Technical Document

Niagara Rdbms Driver Guide

July 4, 2023

niagara4

Niagara Rdbms Driver Guide

Tridium, Inc.
3951 Westerre Parkway, Suite 350
Richmond, Virginia 23233
U.S.A.

Confidentiality

Trademark notice

Copyright and patent 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 Niagara technical documentation library. Released versions of Niagara software include a complete collection of technical information that is provided in both online help and PDF format. The information in this document is written primarily for Systems Integrators. To make the most of the information in this book, readers should have some training or previous experience with Niagara software, as well as experience working with JACE network controllers.

Document content

This document explains how to install and configure the RdbmsNetwork and database connections.

CAUTION: Protect against unauthorized access by restricting physical access to the computers and devices that manage your building model. Set up user authentication with strong passwords, and secure components by controlling permissions. Failure to observe these recommended precautions could expose your network systems to unauthorized access and tampering.

Document change log

This topic summarizes the releases of this document.

July 4, 2023

- Added new topic "Dynamic ports in the RdbSqlServer" in the "Installation and configuration" chapter.
- Added "TLS Min Protocol" and "Verify Subject in Certificate" properties in "rdbSqlServer-SqlServerDatabase" component topic.
- Added "TLS Min Protocol" and "Verify Subject in Certificate" properties in "rdbMySQL-MySQLDatabase" component topic.
- Added "TLS Min Protocol" and "Verify Subject in Certificate" properties in "rdbOracle-OracleDatabase" component topic.
- Updated "Configuring Supervisor database properties" section.
- Updated "Security best practices" chapter.

January 19, 2022

Updated the extra connection properties description.

November 9, 2021

- Added a note to the "Security best practices" section recommending to configure minimum TLS version levels for specific databases.
- Added a new section to "Archive History Provider".
- Added "Batch history capacity" section.

October 4, 2021

Moved **`alarmOrion`** topics to *Niagara Alarms Guide*.

July 27, 2021

Added "Archive history limits" section and the component topic, "rdb-HistoryServiceArchiveProvider."

This document contains:

- Recommended security best practices.

- An explanation of the batch query optimization option (**`useBulkCopyForBatchInsert`** and **`rewriteBatchedStatements`**.
- Updates to "Point query example" topic to include title change, new screen capture and improved explanation.

June 24, 2021

Added a component information for the "alarmOrion-OrionArchiveAlarmProvider".

January 22, 2021

Updated "Importing history data from an Rdbms database" and, generally, improved the history import descriptions.

January 14, 2021

Added a note in the "Database requirement" and "MySQLDatabase (rdbMySQL-MySQLDatabase)" topics that indicates the latest supported MySQL connector version.

October 21, 2020

- Added information about the **Set Hsql Password** action.

June 5, 2020

- Minor corrections to several component topics to support online help.
- Added information specifying that the `mySql Connector/J` connector must be renamed when installed.
- Added missing property (`Connector`) on the **`SqlServerDatabase`** component.

October 25, 2019

In the topic, "About this guide", added a caution note alerting customers to restrict access to all computers, devices, field buses, components, etc., that manage their building model.

August 7, 2019

Updated for Niagara 4.8.

September 17, 2018

Initial publication for Niagara 4.

Related documentation

This topic identifies other documents that provide information about this driver.

The following documents are related to the content in this document and may provide addition information on the topics it covers:

- *Getting Started with Niagara*
- *Niagara Drivers Guide*

Chapter 1 Installation and configuration

Topics covered in this chapter

- ♦ Database security best practices
- ♦ Modules
- ♦ Database requirement
- ♦ Prerequisite checklist
- ♦ Installing the network and database
- ♦ Configuring Supervisor database properties
- ♦ Dynamic ports in the RdbSqlServer
- ♦ Testing the database connection
- ♦ Database connection troubleshooting
- ♦ Oracle database and JDK compatibility

The RDBMS (Relational DataBase Management System) driver, is a non-field bus driver that uses a network architecture similar to other framework drivers.

This driver has many properties and extensions in common with other field-bus-type network drivers. However, there are several distinctive RDBMS driver characteristics:

- There is no separate **RdbmsNetwork** driver component palette. Each of the other palettes (**`rdbHsqlDb`**, **`rdbMySQL`** **`rdbOracle`** and **`rdbSqlServer`**) provide the same **RdbmsNetwork** component. The **`rdb`** palette provides history-related components.
- The RDBMS point device extension is unlike point device extensions associated with other drivers. This point device extension uses the **Rdbms Point Query** component to filter database records and provide candidate records for proxy points.
- The **RdbmsNetwork** does not have a tuning policies component.

Figure 1 Worker properties for tuning

Some measure of tuning is provided with the RDBMS **Worker** component, which is available under individual RDBMS device drivers.

Database security best practices

Network security is a number one priority for all IT departments.

On the database side:

- Create a database user that has the least amount of access needed to accomplish database tasks.
- Work with your IT department to secure (harden) the computer on which the relational database is installed.
- Change your database configuration to permit connections that use the latest TLS version protocols.

  IMPORTANT: For security reasons, each database connection must support the latest TLS connection protocol. TLS 1.0 and TLS 1.1 connection protocols no longer meet our security standards. Coordinate with your database administrator to make sure that your database supports the latest TLS version.

  The following table gives an overview of TLS versions supported by different databases and provide information about the client side setup:

DB type	Supported TLS versions	Connection property (if any)	Client-side configuration information
MySQL	TLSv1, TLSv1.1, TLSv1.2, TLSv1.3	enabledTLSProtocols	For Connector/J 8.0.26 and later: TLSv1 and TLSv1.1 were deprecated in Connector/J 8.0.26 and removed in release 8.0.28; Connector/J 8.0.26 or earlier must be used if TLSv1 or TLSv1.1 is required. For more information, see **MySQL Connector/J 8.0 Developer Guide→Connector/J Reference→Connecting Securely Using SSL**.
Oracle	undetermined | 1.0 |1.1 | 1.2	**`oracle.net.ssl_version`** or **`SSL_VERSION in sqlnet.ora/listener.ora`**	For information about how to configure the version of SSL to be used, see Oracle Database Security Guide at `https://docs.oracle.com` and choose **C Kerberos, SSL, and RADIUS Authentication Parameters→Secure Sockets Layer Version Parameters.**
MS SQL Server	TLSv1, TLSv1.1, TLSv1.2, TLSv1.3		For more information about how to enable TLS 1.2 support for SQL Server 2017 on Windows, SQL Server 2016, SQL Server 2008, SQL Server 2008 R2, SQL Server 2012, and SQL Server 2014, see Microsoft Support at `https://support.microsoft.com` and choose Knowledge Base Article KB3135244 *TLS 1.2 support for Microsoft SQL Server*.
HSQLDB	TLSv1, TLSv1.1, TLSv1.2, or TLSv1.3		HSQLDB is used internally only as a file system DB. For external (future use case), use the following Hsqldb TLS URL prefixes: – `jdbc:hsqldb:hsqls://` – `jdbc:hsqldb:https://`

On the Niagara side:

- Use encrypted and authenticated connections (Refer to the *Niagara Station Security Guide*).
- Do not enable the **`Sql Scheme Enabled`** property. This property is on the **MySQLDatabase Property Sheet** (to find, expand **Config→Drivers→RdbmsNetwork**, and double-click the **MySQLDatabase** node.
- If you are a Niagara Enterprise Security user, define a strong **`Passkey`** to protect your network `PIN`. To configure the **`Passkey`**, expand **Config→Drivers→RdbmsNetwork**, expand your MySql database and double-click **Rdb Security Settings**.

Modules

The RDBMS driver requires a set of modules, including **`rdb`** and **`orion`**. The specific rdbDatabase module you require depends on your database configuration.

Palette name	Network component(s) in the Nav tree	.jar file name in the modules folder	Function
alarmOrion	OrionAlarmService, Converters	alarmOrion-rt.jar	Provides the OrionAlarmService, which replaces the standard AlarmService so that a station can support an RDBMS alarms database. The *Niagara Alarms Guide* documents the components in this module.
orion	OrionService, DynamicTable, OrionMigrator	orion.jar	Provides an Orion database system. Developed at Purdue University, this general-purpose uncertain database system unifies the modeling of probabilistic data across applications. (source: orion.cs.purdue.edu)

Palette name	Network component(s) in the Nav tree	.jar file name in the modules folder	Function
rdb	HistoryUnicodeUpdater and HistoryTimezoneUpdater	rdb.jar	Defines database ords used to configure exported data.
rdbHsqlDb (optional)	RdbmsNetwork, RdbmsFolder, HsqlDbDatabase	rdbHsqlDb.jar	Supports an open source relational database management system written in Java. (source: Wikipedia). This database resides in a remote controller station.
rdbMySQL (optional)	RdbmsNetwork, RdbmsFolder, MySQLDatabase	rdbMySQL.jar	Supports an open-source relational database management system originally developed by Michael Widenius whose daughter's name was My. SQL stands for Structured Query Language (source: Wikipedia). This database resides in a Supervisor station.
rdbOracle (optional)	RdbmsNetwork, RdbmsFolder, OracleDatabase	rdbOracle.jar	Supports a multi-model database management system produced and marketed by Oracle Corporation (source: Wikipedia). This database resides in a Supervisor station.
rdbSqlServer (optional)	RdbmsNetwork, RdbmsFolder, SqlServerDatabase	rdbSqlServer.jar	Supports a relational database management system developed by Microsoft (source: Wikipedia). This server may be in the Supervisor station or in another network location.

Database requirement

The RDBMS driver connects Niagara stations to an RDBMS running in the Supervisor station. The purpose of this database is to import and export historical data and to populate control points with the results of SQL queries against the database. The RDBMS driver supports four third-party databases: MySQL, OracleDatabase or SqlServerDatabase, and HsqlDbDatabase.

An HsqlDbDatabase is configured at the factory to run in each remote controller. HSQLDB (Hyperthreaded Structured Query Language Database) is maintained by the HSQL Development Group and is available under a BSD-type (free) license. For more information about HSQLDB, refer to: `http://hsqldb.org/`. The driver supports no other third-party database in a remote controller.

One of the other third-party databases is required to run in the Supervisor PC. It is beyond the scope of this document to describe how to set up one of these databases.

Using Workbench, you install, configure and test an RDBMS driver connection to one of these database servers:

NOTE: Refer to the *Niagara 4 Installation Guide* for the latest information about supported database versions and supported operating systems.

MySQL database

The MySQL database is an Oracle Corporation RDBMS that requires a GPL (General Purpose License) or proprietary license. In addition to installing the database on your Supervisor computer, the MySQL database requires a Java Data Base Connectivity (JDBC) connector (Connector/J).

You download the connector from: `http://dev.mysql.com`. It may be named `mysql-connector-java-x.x.x.jar`, where `x.x.x` is the version number.

NOTE: The rdbMySQL-MySQLDatabase component was tested with the MySQL connector version "`mysql-connector—java-8.0.24`". Use of earlier versions of this connector is not recommended and not supported.

You may verify with the framework release documentation that the version you downloaded is compatible with the framework, then change the connector name to this generic name: `mysql-connector-java.jar` and copy it to this folder: `C:\Niagara\Niagara.home\jre\lib\ext`, where `C:` represents your drive, and `Niagara\Niagara_home` represents the location and version number of your unique N4 installation.

OracleDatabase

The OracleDatabase is an Oracle Corporation RDBMS that requires a proprietary license. The supported version may be more recent than version 9i. Contact your support channel for more up-to-date information. For more information about Oracle, refer to: `http://www.oracle.com/database/index.html`.

SqlServerDatabase

The SqlServerDatabase is a Microsoft RDBMS that requires a proprietary license. For more information about Microsoft SQL Server, refer to: `http://www.microsoft.com/sql/default.mspx`.

The RDBMS driver does not support Windows Authentication alone. You must have selected SQL Server Authentication (mixed mode) for any user of this database. This is an SQL Server login property, not a Niagara property.

Secure database connection

To prevent malicious hacker attacks on the database or station, the station must be able to authenticate the database server (especially if it is remote) and communication between station and server must be encrypted.

Prerequisite checklist

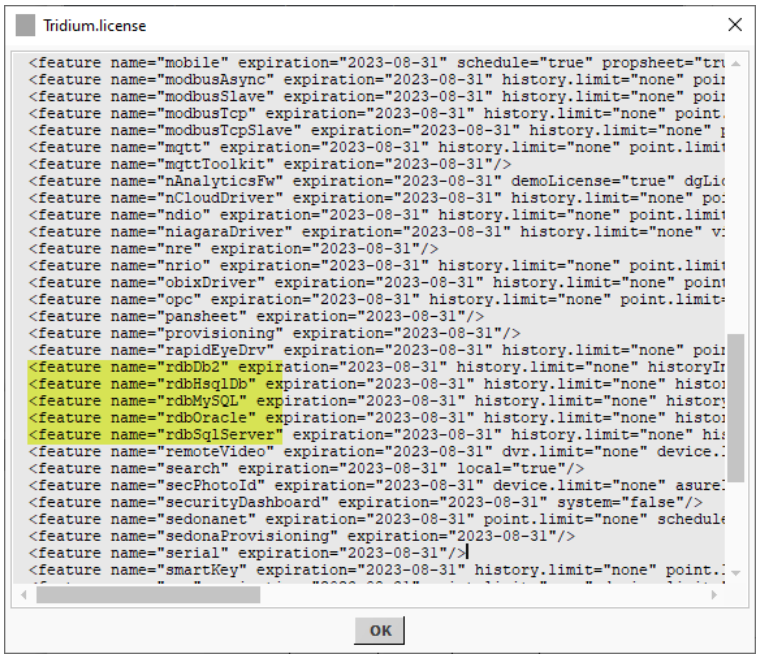
To successfully install and use the RDBMS driver and specific supported rdb database your installation needs to meet specific requirements.

Use of any of the databases mentioned in this document is subject to the terms and conditions of the respective database supplier. For additional copyright and licensing information, please refer to the individual supplier's documentation.

In addition to the right to use a specific database, you must have:

- Niagara and Workbench running on a PC/laptop.
- A Niagara license for a specific database type in the license file.

Figure 2 License file showing database licenses



The screen capture shows the database licenses in a `tridium.license` file.

- A relational database installed on the Supervisor PC: SqlServer, Oracle or MySQL. Embedded (remote) controllers support only the HsqlDbDatabase database. How to install a third-party relational database is beyond the scope of this guide. Look for help on the Internet.
- The IP address to establish a network connection to the database host (this can be localhost for your PC).
- If you are using Oracle's MySQL database, you need the MySQL Connector installed in your `$niagara.home/jre/lib/ext/` folder. You must rename the connector to mysql-connector-java.jar.
- Appropriate rights for the required database access.
- PKI (Public Key Infrastructure)-TLS (Transport Layer Security) secure communication provided by a server certificate for the database connection and the root CA (Certificate Authority) certificate used to sign the server certificate in the platform/station's Trust Store. While you can disable secure communication, the best practice is to always enable and implement secure connection that both encrypts and authenticates the database server.

  If you are using a certificate that was not signed by a CA in the System Trust Store (that is, if your company serves as its own CA), you must import the root CA (signer) certificate into your station's User Trust Store. The server certificate for the database, which you select from the **My SQL Server Cert** drop–down list on the database **Property Sheet** must have been signed by a third-party root CA certificate in the System Trust Store or by your company's root CA certificate in the User Trust Store.

You must know:

- The name of the database instance
- The user name and password to log in to the database
- If the database is using a non-default port number, the port number
- The name of the root CA certificate used to sign the server certificate presented by the database server. This certificate must be in the platform/station Trust Store. The server certificate is in the platform/station's User Key Store.

Installing the network and database

The network connects the Supervisor PC to local area network devices. The third-party database runs on Supervisor PC. The supported database components are available through three rdb palettes: rdbMySQL, rdbOracle, and rdbSqlServer.

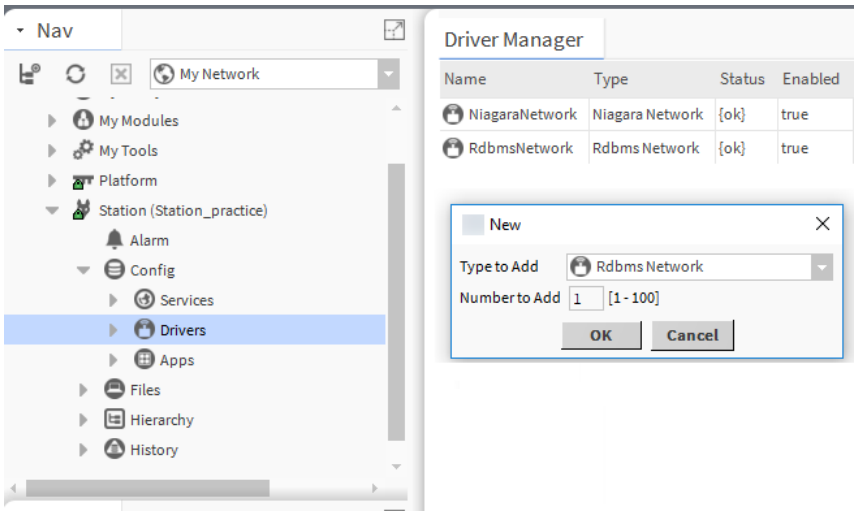
Prerequisites: You are working in Workbench running on a PC that is connected to the device network. A third-party database is installed on the PC and ready to use.

Step 1 Double-click the station's **Drivers** container.

The **Driver Manager** view opens.

Step 2 Click the **New** button.

The **New** network window opens.



For information about creating new networks, see the *Niagara Drivers Guide*.

Step 3 Select `Rdbms Network` from the drop-down list, the number of networks to add, and click **Ok**.

The **New** window used to name the network(s) opens.

Step 4 Change the default name(s) or use the default name(s) and click **Ok**.

You should have network named **RdbmsNetwork** (or whatever you named it) under your **Drivers** folder showing a status of `{Ok}` with the **`Enabled`** property set to `true`.

Step 5 To establish a connection to the RDBMS, right-click the **RdbmsNetwork** you just created, click **ActionsPing**.

The connection reports {ok} if a database exists.

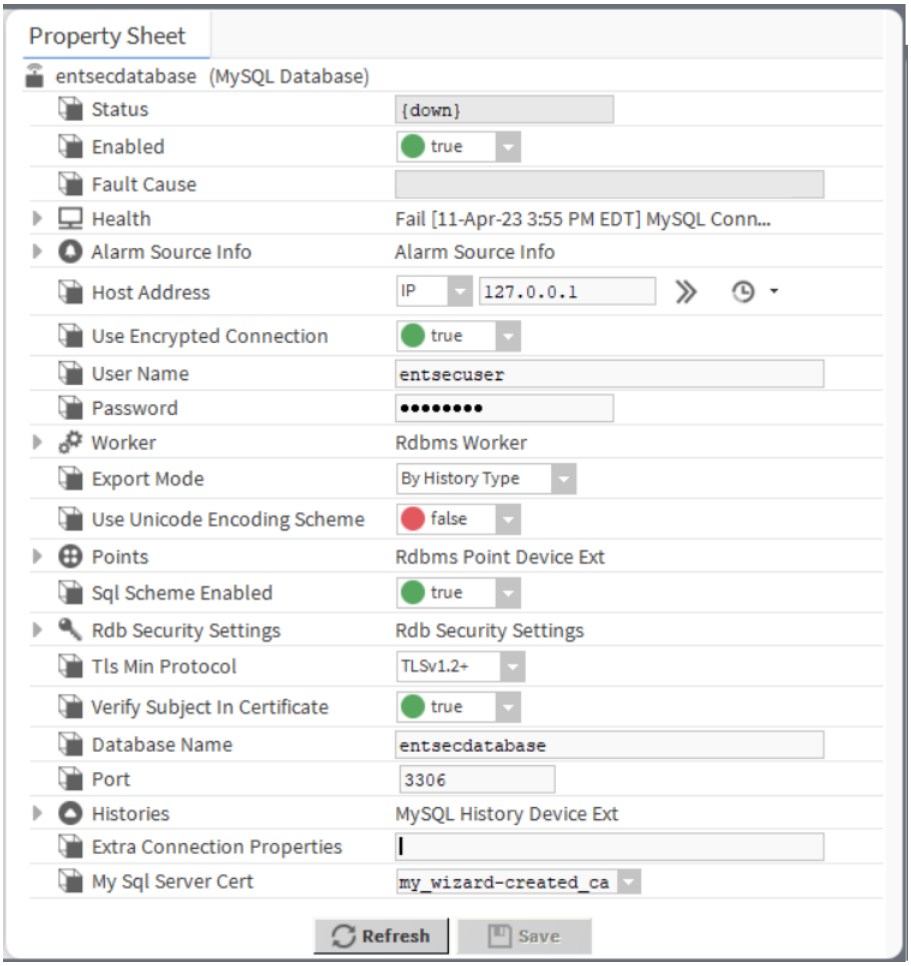
Configuring Supervisor database properties

Database configuration establishes names, credentials and other properties that enable the Supervisor station to connect to the external, third-party database.

Prerequisites: You are using Workbench and are connected to your Supervisor station. You have installed a third-party database that supports secure communication. The database is running.

Step 1 Right-click your database name under **Drivers→RdbmsNetwork** in the Nav tree, and click **Views→Property Sheet**.

The **Property Sheet** opens.



Above is an example using the MySQLDatabase **Property Sheet**.

The *Components* chapter in this guide documents all properties. The following table indicates which properties are available based on the rdb database device.

Table 1 RdbmsNetwork database properties

Property	HsqlDb (controller only)	MySQL	Oracle	SqlServer
Status (Information)				
Enabled	√	√	√	√
Fault Cause (Information)				
Health (Information)				
Alarm Source Info	√	√	√	√
Host Address	-	√	√	√
User Name	√	√	√	√
Password	√	√	√	√
Worker	√	√	√	√
Export Mode	√	√	√	√

Property	HsqlDb (controller only)	MySQL	Oracle	SqlServer
Use Unicode Encoding Scheme	√	√	√	√
Timestamp Storage	√	-	√	√
Points	√	√	√	√
Sql Scheme Enabled	√	√	√	√
Tls Min Protocol	-	√	√	√
Verify Subject in Certificate	-	√	√	√
Base Directory	√	-	-	-
Database Name	√	√	-	-
Instance Name	-	-	-	√
Service Name	-	-	√	-
Port	-	√	√	√
Histories	-	√	√	√
Extra Connection Properties	-	√	-	√
Version	-	-	-	√

Step 2 Configure at least these properties:

- **User Name** and **Password** (database credentials)
- **Database Name**
- **Host Address**

Step 3 To configure full UTF-8 Unicode support (NVARCHAR columns instead of VARCHAR columns), set **Use Unicode Encoding Scheme** to `true`.

To preserve compatibility with legacy systems, this property defaults to `false` and must be changed before you connect to your database for the first time.

Most databases (except MySQL) have a property called **Timestamp Storage**. This property can configure the driver to update and export history timestamps using Coordinated Universal Time (UTC).

Step 4 As a best practice, set **Export Mode** to `By History Type` as most database administrators would rather manage a few tables instead of thousands of individual tables.

When exporting by `By History Type`, the driver exports histories of the same data type (Boolean, Numeric, Enum, String) to the same table. For example, all numeric type histories export to a table named HISTORYNUMERICTRENDRECORD. This table contains an additional column named HISTORY_ID, which stores the history ID reference (ORD) from the station for each record in the table.

Step 5 To configure UTC timestamps, change **Timestamp Storage** from the default (`Dialect Default`) to one of the UTC options.

With `Dialect Default`, you must set both **Use Last Timestamp** and **Use History Config Time Zone** to `true`. This exports histories using the **History Config Timezone**. If you set the two properties to `false`, the driver exports histories using the station timezone.

Selecting the `Utc Timestamp` takes precedence over the **Use Last Timestamp** and **Use History Config Time Zone** properties in the history device extension, rendering them effectively irrelevant.

Switching back and forth between `Dialect Default` and `Utc Timestamp` pollutes the database with inconsistent timestamps, which can negatively affect any query you run to determine whether or not to export newer records.

Step 6 Select the minimum TLS protocol, which can be negotiated when establishing encrypted communications with the database server.

Step 7 Enable **`Verify Subject in Certificate`** for the Rdbms driver to verify that the existing Subject in the server's certificate matches the expected value.

Step 8 Select the server certificate for the database from the **`My Sql Server Cert`** drop–down list at the bottom of the **Property Sheet**.

This certificate must have been signed by the root CA certificate of a third-party CA (Certificate Authority) or by your company's own root CA certificate (if your company serves as its own CA). If your company serves as its own CA, its root CA certificate must be in the platform/station's User Trust Store. For certificate management, refer to the *Niagara Station Security Guide*

Step 9 To complete the configuration, click **Save**.

Dynamic ports in the RdbSqlServer

This topic describes how to configure your **`rdbSqlServer`** module so that you can connect to a database using dynamic ports. You need to set a few properties in the module property sheet and also configure the Sql Server Browser.

`SqlServerDatabase` supports connecting to an SQL Server Named Instance using dynamic port discovery. To configure the use of dynamic ports, the port property must be set to `0`, and the **`instance Name`** parameter value corresponding to the desired Named Instance must be specified in the **Extra Connection Properties**.

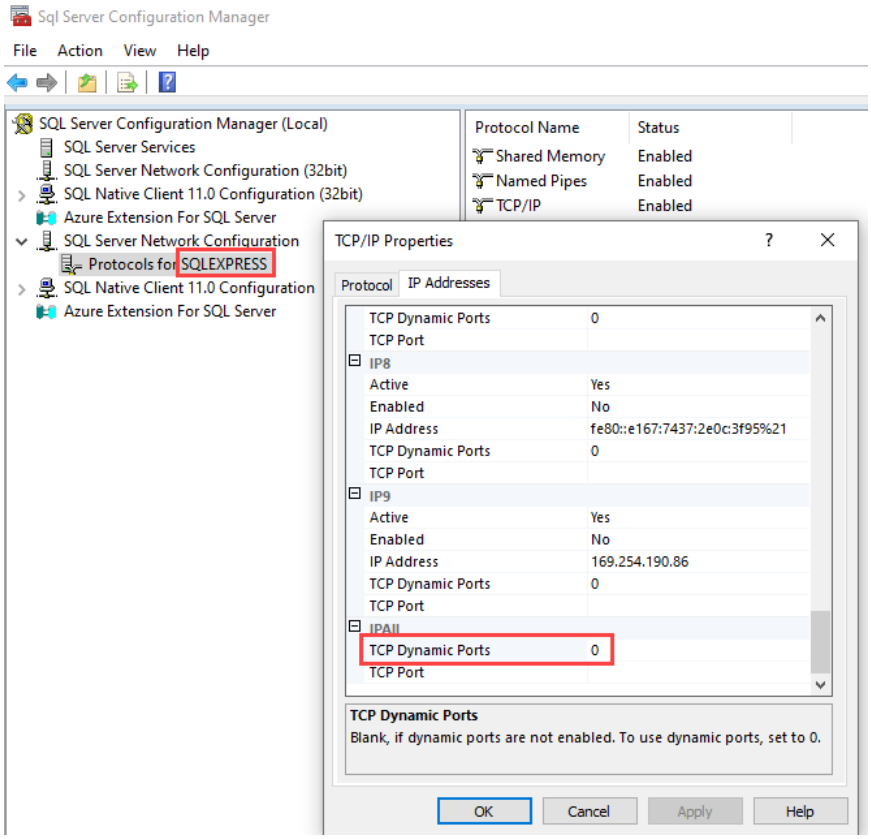
Using dynamic ports with SqlServerDatabase

SQL Server allows you to install multiple database instances on a single server. Each instance has a specific name that helps identify it. When connecting to a named instance of SQL Server, you have two options: you can either specify the port number associated with the named instance, or you can directly specify the instance name. If you don't provide an instance name or port number, the connection will be made to the default instance.

When SQL Server starts up and is configured to listen on dynamic ports, it checks with the operating system to find an available port and opens an endpoint for that port. To establish a connection, incoming connections need to specify the same port number. Since the port number can change each time SQL Server starts, the SQL Server Browser Service is provided to monitor the ports and redirect incoming connections to the current port for that particular instance. If a port value is not provided in the connection string, JDBC (Java Database Connectivity) takes care of discovering the port for a named instance.

To enable the dynamic port feature, you can use the SQL Server Browser to configure the named instance of the database and set the value of the TCP DYNAMIC PORTS property to `0`. When connecting to an MS SQL Server instance, if the port property is set to `0`, the JDBC connection is handed over to SQL Server, and the JDBC driver excludes the port number from the connection. It then signals the JDBC driver to utilize the SQL Server Browser service to locate the instance identified by the **`instanceName`** connection. You can refer to the image below for an example.

Figure 3 Example Sql Server Browser settings for TCP Dynamic Ports



Configure the rdbSqlServer-SqlServerDatabase properties as described above and as described above. See the "rdbSqlServer-SqlServerDatabase" topic for more details.

Testing the database connection

This procedure applies to any type of rdb Database.

Prerequisites: You are working in Workbench

Step 1 Right-click **RdbmsNetwork→RdbmsDatabase Device Extension** in the Nav tree, and click **Actions→Ping**.

If a valid connection to the database is made, the Health property displays `{Ok}`.

Step 2 If the Health property displays `{Fail}`, a connection is not made and you should examine the **`Last Fail Cause`** property for details.

Database connection troubleshooting

This topic includes some general, as well as database-specific, suggestions for some of the more common problems with establishing a network connection to the remote RDBMS databases.

Things to check on the database side

- Check with the database administrator (or owner of the database) to make sure that your login credentials have sufficient authorization for establishing a remote connection to the database.
- Make sure the database is running and is correctly configured.

Things to check on the Workbench Property Sheet

- Make sure that both the **RdbmsNetwork** and database have their `Enabled` properties set to `true`.
- Check that you have the correct **User Name** and **Password**. These are the credentials required by the third-party database, not the credentials to log into the platform or station.
- Confirm the name of the database. This could be the **Database Name**, **Instance Name** or **Service Name**.
- Check that you have set the correct **Port** number. Default port numbers may not have been used when the database instance was initially configured.

MySQLDatabase connections

Make sure that you have installed the correct MySQL connector version.

NOTE: The rdbMySQL-MySQLDatabase component was tested with the MySQL connector version "`mysql-connector-java-8.0.24`". Use of earlier versions of this connector is not recommended and not supported.

If you see the following, or similar, error it could mean that you are running an incompatible version of the connector. `java.lang.NoClassDefFoundError: Could not initialize class com.mysql.cj.protocol.a.authentication.AuthenticationLdapSaslClientPlugin`.

SqlServer RDBMS connections

- If the RDBMS server is running a named instance of the database that you are trying to connect to, make sure that you have the correct **Instance Name** which is provided during the sql server installation. If **Instance Name** is empty, the driver ignores the property and defaults to the database assigned in the SqlServer.
- For a station to connect, the Microsoft SQL Server instance must be configured for SQL Server Authentication. This is a property to configure in the third-party database. It is not a Niagara property.

