on which this deviceExt resides.
Global Index Last Attempt	read-only	Timestamp of the last attempted global index operation, in the scope of the individual NiagaraStation on which this deviceExt resides.
Global Index Last Success	read-only	Timestamp of the last successful global index operation, in the scope of the individual NiagaraStation on which this deviceExt resides.
Global Index Last Failure	read-only	Timestamp of the last failed global index operation, in the scope of the individual NiagaraStation on which this deviceExt resides.
Global Index Fault Cause	read-only	Indicates the reason why the global index operation is in fault, in the scope of the individual NiagaraStation on which this deviceExt resides.
Accepts Global Index Requests	true (default), false	When “true”, this NiagaraStation is included in the global system indexing operation. If set to “false”, this NiagaraStation is excluded from the global system indexing operation. You might want to do this for licensing reasons, to specify individual NiagaraStations to exclude.

Property	Value	Description
Received Exported Index Last Attempt	read-only	Timestamp of the last attempted received export index operation. This happens only if the remote NiagaraStation is configured to export its index data to the supervisor.
Received Exported Index Last Success	read-only	Timestamp of the last successful received export index operation. This happens only if the remote NiagaraStation is configured to export its index data to the supervisor.
Received Exported Index Last Failure	read-only	Timestamp of the last failed received export index operation. This happens only if the remote NiagaraStation is configured to export its index data to the supervisor.
Received Exported Index Fault Cause		Indicates the reason why the last received export index operation is in fault. This happens only if the remote NiagaraStation is configured to export its index data to the supervisor.
Received Exported Index From Remote Station	true, false (default)	If a NiagaraStation is configured to export its index data to the Supervisor, the Supervisor will accept it only if this property is set to “true”. This prevents security issues caused by any unauthorized remote NiagaraStation pushing its data to the Supervisor. The Supervisor must be explicitly configured to accept it. By default, this is “false” (so that the Supervisor will not accept exports by default).

systemIndex-NiagaraSystemIndexExport

Installed on a NiagaraStation, this component is used to push entity data from the remote NiagaraStation to the Supervisor, putting the data into the Supervisor’s SystemDb. It can be scheduled to execute periodically to re-index the remote station.

This component is found in the **`niagaraSystemIndex`** palette.



Properties

In addition to standard properties (e.g. Status, State, Enabled, Fault Cause, Alarm Source Info, and etc.), this component provides the following configuration properties.

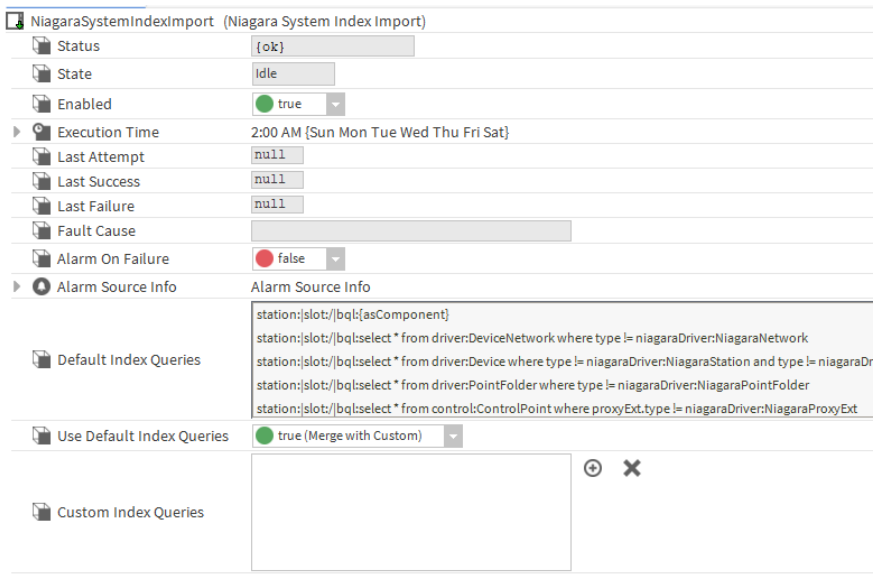
Property	Value	Description
Last Attempt	read-only, null (default)	Reports the date and time of the last attempted execution.
Last Success	read-only, null (default)	Reports the last time the station successfully performed this function.
Last Failure	read-only, null (default)	Reports the last time the system failed to perform this function. Refer to **Fault Cause** for details.
Execution Time	additional properties	See following sections.
Alarm on Failure	true (default), false	Enables/disables the generation of an alarm on indexing failure.
Alarm Source Info	additional properties	Contains a set of properties for configuring and routing alarms when this component is the alarm source. For property descriptions, refer to the *Alarms Guide*
Default Index Queries	BQL queries (default)	Provides pre-configured default query ORDs. It is a read-only OrdList (frozen property) on several indexing components: LocalSystemIndexer, NiagaraNetworkSystemIndexSource, NiagaraSystemIndexExport, and NiagaraSystemIndexImport). For more details, refer the separate topic in this guide titled “Default Index Queries”.
Use Default Index Queries	`true` (Merge with Custom) (default), `false` (Custom Only)	If true, indexing uses the **Default Index Queries** and merges those with any Custom Index Queries that have been entered. If false, indexing uses only the **Custom Index Queries** that have been entered.
Custom Index Queries	empty (default), text	Provides an admin-writable OrdList, a frozen property with which to specify additional query ORDs that the framework merges with the **Default Index Queries** (unless **Use Default Index Queries** is set to `false`). This property is empty unless you have entered custom queries. Click (+) to enter a new custom query.

