Technical Document

Niagara Rdbms Driver Guide

June 5, 2020

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. In order to make the most of the information in this book, readers should have some training or previous experience with Niagara 4 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 .

June 5, 2020

- Minor corrections to several component topics to support online help.
- Added information specifying that the mySql Connector/J 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

- ♦ Modules
- ♦ Database requirement
- ♦ Prerequisite checklist
- ♦ Installing the network and database
- ♦ Configuring Supervisor database properties
- ♦ Testing the database connection
- ♦ Database connection troubleshooting
- ♦ Replacing the standard alarm database

The Rdbms 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. The rdb palette provides history-related components.

  In addition to being available as a **`Type [of network] to add`** when you double-click the **Driver** node in the Nav tree, and click **New** at the bottom of the **Driver Manager** view, the driver is also located on each of the individual **`rdb`** database module palettes.

- 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.

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.
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)
rdb	HistoryUnicodeUpdater and HistoryTimezoneUpdater	rdb.jar	Defines database ords used to configure exported data.

Palette name	Network component(s) in the Nav tree	.jar file name in the modules folder	Function
rdbHsqlDb (optional)	RdbmsNetwork, RdbmsFolder, HsqlDbDatabase	rdbHsqlDb.jar	Supports an open source relational database management system written in Java. (source: Wikipedia)
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)
rdbOracle (optional)	RdbmsNetwork, RdbmsFolder, OracleDatabase	rdbOracle.jar	Supports a multi-model database management system produced and marketed by Oracle Corporation. (source: Wikipedia)
rdbSqlServer (optional)	RdbmsNetwork, RdbmsFolder, SqlServerDatabase	rdbSqlServer.jar	Supports a relational database management system developed by Microsoft. (source: Wikipedia)

Database requirement

The Rdbms driver connects Niagara stations to an RDBMS (Relational Database Management System) 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.

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 4.4 or later and Workbench running on a PC
- A Niagara license for a specific database type in the license file
- A relational database installed on the Supervisor PC: SqlServer, Oracle, MySQL, or HsqlDb. Embedded (remote) controllers support only the HsqlDbDatabase database.
- An IP network connection to the database host
- Appropriate rights for the required database access
- Secure communication with certificates configured for the database. While secure communication can be disabled, the best practice is to always enable and implement secure connections that both encrypt and authenticate the database server (TLS/SSL).

You must know:

- The user name and password to log in to the database
- If you are trying to connect to a named instance of a database, the name of the database instance
- If the database is using a non-default port number, the port number so you can configure your rdb database device correctly
- The platform/station Trust store requires the root CA certificate used to sign the server certificate presented by the database server.

Installing the network and database

The network connects to 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.

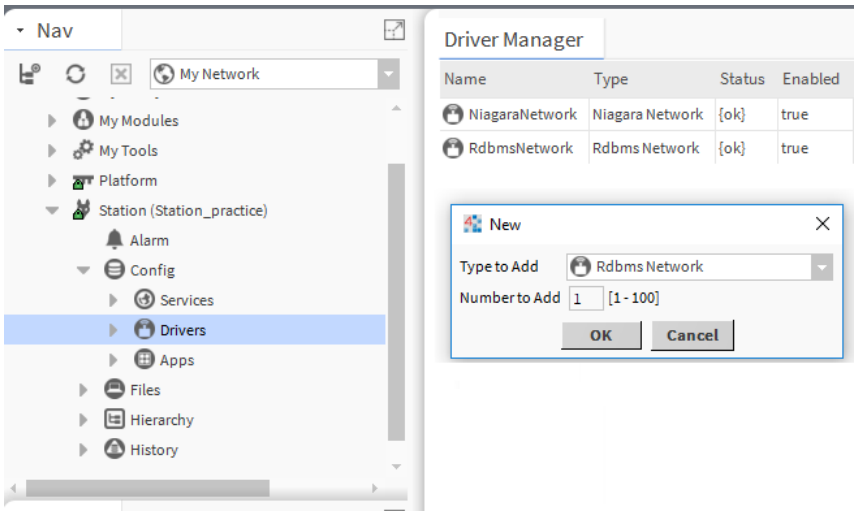
Prerequisites: You are working in Workbench running on a PC that is connected to the device network.