NOTE: By default, SqlServerExress provides named instances for databases. The default name provided is "SQLEXPRESS."

To monitor the performance of an instance of SQL Server or troubleshoot problems with the queries being submitted to the database by the station, alongside the usual Platform-Application Director output it is often useful to use an SQL Profiler.

Oracle database and JDK compatibility

If your Oracle database reports that it is down or in fault, follow this procedure to upgrade the database.

Oracle versions 12.1 and later are compatible with ojdbc6.jar for JDK 6 and ojdbc7.jar for JDK 7. Oracle no longer supports ojdb14.jar (for JDK 1.4) or lower for clients to connect through the authenticated protocol of a 12.1 version database. For up-to-date Oracle compatibility information, refer to https://www.oracle.com/database/technologies/faq-jdbc.html

Step 1 Download the latest software from Oracle. For example, `ojdbc8.tar.gz` and extract the compressed file.

Step 2 Paste the extracted .jar file into `niagara_home/jre/lib/ext`.

Step 3 Start or restart the Supervisor station.

Step 4 Ping the Oracle database.

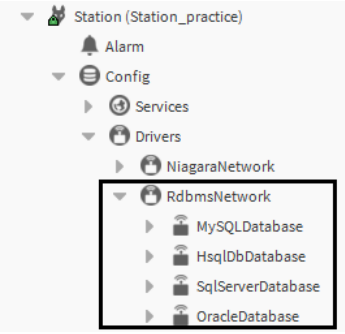
Chapter 2 Data management

Topics covered in this chapter

- Discovering and adding points
- Adding and configuring a new point query
- Editing an existing query
- Point query example
- Rdb Archive History Provider
- Batch history capacity
- Exporting history data to an RDBMS
- Importing history data from an Rdbms database
- Unix time conversion for MySQL
- Updating an existing database to support Unicode and UTC
- Updating existing Orion databases to support Unicode

The RDBMS driver manages the relationship between the external database and the station importing point data and exporting historical data for archiving and further analysis. In the process, data can be transformed and manipulated.

Figure 4 Database devices in the Nav tree



Under the **RdbmsNetwork**, each device component represents a specific type of relational database and should be located under the **RdbmsNetwork** driver.

Discovering and adding points

The discovery process uses the Rdbms Point Query **`Sql`** property to query the targeted rdb database and return only those points that satisfy the query.

Step 1 Click the **Discover** button (at the bottom of the view). This executes the query (as defined in the Rdbms Point Query **`Sql`** property) and any discovered points appear in the **Discovered** pane at the top of the view.

Step 2 Double-click on the desired Rdbms Point Query component.

The **Rdbms Point Query Manager** view opens.

Step 3 At the bottom of the view, click the **Add** button.

The **Add** window opens, with all selected points in the top pane of the window.

The Point Manager's **Add** button is available when you select (highlight) one or more data item in the top **Discovered** pane. The toolbar has an available `Add` tool, and the Manager menu has an `Add` command. Also, you can simply double-click a discovered item to bring it up in the Add window

Step 4 Configure the properties and click **OK**.

Adding and configuring a new point query

You create proxy points under the **Rdbms Device Extension** for any of the database device types. As with device objects in other drivers, each RdbmsNetwork device has a single **Points** extension.

Prerequisites: The driver and database are installed.

The **Rdbms Point Query Manager** works differently than other Point Manager views because of the way it uses the **Rdbms Points Query** component. This component (using its **`Sql`** property) filters the data to provide the candidate records that are available for adding as proxy points in the manager view.

Although the default view of the **Rdbms Point Device Extension** is the **Rdbms Point Device Ext Manager**, you may often need to use the **Property Sheet** view and the **Rdbms Query** views.

In this procedure, the term "rdb Database" represents any valid Rdbms Database Device Extension.

Step 1 Expand **Drivers→RdbmsNetwork** in the Nav tree, expand your rdb Database node and double-click the **Points** node.

The **Point Device Ext Manager** view opens.

Step 2 Click the **New** button at the bottom of the view.

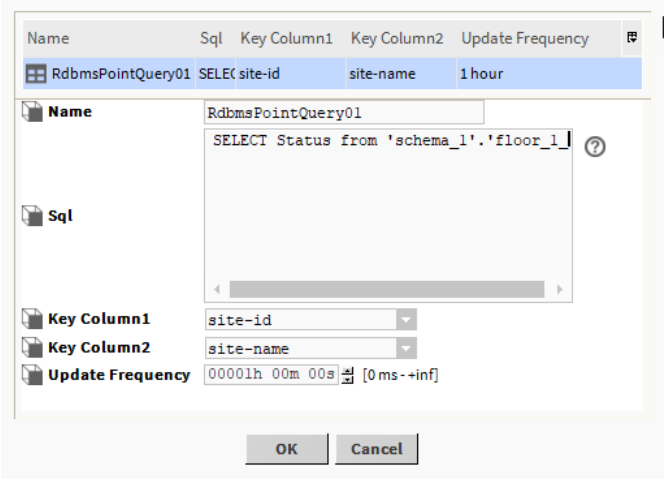
The **New** window opens.

Step 3 Select type of **RdbmsPointQuery** component to add, indicate the number of query components to add, and click **Ok**.

NOTE: If you add more than one, you can batch-edit most of the properties to configure them all at once.

A second **New** window opens.

Figure 5 Rdbms Point Query view



Step 4 Enter the query in the **Sql** property.

NOTE: The **Sql** property is BFormat. You can add BFormat syntax to the query string that processes before the driver sends the query to the database.

Step 5 Use **`Key Column 1`** and **`Key Column 2`** to create a unique identifier for the point.

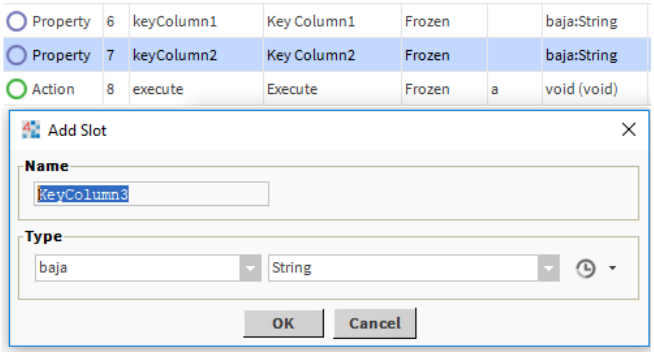
If you leave the key columns blank, the driver automatically uses the first column in the row as the primary key.

Key Column 2 is optional. Use it when you need an additional data item to establish a unique composite key.

The key columns you define (using the **Key Column 1** and **2** properties) may not actually be primary keys, although they often are.

An example situation where a single column cannot uniquely identify each row in a table might be a table of fan motor types with columns for "manufacturer", "model" and "maximum speed". To identify each row, you need to look at both the manufacturer and the model. These two columns would be the Key Column1 and Key Column2 columns. Only with both of them can you identify any given row, since individually neither column is unique.

Step 6 If you need more than two key columns to specify a unique key, add another key column slot from the RdbmsPointQuery **Slot Sheet** view.



Step 7 Configure the other query properties and click **Ok**.

The driver adds the new **RdbmsPointQuery** component(s) under the **Points Device Extension** node in the Nav tree and displays them in the **Rdbms Point Device Ext Manager** view.

Editing an existing query

You can change a query that already exists. The **Rdbms Query View** that opens when you edit an existing query is, typically, the most convenient place to work with queries as you are developing them because the query executes immediately and the lower pane displays the results as soon as you click the **Run** button (or with some delay, depending on database size and network connection speed). If there are errors in the query, an error window opens with an error message.

Prerequisites: The query you are working on already exists.

Step 1 Expand the **Drivers→RdbnsNetwork**, expand your rdbDatabase and the **Points** folder it contains, right-click the **RdbmsPointQuery** node, and click **Views→RdbmsQueryView**.

The **Rdbms Query View** opens.

NOTE: This view executes on display. So as soon as you open this view the saved query executes and displays results in the **Query Results** pane.

Step 2 To edit the saved query, select and work with the text in the editor box (top box).

Each valid query entry in the **Sql** property returns a set of data. Two properties define the query:

- **Sql** is a large text editor that displays the text of the Query (if any).
- **Update Frequency** displays when (in hours, minutes, and seconds) the driver executes the query and updates the control points.

Step 3 To test-run the query, click **Run**.

Step 4 To save the query for future use, click **Save**.

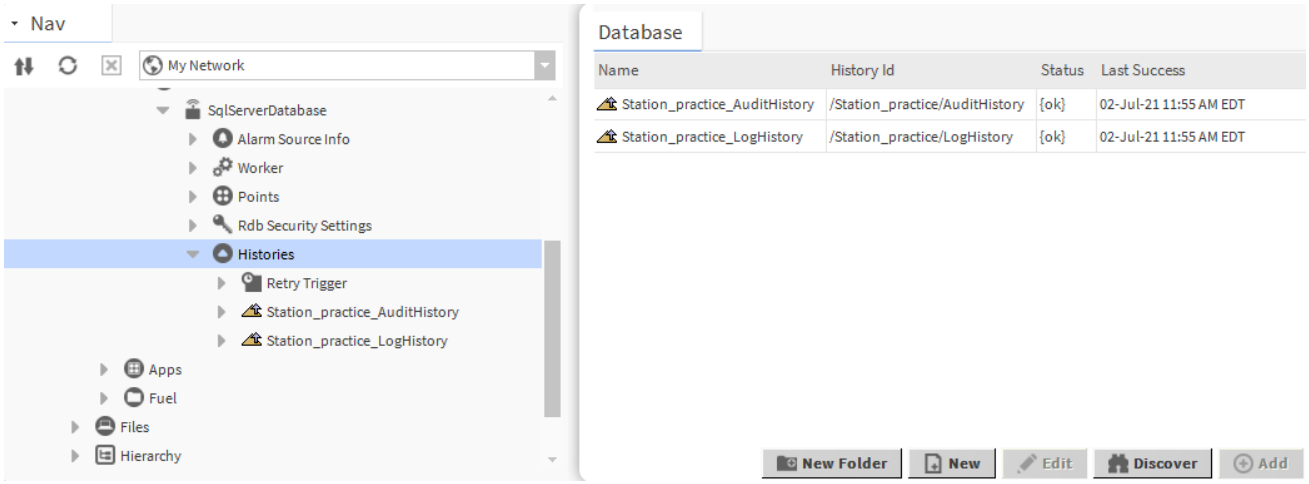
CAUTION: If you change (or refresh) the view without clicking **Save**, any unsaved changes are lost. No warning is given.

Point query example

This hypothetical example illustrates how the **RdbmsNetwork** proxy points might be created and used.

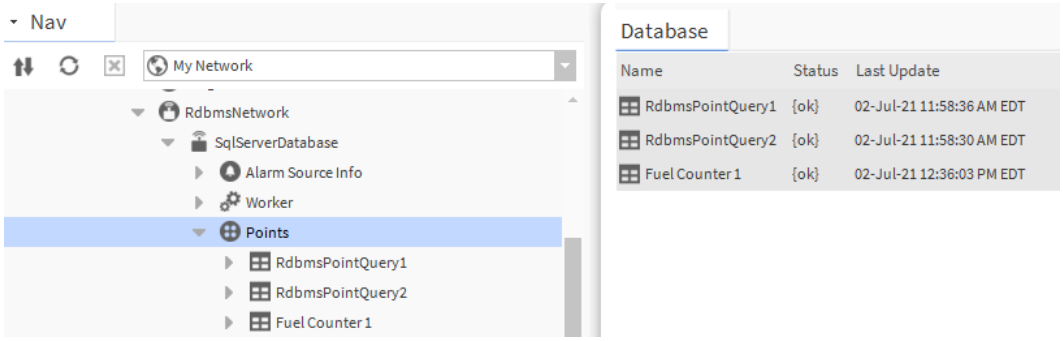
A nationwide convenience store corporation uses a remote SqlServer database to archive fuel sales records from each of its stores. The database stores records for three types of fuel every 15 minutes. To graphically display updated information over the Internet, the store uses the **RdbmsNetwork** and the **Point Device Extension**, as demonstrated below:

1. Each store exports transaction histories from its controller via a Supervisor, to a central SqlServer database using the **Sql Server History Device Ext** and the **History Export Manager** view.

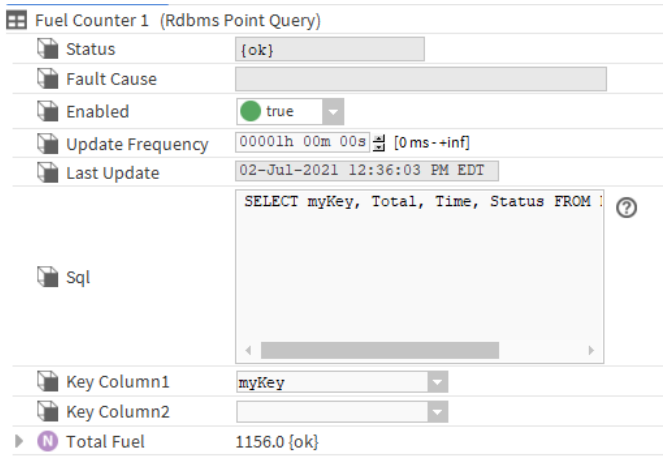


NOTE: Embedded controllers support only the **HsqlDbDatabase**.

2. For each store, a Supervisor station creates proxy points for each fuel type, by creating and configuring an **RdbmsPointQuery** for each fuel type.



3. An Sql query displays the total fuel sold as of the latest update.

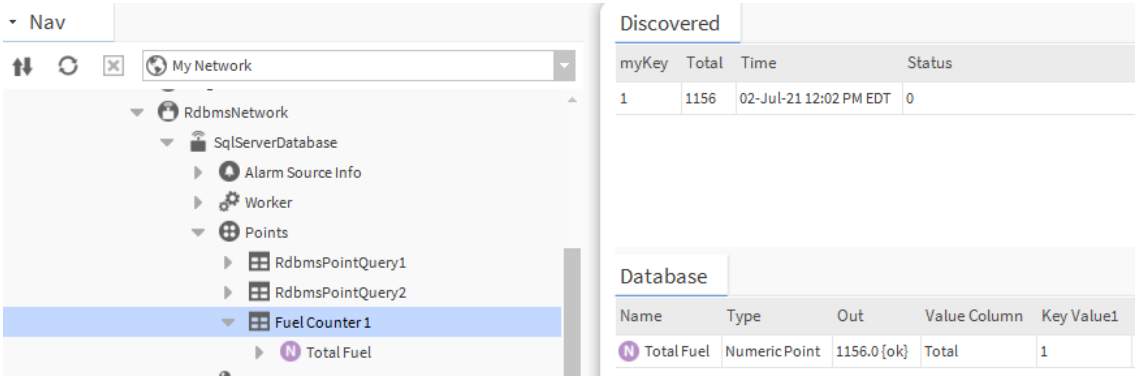


This counter is scheduled to update every 15 minutes, providing a total that updates every 15 minutes along with the time of the update.

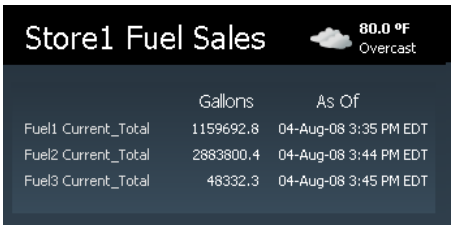
In this example, these Sql parameters are optional:

- "1 AS myKey" creates a new column as a key column and is not required.
- "1 AS Total", "AS Time", and "AS Status" create titles for the respective columns. If these optional parameters are not used, columns are titled "column1", "column2", "column3", respectively.

4. Data are discovered and added under the **Rdbms Point Device Extension** node.



5. Once added to the **RdbmsNetwork**, these proxy points are used to graphically display total gallons of each fuel type sold as of the update time.



Rdb Archive History Provider

The Rdb Archive History Provider feature was added starting in Niagara 4.11. It allows queries against local history records to be supplemented by archived history records that have been previously exported to a remote RDBMS using the standard RDBMS drivers (rdbSqlServer, rdbMySQL, and rdbOracle). Because it plugs in at a low level in the framework architecture (for example, the history module), existing views that query histories can benefit from this functionality without any additional changes.

This feature can allow you to seamlessly compare baseline years of historical trends with new data using a WebChart.

The standard RDBMS drivers (rdbSqlServer, rdbMySQL, and rdbOracle) can export local history records periodically to a remote database. The Rdb Archive History Provider plugs in to the station's **HistoryService**, making it possible for any existing views that query histories to benefit from both local and archived records.

While the station still stores local histories, once the driver exports those history records to a relational database, you can reduce the capacity of those local histories to free up resources in your Supervisor station. At history query time, the Rdb Archive History Provider can easily retrieve those exported (older) archived history records residing in an Oracle, SQL server or MySQL database.

License update prerequisite

To use the Rdb Archive History Provider, your license needs two updates:

- A general **`historyArchive`** license feature that covers any archive history provider implementation
- A specific **`rdbHistoryArchive`** license feature that covers your chosen RDB using the Rdb Archive History Provider against any supported RDBMS.

Optimizing pre-Niagara 4.11 database tables

If your archived histories predate Niagara 4.11, you may choose to run a job that optimizes your relational database's existing table indexes. Once optimized, you do not need to perform this optional index migration again in the future. It is not required and has no impact on the functionality of the Rdb Archive History Provider, but you may notice that the history query performance is not optimal when the indexes are not present and/or migrated.

Prerequisites: You are working in Workbench running on a PC, are connected to a Supervisor station and have access to a relational database on your company's network.

Although this step is not required, optimizing the existing table indexes can maximize query performance. Since this procedure affects the tables in the remote relational database, consult your DBA (Database Administrator) to make sure that your database has available space to add indexes if they are not already present. It is also recommended that you back up your database before you perform the migration in case a failure occurs during index migration.

Step 1 In the Nav tree, expand **Config→Drivers→RdbmsNetwork**, expand your RDB node, right-click **Histories** and click **Actions→Migrate to Optimized Data Indexes**.

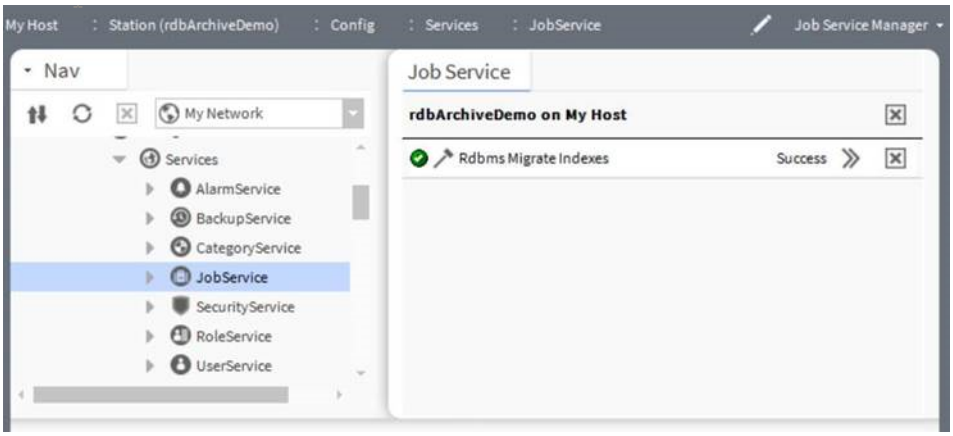
The driver prompts you to confirm that you intend to run this job.

Step 2 To continue, click **Yes**.

Invoking this action kicks off a job in the **JobService** where you can monitor the migration progress and see the results.

Step 3 To monitor job status, navigate to **Config→Services** and double-click **JobService**.

The **Job Service Manager** opens and displays a row for Rdbms Migrate Indexes.



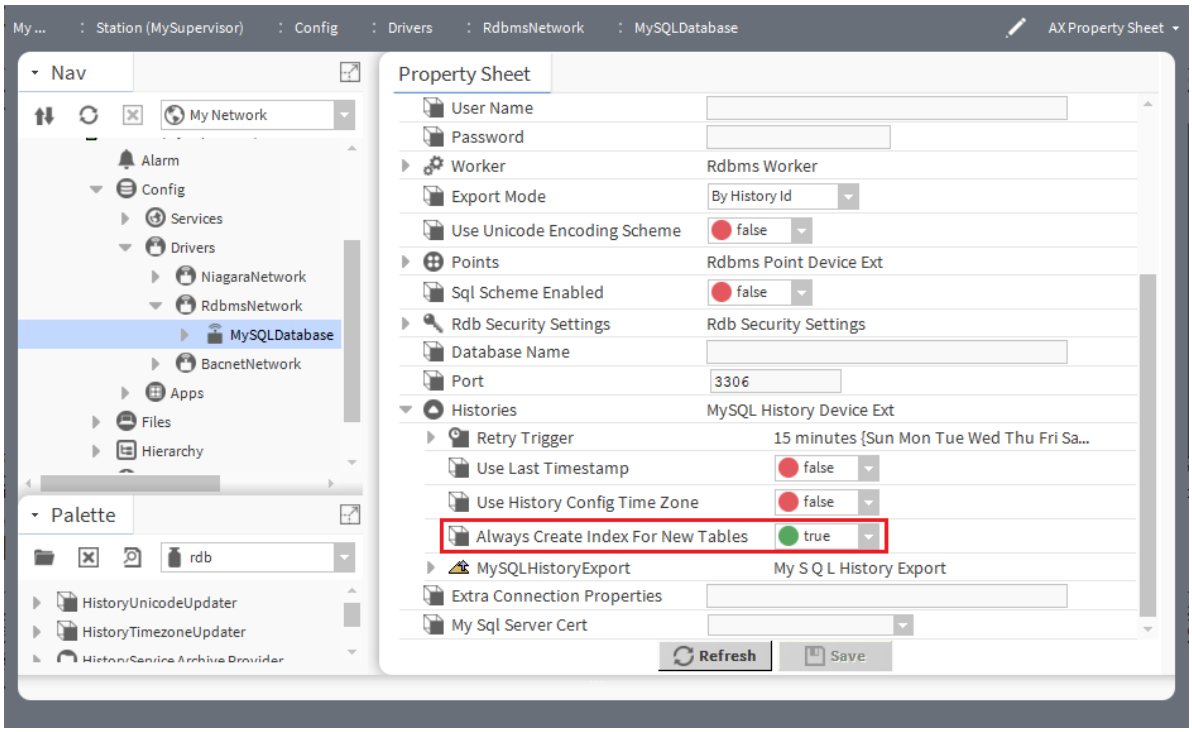
Configuring the driver to always create an index on exported history tables

The RDBMS driver can be configured to always create an index in the target database when exporting history data. You can set the **`Always Create Index For New Tables`** property (added in Niagara 4.11) on the Rdb Archive History Provider component to `true` to ensure that an index in the created RDBMS tables for exported histories is always created regardless of how other properties are configured. For example, if this property is set to `false` and the **`Use Last Timestamp`** property is set to `true`, indexes may not get created on the exported tables. There is no creation of indexes when **`Use Last Timestamp`** is set to `true`, regardless of this property setting. Setting this property to `true` in that scenario would cause any new tables created going forward to also have a proper index created. You can subsequently use the **The Migrate To Optimized Table Indexes** action against existing exported tables (also those prior to Niagara 4.11) that you want to migrate to ensure that updated indexes are created in the RDBMS. Creating indexes for exported tables can help maximize query performance.

Prerequisites: You are working in Workbench running on a PC, are connected to a Supervisor station and have access to a relational database on your company's network.

Step 1 Navigate to **Config→Drivers→RdbmsNetwork**, double-click on your RDB component and expand **Histories**.

The **Histories** properties open.



Step 2 To configure the station to always create an index on export, change **`Always Create Index For New Tables`** to `true` and click **Save**.

The next time you export, the station creates an index on the exported table unless one already existed.

Setting up an Rdb Archive History Provider

For history query purposes, an archive history provider pulls archive data into a station from a remote database on-demand (it does not persist the retrieved archive data locally, but only uses the data in the displayed query results).

Prerequisites: You are working in Workbench connected a Supervisor station.

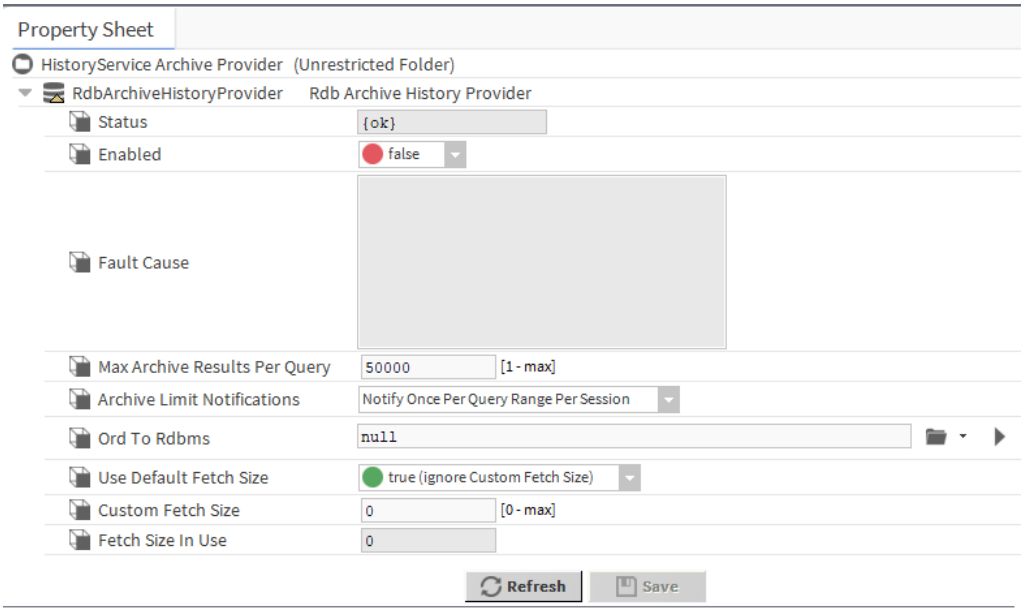
Step 1 Open the **`rdb`** palette and expand the **HistoryService Archive Provider** folder.

Step 2 In the station, expand **Config→Services→HistoryService**.

The **HistoryService** contains an **Archive History Providers** container.

Step 3 From the palette, add a **RdbArchiveHistoryProvider** to the **Archive History Providers** container under **HistoryService** and double-click the provider you just added.

The component's **Property Sheet** opens.



This **Property Sheet** configures the Rdb Archive History Provider.

Step 4 Use the button to the right of the **`Ord to Rdbms`** property to open the **Component Chooser**, locate the relational database in your driver network and click **Save**.

Step 5 Configure **`Max Archive Results Per Query`** and **`Archive Limit Notifications`** if needed.

`Max Archive Results Per Query` determines the maximum number of history records to read from the RDBMS for any history time range query that taps into it. If more history records are available beyond this limit at history query time, the **`Archive Limit Notifications`** property defines the behavior of a subset of Workbench views, but not all of them. Web Chart and HTML5 History Table views (accessible from the browser and Workbench) provide their own notification when a history query exceeds this limit. When the limit is reached for a query, in addition to the warning, you get truncated archive history results that always consider the most recent history records first.

The **`Archive Limit Notifications`** specifies what happens when a history query made from a Workbench user connected to the station exceeds the **`Max Archive Results Per Query`** limit.

NOTE: This setting does not apply to HTML5 history views, including HTML5 views accessed within Workbench, such as the Web Chart view. It only applies to native Workbench views that perform history queries, such as the AX History Chart or AX History Table views.

Step 6 To complete the configuration, click **Save**

Chart example, local data

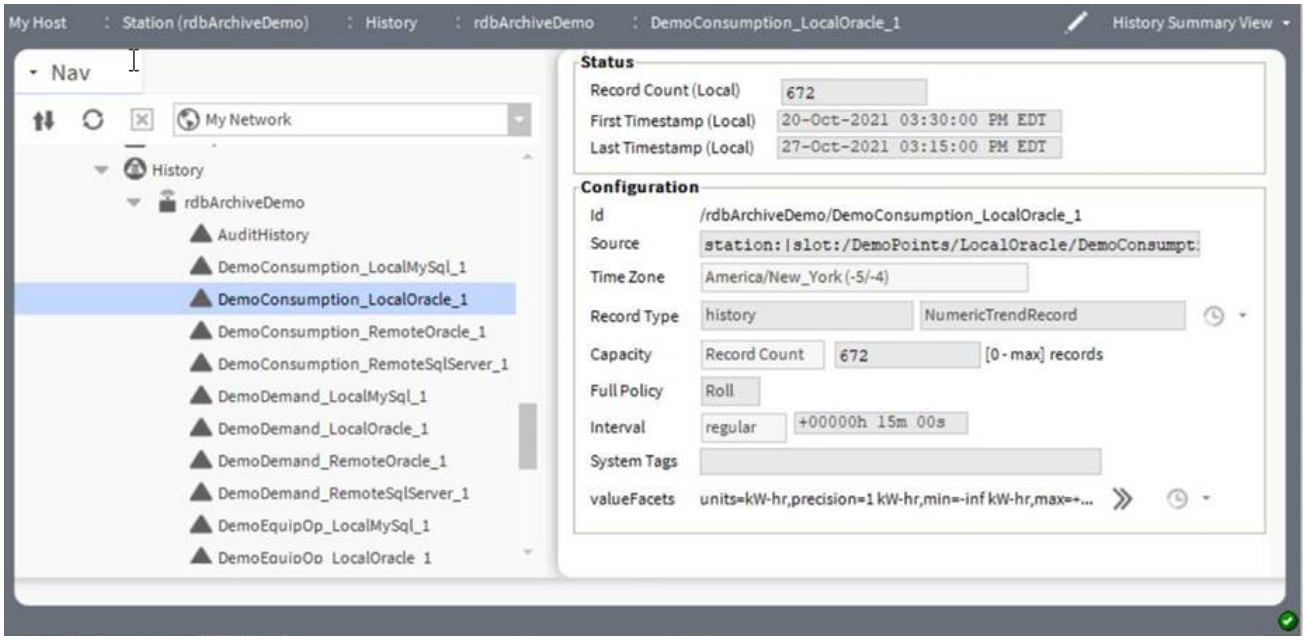
The history being charted in this example has not been configured yet to pull data from an Rdb Archive History Provider.

This history has only a week's worth of local history data to display, even though the time range is configured for year-to-date. More records could be available from an archive.

Figure 6 Local data

The **History Summary View** confirms the local history data is confined to a 672 record capacity (rolling). To access this view, click the drop-down list in the upper right corner of the chart.