niagaraSystemIndex-NiagaraSystemIndexImport

Installed on the Supervisor under either a direct NiagaraStation's "System Indexer" device extension or a reachable station's "System Indexer" device extension, this component is used to pull entity data from an individual NiagaraStation, putting the data into the Supervisor’s SystemDb. The Supervisor must be explicitly configured to accept it.

It can be scheduled to execute periodically to re-index the remote station.

This component is found in the **niagaraSystemIndex** palette.



Properties

In addition to standard properties (for example, Status, State, Enabled, Fault Cause, Alarm Source Info), this component provides the following configuration properties.

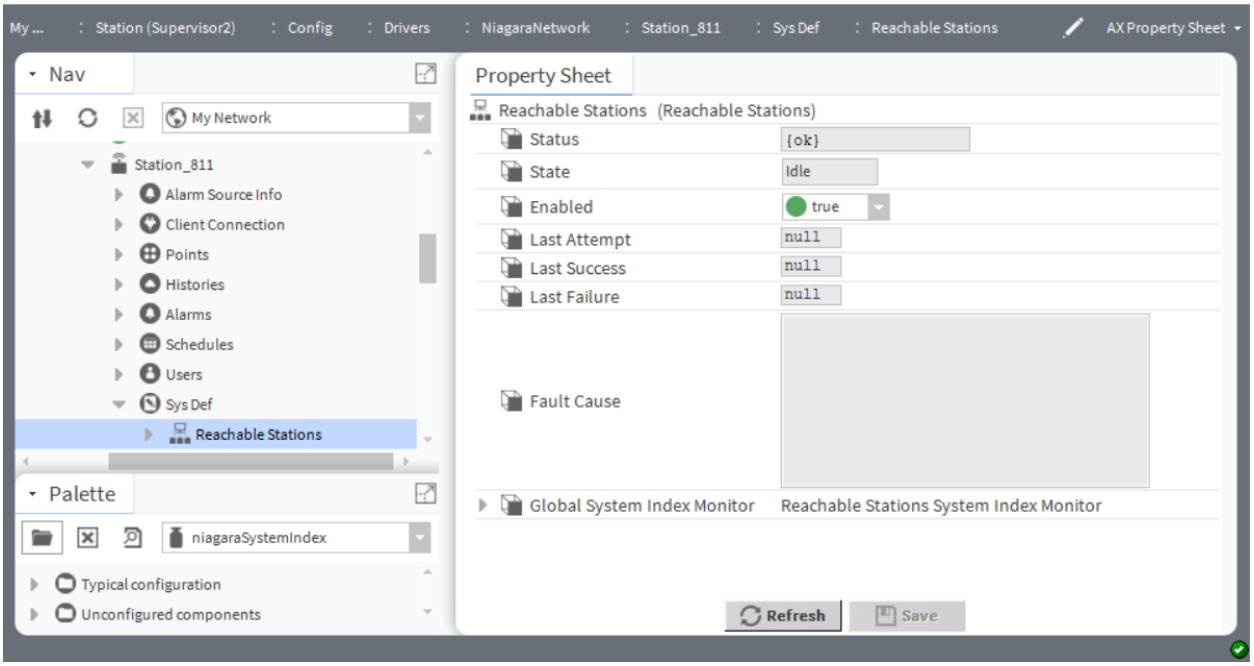
Property	Value	Description
Last Attempt	read-only, null (default)	Reports the date and time of the last attempted execution.
Last Success	read-only, null (default)	Reports the last time the station successfully performed this function.
Last Failure	read-only, null (default)	Reports the last time the system failed to perform this function. Refer to **`Fault Cause`** for details.
Execution Time	additional properties	See following sections
Alarm on Failure	true (default), false	Enables/disables the generation of an alarm on indexing failure.
Alarm Source Info	additional properties	Contains a set of properties for configuring and routing alarms when this component is the alarm source. For property descriptions, refer to the *Alarms Guide*
Default Index Queries	BQL queries (default)	Provides pre-configured default query ORDs. It is a read-only OrdList (frozen property) on several indexing components: LocalSystemIndexer, NiagaraNetworkSystemIndexSource, NiagaraSystemIndexExport, and NiagaraSystemIndexImport). For more details, refer the separate topic in this guide titled “Default Index Queries”.