Step 1 Double-click the station's **Drivers** container, to bring up.

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, double-click the **RdbmsNetwork** you just created, click **New** at the bottom of the **Device Manager** view, select the number of databases (usually just one), and click **Ok**.

The window to name the database opens.

Step 6 To add the database, click **Ok**.

You should have an database under your **RdbmsNetwork** node in the Nav tree.

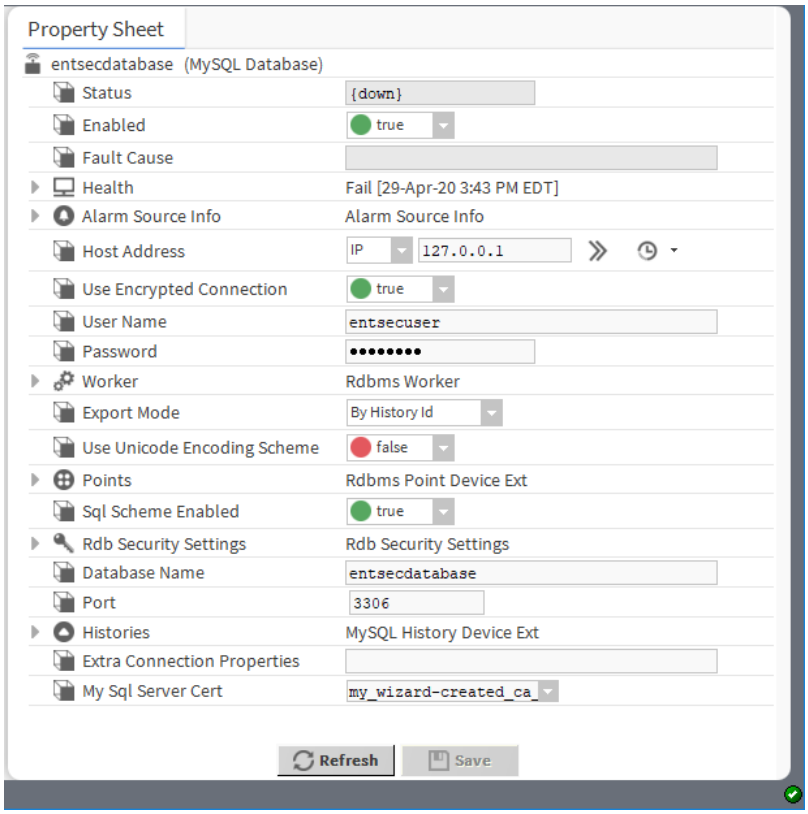
Configuring Supervisor database properties

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, configured it, it supports secure communication and it 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.roperties



Above is an example using the MySQLDatabase Property Sheet.

Step 2 Select `true` for **`Sql Scheme Enabled`**.

Step 3 Set values for the other properties applicable to the specific database.

The *Properties Dictionary* in this guide documents all properties. The following table indicates which properties are available based on the rdbDatabase device.

Table 1 RdbmsNetwork database properties

Property	HsqlDb	MySQL	Oracle	SqlServer
Status (Information)				
Enabled	X	X	X	X
Fault Cause (Information)				
Health (Information)				
Alarm Source Info	X	X	X	X
Host Address	-	X	X	X
User Name	X	X	X	X
Password	X	X	X	X
Worker	X	X	X	X
Export Mode	X	X	X	X
Use Unicode Encoding Scheme	X	X	X	X
Timestamp Storage	X	-	X	X

Property	HsqlDb	MySQL	Oracle	SqlServer
Points	X	X	X	X
Sql Scheme Enabled	X	X	X	X
Base Directory	X	-	-	-
Database Name	X	X	-	-
Instance Name	-	-	-	X
Service Name	-	-	X	-
Port	-	X	X	X
Histories	-	X	X	X
Extra Connection Properties	-	X	-	X
Version	-	-	-	X

Step 4 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`. It 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 5 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 6 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 7 Define a name for the database. To complete the configuration, click **Save**.