Figure 7 Local History Summary View



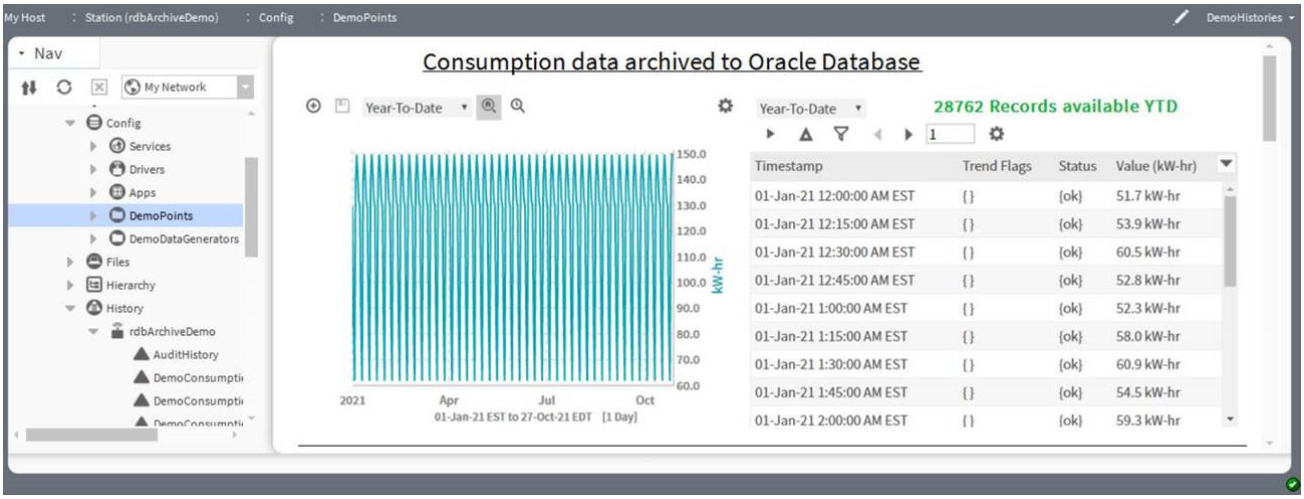
The local data consists of only 672 records, basically a week's worth of 15–minute interval data.

PX example, local and archive data

All views that use history queries can benefit from the archive history provider.

The following PX graphic looks at history data that was pulled from both local and archived sources using an archive provider for an Oracle database. You can configure MySQL and SQL server databases accordingly. All that is required is to set up the Rdb Archive History Providers to reference those databases as well.

Figure 8 PX report drawing from local and archived history data



To debug queries, you can turn on a new "rdb.archiveHistoryProvider" logger to FINE level.

Batch history capacity

The Archive History Provider feature allows queries against local and archived history records. The archived records come from an external data store, typically a relational database. Once the provider is configured and operational, and history data are available from an external data store, you have the opportunity to update the capacity of locally-stored histories to reduce local storage requirements.

Local histories are faster to query than archived histories, so you should consider a local history capacity setting that balances your local storage requirements with your common history query time ranges. You should also choose a local history capacity that is acceptable even on occasions when the archive data source is temporarily unavailable (for example, when the remote archive data source is down for maintenance).

Changing each capacity property individually could be a tedious process. This section documents how to use existing tools to make updating the capacity properties as easy as possible.

A locally-stored history can be:

- A local history residing on the station
- A remote history imported to the local station
- A remote history exported to the local station

Each of these requires a different set of steps, which are documented in the three task topics that follow.

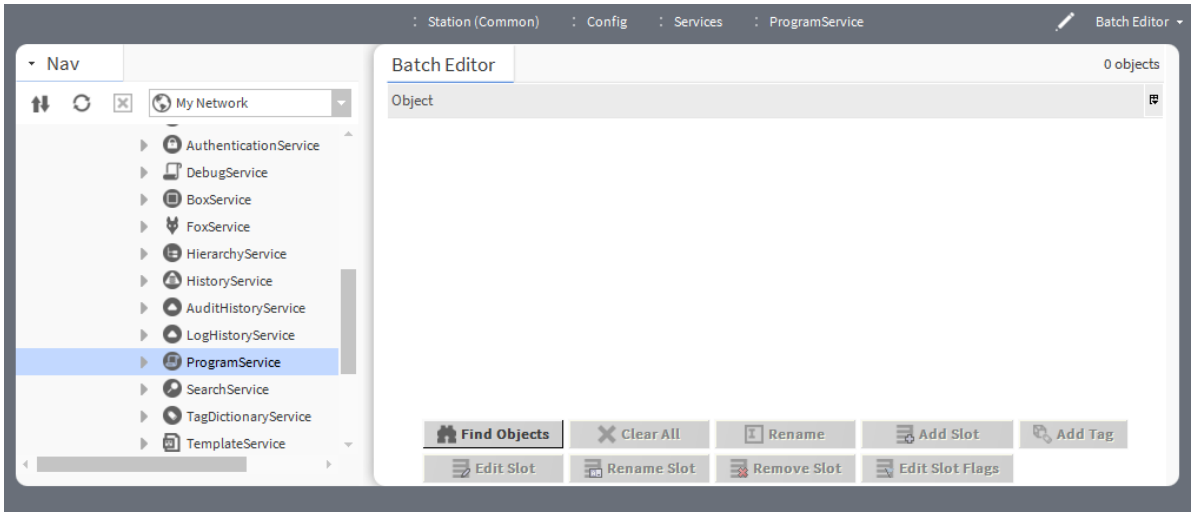
Updating the capacity property of multiple local histories

A station's storage capacity for multiple local histories is limited. To reduce local storage requirements, this procedure uses the **Batch Editor** of the **ProgramService** to configure at one time how many history records a station can store.

Prerequisites: You are connected to a station. The Archive History Provider is configured and operational and history data are available from multiple local data stores.

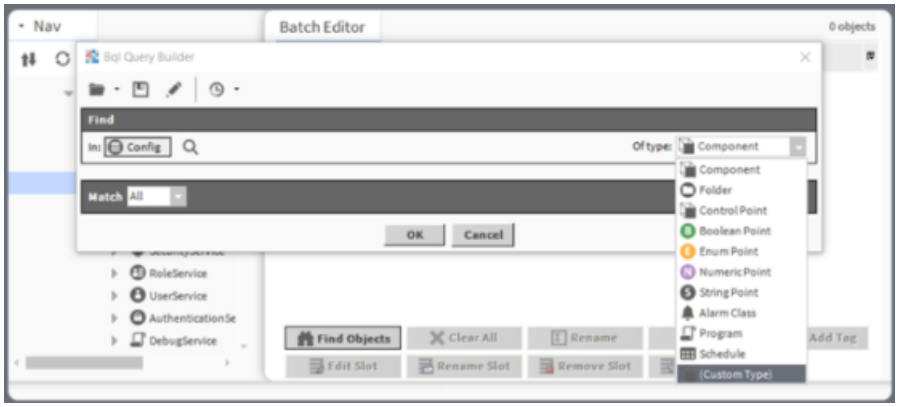
Step 1 Expand **Station→Config→Services**, and double-click **ProgramService**.

The **Batch Editor** opens.



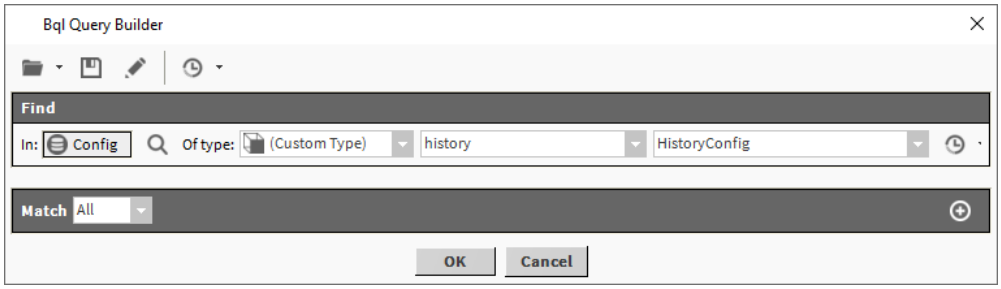
Step 2 To locate the histories, click **Find Objects**.

The **Bql Query Builder** opens.



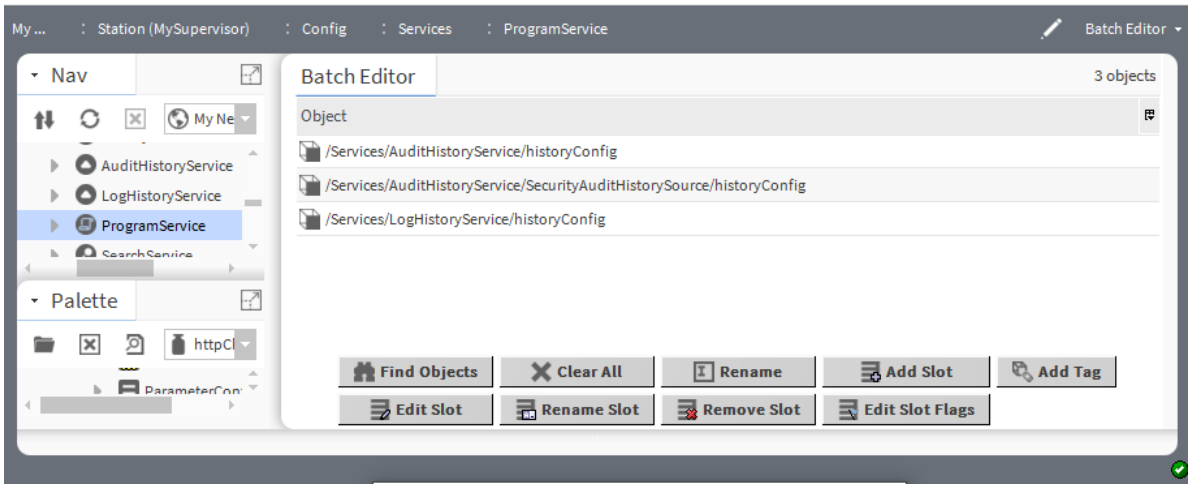
Step 3 For **`Of Type`**, select **(Custom Type)** from the drop-down list.

`Of Type` moves to left end and two more drop-down lists appear.



Step 4 Select `history` and `HistoryConfig` from the drop-down lists and click **OK**.

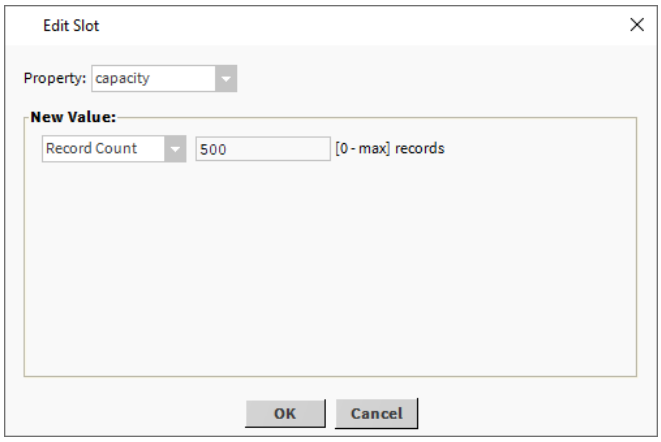
The **Batch Editor** displays the history files it found.



The screen capture shows two audit history files and a single log history file.

Step 5 To configure the number of records to store for a specific history, select the history row and click the **Edit Slot** button at the bottom of **Batch Editor**.

The **Edit Slot** window opens.



Step 6 In the **New Value** pane, fill in the number of records and click **OK**.

The **Batch Editor Results** opens along with the capacity values for each history.

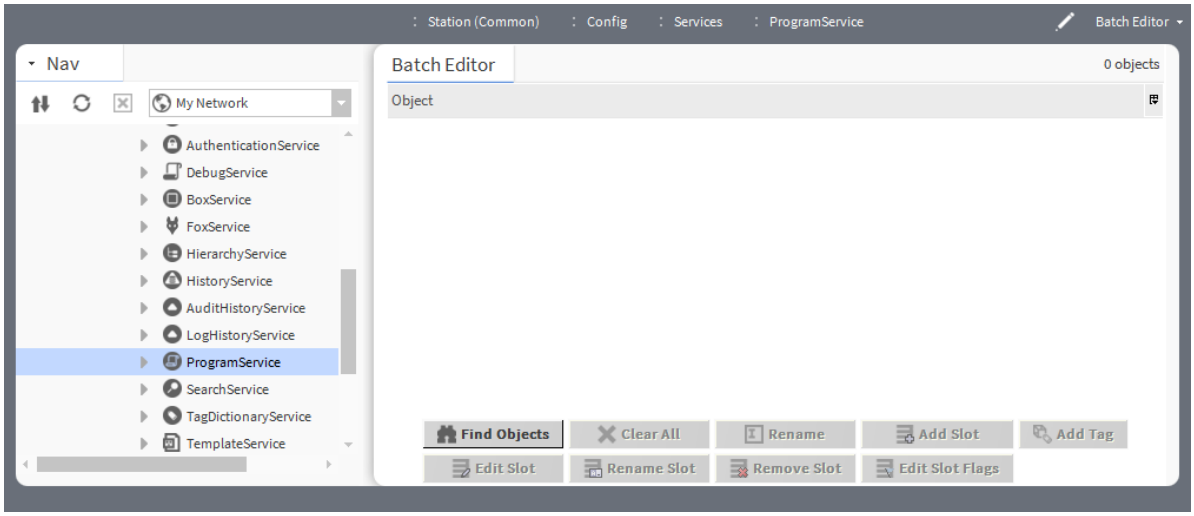
Updating the capacity property of multiple imported histories

A way to include multiple archived histories in the local station is to import them as a batch from the remote database. This procedure uses the **Batch Editor** to configure how many archived histories to import from a remote database.

Prerequisites: You are connected to a remote station that is ready to receive (import) history data. The Archive History Provider is configured and operational.

Step 1 Expand **Station→Config→Services**, and double-click **ProgramService**.

The **Batch Editor** opens.

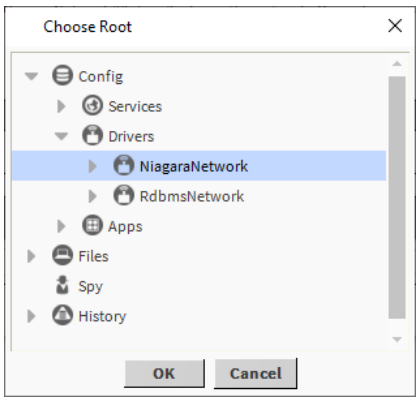


Step 2 To locate the histories to configure, click **Find Objects**.

The **Bql Query Builder** opens.

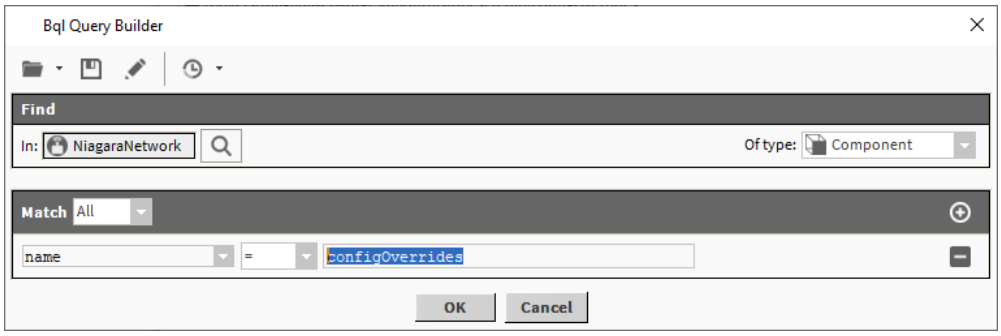
Step 3 Click the search icon ( ) beside the **In** property.

The **Choose Root** window opens.



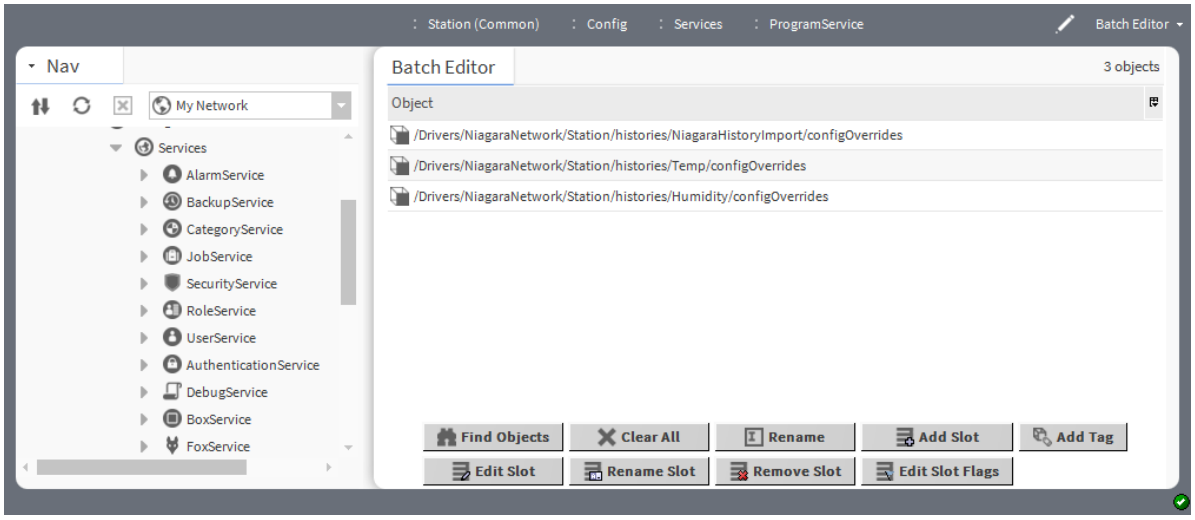
Step 4 Select **NiagaraNetwork** and click **OK**.

The **Bql Query Builder** selects the **NiagaraNetwork**.



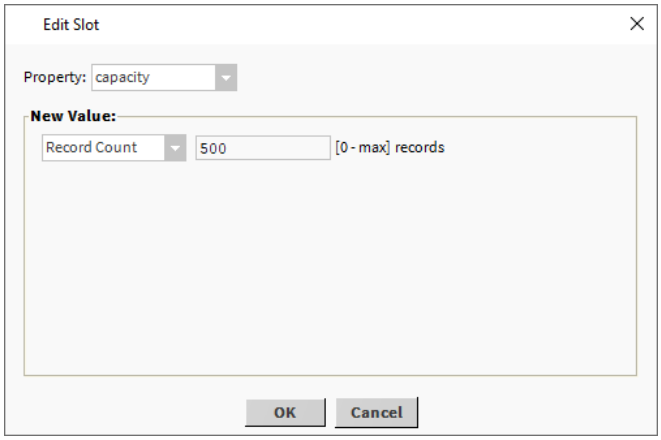
Step 5 To add a search criterion, click the add icon (⊕) on the **Match** bar, change the drop-down list to `name`, type `configOverrides` and click **OK**.

The **Batch Editor** displays the files it found based on your search criteria.



Step 6 To configure the number of records to store for a specific history, select the history row and click the**EditSlot** button at the bottom of the **Batch Editor**.

The **Edit Slot** window opens.



Step 7 In the **New Value** pane, fill in the number of records and click **OK**.

The **Batch Editor Results** window opens along with the new values for capacity.

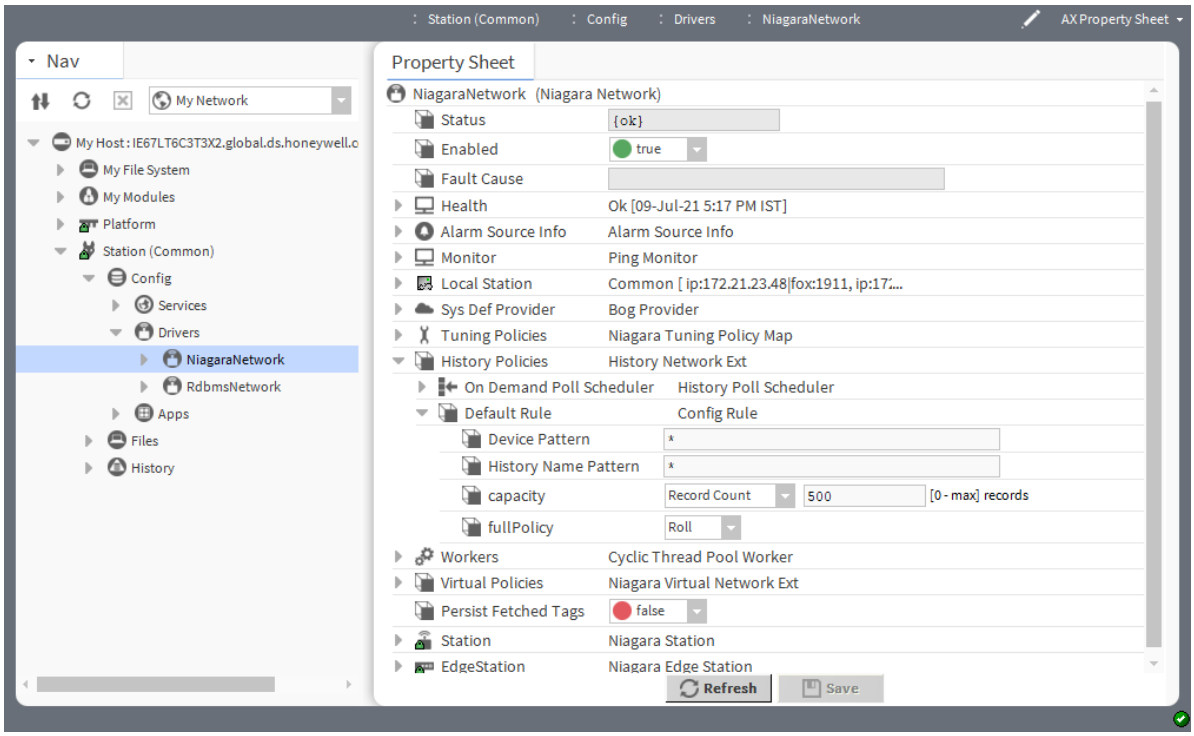
Updating the capacity of remote exported histories

A way to include multiple archived histories in the local station is to export them as a batch from the remote database. Configuring the **`Default Rule`**, a property of the station's **`History Policies`**, controls the number of archived history records that a remote database can export to a station.

Prerequisites: You are connected to a remote station that is ready to receive the exported history data. The Archive History Provider is configured and operational.

Step 1 Expand **Station→Config→Drivers**, right-click **NiagaraNetwork** and click **Views→AX Property Sheet**.

The **Property Sheet** opens.



`History Policies` has a **`Default Rule`** (you can add additional config rules) and each config rule has a **`capacity`** property.

Step 2 In the **capacity** property, fill in the number of history records and click **Save**.

The updates to the capacity are effective from the next time the remote station exports histories.

Exporting history data to an RDBMS

Each RDBMS device has an associated History Device Extension, which you can use to create export history descriptors for exporting data to an RDBMS (MySQL, Oracle, and SqlServer). You can export data manually or on a regular basis.

Prerequisites: You are working in Workbench running on a PC, are connected to a Supervisor station and have access to a relational database on your company's network. Your target database is licensed.

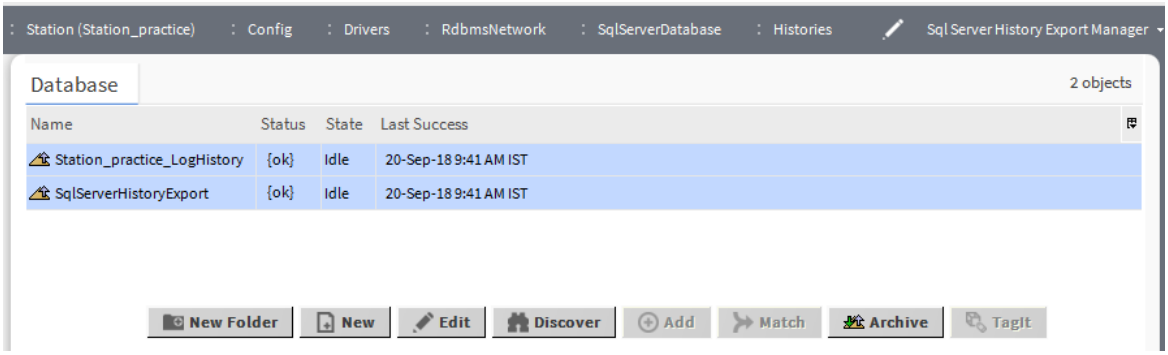
The driver supports two database schema, which provide data export by ID or by type. The **RdbmsNetwork** driver creates the database tables and schema automatically as it exports each history.

Your DBA (Database Administrator) may consider an export by type to be more suitable for the post process ETL (Extracting, Transforming and Loading) of the data beyond the flat file produced by the driver. This is because the driver creates fewer tables and requires fewer permissions on the database.

In the following steps, the term RDB refers to any of the database types. The data export by History ID is the default mode, which is more efficient when using the Rdb Archive History Provider to read the data back into the station at query time.

Step 1 In the Nav tree, expand **Config→Drivers→RdbmsNetwork**, expand your RDB node and double-click the **Histories** node.

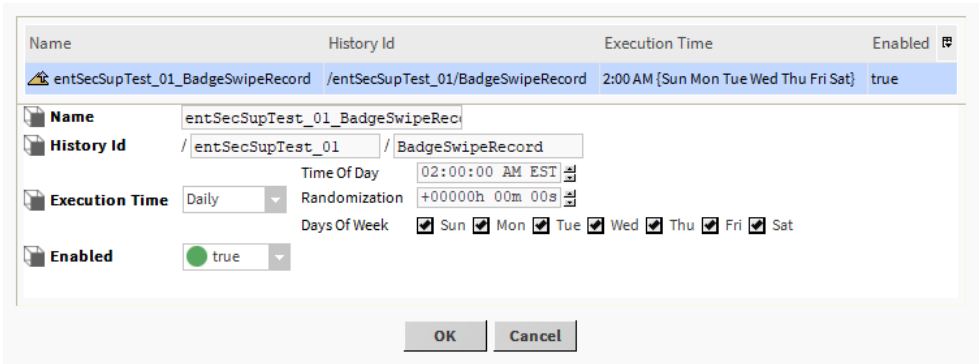
The **History Export Manager** opens.



Step 2 Do one of the following:

- To create a new export descriptor, click the **New** button.
- To discover existing histories that are available to export, click the **Discover** button, find and select a history whose data you want to export.

If you clicked **New**, the **New** window opens.



Step 3 Configure the properties, and click **OK**.

The new history export descriptor(s) appear(s) in the **Database** (lower) pane of the **History Export Manager**.

Step 4 To initiate an export action, do one of the following:

- Select one or more history descriptors in the **Database** pane and click the **Archive** button.
- Right-click on a single history descriptor in the **Database** pane and click **Actions→Execute**.
- Using the **`Daily`** or **`Interval`** settings, as set in the **New** or **Edit** windows, allow the export to occur, as scheduled.

The **Database** pane displays the status and time of the last export action in the Status and Last Success columns, respectively. Each export descriptor appears under the **Histories** node in the Nav tree.

The first time you export histories, the driver creates meta tables. The HISTORY_CONFIG and HISTORY_TYPE meta tables include a column for DB_TIMEZONE. This column is based on the **`Timestamp Storage`** property on the rdb Database device. The DB_TIMEZONE column stores the actual timezone that the database is currently using to store timestamps (this is different from the pre-existing TIMEZONE column, which reflects the timezone in which the history was created but not the one in which it is being stored). When records are being exported in UTC mode, the DB_TIMEZONE column updates accordingly.

For more detailed information about the **History Export Manager**, refer to the *Niagara Drivers Guide*.

Export by history ID

The driver stores each exported history in a dedicated table with a name that corresponds to the history name in the framework. To ensure uniqueness and to make it possible to trace a history back to its source via the descriptive HISTORY_CONFIG table, the driver adds an incrementing number as a suffix to duplicate names.

NOTE: For an Oracle database, the maximum length of a table name is 30 characters.

Numeric writable export example

Figure 9 Example

By History ID Numeric Writable from table RDBSTATION_NUMERICWRITABLE

ID	TIMESTAMP	TRENDFLAGS	STATUS	VALUE	TRENDFLAGS_TAG	STATUS_TAG
2	07-NOV-13 15.48.00.023000000	1	0	2.476382732	{start}	{ok}
3	07-NOV-13 15.50.01.027000000	0	0	0.086565435	{ }	{ok}
4	07-NOV-13 15.52.01.013000000	0	0	0.040000558	{ }	{ok}
5	07-NOV-13 15.54.00.005000000	0	0	1.321054816	{ }	{ok}
6	07-NOV-13 15.56.00.022000000	0	0	1.294879913	{ }	{ok}
7	07-NOV-13 15.58.00.018000000	0	0	1.195041418	{ }	{ok}

In this example:

- The RdbmsNetwork maintains the ID and uses it for database indexing. The RdbmsNetwork driver does not use this value in a station.
- TIMESTAMP records when the value was logged and can be localized to the exporting station or the source station. This choice is stored in the HISTORY_CONFIG table.
- The VALUE column data type changes according to the type of the point exported, for example, double, float, enum, string or boolean.
- VALUE can be `{null}`, for example, where `{NaN}` is recorded in the source history.
- The driver does not export point facets (units) from the database.

The HISTORY_CONFIG table keeps track of the exported tables as in this example:

Figure 10 History_config table

By History ID Numeric Writable from table RDBSTATION_NUMERICWRITABLE Data Types

COLUMN NAME	DATA TYPE	NULLABLE	DATA DEFAULT	COLUMN ID	PRIMARY KEY
ID	NUMBER	No		1	Yes
TIMESTAMP	TIMESTAMP(6)	Yes		2	
TRENDFLAGS	NUMBER	Yes		3	
STATUS	NUMBER	Yes		4	
VALUE	FLOAT	Yes		5	
TRENDFLAGS_TAG	VARCHAR2(500 BYTE)	Yes		6	
STATUS_TAG	VARCHAR2(500 BYTE)	Yes		7	

Note that Boolean exports use a datatype of Char(1 Byte) for the VALUE column

History_Config when using BY_HISTORY_ID exports

Figure 11 Example

History_Config when using BY_HISTORY_ID exports

ID	ID_	HISTORYNAME	SOURCE	SOURCEHANDLE	TIMEZONE	INTERVAL_	SYSTE
1	/rdbStation/AuditHistory		station:|h:11	null	Europe/London (+0/+1)	true:60000	
2	/rdbStation/BooleanWritable		station:|slot:/BooleanWritable/BooleanInterval	h:12b2	Europe/London (+0/+1)	false:240000	
3	/rdbStation/LogHistory		station:|h:13	null	Europe/London (+0/+1)	true:60000	
4	/rdbStation/NumericWritable		station:|slot:/NumericWritable/NumericInterval	h:12a8	Europe/London (+0/+1)	false:120000	
5	/rdbStation/StoreDoor		station:|slot:/StoreDoor/BooleanCov	h:12b8	Europe/London (+0/+1)	true:60000	

YSTEMTAGS	VALUEFACETS	TABLE_NAME	DB_TIMEZONE
		RDBSTATION_AUDITHISTORY	Europe/London (+0/+1)
	trueText=s:true|falseText=s:false	RDBSTATION_BOOLEANWRITABLE	Europe/London (+0/+1)
		RDBSTATION_LOGHISTORY	Europe/London (+0/+1)
	units=u:null;;;;|precision=i:1|min=d:-inf|max=d:+inf	RDBSTATION_NUMERICWRITABLE	Europe/London (+0/+1)
	trueText=s:Open|falseText=s:Closed	RDBSTATION_STOREDOOR	Europe/London (+0/+1)

The data types of this example are:

Figure 12 History config data types

Data Types of History_config

COLUMN NAME	DATA TYPE	NULLABLE	COLUMN ID	PRIMARY KEY
ID	NUMBER	No	1	Yes
ID_	VARCHAR2(500 BYTE)	Yes	2	
HISTORYNAME	VARCHAR2(500 BYTE)	Yes	3	
SOURCE	VARCHAR2(500 BYTE)	Yes	4	
SOURCEHANDLE	VARCHAR2(500 BYTE)	Yes	5	
TIMEZONE	VARCHAR2(500 BYTE)	Yes	6	
INTERVAL_	VARCHAR2(500 BYTE)	Yes	7	
SYSTEMTAGS	VARCHAR2(500 BYTE)	Yes	8	
VALUEFACETS	VARCHAR2(500 BYTE)	Yes	9	
TABLE_NAME	VARCHAR2(500 BYTE)	Yes	10	
DB_TIMEZONE	VARCHAR2(500 BYTE)	Yes	11	

The SOURCE column shows the origin of the history. A space separates the source point ORD and the route taken by the exported data to Supervisor. It is occasionally necessary to extend the length of the SOURCE column when station naming conventions make the ORDs very long. The DBA performs this using SQL or database management tools. This condition exhibits as a “Data Truncation” error in the application director output of the station.