Property	Value	Description
Use Default Index Queries	`true` (Merge with Custom) (default), `false` (Custom Only)	If true, indexing uses the **`Default Index Queries`** and merges those with any Custom Index Queries that have been entered. If false, indexing uses only the **`Custom Index Queries`** that have been entered.
Custom Index Queries	empty (default), text	Provides an admin-writable OrdList, a frozen property with which to specify additional query ORDs that the framework merges with the **`Default Index Queries`** (unless **`Use Default Index Queries`** is set to `false`). This property is empty unless you have entered custom queries. Click (+) to enter a new custom query.

niagaraDriver-ReachableStations

As of EC-Net 4.13 this component is a container of reachable stations, which define all of the downstream NiagaraStations that are routable starting at the NiagaraStation in which this container resides. It has an action to update the reachable, downstream NiagaraStations that are currently available and model them as child **ReachableStationInfo** instances.

Figure 9 Reachable Stations property



This component is contained on the EC-Net station under the **SysDef** device extension.

Property	Value	Description
Status	read-only	Indicates the condition of the component at the last check.

State	read-only	A frozen property on a component, indicates the current state of its **Update Reachable Stations** action.
		`Idle` (default)
		`Pending`
		`In Progress`

Property	Value	Description
Last Attempt	read-only, null (default)	Reports date/timestamp of the last attempt to update the reachable stations.
Last Success	read-only, null (default)	Reports date/timestamp of the last successful update of reachable stations.
Last Failure	read-only, null (default)	Reports date/timestamp of the last failed update of reachable stations.