The INTERVAL column shows the collection interval of the source history extension in milliseconds, the string prefix is “false” for Interval type histories.

Export by history type

In this method, the driver creates a single table for each type of record exported, recording the source point in that table

This HISTORY_NUMERIC_TREND_RECORD is an example of the table.

Figure 13 History type example

HISTORYNUMERICTRENDRECORD

ID	TIMESTAMP	TRENDFLAGS	STATUS	VALUE	HISTORY_ID	TRENDFLAGS_TAG	STATUS_TAG
55	07-NOV-13 17.34.00.022000000	0	0	1.717286944	/rdbStation/NumericWritable	{ }	{ok}
56	07-NOV-13 17.36.00.007000000	0	0	1.563038707	/rdbStation/NumericWritable	{ }	{ok}
57	07-NOV-13 17.38.00.028000000	0	0	1.413464427	/rdbStation/NumericWritable	{ }	{ok}
58	07-NOV-13 17.40.01.018000000	0	0	0.001343357	/rdbStation/NumericWritable	{ }	{ok}
59	07-NOV-13 17.42.00.517000000	1	0	2.121934891	/rdbStation/AnotherNumericValue	{start}	{ok}
60	07-NOV-13 17.42.00.015000000	0	0	2.121934891	/rdbStation/NumericWritable	{ }	{ok}
61	07-NOV-13 17.42.30.597000000	0	0	0.911328733	/rdbStation/AnotherNumericValue	{ }	{ok}
62	07-NOV-13 17.43.00.667000000	0	0	0.148435533	/rdbStation/AnotherNumericValue	{ }	{ok}

The data types of this descriptive table are:

Figure 14 History numeric trend data types

Datatypes of example HISTORYNUMERICTRENDRECORD table

COLUMN NAME	DATA TYPE	NULLABLE	COLUMN ID	PRIMARY KEY
ID	NUMBER	No	1	Yes
TIMESTAMP	TIMESTAMP(6)	Yes	2	
TRENDFLAGS	NUMBER	Yes	3	
STATUS	NUMBER	Yes	4	
VALUE	FLOAT	Yes	5	
HISTORY_ID	VARCHAR2(500 BYTE)	Yes	6	
TRENDFLAGS_TAG	VARCHAR2(500 BYTE)	Yes	7	
STATUS_TAG	VARCHAR2(500 BYTE)	Yes	8	

The point type to table mapping required to achieve this type of export are described in the HISTORY_ TYPE_MAP table:

Figure 15 History type map

Data from HISTORY_TYPE_MAP

ID	ID_	TIMEZONE	RECORDTYPE	VALUEFACETS	TABLE_NAME	DB_TIMEZONE
2	/rdbStation/AuditHistory	Europe/London (+0/+1)	history:AuditRecord		HISTORYAUDITRECORD	Europe/London (+0/+1)
3	/rdbStation/BooleanWritable	Europe/London (+0/+1)	history:BooleanTrendRecord	trueText=s:true|falseText=s:false	HISTORYBOOLEANTRENDRECORD	Europe/London (+0/+1)
4	/rdbStation/LogHistory	Europe/London (+0/+1)	history:LogRecord		HISTORYLOGRECORD	Europe/London (+0/+1)
5	/rdbStation/NumericWritable	Europe/London (+0/+1)	history:NumericTrendRecord	units=u:null;;;;|precision=i:1|min=d:-inf|max=d:+inf	HISTORYNUMERICTRENDRECORD	Europe/London (+0/+1)
6	/rdbStation/StoreDoor	Europe/London (+0/+1)	history:BooleanTrendRecord	trueText=s:Open|falseText=s:Closed	HISTORYBOOLEANTRENDRECORD	Europe/London (+0/+1)
7	/rdbStation/AnotherNumericValue	Europe/London (+0/+1)	history:NumericTrendRecord	units=u:null;;;;|precision=i:1|min=d:-inf|max=d:+inf	HISTORYNUMERICTRENDRECORD	Europe/London (+0/+1)

The data types of this example are:

Figure 16 History type map data types

Datatypes for HISTORY_TYPE_MAP				
COLUMN NAME	DATA TYPE	NULLABLE	COLUMN ID	PRIMARY KEY
ID	NUMBER	No	1	Yes
ID_	VARCHAR2(500 BYTE)	Yes	2	
TIMEZONE	VARCHAR2(500 BYTE)	Yes	3	
RECORDTYPE	VARCHAR2(500 BYTE)	Yes	4	
VALUEFACETS	VARCHAR2(500 BYTE)	Yes	5	
TABLE_NAME	VARCHAR2(500 BYTE)	Yes	6	
DB_TIMEZONE	VARCHAR2(500 BYTE)	Yes	7	

Status and trend flags

Common to both types of History Export, the STATUS and STATUS_FLAG columns represent the state of the point at the time the record was recorded where STATUS is the sum of the possible states.

Status flags

State #	State (tag)	State #	State (tag)	State #	State (tag)
0	{ok}	4	{down}	32	{overridden}
1	{disabled}	8	{alarm}	64	{null}
2	{fault}	16	{stale}	128	{unackedAlarm}

For example:

STATUS	STATUS_TAG
15	{disabled, fault, down, alarm}
136	(a;ar,. imacledA;ar,}

Trend flags

The TRENDFLAGS and TRENDFLAGS_TAG columns record event information about the history record, using the same 'sum' method described above.

Trend #	Trend (tag)	Trend #	Trend (tag)	Trend #	Trend (tag)
1	{start}	4	{Hidden}	16	{Interpolated}
2	{OutofOrder}	8	{Modified}		

Importing history data from an Rdbms database

Each RDBMS device (MySQL, Oracle, and SqlServer) has an associated history device extension, which you can configure and use to create one or more import descriptors. Using an import descriptor you can pull data that did not originate in your station into your station for modeling as a Niagara history. In the following steps, the acronym RDB refers to any of these relational database types.

Step 1 Expand **Drivers→RdbmsNetwork**, expand your RDB, right-click the **Histories** node in the Nav tree and click **Views→Rdbms History Import Manager**.

The **Rdbms History Import Manager** view opens.

Step 2 To create an import descriptor, click the **New** button.

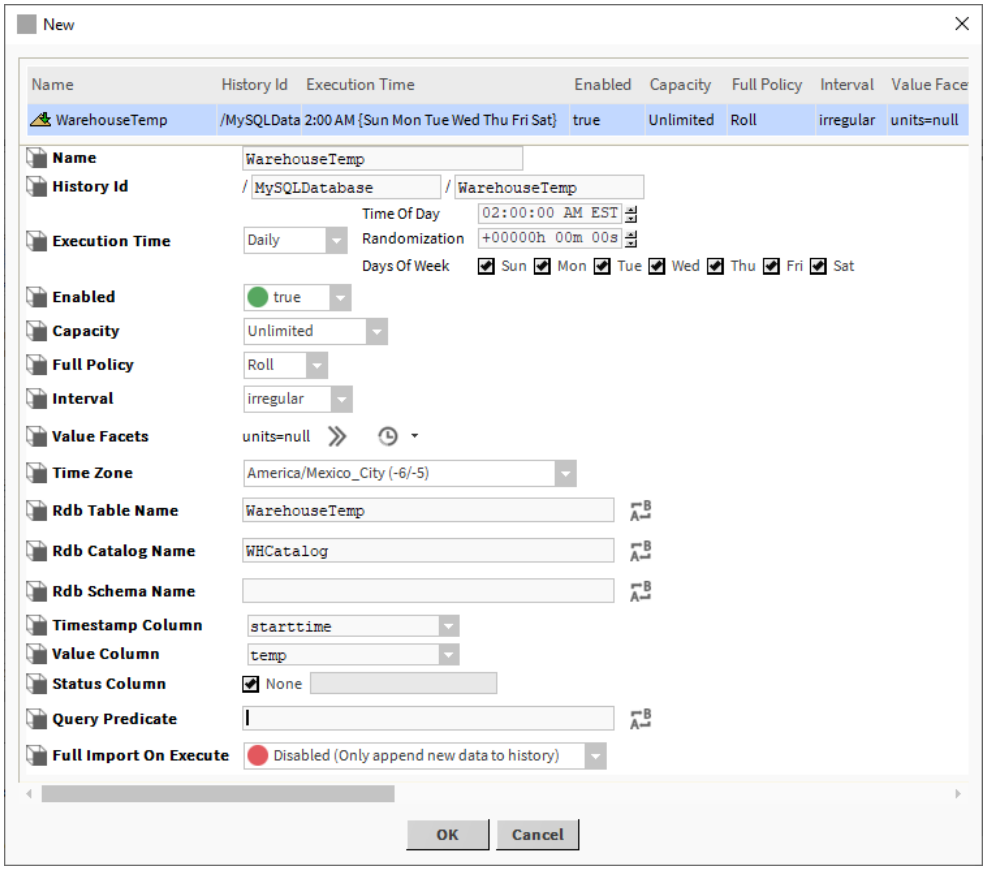
As an alternative to the **New** button and window, you can click the **Discover** button and select a discovered RDB table that you want to import and model as a Niagara history.

If you clicked **New**, the **New** window opens.

Step 3 Select `Rdb History Import` (the default) from the **Type to Add** drop-down list and use the **Number to Add** property to enter the number of import descriptors to add and click **OK**.

You need to add one unique import descriptor for each RDB table that contains the data to import.

A second **New** window opens.



The table at the top of the window should be ordered by timestamp as are other system histories.

In addition to **Enabled** set to `true` (the default), you need to configure at least these properties and click **OK**:

- **Name** provides a name for the descriptor.
- **History Id** defines the history device name and history name.
- **Execution Time** configures when to perform the import.

If you discover these descriptors, rather than create them individually, the system populates the **Timestamp Column**, **Value Column** and **Status Column** properties, otherwise, you can enter column values.

The new history import descriptor(s) appear in the **Database** (lower) pane and each import descriptor appears under the **History** node in the Nav tree.

Step 4 To initiate an import action, do one of the following:

- Select one or more history descriptors in the database pane and click the **Archive** button.

- Right-click on a single history descriptor in the database pane and click **Actions→Execute**.
- Do nothing. Based on the **Daily** or **Interval** settings, as set in the **New** or **Edit** windows, the import occurs as scheduled.

The **Database** pane displays the status and time of the last import action in the Status and Last Success columns, respectively.

To view the history data you imported, use a history report (expand **History**, expand the station and double-click a history name), PX view or chart.

Unix time conversion for MySQL

Unix time is a system for describing instants in time, defined as the number of seconds that have elapsed since 00:00:00 Coordinated Universal Time (UTC), Thursday, 1st January 1970. Unix time is a single signed integer number with which the driver time-stamps the Database records.

To convert Unix time into a readable date format in MySQL add the following query statement: FROM_UNIXTIME(TIMESTAMP/1000). This statement results in a timestamp with this syntax: YYYY-MM-DD HH:MM:SS.ssss

Updating an existing database to support Unicode and UTC

The computing industry standard known as Unicode handles text expressed in most of the world's writing systems. UTC (Coordinated Universal Time) is the primary time standard by which the world regulates clocks. It does not observe daylight saving time. This procedure updates a database to support Unicode and UTC.

Prerequisites: Before connecting to your database for the first time you enabled **Use Unicode Encoding Scheme**.

Step 1 Back up the database(s) to update.

CAUTION: This Unicode and UTC upgrade procedure is one-way only and cannot be reversed. It is highly recommended that you back up the database(s) prior to running the wizard.

Step 2 Right-click the **RdbmsNetwork** node in the Nav tree and click **Update Wizard**

The **Database Update Wizard** window opens to the the **Select Update Procedures** window.

Step 3 Enable one or both properties and click **Next**.

The wizard displays one or more databases to be updated to Unicode.

Step 4 Select the database(s) to update to Unicode and click **Next**.

The wizard completes the update and displays the results for review.

Step 5 To continue, click **Next**.

If you enabled both options, the wizard displays one or more databases to be updated to UTC.

Step 6 Select the database(s) to update to UTC and click **Next**.

If you updated UTC, the wizard opens the **Timezone Update — Timestamp Storage Policy** view.

Step 7 To define the storage policy, select one or the other of the options and click **Next**.

The wizard completes the update and displays the results for review.

Step 8 To complete the update, click **Finish**.

If you updated to Unicode, the wizard automatically sets the **Use Unicode Encoding Scheme** property on the database **Property Sheet** to `true`.

If you updated to UTC, the wizard automatically sets the **Timestamp Storage** property on the database **Property Sheet** to `true` and sets this property to read-only. To make **Timestamp Storage** writable again, right-click the database device and click **Actions→Allow Dialect Modifications**.

When the procedures and databases have been specified in the Update Wizard, the updates are submitted as jobs to the Job Service. The Update Wizard processes the database tables and changes the data types of string-valued columns to the NVARCHAR data types and adjusts the database timestamps to UTC time. The wizard provides visibility on the progress of individual jobs. At a later stage, you can view the jobs in the Job Service. The update operations either complete entirely or roll back entirely if any error occurs.

Updating existing Orion databases to support Unicode

This procedure is for databases that are currently without UTF-8 support.

Step 1 Right-click the rdb Database node in the Nav tree and click **Views→Property Sheet**.

Step 2 Enable the **`Use Unicode Encoding Scheme`** property (set its value to `true`).

This property must be set to `true` before you configure the Orion properties.

Step 3 Open the **`orion`** palette and drag the **OrionMigrator** to the **RdbmsNetwork** node in the Nav tree.

Step 4 Configure the **OrionMigrator** component properties and click **Save**.

Chapter 3 Components

Topics covered in this chapter

♦ rdb module
♦ rdb database modules
♦ orion module

Components include services, folders and other model building blocks associated with a module. You may drag them to a **Property** or **Wire Sheet** from a palette.

Descriptions included in the following topics appear as context-sensitive help topics when accessed by:

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

rdb module

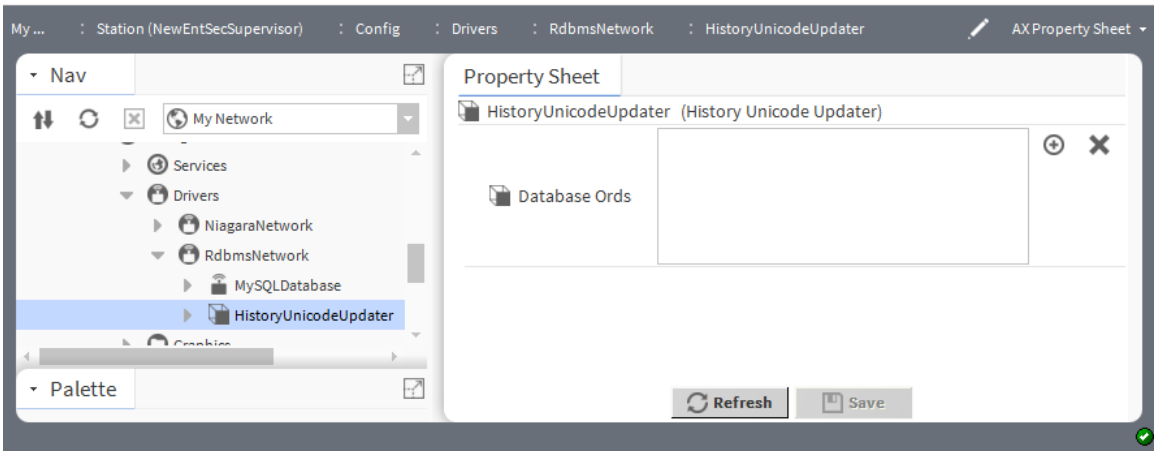
This module provides components for updating RDBMS histories.

These components are required to upgrade an older RDBMS to handle the Unicode text required by most of the world's writing systems and the time standard by which the world regulates clocks.

rdb-HistoryUnicodeUpdater

This component is available on the **rdb** module Palette view. Its granular and discrete functions serve as an alternative to using the Update Wizard. You can use the **Database Ords** property in this component to add one or more database paths to the Updater.

Figure 17 HistoryUnicodeUpdater properties



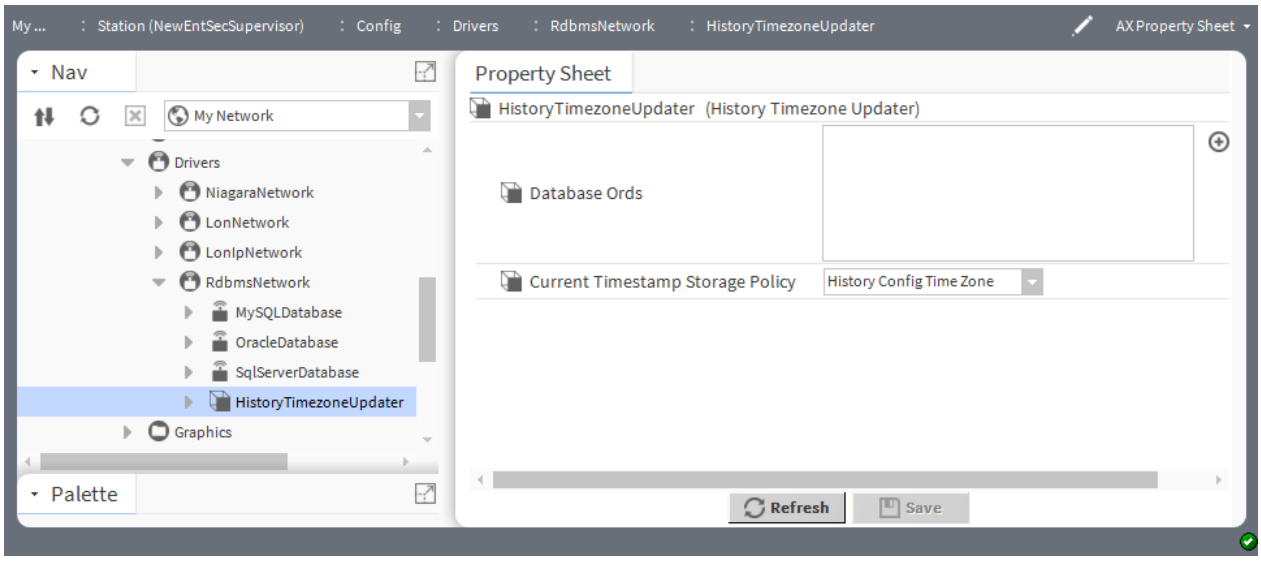
You access this property by double-clicking the **HistoryUnicodeUpdater** node in the Nav tree.

Property	Value	Description
Database Ords	ord	Specifies an ord path to the appropriate updater(s).

rdb-HistoryTimezoneUpdater

This component is available on the **rdb** module Palette view. You can use the **Database Ords** property in this component to add one or more database paths to the Updater.

Figure 18 HistoryTimezoneUpdater properties



You access these properties by double-clicking the **HistoryTimezoneUpdater** node in the Nav tree.

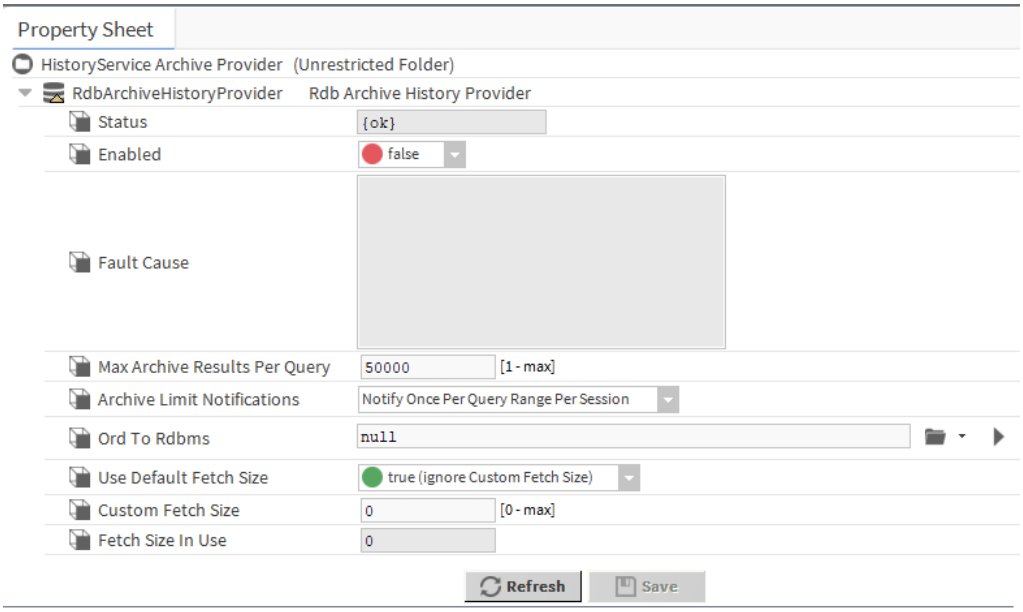
This component's functionality is identical to that achieved by using the Update Wizard. It is available via the palette to provide granular and discrete functionality as an alternative to using the Update Wizard.

Property	Value	Description
Database Ords	ord	Specifies an ord path to the appropriate updater(s).
Current Timestamp Storage Policy	drop-down	Selects the policy that defines the time zone: `Station Time Zone` uses the time zone of the station to update history records. `History Config Time Zone` uses the time zone specified in the History Configuration properties for time zone conversion.

rdb-HistoryServiceArchiveProvider

This component supplements queries against local history records with archived history records that were previously exported to a remote (cloud) RDBMS using the standard drivers: rdbSqlServer, rdbMySQL, and rdbOracle. These archived history records are from an external data store, a relational database.

Figure 19 HistoryServiceArchiveProvider properties



You access these properties by expanding **Config→Services→Archive History Providers** and double-clicking the **RdbArchiveHistoryProvider** node in the Nav tree.

In addition to the standard properties (Status, Enabled, and Fault Cause), these properties are unique to this component.

Property	Value	Description
Max Archive Results Per Query	number	Specifies the number of archived history records to allow per query.
Archive Limit Notifications	drop-down list (defaults to `Notify Once Per Query Range Per Session`)	Selects the message to display when the limit is exceeded for any history query initiated from Workbench. `Notify Once Per Query Range Per Session` limits the frequency of warnings to once per query range per session `Never Notify` turns notification off. `Always Notify` displays a warning every time the query exceeds the maximum archive results.
Ord To Rdbms	ORD	Defines the path to the archive database.
Use Default Fetch Size	`true (ignore Custom Fetch Size)` (default) `false (use Custom Fetch Size)`	Does not use (`true`) or uses (`false`) the **`Custom Fetch Size`** to retrieve archive history records.
Custom Fetch Size	number	Specifies the number of archive history records to retrieve from the database in the cloud.
Fetch Size in Use	read-only	Reports the number of records retrieved when the **`Custom Fetch Size`** is being used.

rdb database modules

These modules provide the primary RDBMS components for setting up the RDBMS network, databases and extensions.

The RDBMS modules include these palettes, each of which supports a relational database:

- **rdbHsqlDb**
- **rdbMySQL**
- **rdbOracle**
- **rdbSqlServer**

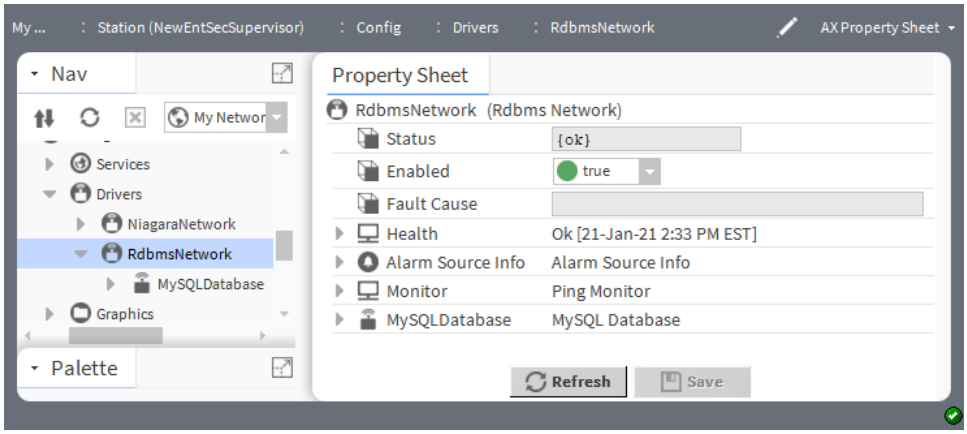
rdb-RdbmsNetwork

Like other framework networks, this component provides a top-level component for all Rdbms driver components. In keeping with the standard framework driver architectural model, many of the **RdbmsNetwork** components, device extensions, and views resemble those in other drivers.

Each of the rdb palettes (**rdbHsqlDb**, **rdbMySQL**, **rdbOracle** and **rdbSqlServer**) provides the **RdbmsNetwork** component.

The **Device Manager** is the view associated with this component.

Figure 20 RdbmsNetwork properties



You access this view by expanding **Config→Drivers** right-clicking **RdbmsNetwork** and clicking **Views→AX Property Sheet**.

Most of these are standard properties: Status, Enabled, Fault Cause, Health, and Alarm Source Info. The *Niagara Drivers Guide* documents the **Monitor** properties. The "rdbMySQL-MySQLDatabase" topic in this guide documents the **MySQLDatabase** properties.

rdb-RdbmsFolder

This component is used to organize databases under an **RdbmsNetwork**.

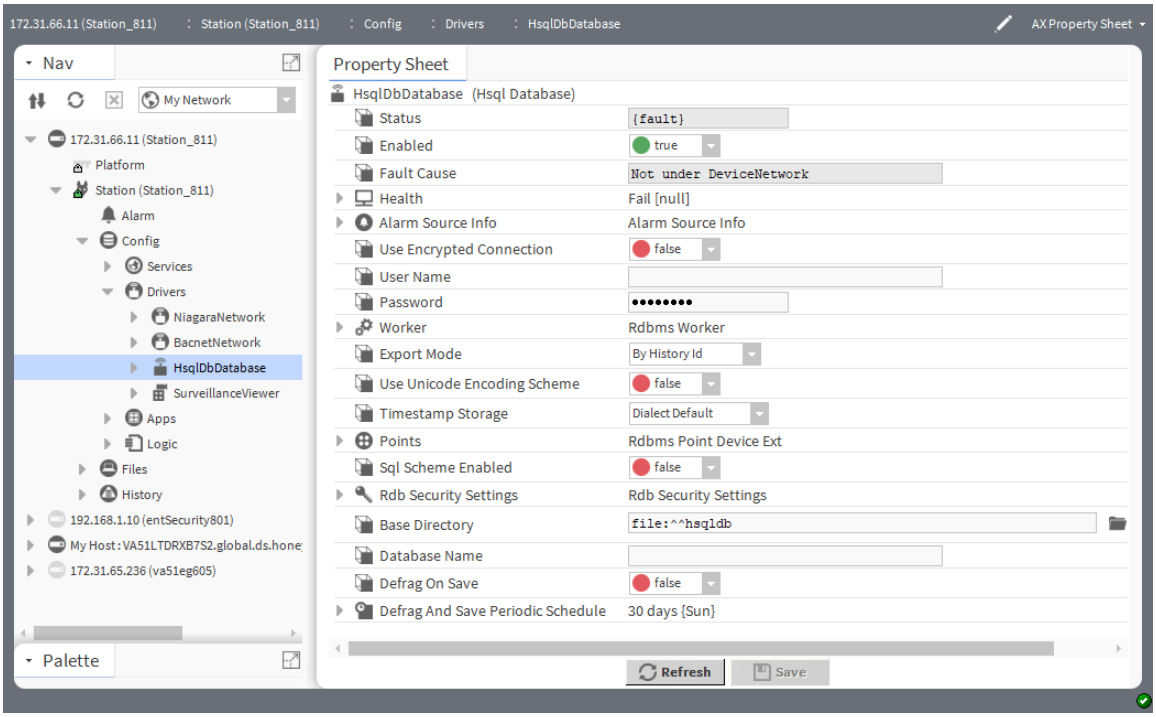
You create this folder using the **New Folder** button in the **Device Manager** view for any **RdbmsNetwork**. The view associated with this component is the **Device Manager**.

rdbHsqlDb-HsqlDatabase

This component models an HsqlDb (Hyperthreaded Structured Query Language Database) relational database in a remote controller station. The HSQL Development Group maintains the standard for this database, which is available under a BSD type (free) license.

The **rdbHsqlDb** palette provides the HsqlDbDatabase components.

Figure 21 HsqlDbDatabase properties



You access the properties to configure the driver for this database by expanding **Config→Drivers→Rdbms-Network**, and double-clicking the **HsqlDbDatabase** node in the Nav tree.

In addition to the standard properties (Status, Enabled, Fault Cause, Health and Alarm Source Info), these properties support this component.

Property	Value	Description
Use Encrypted Connection	`true` or `false` (default)	Indicates if the connection between the station and the database is secure (`true`) or not secure (`false`). To ensure that your system cannot be hacked, leave this property set to `true`. Change it only if your database does not support data encryption.
User Name	text	Defines the user name used to log in to the database. Login credentials must provide sufficient database privileges to allow you to perform one or more (depending on database type) of these commands: CREATE TABLE, CREATE INDEX, CREATE SEQUENCE
Password	text	Defines the password required to log in to the database. The **Confirm** property must be an exact match to the **Password** property.
Worker	additional properties	The "rdb-RdbmsWorker" topic documents these properties.
Export Mode	drop-down list	Specifies how histories are exported to the specified database.