Actions

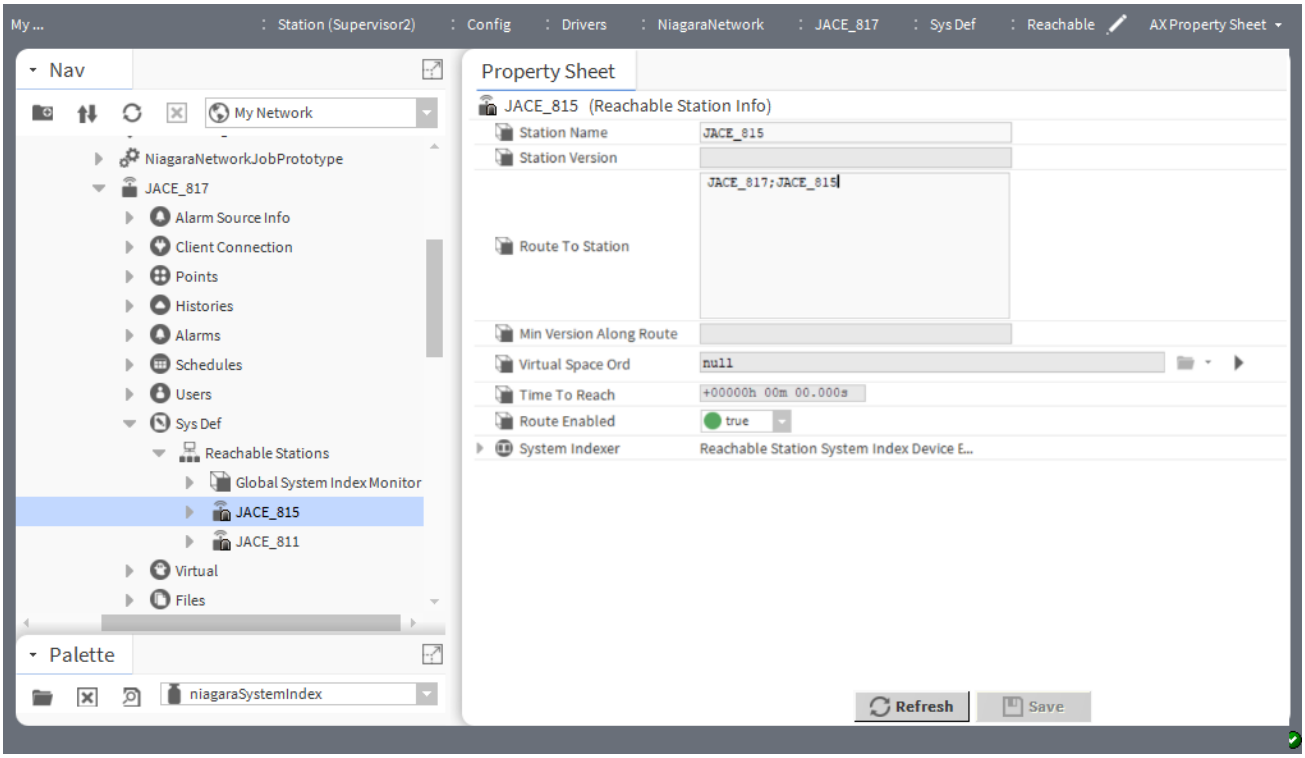
Update Reachable Stations:Manually updates the reachable stations for a particular starting route (NiagaraStation in the NiagaraNetwork).It triggers the update process of reachable, downstream NiagaraStations that are currently available and models them as child ReachableStationInfo instances.

niagaraDriver-ReachableStationInfo

This component appears automatically for any discovered, reachable station. It provides route information and allows you to enable or disable a particular route to a reachable station.

The ReachableStationInfo components get automatically placed under **“ Station”→Sys Def→ReachableStations**.