Property	Value	Description
		`By History Id` exports one table per History Id. This is the default value setting. `By History Type` exports one table per History Type. This option may make the data easier to query once exported.
Use Unicode Encoding Scheme	`true` or `false` (default)	Creates history table schemas with the Universal character set Transformation Format (UTF-8) or Unicode data types for string-valued columns to store Asian character sets. `false` maintains backward compatibility with legacy history export mechanisms. `true` enables the NVARCHAR data type for any column in a table that expects string data. **NOTE:** An Update Wizard is available to upgrade existing databases to support Unicode.
Timestamp Storage	drop-down list	Exports or updates history timestamps to Coordinated Universal Time (UTC) enabling the export of history records from different timezones into a common database and be chronologically correct and independent of any specific source timezone characteristics. In other words, exported histories show the timestamp data from where the history is actually stored, making useable histories with a consistent timestamp. `Dialect Default` maintains compatibility with legacy history export mechanisms. This property does not apply to the MySQL device. `Local Time Stamp` applies to Orion databases only and does not apply to history exports. `Utc Timestamp` exports all subsequent histories with UTC timestamps. `Utc Millis` applies to Orion databases only and does not apply to history exports. **NOTE:** An Update Wizard is available to upgrade existing databases to support Unicode.
Points	folder	Holds all points.
Sql Scheme Enabled	`true` or `false` (default)	Permits (`true`) and prohibits (`false`) the use of ORDs that contain `sql:`. If a BFormat contains `sql:`, users can directly query the database bypassing framework access control and make direct changes to the database. This is a security risk. Best security practice sets this property to `false`.
Rdb Security Settings	additional properties	The “rdb-RdbSecuritySettings” topic documents these properties.
Base Directory	chooser	Defines the path that points to the location of the Hsql database. A typical configuration is to create a folder directly under the station (in the file space). For example, if the folder is named **hsqldb**, the filepath to this folder would be: `file:^hsqldb`.

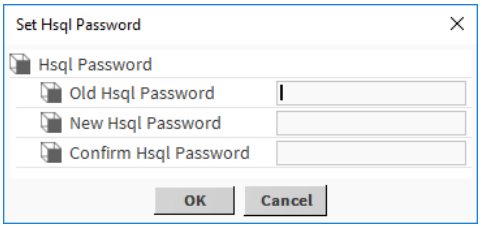
Property	Value	Description
Database Name	text	Provides the name of the database.
Defrag On Save	`true` or `false` (default)	Configures the driver to defragment when it saves the database.
Defrag And Save Periodic Schedule	control-TimeTrigger with multiple properties	*Getting Started with Niagara* documents these properties.

Actions

This component supports the following actions:

- **Ping** sends a message to a network object (device, database, etc). The message provokes a response, which indicates the current state of the object.
- **Allow Dialect Modifications** makes the **`Timestamp Storage`** property writable within the **RdbmsNetwork Update Wizard**. When the wizard converts a database to UTC (Coordinated Universal Time), it sets this property to read-only.
- **Defrag And Save** sequences through the database to remove deleted records and save the resulting defragmented database.
- **Hsql Password** opens a Password window that is used to change the Hsql password.

Figure 22 Set Hsql Password window



Although a new or upgraded HsqlDbDatabase automatically connects to the station using the factory-default password, you should always set a new strong password of your own in each controller station.

The new password must meet the following requirements:

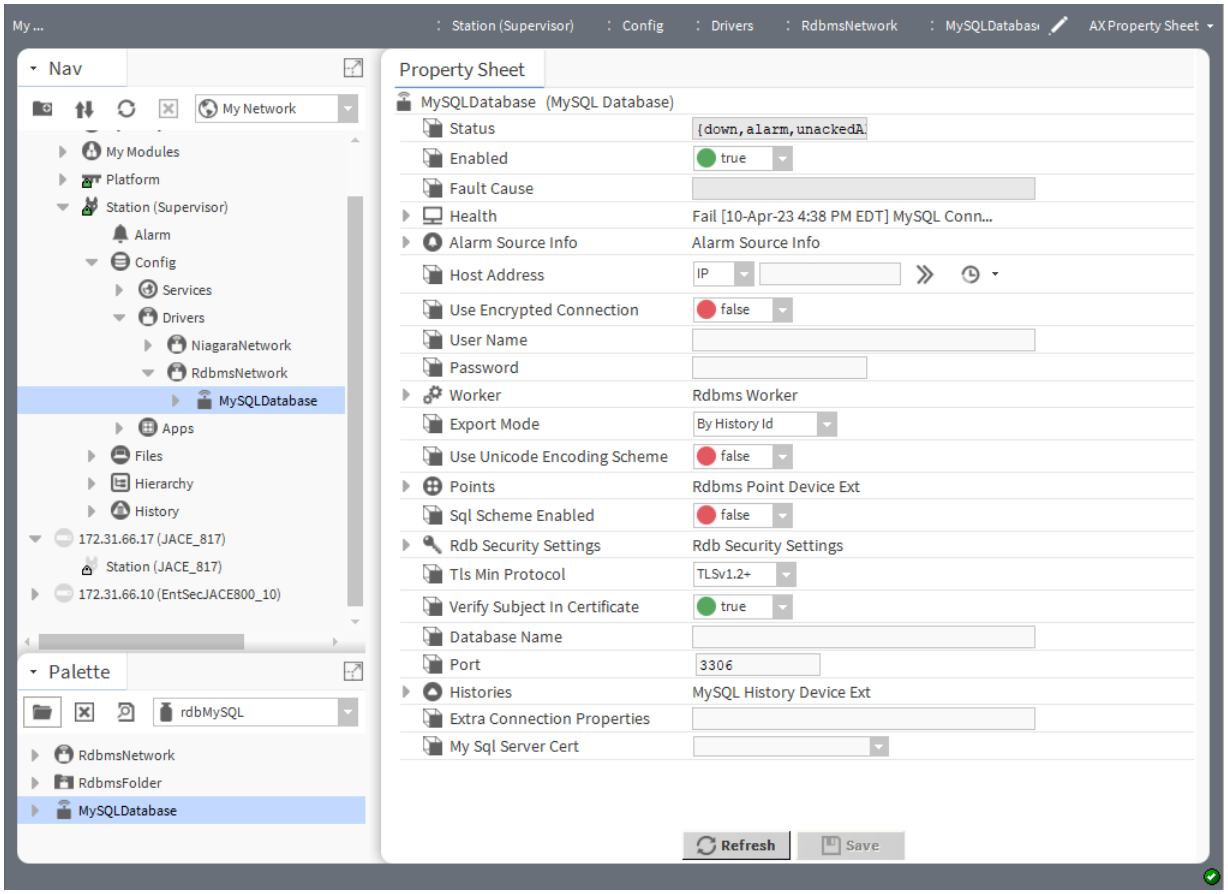
- Length: at least 10 characters
- Complexity: at least 1 digit, 1 upper case character and 1 lower case character.
- Allowed character types:
  - lowercase letters
  - uppercase letters
  - special characters: `@ # ! $ & + > < ] [ ) (`

rdbMySQL-MySQLDatabase

This component models a MySQL relational database, which is an Oracle Corporation relational database management system (RDBMS) that requires a GPL or proprietary license. It is available in the **`rdbMySQL`** palette.

NOTE: The rdbMySQL-MySQLDatabase component was tested with the MySQL connector version "`mysql-connector-java-8.0.24`". Use of earlier versions of this connector is not recommended and not supported.

Figure 23 MySQLDatabase properties



You access the properties to configure the driver for this database by expanding **Config→Drivers→Rdbms-Network**, and double-clicking the **MySQLDatabase** node.

In addition to the standard properties (Status, Enabled, Fault Cause, Health and Alarm Source Info), these properties support this component.

Property	Value	Description
Host Address	IP address	Sets the IP address or hostname of the computer platform where the database resides. A Dialup selection option is available, if required. This property does not apply to the MySQL device.
Use Encrypted Connection	`true` or `false` (default)	Indicates if the connection between the station and the database is secure (`true`) or not secure (`false`). To ensure that your system cannot be hacked, leave this property set to `true`. Change it only if your database does not support data encryption.
User Name	text	Defines the user name used to log in to the database. Login credentials must provide sufficient database privileges to allow you to perform one or more (depending on database type) of these commands: CREATE TABLE, CREATE INDEX, CREATE SEQUENCE

Property	Value	Description
Password	text	Defines the password required to log in to the database. The **`Confirm`** property must be an exact match to the **`Password`** property.
Worker	additional properties	The "rdb-RdbmsWorker" topic documents these properties.
Export Mode	drop-down list	Specifies how histories are exported to the specified database. `By History Id` exports one table per History Id. This is the default value setting. `By History Type` exports one table per History Type. This option may make the data easier to query once exported.
Use Unicode Encoding Scheme	`true` or `false` (default)	Creates history table schemas with the Universal character set Transformation Format (UTF-8) or Unicode data types for string-valued columns to store Asian character sets. `false` maintains backward compatibility with legacy history export mechanisms. `true` enables the NVARCHAR data type for any column in a table that expects string data. **NOTE:** An Update Wizard is available to upgrade existing databases to support Unicode.
Timestamp Storage	drop-down list	Exports or updates history timestamps to Coordinated Universal Time (UTC) enabling the export of history records from different timezones into a common database and be chronologically correct and independent of any specific source timezone characteristics. In other words, exported histories show the timestamp data from where the history is actually stored, making useable histories with a consistent timestamp. `Dialect Default` maintains compatibility with legacy history export mechanisms. This property does not apply to the MySQL device. `Local Time Stamp` applies to Orion databases only and does not apply to history exports. `Utc Timestamp` exports all subsequent histories with UTC timestamps. `Utc Millis` applies to Orion databases only and does not apply to history exports. **NOTE:** An Update Wizard is available to upgrade existing databases to support Unicode.
Points	folder	Holds all points.
Sql Scheme Enabled	`true` or `false` (default)	Permits (`true`) and prohibits (`false`) the use of ORDs that contain `sql:`. If a BFormat contains `sql:`, users can directly query the database bypassing framework access control and make direct changes to the database. This is a security risk. Best security practice sets this property to `false`.

Property	Value	Description
Rdb Security Settings	additional properties	The "rdb-RdbSecuritySettings" topic documents these properties.
Tls Min Protocol	drop-down menu	Selects the minimum TLS protocol that can be negotiated when establishing encrypted communications with a database server.
Verify Subject In Certificate	`true` (default) or `false`	If set to `true`, the Rdbms driver verifies that the **`Subject`** present in the server's certificate matches an expected value. The value used for the comparison to the **`Subject`** in the server's certificate is specific to each Rdbms device type.
Database Name	text	Provides the name of the database.
Port	number (defaults to 3306)	Specifies the port number to use when connecting with the database. Defaults are: HsqlDbDatabase, no port specified because this rdb is for local database use only: MySQLDatabase: 3306 OracleDatabase: 1521 SqlServerDatabase: 1433
Histories	additional properties	"rdbMySQL-MySQLHistoryDeviceExt" documents these properties.
Extra Connection Properties	normally left blank; if used takes the format: parameter1=value1;parameter2=value2;...etc.	Configures additional properties that are not required for a normal connection to the database. The driver passes this information in plain text to the database. You can set certain configuration properties as part of the **`extra connection properties`** of the RDBMS database to optimize the batch processing of the insert queries, specifically the history export queries. To enable the JDBC driver of a MySQL Server to pack as many queries as possible into a single network packet, lowering network overhead, set **`rewriteBatchedStatements`** set to `true`.
My Sql Server Cert	drop-down list	Defines the database server certificate for TLS secure communication. If the field is empty, no Subject Identity is checked. It is implied that the certificate presented by the MySQL server is already trusted. Otherwise, the Subject's Common Name (CN) from the certificate alias configured in the **`My Sql Server Cert`** property is used.

Actions

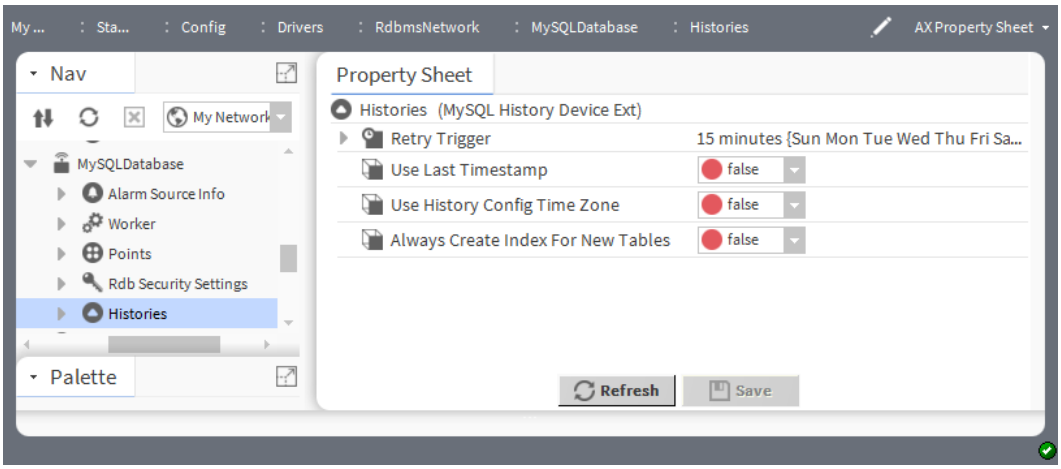
This component supports the following actions:

- **Ping** sends a message to a network object (device, database, etc). The message provokes a response, which indicates the current state of the object.
- **Allow Dialect Modifications** makes the **`Timestamp Storage`** property writable within the **RdbmsNetwork Update Wizard**. When the wizard converts a database to UTC (Coordinated Universal Time), it sets this property to read-only.

rdbMySQL-MySQLHistoryDeviceExt

This component is the MySQL implementation of HistoryDeviceExt. It is a child of a **MySQLDatabase**.

Figure 24 MySQL History properties



You access these properties by expanding **Config→Drivers→RdbmsNetwork→MySQLDatabase** followed by right-clicking **Histories** and clicking **Views→AX Property Sheet**

Property	Value	Description
Retry Trigger	additional properties	Defines how frequently to attempt a failed operation again. This continues until successful execution occurs. Appears in the Nav tree but not in any manager view and is unique in that it requires no linking of its output for operation. *Getting Started with Niagara* documents **`Retry Trigger`** properties.
Use Last Timestamp	`true` or `false` (default)	Enables and disables the reporting of the last date and time on a history record. `false` queries the database for the last timestamp in the database every time an export descriptor processes data. Depending on the size of the database table and the number of export descriptors being processed, this can be a time consuming activity. `true` stores the last timestamp as a property on the export descriptor each time it processes the export. This efficiency-related setting helps with overall system performance when exporting data.

Property	Value	Description
Use History Config Time Zone	`true` or `false` (default)	Configures how to normalize the timestamp before writing it to the external database. It includes the time zone. `false` normalizes values to the Supervisor station's time zone. `true` writes the original timestamp value (retained by the Supervisor) to the external database. Unfortunately, this option ignores the time zone. This causes difficulties with local time zones, especially in relation to daylight savings time. The **`Timestamp Storage`** property exports or updates history timestamps to Coordinated Universal Time (UTC). **NOTE:** The effect of these History properties may be irrelevant depending upon the setting of the**`Timestamp Storage`**
Always Create Index For New Tables	`true` or `false` (default)	Configures histories regarding index creation for new tables. It specifies whether indexes should always be created for new exported history tables. `true` ensures that an index on created RDBMS tables for exported histories is always created, no matter what the configuration of other properties is. `false` may not create a new index when exporting new tables. The driver may still create indexes based on the configuration of other properties (for example, if **`Use Last Timestamp`** is `false`, indexes get created no matter what). If this property is `false` and the **`Use Last Timestamp`** property is set to `true`, an export may not create indexes for the exported tables. Setting this property to `true` in that scenario would cause any new tables created going forward to also have a proper index created. You can subsequently use the **Migrate To Optimized Table Indexes** action against existing exported tables (also those prior to Niagara 4.11) to migrate them and ensure that updated indexes are created in the RDBMS. Creating indexes for exported tables can help maximize query performance.

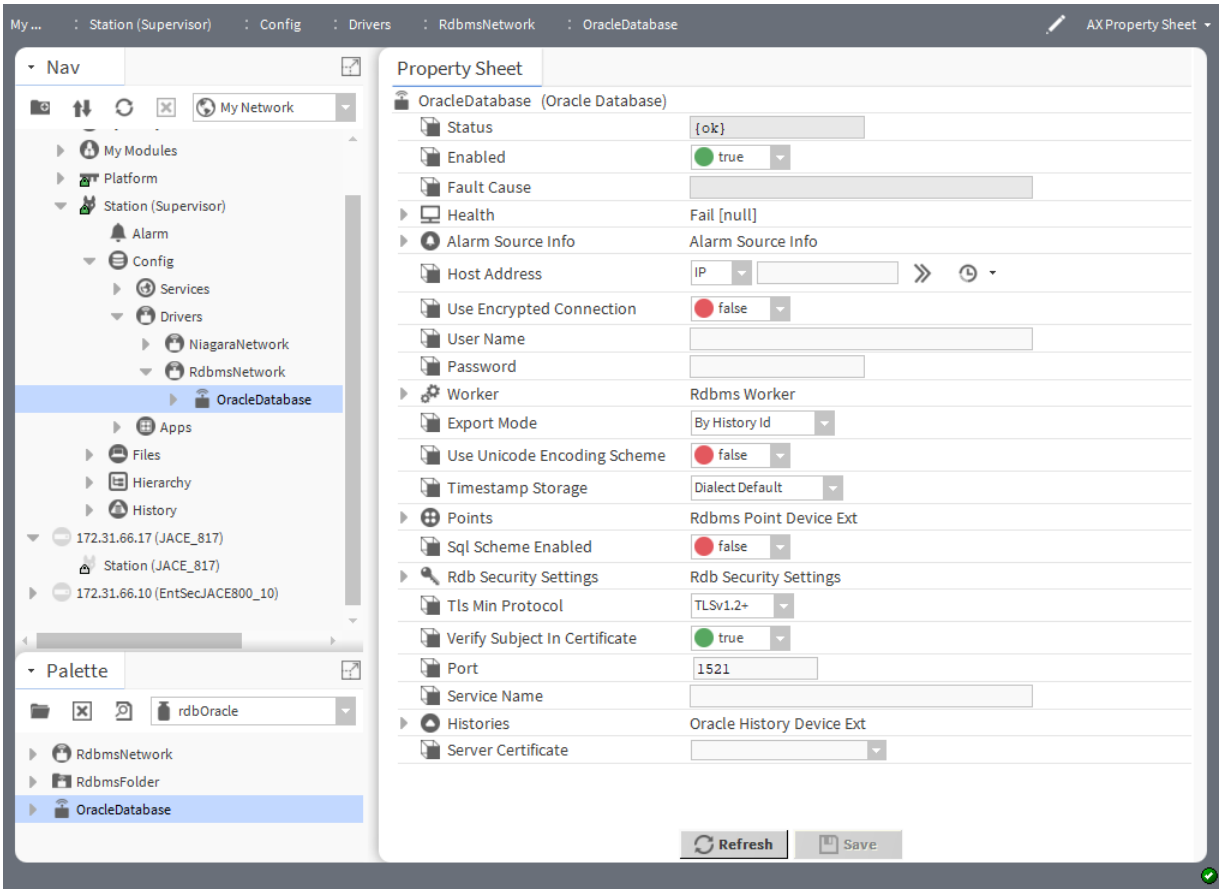
Actions

- **Retry** downloads histories again.
- **Migrate To Optimized Table Indexes** (introduced in Niagara 4.11) can be used against existing RDBMS installations (prior to Niagara 4.11) to migrate the existing RDBMS tables and ensure that updated indexes are created in the RDBMS. Coordinate with your DBA before you execute this action. The updated indexes maximize query performance, but this action is not mandatory. Since this optional action affects the tables in the existing RDBMS, this migration does not happen automatically but requires you to manually invoke the action. Invoking this action kicks off a job in the **JobService** where you can monitor the migration progress and results.

rdbOracle-OracleDatabase

This component models an Oracle relational database management system (RDBMS) that requires a proprietary license. It is available in the **`rdbOracle`** palette.

Figure 25 Oracle Database properties



You access the properties to configure the driver for this database by expanding **Drivers→RdbmsNetwork**, and double-clicking the **OracleDatabase** node in the Nav tree.

Property	Value	Description
Host Address (database device components)	IP address	Sets the IP address or hostname of the computer platform where the database resides. A Dialup selection option is available, if required. This property does not apply to the MySQL device.
Use Encrypted Connection	`true` or `false` (default)	Indicates if the connection between the station and the database is secure (`true`) or not secure (`false`). To ensure that your system cannot be hacked, leave this property set to `true`. Change it only if your database does not support data encryption.
User Name (database device components)	text	Defines the user name used to log in to the database. Login credentials must provide sufficient database privileges to allow you to perform one or more (depending on database type) of these commands: CREATE TABLE, CREATE INDEX, CREATE SEQUENCE
Password (database device components)	text	Defines the password required to log in to the database. The **`Confirm`** property must be an exact match to the **`Password`** property.

Property	Value	Description
Worker	additional properties	The "rdb-RdbmsWorker" topic documents these properties.
Export Mode (database device components)	drop-down list	Specifies how histories are exported to the specified database. `By History Id` exports one table per History Id. This is the default value setting. `By History Type` exports one table per History Type. This option may make the data easier to query once exported.
Use Unicode Encoding Scheme (database device components)	`true` or `false` (default)	Creates history table schemas with the Universal character set Transformation Format (UTF-8) or Unicode data types for string-valued columns to store Asian character sets. `false` maintains backward compatibility with legacy history export mechanisms. `true` enables the NVARCHAR data type for any column in a table that expects string data. **NOTE:** An Update Wizard is available to upgrade existing databases to support Unicode.
Timestamp Storage	enum property drop-down list (defaults to `Dialect Default`)	Exports or updates history timestamps to Coordinated Universal Time (UTC) enabling the export of history records from different timezones into a common database and be chronologically correct and independent of any specific source timezone characteristics. In other words, exported histories show the timestamp data from where the history is actually stored, making useable histories with a consistent timestamp. `Dialect Default` maintains compatibility with legacy history export mechanisms. This property does not apply to the MySQL device. `Local Time Stamp` applies to Orion databases only and does not apply to history exports. `Utc Timestamp` exports all subsequent histories with UTC timestamps. `Utc Millis` applies to Orion databases only and does not apply to history exports. **NOTE:** An Update Wizard is available to upgrade existing databases to support Unicode.
Points	folder	Holds all points.
Sql Scheme Enabled (database device components)	`true` or `false` (default)	Permits (`true`) and prohibits (`false`) the use of ORDs that contain `sql:`. If a BFormat contains `sql:`, users can directly query the database bypassing framework access control and make direct changes to the database. This is a security risk. Best security practice sets this property to `false`.
Rdb Security Settings	additional properties	The "rdb-RdbSecuritySettings" topic documents these properties.

Property	Value	Description
Tls Min Protocol	drop-down menu	Selects the minimum TLS protocol that can be negotiated when establishing encrypted communications with a database server.
Verify Subject In Certificate	`true` (default) or `false`	If set to `true`, the Rdbms driver verifies that the **`Subject`** present in the server's certificate matches an expected value. The value used for the comparison to the **`Subject`** in the server's certificate is specific to each Rdbms device type.
Port	number (the default value depends on the rdb Database type)	Specifies the port number to use when connecting with the database. Defaults are: HsqlDbDatabase, no port specified because this rdb is for local database use only: MySQLDatabase: 3306 OracleDatabase: 1521 SqlServerDatabase: 1433
Service Name	text	Refers to the Oracle SID or System Identifier, which uniquely identifies a database instance. It allows you to configure a connection to an Oracle database if the user has not been previously set up in the Oracle Database with a default SID, or to connect to an SID that is not the default. If a default SID has been set up and you need to connect to it, you can leave this property blank.
Histories	additional properties	"rdbMySQL-MySQLHistoryDeviceExt" documents these properties.
My Sql Server Cert	drop-down list	Defines the database server certificate for TLS secure communication. If the field is empty, a Common Name (CN) is formed based on the host name from the **`Host Address`** property. Otherwise, the Distinguished Name (DN) from the certificate alias configured in the **`Server Certificate`** property is used.

Actions

This component supports the following actions:

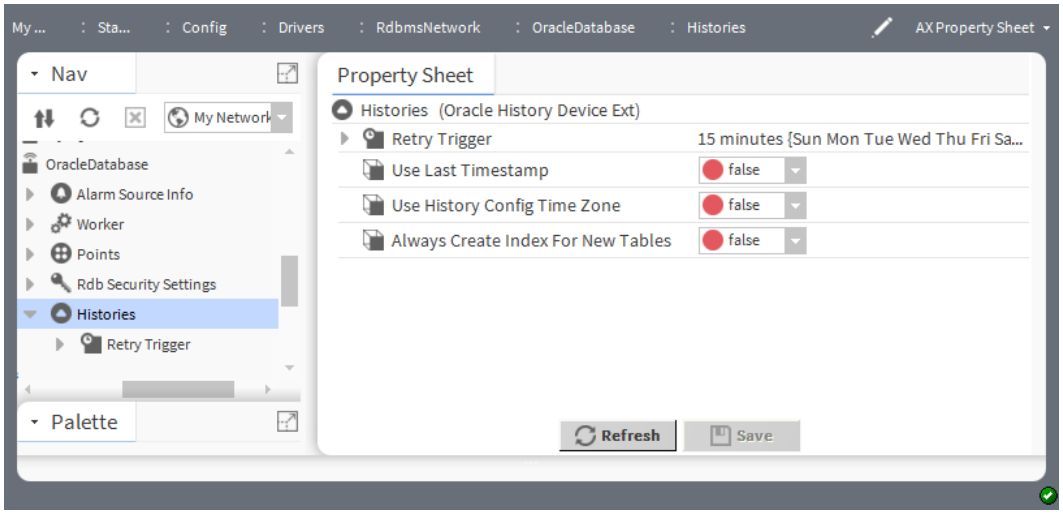
- **Ping** sends a message to a network object (device, database, etc). The message provokes a response, which indicates the current state of the object.
- **Allow Dialect Modifications** makes the **`Timestamp Storage`** property writable within the **RdbmsNetwork Update Wizard**. When the wizard converts a database to UTC (Coordinated Universal Time), it sets this property to read-only.

rdbOracle-OracleHistoryDeviceExt

This component is the Oracle implementation of HistoryDeviceExt, and a child of a the OracleDatabase component.

The History Device Extension is fully described in the *Niagara Drivers Guide*.

Figure 26 Oracle history device extension properties



You access these properties by expanding **Config→Drivers→RdbmsNetwork→OracleDatabase** followed by right-clicking **Histories** and clicking **Views→AX Property Sheet**

Property	Value	Description
Retry Trigger	additional properties	Defines how frequently to attempt a failed operation again. This continues until successful execution occurs. Appears in the Nav tree but not in any manager view and is unique in that it requires no linking of its output for operation. *Getting Started with Niagara* documents **`Retry Trigger`** properties.
Use Last Timestamp	`true` or `false` (default)	Enables and disables the reporting of the last date and time on a history record. `false` queries the database for the last timestamp in the database every time an export descriptor processes data. Depending on the size of the database table and the number of export descriptors being processed, this can be a time consuming activity. `true` stores the last timestamp as a property on the export descriptor each time it processes the export. This efficiency-related setting helps with overall system performance when exporting data.

Property	Value	Description
Use History Config Time Zone	`true` or `false` (default)	Configures how to normalize the timestamp before writing it to the external database. It includes the time zone. `false` normalizes values to the Supervisor station's time zone. `true` writes the original timestamp value (retained by the Supervisor) to the external database. Unfortunately, this option ignores the time zone. This causes difficulties with local time zones, especially in relation to daylight savings time. The **`Timestamp Storage`** property exports or updates history timestamps to Coordinated Universal Time (UTC). **NOTE:** The effect of these History properties may be irrelevant depending upon the setting of the**`Timestamp Storage`**
Always Create Index For New Tables	`true` or `false` (default)	Configures histories regarding index creation for new tables. It specifies whether indexes should always be created for new exported history tables. `true` ensures that an index on created RDBMS tables for exported histories is always created, no matter what the configuration of other properties is. `false` may not create a new index when exporting new tables. The driver may still create indexes based on the configuration of other properties (for example, if **`Use Last Timestamp`** is `false`, indexes get created no matter what). If this property is `false` and the **`Use Last Timestamp`** property is set to `true`, an export may not create indexes for the exported tables. Setting this property to `true` in that scenario would cause any new tables created going forward to also have a proper index created. You can subsequently use the **Migrate To Optimized Table Indexes** action against existing exported tables (also those prior to Niagara 4.11) to migrate them and ensure that updated indexes are created in the RDBMS. Creating indexes for exported tables can help maximize query performance.

Actions

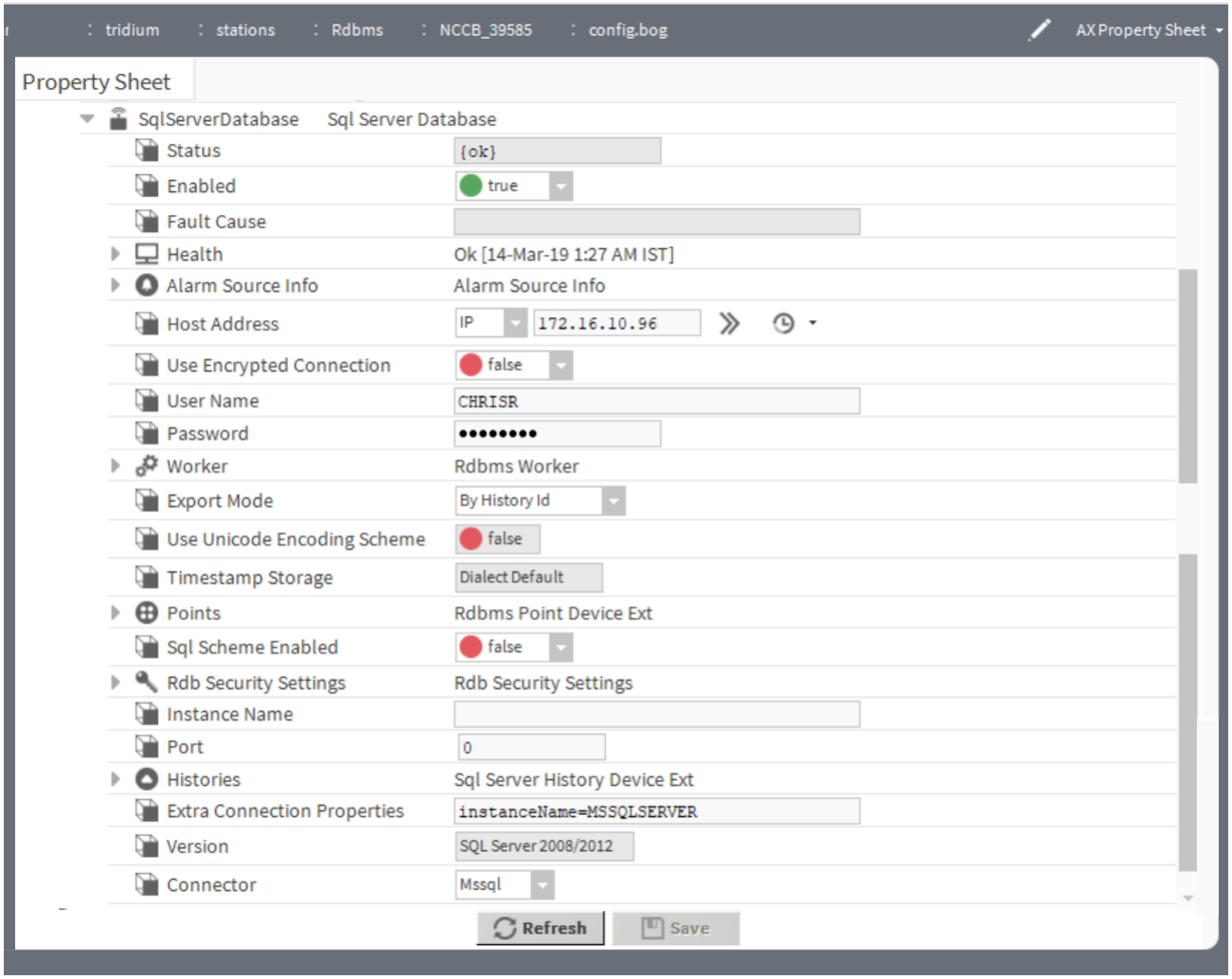
- **Retry** downloads histories again.
- **Migrate To Optimized Table Indexes** (introduced in Niagara 4.11) can be used against existing RDBMS installations (prior to Niagara 4.11) to migrate the existing RDBMS tables and ensure that updated indexes are created in the RDBMS. Coordinate with your DBA before you execute this action. The updated indexes maximize query performance, but this action is not mandatory. Since this optional action affects the tables in the existing RDBMS, this migration does not happen automatically but requires you to manually invoke the action. Invoking this action kicks off a job in the **JobService** where you can monitor the migration progress and results.