Figure 10 Reachable Station Info properties



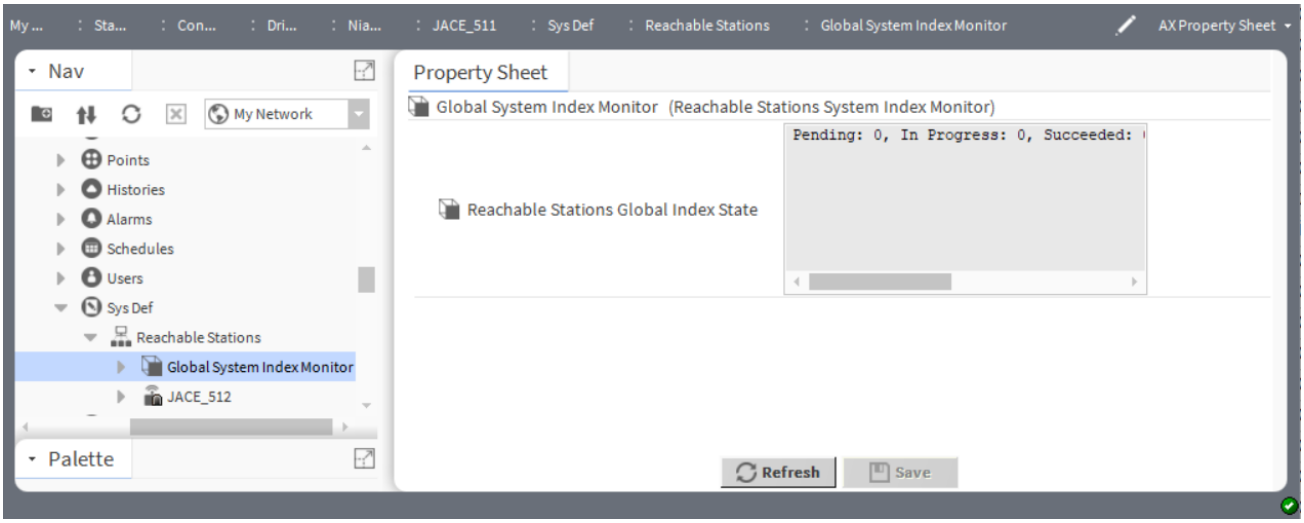
Property	Value	Description
Station Name	read-only	Displays the name of a remote, reachable station.
Station Version	read-only	Specifies the baja version of the destination reachable station.

Property	Value	Description
Route To Station	read-only	The route of intermediate station names (delimited by semi-colons) to reach the remote, reachable station.
Min Version Along Route	read-only	Specifies the minimum baja version along the route of mid-tier stations to reach the destination reachable station.
Virtual Space Ord	read-only	Displays the ORD to the virtual space of the remote, reachable station. If virtuals are not enabled somewhere along the route, this ORD will be NULL.
Time To Reach	read-only	Displays the last amount of time it took to reach the remote, reachable NiagaraStation over the route during the last update.
Route Enabled	`true` (default), `false`	If set to `true`, this particular route to the reachable station is enabled and may be used, for example, for system indexing. If set to `false`, this route is disabled and will be excluded from the set of possible routes to a reachable station during activities such as system indexing.

niagaraSystemIndex-ReachableStationsSystemIndexMonitor

As of EC-Net 4.13, this component is a mixin and automatically added to ReachableStations containers under the Supervisor's NiagaraNetwork when the SystemIndexService component is installed. It is used to monitor the global system index progress of descendant reachable stations under the parent container.