  This action is not a requirement and therefore optional. Also check with your DBA before performing this action since it affects the tables in the existing RDBMS.

rdbSqlServer-SqlServerDatabase

This component models an Microsoft SQL Server database versions: SqlServer 2000, SqlServer 2005, SqlServer 2008, SqlServer 2008 R2 and SqlServer 2012. It is available in the **`rdbSqlServer`** palette.

Figure 27 SqlServerDatabase properties



You access the properties to configure the driver for this database by expanding **Config→Drivers→Rdbms-Network**, and double-clicking the **SqlServerDatabase** node in the Nav tree.

The Rdbms driver does not support Windows Authentication alone. You must have selected SQL Server Authentication (mixed mode) for any user of this database. This is an SQL Server login property, not a Niagara property.

In addition to the standard properties (Status, Enabled, Fault Cause, Health and Alarm Source Info), these properties support this component.

Property	Value	Description
Host Address (database device components)	IP address	Sets the IP address or hostname of the computer platform where the database resides. A Dialup selection option is available, if required. This property does not apply to the MySQL device.
Use Encrypted Connection	`true` or `false` (default)	Indicates if the connection between the station and the database is secure (`true`) or not secure (`false`). To ensure that your system cannot be hacked, leave this property set to `true`. Change it only if your database does not support data encryption.

Property	Value	Description
User Name (database device components)	text	Defines the user name used to log in to the database. Login credentials must provide sufficient database privileges to allow you to perform one or more (depending on database type) of these commands: CREATE TABLE, CREATE INDEX, CREATE SEQUENCE
Password (database device components)	text	Defines the password required to log in to the database. The **`Confirm`** property must be an exact match to the **`Password`** property.
Worker	additional properties	The "rdb-RdbmsWorker" topic documents these properties.
Export Mode (database device components)	drop-down list	Specifies how histories are exported to the specified database. `By History Id` exports one table per History Id. This is the default value setting. `By History Type` exports one table per History Type. This option may make the data easier to query once exported.
Use Unicode Encoding Scheme (database device components)	`true` or `false` (default)	Creates history table schemas with the Universal character set Transformation Format (UTF-8) or Unicode data types for string-valued columns to store Asian character sets. `false` maintains backward compatibility with legacy history export mechanisms. `true` enables the NVARCHAR data type for any column in a table that expects string data. **NOTE:** An Update Wizard is available to upgrade existing databases to support Unicode.
Timestamp Storage	enum property drop-down list (defaults to `Dialect Default`)	Exports or updates history timestamps to Coordinated Universal Time (UTC) enabling the export of history records from different timezones into a common database and be chronologically correct and independent of any specific source timezone characteristics. In other words, exported histories show the timestamp data from where the history is actually stored, making useable histories with a consistent timestamp. `Dialect Default` maintains compatibility with legacy history export mechanisms. This property does not apply to the MySQL device. `Local Time Stamp` applies to Orion databases only and does not apply to history exports. `Utc Timestamp` exports all subsequent histories with UTC timestamps. `Utc Millis` applies to Orion databases only and does not apply to history exports. **NOTE:** An Update Wizard is available to upgrade existing databases to support Unicode.
Points	folder	Holds all points.

Property	Value	Description
Sql Scheme Enabled (database device components)	`true` or `false` (default)	Permits (`true`) and prohibits (`false`) the use of ORDs that contain `sql:`. If a BFormat contains `sql:`, users can directly query the database bypassing framework access control and make direct changes to the database. This is a security risk. Best security practice sets this property to `false`.
Rdb Security Settings	additional properties	The "rdb-RdbSecuritySettings" topic documents these properties.
Tls Min Protocol	drop-down menu	Selects the minimum TLS protocol that can be negotiated when establishing encrypted communications with a database server.
Verify Subject In Certificate	`true` (default) or `false`	If set to `true`, the Rdbms driver verifies that the **`Subject`** present in the server's certificate matches an expected value. The value used for the comparison to the **`Subject`** in the server's certificate is specific to each Rdbms device type.
Instance Name	text	Configures a connection to a SqlServer database if the user has not been previously set up in the SqlServer with a default **`Database Name`** or **`Instance Name`**, or to connect to a name that is not the default. If a default name has been set up and you need to connect to it, you can leave this property blank.
Port	number (the default value depends on the rdb Database type)	Specifies the port number to use when connecting with the database. Defaults are: HsqlDbDatabase, no port specified because this rdb is for local database use only: MySQLDatabase: 3306 OracleDatabase: 1521 SqlServerDatabase: 1433
Histories	additional properties	"rdbMySQL-MySQLHistoryDeviceExt" documents these properties.
Extra Connection Properties (MySQL and SqlServer devices only)	normally left blank; if used takes the format: parameter1=value1;parameter2=value2;...etc.	Configures additional properties that are not required for a normal connection to the database. The driver passes this information in plain text to the database. You can set certain configuration properties as part of the **`extra connection properties`** of the RDBMS database to optimize the batch processing of the insert queries, specifically the history export queries. To enable the JDBC driver of a MySQL Server to pack as many queries as possible into a single network packet, lowering network overhead, set **`rewriteBatchedStatements`** set to `true`.

Property	Value	Description
Version	drop-down list, defaults to `Sql Server 2008`	Determines if the Sql Server database supports the SQL DATE type. The DATE Sql type is supported from Sql server 2008 onwards and the database device needs to know how to translate timestamps when retrieving records. Set this property to match the version of the database you are connecting to. `Sql Server 2008` specifies that the Version of Sql Server you are connecting to is Sql Server 2008. **NOTE:** Use this option to connect to an Sql version that is more recent than Sql Server 2008, such as Sql Server 2012. `Sql Server 2005` specifies that the Version of Sql Server you are connecting to is Sql Server 2005. `Sql Server 2000` specifies that the Version of Sql Server you are connecting to is Sql Server 2000.
My Sql Server Cert	drop-down list	Defines the database server certificate for TLS secure communication.

Actions

This component supports the following actions:

- **Ping** sends a message to a network object (device, database, etc). The message provokes a response, which indicates the current state of the object.
- **Allow Dialect Modifications** makes the **`Timestamp Storage`** property writable within the **RdbmsNetwork Update Wizard**. When the wizard converts a database to UTC (Coordinated Universal Time), it sets this property to read-only.

Dynamic ports in the RdbSqlServer

`SqlServerDatabase` supports connecting to an SQL Server Named Instance using dynamic port discovery. To configure the use of dynamic ports, the port property must be set to `0`, and the **`instance Name`** parameter value corresponding to the desired Named Instance must be specified in the **Extra Connection Properties**.

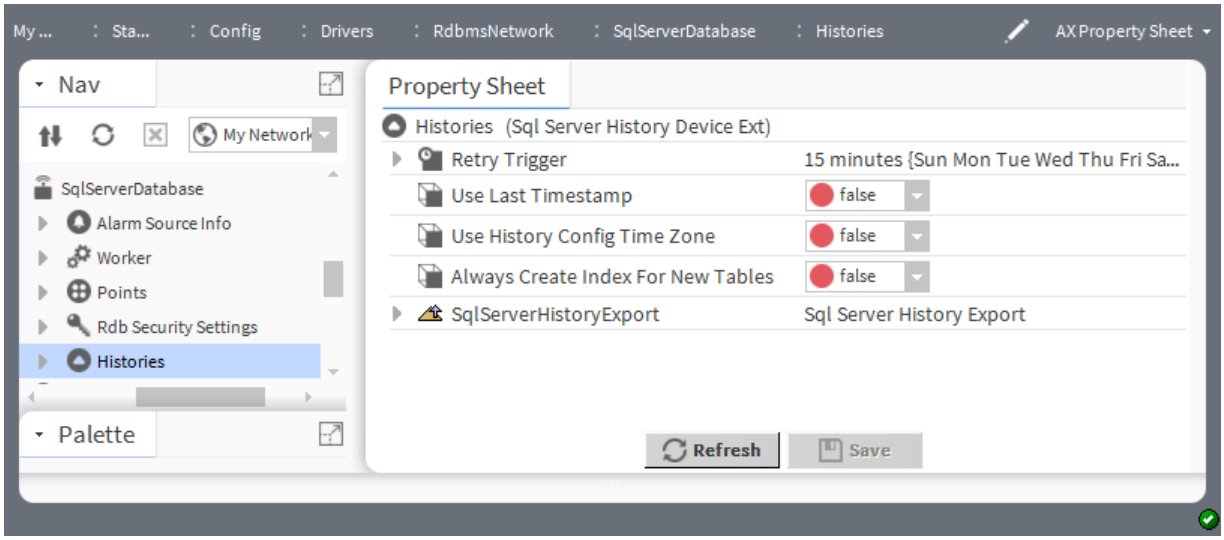
For example, to connect to an SQL Server Named Instance, named as `MSSQLSERVER` that is configured to use dynamic ports, set the **`Port`** property to `0` and add `instanceName=MSSQLSERVER` to the **`ExtraConnectionProperties`** property.

Refer to the "Dynamic ports in the RdbSqlServer" for more details.

rdbSqlServer-SqlServerHistoryDeviceExt

This component is the SqlServer implementation of HistoryDeviceExt. It is a child of an **SqlServerDatabase**.

Figure 28 SqlServer History properties



You access Histories properties by expanding **Config→Drivers→RdbmsNetwork→SqlServerDatabase** followed by right-clicking **Histories** and clicking **Views→AX Property Sheet**

This container itself contains the **`Retry Trigger`** container.

Property	Value	Description
Retry Trigger	additional properties	Defines how frequently to attempt a failed operation again. This continues until successful execution occurs. Appears in the Nav tree but not in any manager view and is unique in that it requires no linking of its output for operation. *Getting Started with Niagara* documents **`Retry Trigger`** properties.
Use Last Timestamp (OracleDatabase device only)	`true` or `false` (default)	Enables and disables the reporting of the last date and time on a history record. `false` queries the database for the last timestamp in the database every time an export descriptor processes data. Depending on the size of the database table and the number of export descriptors being processed, this can be a time consuming activity. `true` stores the last timestamp as a property on the export descriptor each time it processes the export. This efficiency-related setting helps with overall system performance when exporting data.

Property	Value	Description
Use History Config Time Zone (Oracle-Database device only)	`true` or `false` (default)	Configures how to normalize the timestamp before writing it to the external database. It includes the time zone. `false` normalizes values to the Supervisor station's time zone. `true` writes the original timestamp value (retained by the Supervisor) to the external database. Unfortunately, this option ignores the time zone. This causes difficulties with local time zones, especially in relation to daylight savings time. The **`Timestamp Storage`** property exports or updates history timestamps to Coordinated Universal Time (UTC). **NOTE:** The effect of these History properties may be irrelevant depending upon the setting of the**`Timestamp Storage`**
Always Create Index For New Tables	`true` or `false` (default)	Configures histories regarding index creation for new tables. It specifies whether indexes should always be created for new exported history tables. `true` ensures that an index on created RDBMS tables for exported histories is always created, no matter what the configuration of other properties is. `false` may not create a new index when exporting new tables. The driver may still create indexes based on the configuration of other properties (for example, if **`Use Last Timestamp`** is `false`, indexes get created no matter what). If this property is `false` and the **`Use Last Timestamp`** property is set to `true`, an export may not create indexes for the exported tables. Setting this property to `true` in that scenario would cause any new tables created going forward to also have a proper index created. You can subsequently use the **Migrate To Optimized Table Indexes** action against existing exported tables (also those prior to Niagara 4.11) to migrate them and ensure that updated indexes are created in the RDBMS. Creating indexes for exported tables can help maximize query performance.

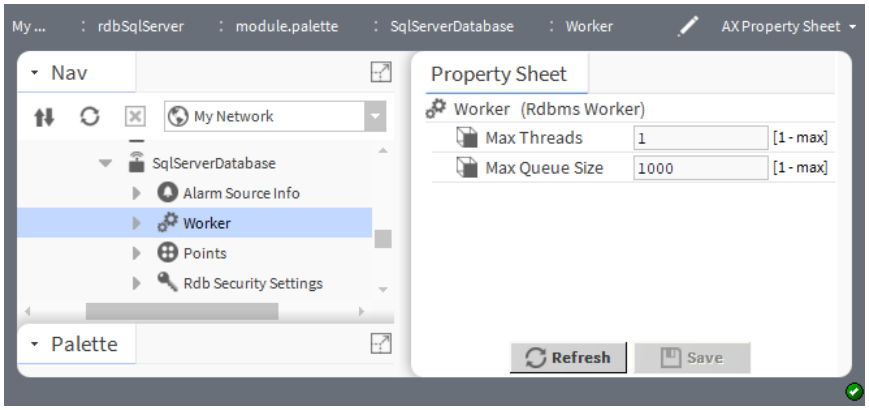
Actions

- **Retry** downloads histories again.
- **Migrate To Optimized Table Indexes** (introduced in Niagara 4.11) can be used against existing RDBMS installations (prior to Niagara 4.11) to migrate the existing RDBMS tables and ensure that updated indexes are created in the RDBMS. Coordinate with your DBA before you execute this action. The updated indexes maximize query performance, but this action is not mandatory. Since this optional action affects the tables in the existing RDBMS, this migration does not happen automatically but requires you to manually invoke the action. Invoking this action kicks off a job in the **JobService** where you can monitor the migration progress and results.

rdb-RdbmsWorker

This child component of all rdb Database devices manages the queue and worker for asynchronous operations on a single database.

Figure 29 Worker properties



This component is available under any of the rdb (relational database) types: **HsqlDbDatabase**, **MySQLDatabase**, **OracleDatabase**, and **SqlServerDatabase**.

This component provides two properties for setting the maximum number of concurrent threads and for setting the maximum allowable queue size for the database connection. These worker properties are not pooled at the Network level, as with the **NiagaraNetwork** driver. Each database driver has its own thread pool setting.

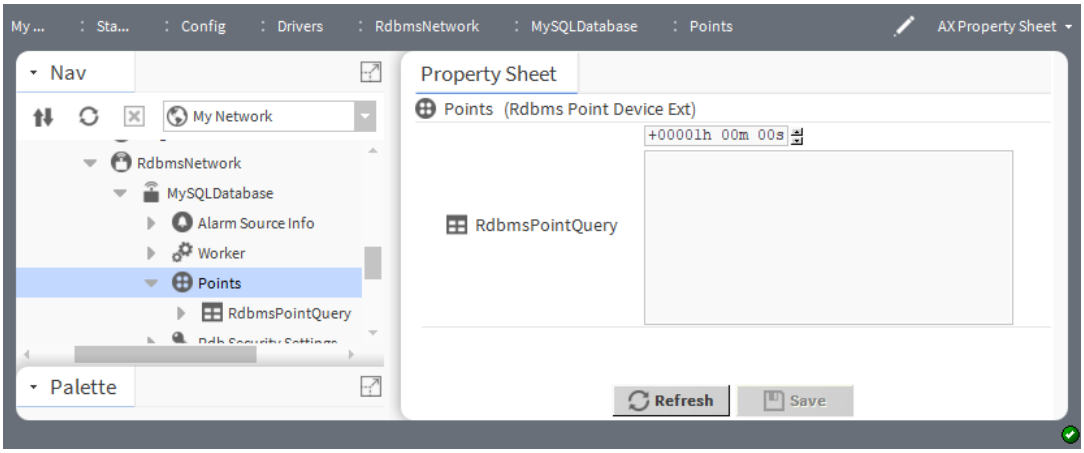
Property	Value	Description
Max Threads	number (defaults to one thread)	Tunes large networks (those with many station components) to process more than a single thread at a time. It is the only visible part of a shared thread-pool scheme for scaling large-jobs and allows the local station's thread pool to grow uncapped. Each thread uses one JDBC Connection to communicate with the database, so there are as many connections created as there are threads. Normally you increase this value to between 20 and 50 to improve performance when handling large volumes of data.
Max Queue Size	number (defaults to 1000)	Specifies the maximum number of items that can be queued for worker processing. In a few cases, particularly in a large system, increasing this size may be beneficial. Queue size does not allocate a fixed memory size. Rather, the amount of memory used is dynamic, changing as message items are added and removed from the queue.

rdb-RdbmsPointDeviceExt

This component provides point import capability for the various relational database drivers using methods and views that are similar to other proxy point driver views. It is available under most of the rdb (relational database) types, including **OracleDatabase**, **SqlServerDatabase**, and **MySQLDatabase**.

The **Points** folder (Rdbms Point Device Extension) is unlike point device extensions associated with other driver types. This point device extension uses the Rdbms Point Query component to filter database records to provide candidate records for proxy points.

Figure 30 Rdbms Point Query properties



To access these properties, expand **Config→Drivers**, expand the rdb database and double-click **Points**.

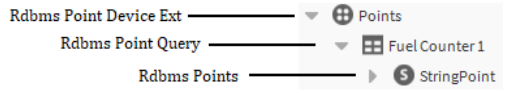
General information

With this component you can represent relational database cell values as proxy points. The **RdbmsNetwork** driver requires no configuration.

Rdb database devices resemble typical field-bus functionality in that Workbench represents them as devices. You can view all devices that are under the **RdbmsNetwork** in a **Device Manager** view, or in the Nav tree, with each device that is installed under the network representing an individual database type and connection.

The Rdbms Point Device Extension provides point import capability for the various relational database drivers using methods and views that are similar to other proxy point driver views. Using this point device extension to import points, you can represent relational database cell values as proxy points, as shown below.

Figure 31 Rdbms points



The Rdbms Point Device Extension (RdbmsPointDeviceExt) may contain one or more Rdbms Point Query properties (depending on how many you add). Individual points are then added under each individual **`Rdbms Point Query`** property using the discover and add process available in the **Rdbms Point Query Manager** view. Rdbms Points are always organized under their parent **Rdbms Point Query** in the Nav tree and are also displayed in the **Database** pane of the **Rdbms Point Query Manager** view.

Point container similarities (compared to other drivers)

Like other Point Device Extensions, the Rdbms Point Device Extension:

- is a required (frozen) slot on the Rdbms Point Device component - it's always there, you cannot delete it.
- displays as a typical Points node under its appropriate RdbmsNetwork driver.
- is a container component, having several special views.
- is a parent of proxy points.
- has a default manager view (the Rdbms Point Device Ext Manager view) and uses Discover, Add, and New buttons to add proxy points.

Point container differences (compared to other drivers)

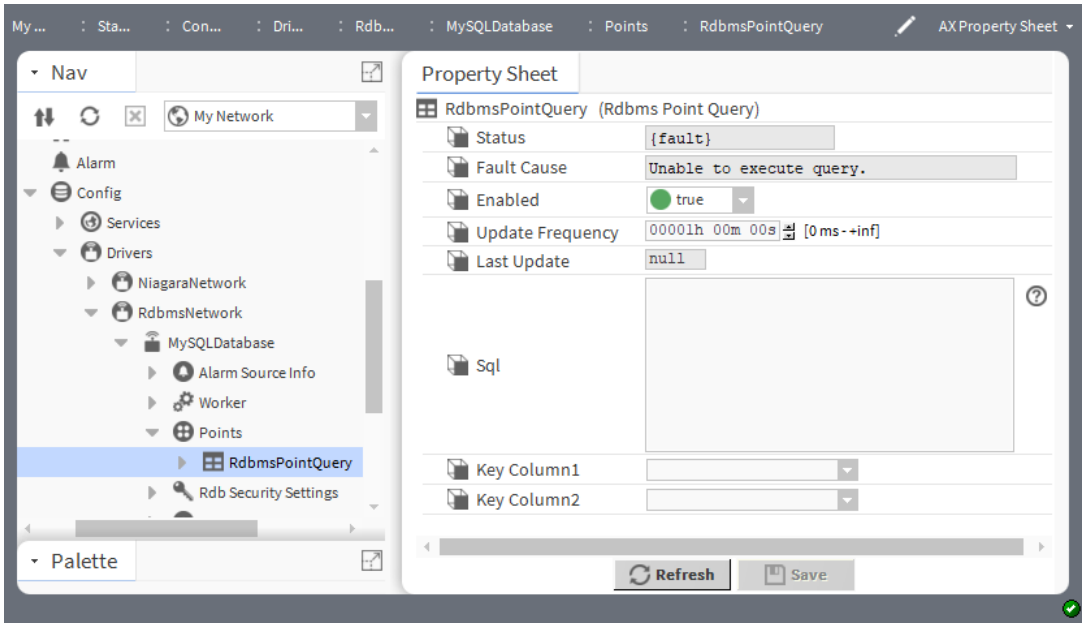
Unlike other Point Device Extensions, the Rdbms Point Device Extension:

- has a unique default view - the **Rdbms Point Device Extension Manager** view.
- has a unique child component - the Rdbms Point Query component.
- has proxy points under the Rdbms Point Device Extension that are read-only; you cannot write to the Rdbms using these points.
- has proxy points under the Rdbms Point Device Extension that use recorded database values not live values. It is possible to have points from the database update very frequently, depending on database archiving and updating parameters, but the data are always coming from a secondary source - the database, not a control point.

rdb-RdbmsPointQuery

This component is a container and a child of the Rdbms Point Device Extension (**Points** folder). You use it to construct and execute queries against any database to which you have access and sufficient privileges.

Figure 32 RdbmsPointQuery properties



To add an Rdbms Point Query to the Points folder under your rdb database, expand **Config→Drivers→RdbmsNetwork**, expand a database node, double-click the **Points** folder, click **New**, create the query and click **OK**.

To access a query, expand the **Points** folder, right-click the **RdbmsPointQuery** node, and click **Views→AX Property Sheet**.

The **`Sql`** property is a text editor used to create and display the query statement. Named columns are optional. Unnamed columns are displayed as "column1, column2, ..." and so on. Key columns are optional. If no key column is specified, the driver uses the first column as the primary key.

You can execute this query manually and also set a regular interval time for updates using the **`Update Frequency`** property.

In a addition to the standard properties (Status, Fault Cause and Enabled), these properties support point queries.

Property	Value	Description
Update Frequency	hours, minutes, seconds	Dictates how often the driver automatically executes the query.
Last Update	date and time	Indicates the last time the driver executed the query.
Sql	text editor	Provides an editor (one of several available) for viewing and editing the query statement. The query provides the criteria for database point discoveries initiated from the **Rdbms Point Query Manager** view and other views. **NOTE:** This property is actually BFormat. This means that you can add additional syntax to the SQL string that processes before the driver sends the query to the database.
Key Column1		
Key Column2		

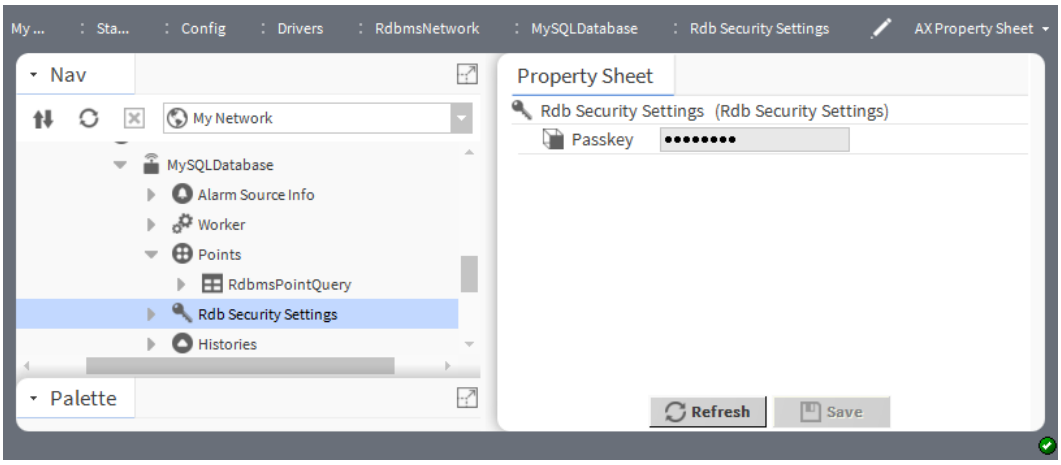
Action

- **Execute** runs the application.

rdb-RdbSecuritySettings

This property configures security settings for the relational database.

Figure 33 Rdb Security Settings properties



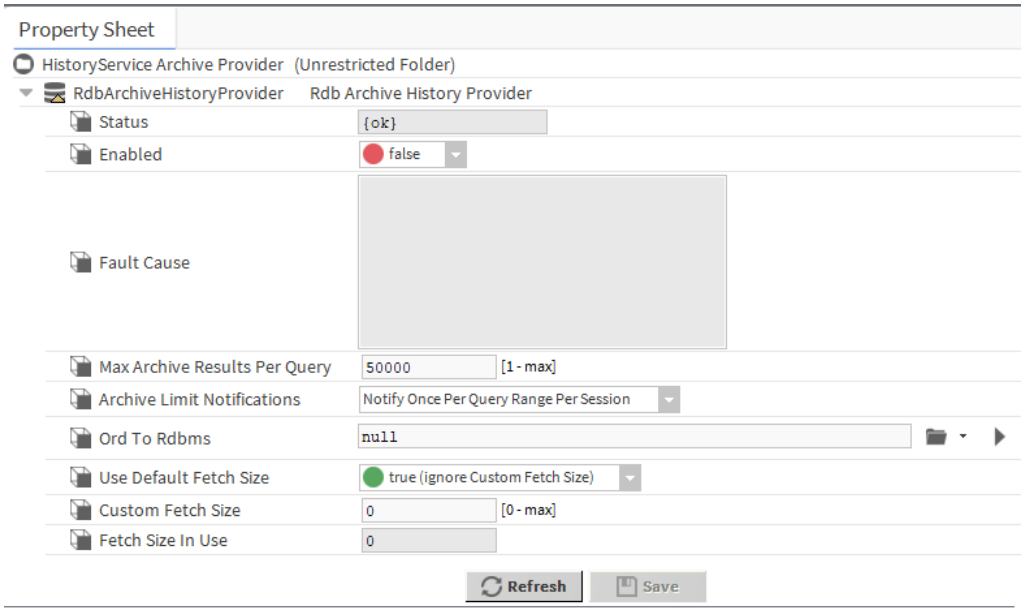
To access this property, expand **Config→Drivers→RdbmsNetwork**, expand a MySql database and double-click **Rdb Security Settings**.

Property	Value	Description
Passkey	text	Configures a secure key, which Niagara Enterprise Security uses to encrypt a PIN.

rdb-HistoryServiceArchiveProvider

This component supplements queries against local history records with archived history records that were previously exported to a remote (cloud) RDBMS using the standard drivers: rdbSqlServer, rdbMySQL, and rdbOracle. These archived history records are from an external data store, a relational database.

Figure 34 HistoryServiceArchiveProvider properties



You access these properties by expanding **Config→Services→Archive History Providers** and double-clicking the **RdbArchiveHistoryProvider** node in the Nav tree.

In addition to the standard properties (Status, Enabled, and Fault Cause), these properties are unique to this component.

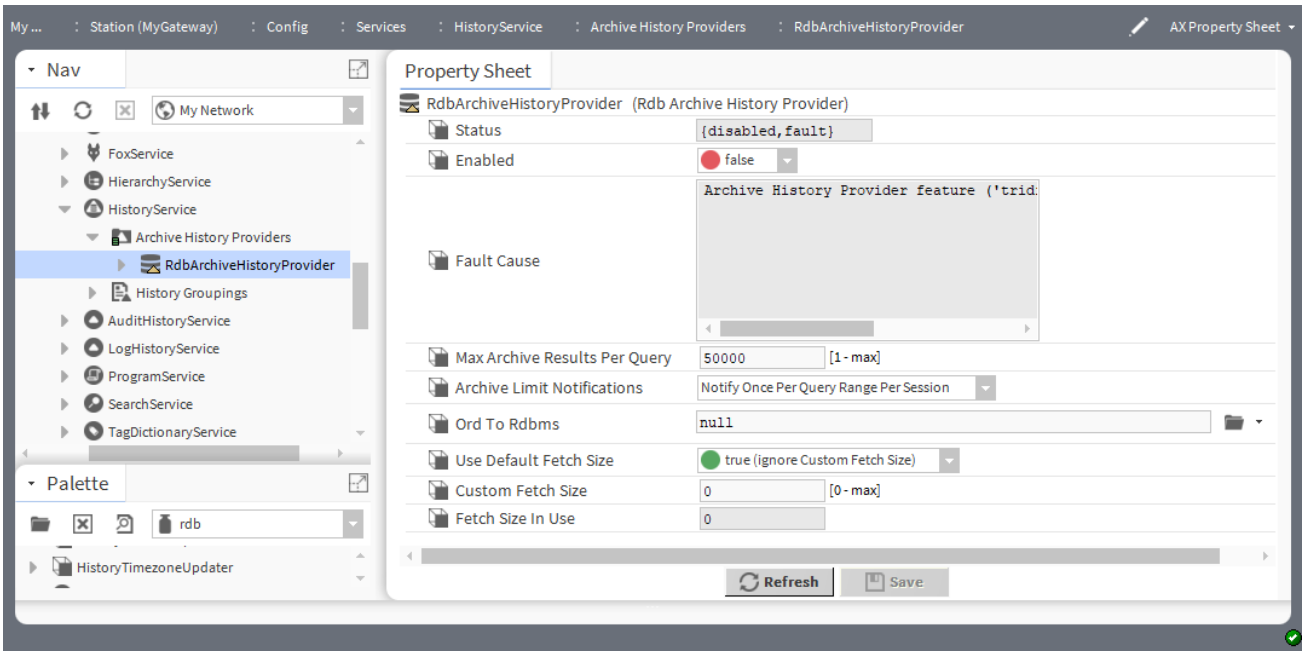
Property	Value	Description
Max Archive Results Per Query	number	Specifies the number of archived history records to allow per query.
Archive Limit Notifications	drop-down list (defaults to `Notify Once Per Query Range Per Session`)	Selects the message to display when the limit is exceeded for any history query initiated from Workbench. `Notify Once Per Query Range Per Session` limits the frequency of warnings to once per query range per session `Never Notify` turns notification off. `Always Notify` displays a warning every time the query exceeds the maximum archive results.
Ord To Rdbms	ORD	Defines the path to the archive database.
Use Default Fetch Size	`true (ignore Custom Fetch Size)` (default) `false (use Custom Fetch Size)`	Does not use (`true`) or uses (`false`) the **`Custom Fetch Size`** to retrieve archive history records.
Custom Fetch Size	number	Specifies the number of archive history records to retrieve from the database in the cloud.
Fetch Size in Use	read-only	Reports the number of records retrieved when the **`Custom Fetch Size`** is being used.

rdb-RdbArchiveHistoryProvider

When installed and configured to reference an existing RDBMS device that has already exported local history data to a remote relational database using Niagara's standard RdbmsHistoryExport descriptors, this component supplements history time queries with archived history records retrieved back (on-demand) from the remote RDBMS.

This provider is in the **rdb** module's palette. To add it to your station, expand **HistoryService Archive Provider** in the palette and drag this component to the **Config→Services→HistoryService** in the Nav tree.

Figure 35 Rdb Archive History Provider properties



To access this **Property Sheet**, expand **Config→Services→HistoryService→Archive History Providers** and double-click **RdbArchiveHistoryProvider**.

In addition to the standard properties (Status, Enabled, and Fault Cause), these properties configure the archive history provider.

Property	Value	Description
Max Archive Results Per Query	number (defaults to 50,000)	Determines the maximum number of history records to read from the RDBMS for any history time range query that taps into it. If more history records are available beyond this limit at history query time, the **Archive Limit Notifications** property defines the behavior of a subset of Workbench views but not all of them. When the limit is reached for a query, in addition to the warning, you get truncated archive history results that always consider the most recent history records first. Web chart and HTML5 History Table views (accessible from a browser and Workbench) provide their own notification when a history query exceeds this limit.
Archive Limit Notifications	drop-down list (defaults to `Notify Once Per Query Range Per Session`)	Specifies the Workbench notification behavior when the **Max Archive Results Per Query** limit is exceeded for a history query made from a Workbench user connected to the station.

Property	Value	Description
		NOTE: This property does not apply to HTML5 history views including HTML5 views accessed within Workbench, such as the Web Chart view. It only applies to native Workbench views that perform history queries, such as the AX History Chart or AX History Table views. `Notify Once Per Query Range Per Session` limits the frequency of warnings to once per history query range per session. This setting attempts to balance providing notifications while reducing too much spam. `Never Notify` Turns notification off completely. `Always Notify` displays a warning every time a history query's results exceed the Max Archive Results Per Query limit.
Ord To Rdbms	ORD with choosers	References the RDBMS device instance in the local station from which to retrieve archive history data on demand.
Use Default Fetch Size	`true` (default) or `false`	It is not common to change this property from the default value. In rare cases, it may be necessary to tweak the fetch size for retrieving history records from a connection to the RDBMS for performance reasons. `true`, uses the default fetch size specified by the referenced RDBMS device when making a connection to retrieve history data from the RDBMS. `false` uses the **`Custom Fetch Size`** value instead.
Custom Fetch Size	number	It is not common to change this property from the default value. In rare cases, it may be necessary to tweak the fetch size for retrieving history records from a connection to the RDBMS for performance reasons. This property can specify a custom fetch size to use when making a connection to retrieve history data from the RDBMS. This property value applies only when the **`Use Default Fetch Size`** property is set to `false`.
Fetch Size To Use	read-only, A value of zero is acceptable (and common).	Based on the current value of the **`Use Default Fetch Size`** and **`Custom Fetch Size`** properties, this property indicates the current fetch size in use for any connections made to the configured RDBMS when retrieving archived history data.

orion module

This module configures the framework's Orion database. This general-purpose uncertain database system unifies the modeling of probabilistic data across the framework.

Orion API

The Orion API (Application Programming Interface) provides commands to embed in applications that invoke orion module functions.

BRdbms

This function existed in earlier Niagara versions of the framework to model an Rdbms database for supporting SQLServer and Oracle database implementations. Applications (including stations running on smaller

controllers) are able to implement O/R mapping and maintain database independence by running a local instance of any of the following RDBMS types: HSQLDB, MySQL, Oracle, or SQL Server

BOrionService

This service allows a station and its applications to use a local Orion relational database running in controllers (including embedded controllers) to manage and display distributed-system (or distributed-application) information. This managed and configurable information includes certain types of component data that are well suited for the relational model.

BIOrionApp

This is an installable application with a set of Orion object types for managing data in the Orion database.

BOrionSpace

This function provides component space for storing Orion objects. Other types of space include: History, File, and Virtual space. An example ORD using the orion space is: `local:|fpx:|propm://HsqlDatabase/hierarchy/NodeType`.

BOrionObject

This is a common subclass for all objects stored in the Orion database. It represents an Orion object and its associated database.

OrionSession

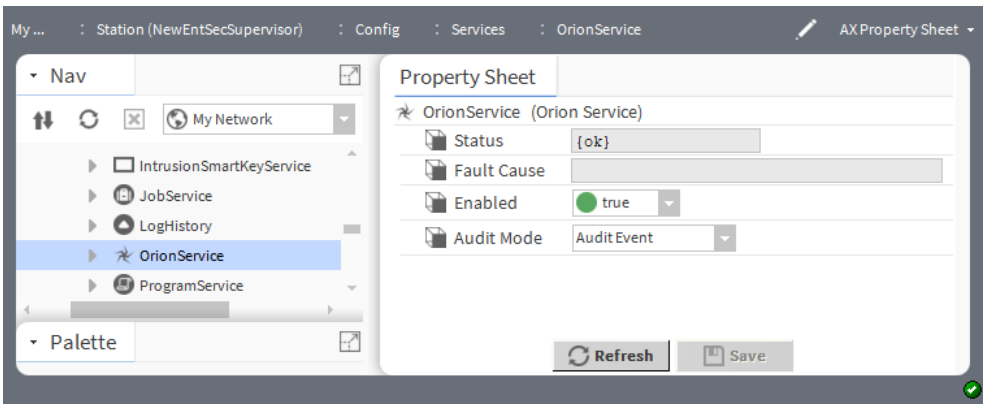
This is a CRUD (Create, Read, Update and Delete) interface for interacting with the Orion database and managing object persistence.

orion-OrionService

This component under the **Services** node in the Nav tree enables the Orion database in a station. It uses a local relational database running in a controller (including embedded controllers) to manage and display distributed-system (or distributed-application) information. This managed and configurable information includes certain types of component data that are well suited for the relational model. It is available in the **`orion`** palette.

Orion is an Object-Relational (O/R) mapping architecture provided to support distributed-applications, large systems, and other applications that may benefit from having relational data modeled as framework objects. Object-Relational Mapping does not replace the config.bog (station) file, but provides for the creation of a new space, the Orion space, that is stored outside the station database file (like histories, files, and modules). This functionality includes a means for alternative and multiple system hierarchy displays that can be used for data presentation, system identification, and navigation.

Figure 36 Orion Service properties



To access these properties, expand **Services**, and double-click the **OrionService** node in the Nav tree.

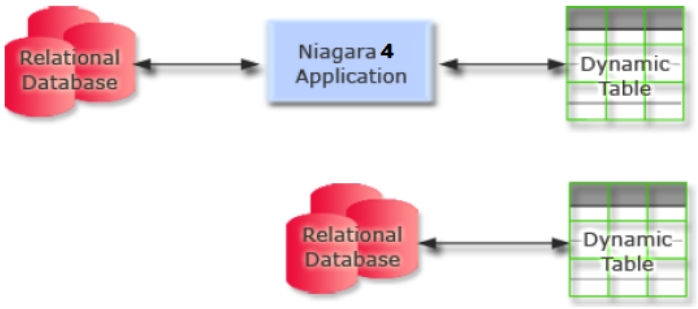
In addition to the standard properties (Status, Fault Cause and Enabled), one property supports this component.

Property	Value	Description
Audit Mode	drop-down list (defaults to `None`)	Specifies if audits are to be sent to a history, database table or both. `None` ignores audit events. `Audit Event` sends audit events to a history. `Database Record` sends audit events to the database table. `Both` sends audit events to both a history and database table.

orion-DynamicTable

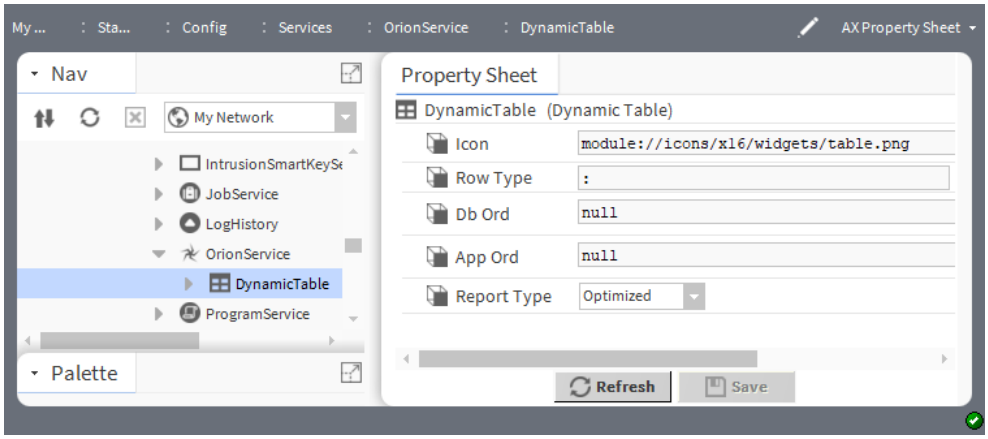
This component configures, retrieves and displays data from relational database tables or from other application services. The Dynamic Table has a configuration view (**Dynamic Table Config**) and a table view (**Dynamic Table**). It is available in the **orion** palette.

Figure 37 Rdbms model with dynamic tables



You drag a **DynamicTable** component from the **orion** palette to the **OrionService** in your station and then configure the component to display data from either an application that is using relational data or directly from a relational database.

Figure 38 Dynamic Table properties



To access these properties, expand **Config→Services→OrionService**, right-click **DynamicTable** and click **Views→AX Property Sheet**.

Property	Value	Description
Icon	file path	Designates a graphic icon to associate with a dynamic table.
Row Type	number	Specifies the **DynamicTable** row in terms of `module` and `type`.
Db Ord	ORD	Specifies an ord path to the relational database. Either this property or the **App Ord** property (but not both) are required for linking the **DynamicTable** to a database. With the **Orion-Service** installed, you can use a chooser to select the database under the **DynamicTable** node.
App Ord	text	Specifies an ord path to an application. Either this property or the **Db Ord** property (but not both) are required for linking the **DynamicTable** to a database.
Report Type	drop-down list	Selects how much data to include in the table. Optimized limits the amount of data. Full Report outputs all data.

orion-FoxOrionDatabase

This component represents the Orion database. It contains the individual Orion Module Types, appears in the Nav tree directly under the OrionRoot node and is not visible in the **orion** palette.

orion-FoxOrionSpace

This component provides the Nav Container View of the Orion database.

To access this component, expand the **Station** node in the Nav tree and double-click **Orion**.

orion-OrionModule

This component represents a module with types registered in an Orion database. The **orion-OrionModule** is not visible in the **orion** palette.

orion-OrionRoot

This component is the root component of an Orion component space. It contains the individual databases that are managed by the OrionService. The orion-OrionRoot component appears in the Nav tree directly under the station when the Orion service is configured and is not visible in the **orion** palette.

orion-OrionType

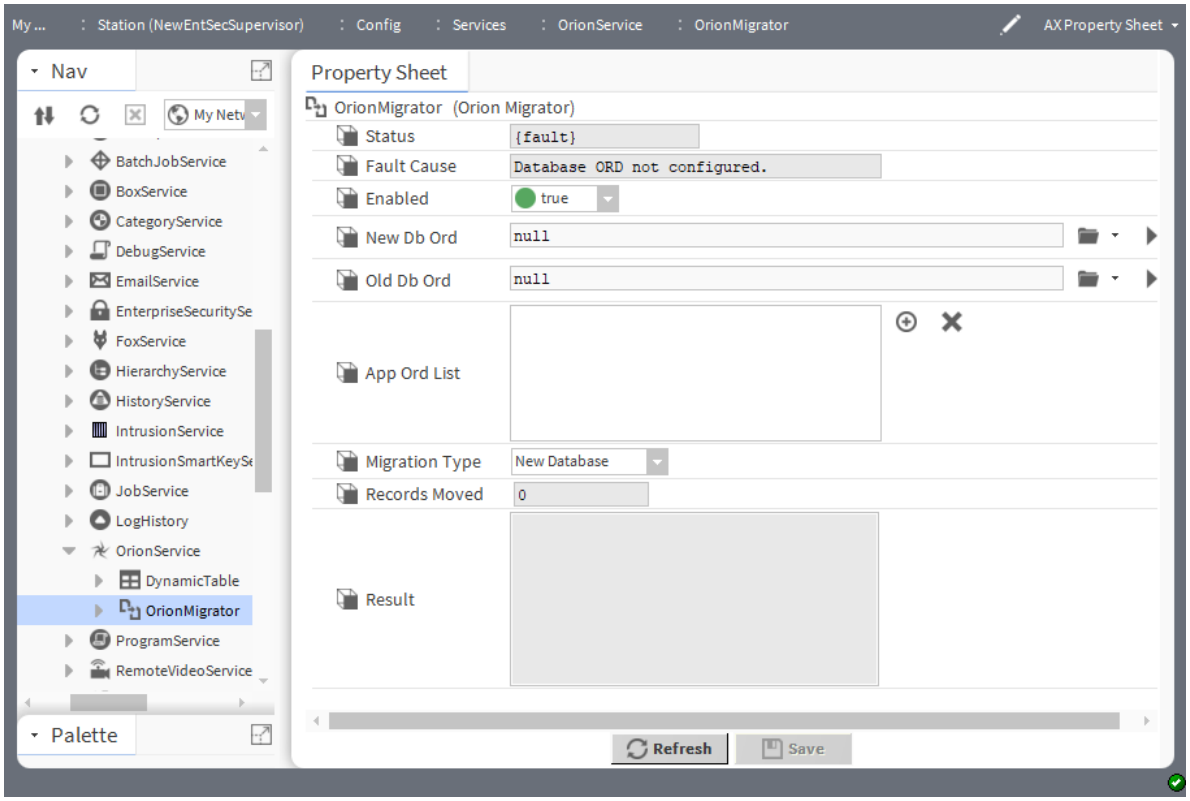
This component is a wrapper for Orion Types, which display in the **Orion Type Summary** view and **Orion Type Table** view. The orion-OrionType component is not visible in the **orion** palette.

orion-OrionMigrator

This component, available in the **orion** palette, is used to migrate existing Orion database data without UTF-8 support into a new RdbmsDevice, which does support the UTF-8 Unicode Encoding Scheme.

You drag an **OrionMigrator** component from the **orion** palette to the **OrionService** in your station.

Figure 39 OrionMigrator properties



To access these properties, expand **Config→Services→OrionService** and double-click **OrionMigrator**.

In addition to the standard properties (Status, Fault Cause and Enabled), these properties support this component.

Property	Value	Description
New Db Ord	ORD selector	Defines the location in the station of the new (target) database.
Old Db Ord	ORD selector	Defines the location in the station of the existing (source) database. This is the database that is already in use.
App Ord List	list	Provides a location to add and delete database ORDs. If you migrate all Orion Apps to the same Rdbms Device, leae this property blank. This also moves the OrionAudit and OrionAppVersion tables to the new database.
Migation Type	drop-down list	Configures what to do based on the condition of the target (new) database. `New Database` is optimized for speed, but throws an exception if any record already exists. This should be your choice for a brand new database. `Insert Only` skips existing records in the target database. `Overwrite` replaces existing records in the target database. If an error occurs, the update fails and stops the migration. `Force Overwrite` replaces existing records in the target database and ignores errors.

Property	Value	Description
		This option is required if you have an Niagara Enterprise Security station with an intrusion zone.
Records Moved	read-only	Reports the number of records migrated.
Result	read-only	Provides additional information about the migration.

Chapter 4 Plugins (views)

Topics covered in this chapter

- Device Manager
- Dynamic Table view
- Dynamic Table Config view
- Orion Db Manager
- Orion Module Types view
- Orion Type Summary view
- Orion Type Table View
- Rdbms History Import Manager
- Point Device Ext Manager
- Rdbms Point Query Manager
- Rdbms Query View
- Rdbms Session View
- MySQL History Export Manager
- Oracle History Export Manager
- SqlServer History Export Manager

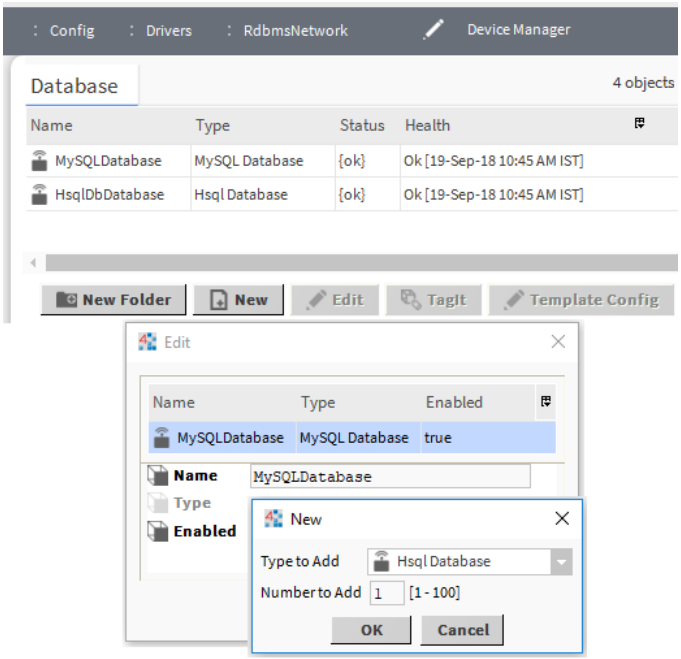
Plugins provide views of components and can be accessed in many ways. For example, double-click a component in the Nav tree to see its default view. In addition, you can right-click on a component and select from its **Views** menu.

For summary documentation on any view, select **Help→On View** (**F1**) from the menu or press **F1** while the view is open.

Device Manager

This view manages the rdb Database devices.

Figure 40 Device Manager view with Add and Edit windows



You access this view by double-clicking the **RdbmsNetwork** node in the Nav tree.

The screen captures show the **Device Manager** view for the **RdbmsNetwork** and examples of the **Edit** and **New** windows for setting up an MySQL database. The driver treats the database as a device in the station.

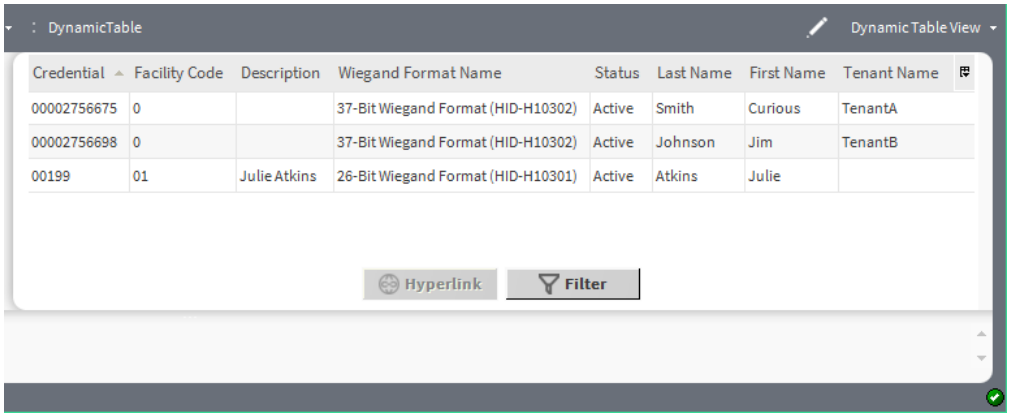
The **New** and **Edit** buttons open **New** and **Edit** windows, which add, configure and monitor rdb database devices.

Individual rdb database devices have different characteristics, features, and properties that are specific to the type of database that they represent. However, most of the setup, configuration, import and export features are similar among all rdb database devices.

Dynamic Table view

This the default view on the **DynamicTable** component. It displays data selected from a database that is specified in the DynamicTable **Property Sheet** view and according to the properties set up in the **Dynamic Table Config** view.

Figure 41 Dynamic Table view



You add a DynamicTable component to a Supervisor station from the **orion** palette, create a table using the **Dynamic Table Config** view, then select `Dynamic Table View` from the drop-down list in the upper right corner.

The **Filter** button at the bottom of the view allows you to display a subset of the table data based on parameters that you can set in the associated **Filters** window. The **Hyperlink** button changes the view to the **Property Sheet** view of the (single) node that you have selected.

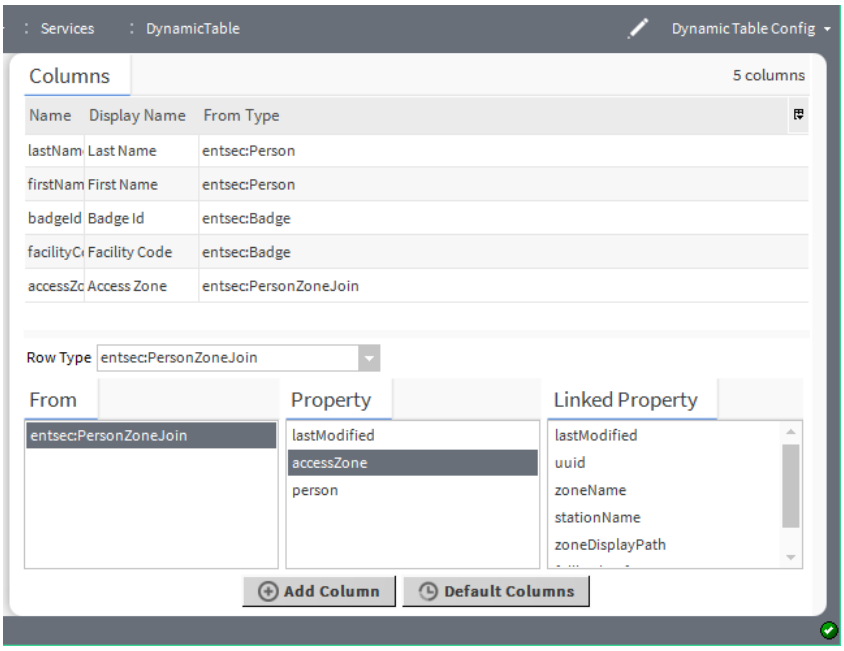
Columns

Column Name	Description
Credential	Displays an identifier.
Facility Code	Identifies the geographic location.
Description	Report additional information.
Wiegand Format Name	Identifies the wiring standard for a card reader.
Status	Reports the current condition of the entity as of the last refresh: {alarm}, {disabled}, {down}, {fault}, {ok}, {stale}, {unackedAlarm}
Last Name	Reports last name associated with the record.
First Name	Reports first name associated with the record.
Tenant Name	Reports the name of the company that occupies the facility.

Dynamic Table Config view

This is a multi-pane view on the DynamicTable component used to configure the columns on a dynamic table view.

Figure 42 Dynamic Table Config view



You access this view by expanding **Services**, right-clicking the **DynamicTable** node and clicking **Views→Dynamic Table Config**.

This view has two major panes: an upper and lower pane. The upper pane contains a table that displays the data records from the row and columns that you designate in the lower pane.

The DynamicTable **Property Sheet** points to a database or application ORD. This database or application is the source for data in the Dynamic Table Config view Row and Column properties. By choosing a Row Type, you can manually add columns to a dynamic table by selecting fields under the From, Property, and Linked Property panes and clicking the **Add Columns** or **Default Columns** buttons.

Columns

Column name	Description
Name	Displays the name of the person.
Display Name	Displays the set display name.
From Type	Displays the software module type.

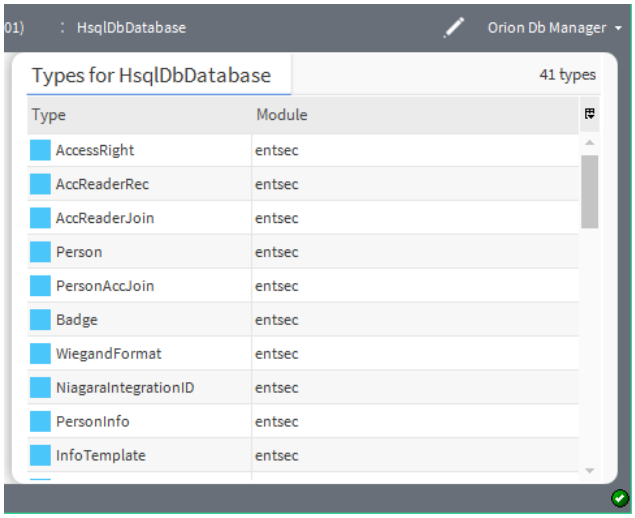
Buttons

- **Add Column** adds a new column.
- **Default Column** adds a column based on the selection of the row type.

Orion Db Manager

This is the view on the FoxOrionDatabase component on a controller station. It displays a table of all the Orion types and their associated module for the selected database.

Figure 43 Orion Db Manager view



This view of the Orion database is directly under the Station. To access it, expand **Station→Orion→ DatabaseName** (such as HsqlDbDatabase).

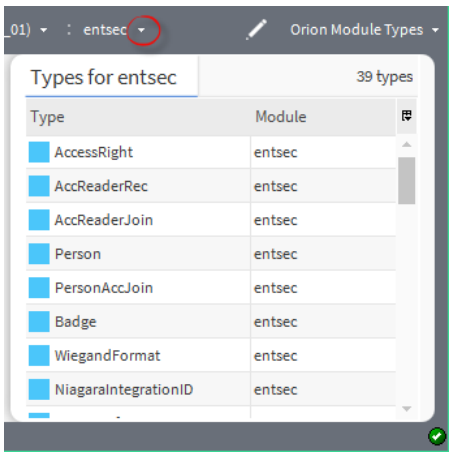
Columns

Column name	Description
Type	Provides the name of the data item.
Module	Identifies the software module name that contains this item.

Orion Module Types view

This is a view on the FoxOrionDatabase component in a remote controller station. It displays a table of all the Orion types and their associated module for the selected database.

Figure 44 Orion Module Types



This view of the Orion database is directly under the Station. To access it, expand **Station→Orion→ Orion-DatabaseName** (such as HsqlDbDatabase), then click the drop-down arrow next to the module name (circled above) and select a type.

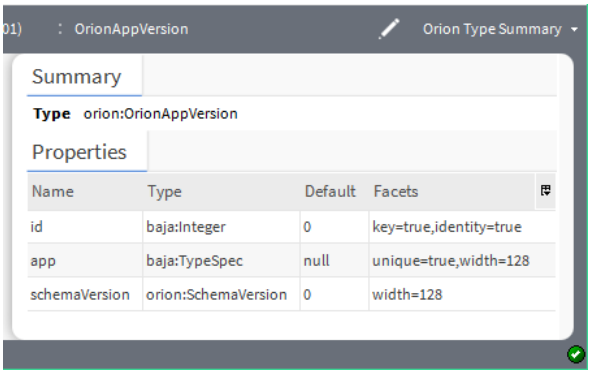
Columns

Column name	Description
Type	Provides the name of the data item.
Module	Identifies the software module name that contains this item.

Orion Type Summary view

This is a view on the **OrionType** component that has an upper and lower section.

Figure 45 Orion Type Summary view



This view shows the Type identification (module:type) and its Super Type in the top section. The Orion Type properties are listed in a table in the lower section of the view.

You access this view by expanding **Station→Orion→DatabaseName** followed by double-clicking orion, and selecting Orion Type Summary from the drop-down list in the upper right corner of the view.

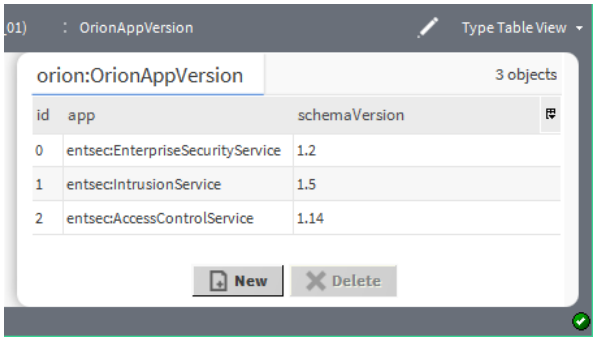
Columns

Column name	Description
Name	Displays the name of the property.
Type	Displays the default instance used.
Default	Displays information.
Facets	Determines the value formatted for display.

Orion Type Table View

This is a view on the **OrionType** component.

Figure 46 Orion Type Table view



This view shows a table of Orion application records.

Columns

Column name	Description
id	Displays a number.
app	Displays the module name and service used.
Schema Version	Displays the version of the schema used.

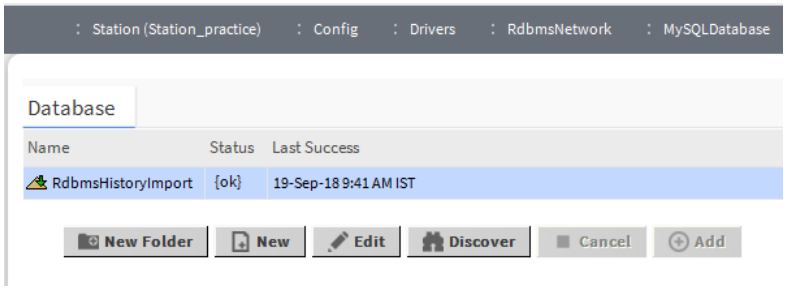
Buttons

- **New** creates a new app version.
- **Delete** removes the selected app version.

Rdbms History Import Manager

This view imports records from an Rdbms database as framework histories. It is a view on the Histories extension of an Rdbms database.

Figure 47 Rdbms History Import Manager View



You access this view by expanding **Drivers→RdbmsNetwork**, expanding your rdb Database, right-clicking the **Histories** node in the Nav tree and clicking **Views→Rdbms History Import Manager**.

Columns

Column name	Description
Name	Identifies the history import.
History ID	Reports the import history ID.
Execution Time	Displays the last time a history was imported.