Figure 11 Reachable Stations System Index Monitor properties



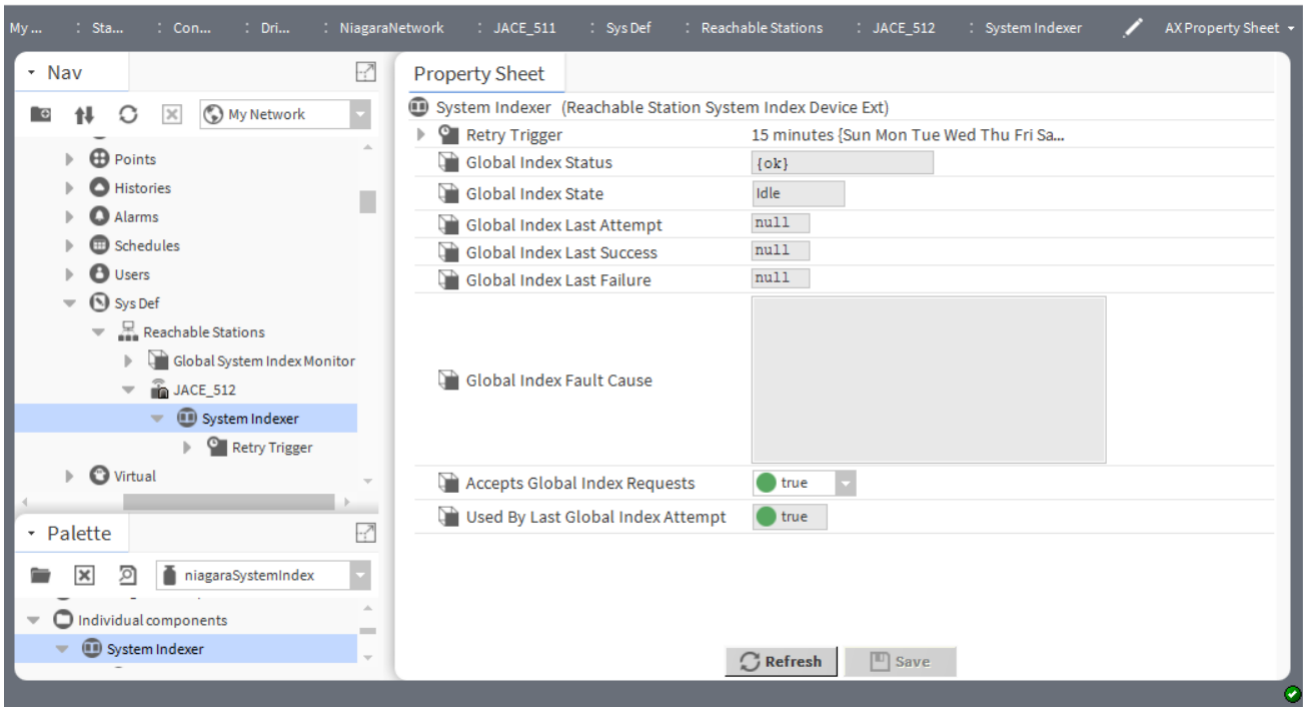
Property	Value	Description
Reachable Stations Global Index State	read-only	Reports the indexing progress of the downstream reachable stations in the following categories: `Pending`, `In Progress`, `Succeeded`, `Failed`, `Unused`, `Ineligible`, `Idle`.

niagaraSystemIndex-ReachableStationSystemIndexDeviceExt

This component is used to enable and configure indexing of individual reachable stations.

The **ReachableStationSystemIndexDeviceExt** component (also named **System Indexer**) is automatically added as a mixin under reachable stations when the **SystemIndexService** component is installed.

Figure 12 Reachable Station System Index Device Ext properties



Property	Value	Description
Retry Trigger	drop-down list, checkboxes, read-only	Specifies trigger mode and reports the time of the last trigger and next trigger.
Global Index Status	read-only	Indicates the status of the last global index operation, in the scope of the individual reachable station on which this deviceExt resides.
Global Index State	`Idle` (default), `Pending`, `In Progress`	Indicates the state of the last global index operation, in the scope of the individual reachable station on which this deviceExt resides.
Global Index Last Attempt	read-only	Timestamp of the last attempted global index operation, in the scope of the individual reachable station on which this deviceExt resides.
Global Index Last Success	read-only	Timestamp of the last successful global index operation, in the scope of the individual reachable station on which this deviceExt resides.
Global Index Last Failure	read-only	Timestamp of the last failed global index operation, in the scope of the individual reachable station on which this deviceExt resides.
Global Index Fault Cause	read-only	Indicates the reason why the global index operation is in fault, in the scope of the individual reachable station on which this deviceExt resides.

Property	Value	Description
Accepts Global Index Requests	`true` (default), `false`	When set to `true`, this reachable station is included in the global system indexing operation. If set to `false`, this reachable station is excluded from the global system indexing operation.
Used By Last Global Index Attempt	read-only	Displays whether this ReachableStationSystemIndexDeviceExt instance was chosen as the selected route during the last global system indexing execution. Only one optimal route to a particular reachable station is chosen at system index time.

Actions

Retry: Repeats the execution of global indexing on this particular reachable station.

Execute Global Index On This Station: Manually executes global indexing on this particular station.

Default Index Queries

This is a set of default queries that are run against stations in the NiagaraNetwork, and optionally against the Supervisor's station (depending on how you have configured the SystemIndexService), during the system indexing process.

You can use these default queries (via the LocalSystemIndexer) to index your local Supervisor station, and/or all stations in the Supervisor's NiagaraNetwork (via the NiagaraNetworkSystemIndexSource), and populate your System Database. By default, the root component of the station is indexed, as well as entities in all other driver networks, as described here.

Additionally, you can use the default queries (via the NiagaraSystemIndexExport/Import) to push/pull entity data from an individual station into the Supervisor's SystemDb.

- `station:|slot:/|bql:{asComponent}`

  Pulls in the root component of the station, which is required in scoped queries.

- `station:|slot:/|bql:select * from driver:DeviceNetwork where type !=niagaraDriver:NiagaraNetwork`

  Pulls in all driver networks (excluding NiagaraNetwork) to index entities in other device networks.

- `station:|slot:/|bql:select * from driver:DeviceFolder where type != niagaraDriver:NiagaraStationFolder`

  Pulls in all driver device folders (excluding NiagaraDriver device folders) to index device folders in other networks.

- `station:|slot:/|bql:select * from driver:Device where type !=niagaraDriver:NiagaraStation and type !=niagaraDriver:NiagaraEdgeStation`

  Pulls in all devices (excluding NiagaraStations and NiagaraEdgeStations) to index devices in other networks.

- `station:|slot:/|bql:select * from driver:PointFolder where type !=niagaraDriver:NiagaraPointFolder`

  Pulls in all driver Point folders (excluding NiagaraDriver point folders) to index point folders in other networks.

- `station:|slot:/|bql:select * from control:ControlPoint where proxyExt.type !=niagaraDriver:NiagaraProxyExt`

  Pulls in all control points (excluding EC-Net proxy points) to index control points from other networks.

- `station:|slot:/|bql:select * from schedule:AbstractSchedule where type !=niagaraDriver:NiagaraAbstractSchedule`

Pulls in all schedules (excluding EC-Net schedules) to index schedules in other networks.

NEQL query examples

The Niagara Entity Query Language (NEQL) provides a mechanism for querying tagged entities in EC-Net applications. This topic provides grammar and syntax examples to help you construct these queries.

Note that NEQL only queries for tags. NEQL supports traversing defined entity relationships as well as parameterized queries and allows you to use the same syntax for queries in hierarchy level definitions as is used in **Search** queries.

In addition to using NEQL queries in a hierarchy definition or search, you might use these queries for testing purposes, say to find devices or points in a station. For example, to find all devices in a station enter the query: `neql:n:device`. Or using the absolute ORD form in EC-Net 4 Pro, press `CTRL`+`L` to open the **ORD** dialog and enter the following:

```
ip:<host>|foxs:|station:|slot:/|neql:n:device
```

NOTE:

In EC-Net, the "`sys:`" ORD scheme can be used to redirect NEQL queries to resolve against the System Database. For example, the following query searches for all devices known to the System Database:

```
ip:<host>|foxs:|sys:|neql:n:device
```

In EC-Net, NEQL queries can be resolved over FOX. Additionally, NEQL results support tables so the results can be displayed in collection tables, reports, Px pages, etc. NEQL can be used in the following cases:

- used on a table on a Px page
- resolved from the ORD Field Editor
- resolved from the browser/path bar when typed in an Hx Profile
- used to generate reports

Finally, NEQL and BQL can be used together (although not supported in **Search**). You can append a BQL query to the end of a NEQL query for additional processing. Do this in any of the cases listed above. The following example shows a BQL `select` query appended to a NEQL query.

```
ip:<host>|foxs:|station:|slot:/|neql:n:device|bql:select toDisplayPathString, status
```

The above query first finds all devices using the NEQL statement, and then the BQL statement processes against the devices returned from the NEQL query, displaying a two–column table with the display path of the devices shown in one column and the status of the devices shown in the other.