Column name	Description
Enabled	Indicates if history import is on (true) or off (false).
Status	Displays the status of the imported history.
State	Displays the current state of the database.
Last Success	Reports the last time the station successfully performed this function.
Last Failure	Reports the last time the system failed to perform this function. Refer to **`Fault Cause`** for details.
Fault Cause	Indicates the reason for a fault.
Capacity	Specifies the number of trend log records (histories) to store in the histories database. When capacity is reached, newer records overwrite the oldest records.
Full Policy	Specifies what happens when a trend log (history) reaches capacity. Applies only if **`Capacity`** is set to `Record Count`. When capacity reaches record count, the newest records overwrite the oldest records. `Roll` ensures that the latest data are recorded. `Stop` terminates recording when the number of stored records reaches capacity. Full policy has no effect if `Capacity` is `Unlimited`.
Interval	Displays the interval in which the history is logged.
Value Facets	Displays the units.
Time Zone	Displays the historical time zone.

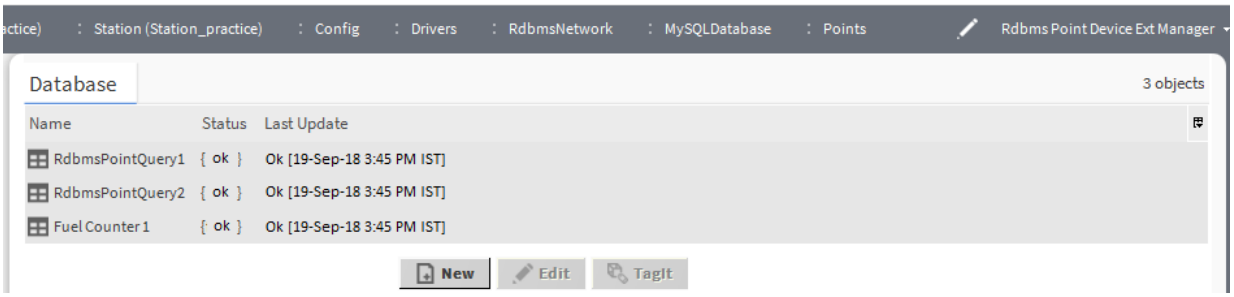
Buttons

- **New Folder** creates a new folder for devices. Each such folder provides its own set of manager views.
- **New** creates a new device record in the database.
- **Edit** opens the device's database record for updating.
- **Discover** runs a discover job to locate installed devices, which appear in the **Discovered** pane. This view has a standard appearance that is similar to all **Device Manager** views.
- **Add** inserts into the database a record for the discovered and selected object.
- **Match** associates a discovered device with a record that is already in the database.
- **Archive** archives the history types.

Point Device Ext Manager

This is the default view of the Rdbms Point Device Extension. It has a single **Database** pane that displays any Rdbms Point Queries that are present.

Figure 48 Point Device Ext Manager view



You access this view by expanding the **Drivers→RdbmsNetwork**, expanding your rdb Database and double-clicking the **Points** folder.

Columns

Column Name	Description
Name	Displays the name of the RDBMS point query
Status	Defines the current status of the point.
Last Update	Displays the status and the last updated date and time.

Using this manager view, you can do the following:

-
- In addition, you can select individual rows to select other menu items to edit the row and go to other views of a selected entry.

Right-click menu

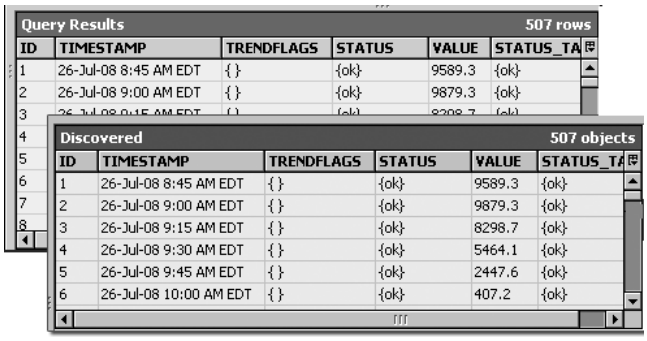
Right–clicking one or more entries in the **Database** pane opens a popup menu with these options:

- **Views** selects the view: **Property Sheet**, **RDBMS Query**, etc.
- **Action** executes the query.
- **New** adds a new query.
- **Rename** changes the name of the query.
- **Cut** removes the selected row(s).
- **Copy** makes a copy of the selected row(s)
- **Paste** inserts the copied row(s).
- **Delete** removes the selected row(s).
- **Duplicate** copies and inserts the selected row as a new table row.
- **Reorder** changes the order of the rows in the table.

Rdbms Point Query Manager

This is the default view for the **RdbmsPointQuery** component under the **Points** folder in the Nav tree. It works in a way that is similar to the standard **Point Manager** view and, like that view, it has a Discovered and Database pane as well as similar buttons at the bottom of the view.

Figure 49 Point Query Manager view



You access this view, once you have created an RdbmsPointQuery by expanding **Drivers→RdbmsNetwork** in the Nav tree, expanding the **Points** folder under your rdb Database node and double-clicking the **RdbmsPointQuery** container.

This view works in a way that is similar to the standard Point Manager view and, like that view, it has **Discovered** and **Database** panes, as well as similar buttons at the bottom of the view.

Columns

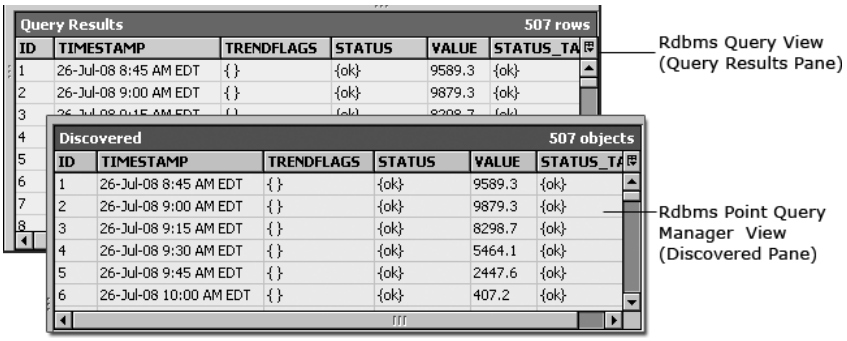
Column name	Description
Name	Identifies the name of the point.
Type	Reports the type of point (Boolean, Numeric, etc.)
Out	Indicates the status of the query.
Value Column	Reports the point value.
Key Value 1	Identifies the primary key.
Key Value 2	Identifies the secondary key.

Buttons

- **New Folder** creates a new folder for devices. Each such folder provides its own set of manager views.
- **New** creates a new device record in the database.
- **Edit** opens the device's database record for updating.
- **Cancel** ends the current discovery job.
- **Add** inserts into the database a record for the discovered and selected object.
- **Match** associates a discovered device with a record that is already in the database.

Query Results vs Discovery

Figure 50 Example of an Rdbms Query compared to the Discovered pane

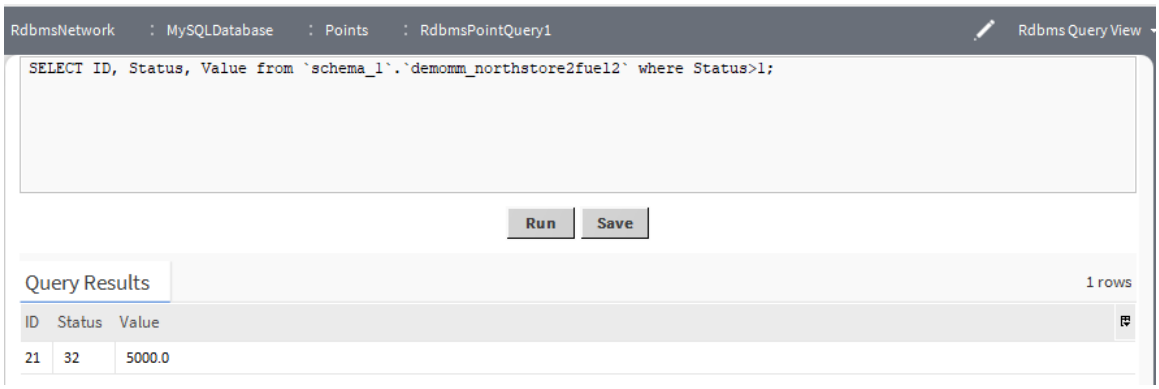


Discovered points represent the results of executing the Rdbms point query. In fact, the results from the data in the **Discovered** pane should match the data presented in the Query Results pane of the **Rdbms Query View**. The **Discover** button executes the query defined by the `Sql` property in the **New Rdbms Point Query** window.

Rdbms Query View

This is a view on the **RdbmsPointQuery** component. It is typically a view for working with queries as you are developing them because queries execute immediately and the lower pane displays the results as soon as you click the **Run** button (or with some delay, depending on database size and network connection speed). If there are errors in the query, an error window opens an error message.

Figure 51 Rdbms Query View



Assuming you have already created at least one query, you access this view for that query by expanding the **Drivers→RdbmsNetwork**, expanding your rdbDatabase, expanding the **Points** folder, right-clicking the **RdbmsPointQuery** node, and clicking **Views→RdbmsQueryView**.

The **Rdbms Point Query** View has two panes and two buttons. The top **Query** pane is a text editor you use to type a query directly into the editor box. Any saved changes are also reflected in the **`Sql`** property of the RdbmsPointQuery **Property Sheet**. Saved changes in the **Property Sheet** are also reflected here.

Buttons

- **Run** executes the query in **Query** pane.
- **Save** saves the results of the query.

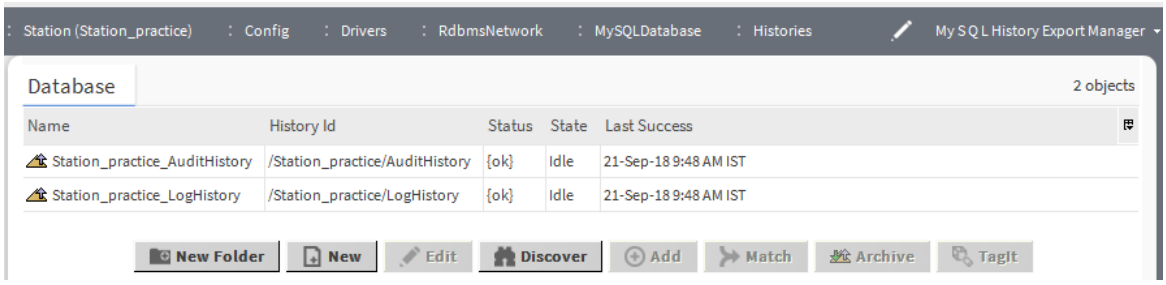
Rdbms Session View

This view provides information about the relational database session.

MySQL History Export Manager

This view discovers, configures and exports history records to a MySQL Rdbms database. It is a view on the MySQL Histories extension.

Figure 52 MySQL History Export Manager view



You access this view by expanding **Drivers→RdbmsNetwork→MySQLDatabase**, right-clicking the Histories node in the Nav tree and clicking **Views→MySQL History Export Manager** node in the Nav tree.

Columns

Name	Description
Name	Identifies the type of history to export.
History ID	Reports the ID for the exported history.

Name	Description
Execution Time	Displays when the export last ran.
Enabled	Indicates if this history export is configured to run.
Status	Reports the condition of the exported history.
State	Displays the current state of the history.
Last Success	Reports when the history last exported successfully.
Last Failure	Reports When the history export failed.
Fault Cause	Indicates the reason the last export failed.

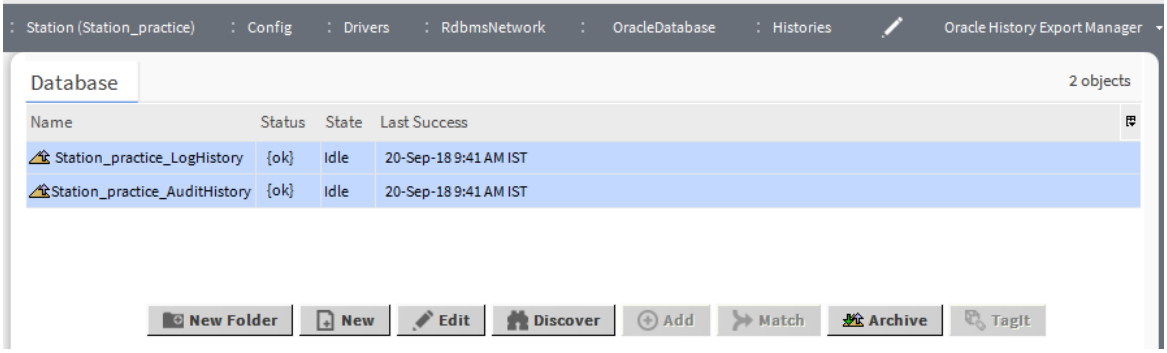
Buttons

- **New Folder** creates a new folder for devices. Each such folder provides its own set of manager views.
- **New** creates a new device record in the database.
- **Edit** opens the device's database record for updating.
- **Discover** runs a discover job to locate installed devices, which appear in the **Discovered** pane. This view has a standard appearance that is similar to all **Device Manager** views.
- **Add** inserts into the database a record for the discovered and selected object.
- **Match** associates a discovered device with a record that is already in the database.

Oracle History Export Manager

This view discovers, configures and exports history records to a Oracle Rdbms database. It is a view on the Oracle Histories extension.

Figure 53 Oracle History Export Manager view



You access this view by expanding **Drivers→RdbmsNetwork→OracleDatabase**, right-clicking the Histories node in the Nav tree and clicking **Views→Oracle History Export Manager** node in the Nav tree.

Columns

Name	Description
Name	Identifies the type of history to export.
History ID	Reports the ID for the exported history.
Execution Time	Displays when the export last ran.
Enabled	Indicates if this history export is configured to run.

Name	Description
Status	Reports the condition of the exported history.
State	Displays the current state of the history.
Last Success	Reports when the history last exported successfully.
Last Failure	Reports When the history export failed.
Fault Cause	Indicates the reason the last export failed.

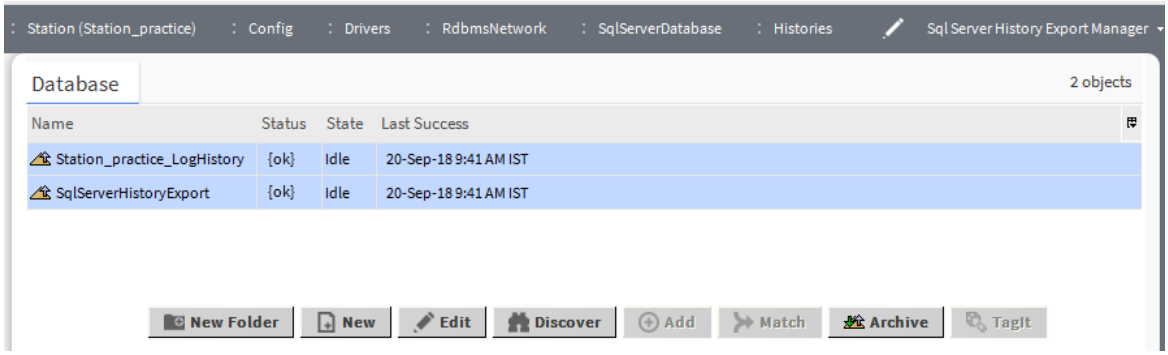
Buttons

- **New Folder** creates a new folder for devices. Each such folder provides its own set of manager views.
- **New** creates a new device record in the database.
- **Edit** opens the device's database record for updating.
- **Discover** runs a discover job to locate installed devices, which appear in the **Discovered** pane. This view has a standard appearance that is similar to all **Device Manager** views.
- **Add** inserts into the database a record for the discovered and selected object.
- **Match** associates a discovered device with a record that is already in the database.
- **Archive** archives the history types.

SqlServer History Export Manager

This view discovers, configures and exports history records to a SqlServer Rdbms database. It is a view on the SqlServer Histories extension.

Figure 54 SqlServer History Export Manager view



You access this view by expanding **Drivers→RdbmsNetwork→SqlServerDatabase**, right-clicking the Histories node in the Nav tree and clicking **Views→SqlServer History Export Manager** node in the Nav tree.

Columns

Name	Description
Name	Identifies the type of history to export.
History ID	Reports the ID for the exported history.
Execution Time	Displays when the export last ran.
Enabled	Indicates if this history export is configured to run.
Status	Reports the condition of the exported history.

Name	Description
State	Displays the current state of the history.
Last Success	Reports when the history last exported successfully.
Last Failure	Reports When the history export failed.
Fault Cause	Indicates the reason the last export failed.

Buttons

- **New Folder** creates a new folder for devices. Each such folder provides its own set of manager views.
- **New** creates a new device record in the database.
- **Edit** opens the device's database record for updating.
- **Discover** runs a discover job to locate installed devices, which appear in the **Discovered** pane. This view has a standard appearance that is similar to all **Device Manager** views.
- **Add** inserts into the database a record for the discovered and selected object.
- **Match** associates a discovered device with a record that is already in the database.
- **Archive** archives the history types.

Chapter 5 Windows

Topics covered in this chapter

- New database windows
- New points windows
- New export histories windows
- New import histories windows
- New query windows

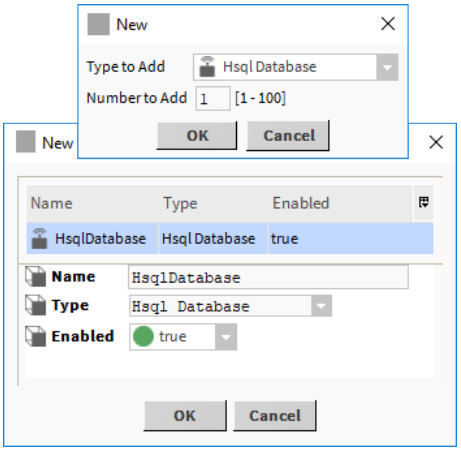
Windows create and edit database records or collect information when accessing a component. You access them by dragging a component from a palette to a nav tree node or by clicking a button.

Windows do not support **On View (F1)** and **Guide on Target** help. To learn about the information each contains, search the help system for key words.

New database windows

These windows configure a new RDBMS.

Figure 55 MySQL History Device Ext properties



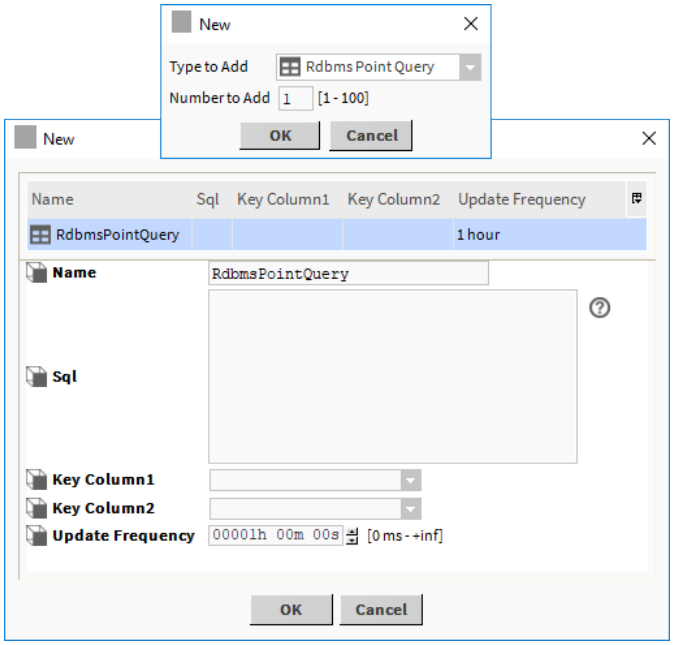
To access these properties, expand **Config→Drivers**, double-click the **RdbmsNetwork** node and click **New**.

Property	Value	Description
Name	text	Supplies a name for the database.
Type	drop-down list	Selects the type of database.
Enabled	`true` (default) or `false`	Indicates if the database is online.

New points windows

These windows add database queries as points to configure what data to download from the database.

Figure 56 Points window properties



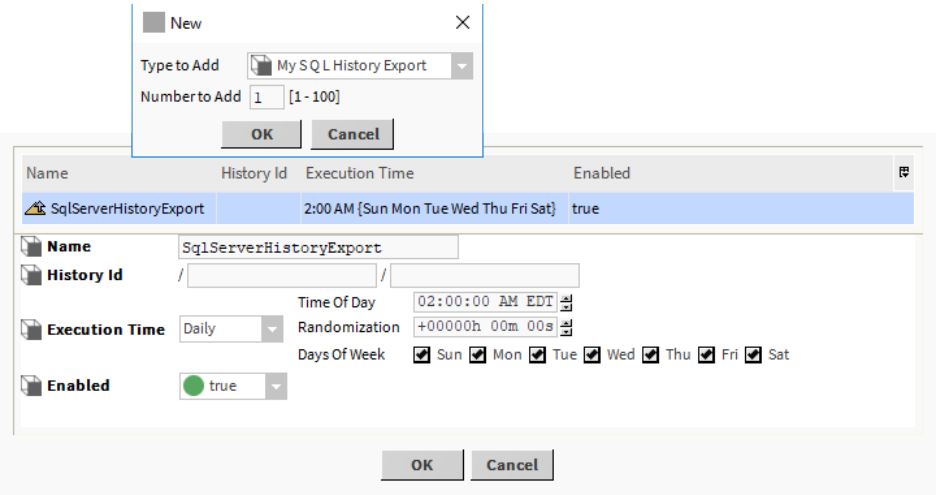
To open these windows, expand **Config→Drivers→RdbmsNetwork**, expand a database node in the network view and click **New**.

Property	Value	Description
Name	text	Provides a name for the import/export task or point query.
Sql	text	Sets up Sql queries using BFormat. Click the question mark for examples and help.
Key Column 1	text	Defines the primary key for imported data. This key uniquely identifies each row in a database table. It might be part of the data record itself (for example, a unique user id) or an extra piece of information that is not related to the actual data record. A primary key can consist of Key Column 1 and 2, creating a composite key.
Key Column 2	text	Augments **`Key Column 1`** when **`Key Column 1`** is not enough to establish a unique key for each imported data record.
Update Frequency	hours minutes seconds	Configures how frequently to execute the query.

New export histories windows

These windows configure when and which histories to export.

Figure 57 New export histories windows



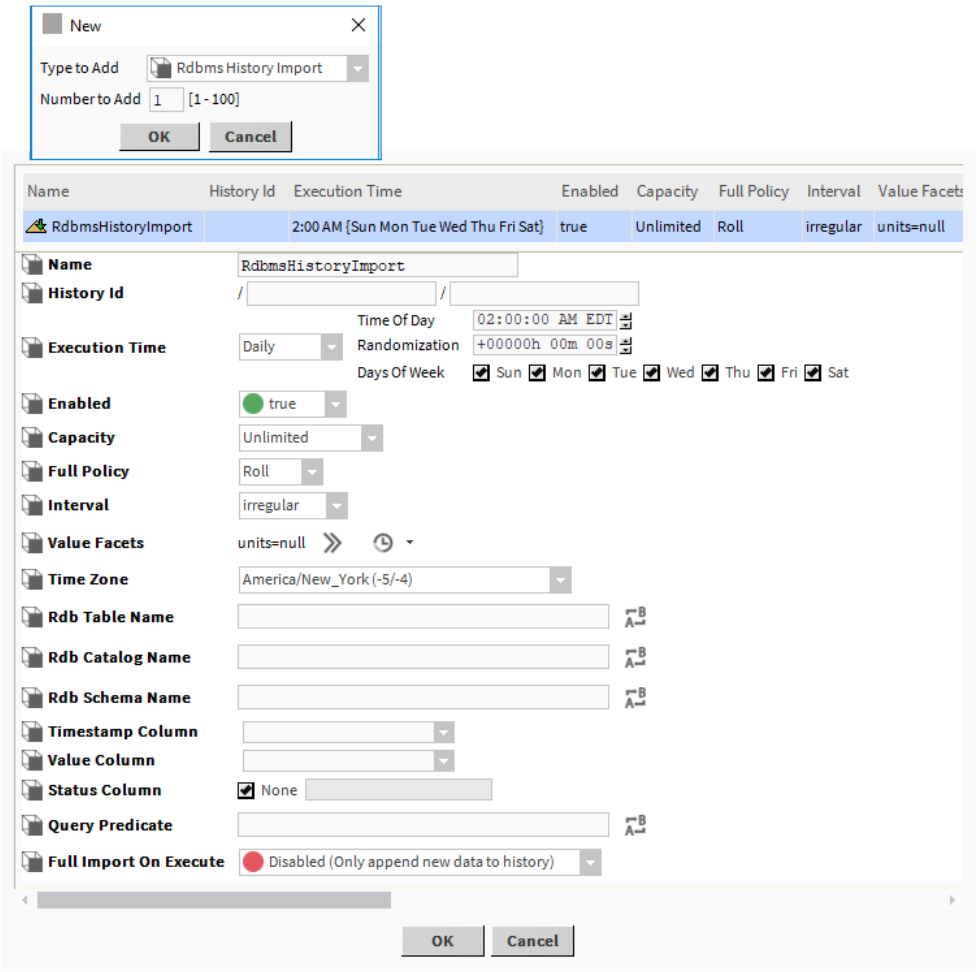
To access these windows, expand **Drivers→RdbmsNetwork**, double-click the **Histories** node in the Nav tree, and click **New**.

Property	Value	Description
Name	text	Provides a name for the import/export task or point query.
History Id	text	Specifies the database name, such as MySQL, Oracle, SqlServer) and history name in two fields.
Execution Time	control time trigger properties	Configures a time trigger that controls when to perform the function. Trigger properties are documented in the *Getting Started with Niagara* guide.
Enabled	`true` (default) or `false`	Activates (`true`) and deactivates (`false`) use of the object (network, device, point, component, table, schedule, descriptor, etc.).

New import histories windows

These windows configure the importing of history data into a Supervisor station RDBMS.

Figure 58 Import history window



To access these windows, expand **Drivers→RdbmsNetwork**, expanding your rdb Database, right-click the **Histories** node in the Nav tree, click **Views→Rdbms History Import Manager**, and click **New**.

Property	Value	Description
Name	text	Provides a name for the import/export task or point query.
History Id	text	Specifies the database name, such as MySQL, Oracle, SqlServer) and history name in two fields.
Execution Time	control time trigger properties	Configures a time trigger that controls when to perform the function. Trigger properties are documented in the *Getting Started with Niagara* guide.
Enabled	`true` (default) or `false`	Activates (`true`) and deactivates (`false`) use of the object (network, device, point, component, table, schedule, descriptor, etc.).
Capacity	drop-down list	Configures the maximum number of records to import into a database. `Unlimited` places no limit on the size of the import.

Property	Value	Description
		`Record Count` opens a property to define the number of records (defaults to 500).
Full Policy	drop-down list (defaults to `Roll`)	Determines what happens when the station database reaches its **`Capacity`**. `Roll` overwrites the oldest records with the new ones. `Stop` stops the import job from recording.
Interval	drop-down list (defaults to `irregular`)	Configures when the system creates a history record. `irregular` specifies no particular frequency for imports. `regular` sets an import action frequency in terms of hours, minutes and seconds.
Value Facets	Config Facets chooser	Configures units.
Time Zone	drop-down list	Records the time zone associated with the history. All the records (rows) are recorded in UTC (Coordinated Universal Time) and there are various localization technologies you may choose to display the time zone data.
Rdb Table Name	text	Redefines the table name to this name on import.
Rdb Catalog Name	text	Redefines the catalog name to this name on import.
Rdb Schema Name	text	Redefines the schema name to this name on import.
Timestamp Column	text	Specifies the Timestamp column for the imported data. If you discovered the histories to import, the driver displays the table columns in the option list for you to choose from. Otherwise, type in the column name in the text field.
Value Column	text	Specifies the Value column for the imported data. If you discovered the data, the driver displays table columns in the option list for you to choose from. Otherwise, type in the column name in the text field.
Status Column	text	Specifies if a status column is included in the imported data or not. To include the status column, clear the **`None`** box and enter the name of the status column. If you discovered data, the driver displays table columns in the option list for you to choose from. Otherwise, type in the column name in the text field.
Query Predicate	text	Uses a query predicate to filter the records to import. For example, you could type in “where Value > 100” to import only those records that have a Value greater than 100. Or, you could type in “where Value between 1 and 100” , which would import records with Values between 1 and 100.
Full import On Execute	drop-down list	Specifies to import either the full database up to the specified **`Capacity`** (`Enabled`) or only the new data on each successive import action (`Disabled`).

New query windows

This window adds point queries under the **Point Device Ext Manager**.

Once added, these Rdbms Point Queries appear in the manager view as well as in the Nav tree.

Figure 59 The New query window

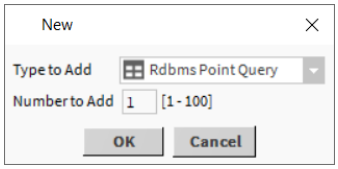
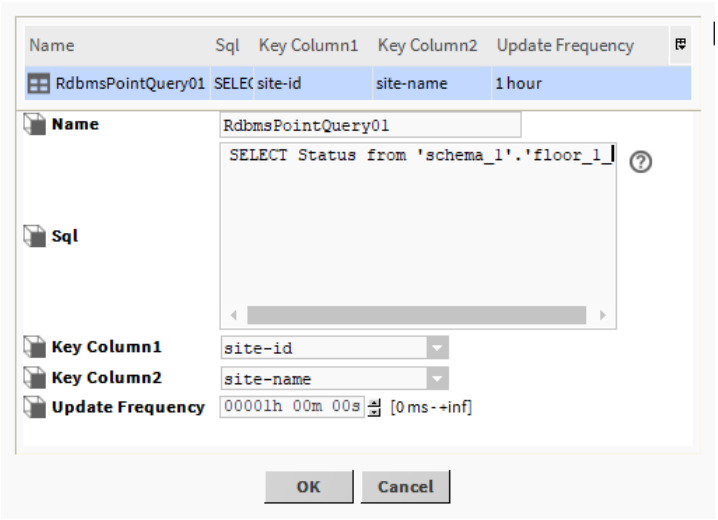


Figure 60 New/edit query properties



You open these windows when you click New or Edit in the **Point Device Ext Manager**.

Property	Value	Description
Type to Add	drop-down list	Selects the type of query to add.
Number to Add	number	Selects how many queries to add.
Name	text	Provides a name for the import/export task or point query.
Sql	text	Sets up Sql queries using BFormat. Click the question mark for examples and help.
Key Column 1	text	Defines the primary key for imported data. This key uniquely identifies each row in a database table. It might be part of the data record itself (for example, a unique user id) or an extra piece of information that is not related to the actual data record. A primary key can consist of Key Column 1 and 2, creating a composite key.
Key Column 2	text	Augments **`Key Column 1`** when **`Key Column 1`** is not enough to establish a unique key for each imported data record.
Update Frequency	hours minutes seconds	Configures how frequently to execute the query.

Index

P

Q

R

S

T

U

V

W