CAUTION: Appending a BQL query to the end of a NEQL query can be a costly operation in terms of processing time. You could experience processing delays when using such queries.

Query examples

The following examples are designed to help you construct queries.

To query for	Syntax example	Returns result
point tag	`n:point`	Any entity with the point tag (in the "n" namespace)
name tag = "foo"	`n:name = "foo"`	Any entity with the "foo" name tag (in the "n" namespace)
type tag = "baja:Folder"	`n:type = "baja:Folder"`	Any entity with the "baja:Folder" type tag (in the "n" namespace)
points that are NumericWritables with hs: coolingCapacity > 4.03	`type = "control:NumericWritable" and hs:coolingCapacity > 4.03`	Any entity with the "control:NumericWritable" type tag (in the "n" namespace) and the coolingCapacity tag (in the "hs" namespace) with a value greater than 4.03

To query for	Syntax example	Returns result
everything	`true`	All entities
entities with names containing “Switch”	`n:name like ".*Switch.*"`	Any entity with a name that contains "Switch" case-sensitive
entities with geoCity tag with value not equal to Atlanta	`n:geoCity != "Atlanta"`	Any entity with the geoCity tag (in the n namespace) with a value that is not Atlanta
where n:pxView is a relation	`n:pxView->n:type`	Any entity with a pxView relation where the endpoint is a niagara type. This is all the entities that have px views.
entities with t:foo but not t:herp	`t:foo and not t:herp`	Any entity with the foo tag (in the t name-space) that does not also have the herp tag (in the t namespace).
points that were built earlier than 2015 or whose primary function is backup	`hs:yearBuilt < 2015 or hs:primaryFunction = "backup"`	Any entity with either the yearBuilt tag (in the hs namespace) with value less than 2015 or the primaryFunction tag (in the hs namespace) with value "backup"
child entities of entities with the floor tag = 2	`n:parent->hs:floor = 2`	Any entity whose parent has the floor tag (in the hs namespace) with value 2

NOTE: Do not create a namespace for a tag dictionary if that namespace is already used for an existing ORD scheme (NEQL or BQL). Since the search service allows for different types of searches (e.g., neql and bql), user must validate the search text and make sure they are not trying to enter an invalid ORD. Precede the query search with "neql:" to specify that it should run a neql query explicitly For Example: “`neql:h:test`”.

NEQL grammar

The following list is the subset of the NEQL grammar that you can use to build NEQL queries for the “Query” portion of Query Level Defs and the “Filter Expression” portion of Relation Level Defs in Hierarchies. For a more comprehensive overview of NEQL and the complete grammar and examples, refer to NEQL documentation in the *EC-Net Developer Guide*.

```
<statement> := <full select> | <filter select> | <traverse>

<full select> := select <tag list> where <predicate>

<filter select> := <predicate>

<traverse> := traverse <relation> (where <predicate)

<tag list> := <tag> (, <tag>)*

<tag> := (<namespace>:)<key>

<relation> := (<namespace>:)<key><direction>

<namespace> := <word>

<key> := <word>

<direction> := -> | <-

<predicate> := <condOr>

<condOr> := <condAnd> (or <condAnd>)*

<condAnd> := <term> (and <term>)*

<term> := <cmp> | <tagPath> | <not>

<cmp> := <comparable> <cmpOp> <comparable> | <like>

<like> := <tagPath> like <regex>

<cmpOp> := = | != | < | <= | > | >=
```

```
<comparable> := <val> | <tagPath>

<val> := <number> | <bool> | <str>

<tagPath> := (<relation>)*<tag>

<not> := not <negatable> | !<negatable>

<negatable> := (<predicate>) | <tag> // note: parens around <predicate> signify ac-
tual paren characters, NOT optional syntax

<number> := <int> | <double>

<bool> := true | false

<str> := "<chars>"

<typeSpec> := <moduleName>:<typeName>

<moduleName> := <word>

<typeName> := <word>

<word> := <chars>

<regex> := "<chars>" // note: some regex edge cases not supported
```

Index

T

U

V