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.

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`**. 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.

Replacing the standard alarm database

The AlarmService manages the standard alarm database. Instead, You can configure an RDBMS (SQL Server, Oracle, or MySQL) to replace a station's standard alarm database.

Use of an RDBMS to store alarm records differs from exporting histories to an RDBMS. Exported histories reside in the station's default history database (`^\history\segN\(etc)`), whereas, configuring for RDBMS storage of alarms replaces use of the station's default alarm database `(^\alarm\alarm.adb)`.

This procedure describes how to set up the RdbAlarmService.

Step 1 Install and configure the RdbmsNetwork driver and test the connection to your desired database.

Step 2 Open the **`alarmRdb`** palette and copy the **rdbAlarmService** component into the station's **Services** folder.

Step 3 To retain child components of any previous (standard) **AlarmService** component (AlarmClasses, Recipients), cut them from the old **AlarmService** component and paste them into the new **RdbAlarmService** component before deleting the original **AlarmService**.

Step 4 Delete the old (standard) **AlarmService** component from the station's **Services** folder.

NOTE: The driver does not migrate existing alarms to the new data store.

Step 5 Right-click on the **rdbAlarmService** in the Nav tree and click **View→Property Sheet** .

The rdbAlarmService **Property Sheet** view opens.

Step 6 Select the `rdbms database device type` from the **`Driver`** option list.

Only those RDBMS database devices that are located under the station's **RdbmsNetwork** node appear as options in the list.

Step 7 To complete the configuration, click **Save**.

The new **RdbAlarmService** is available for use.

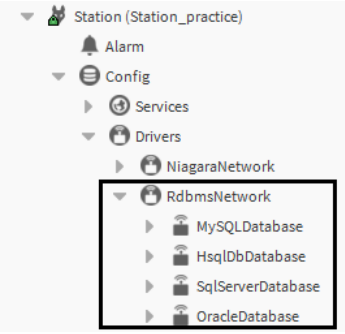
Chapter 2 Data management

Topics covered in this chapter

- Discovering and adding points
- Adding and configuring a new point query
- Editing an existing query
- Rdbms implementation example
- Exporting history data to an Rdbms database
- Unix time conversion for MySQL
- Importing history data from an Rdbms database
- 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 2 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.

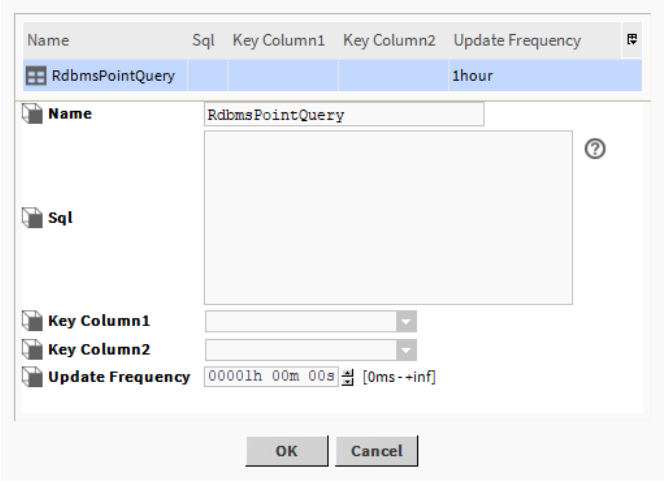
The **New** window opens.

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 3 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 Configure any 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.

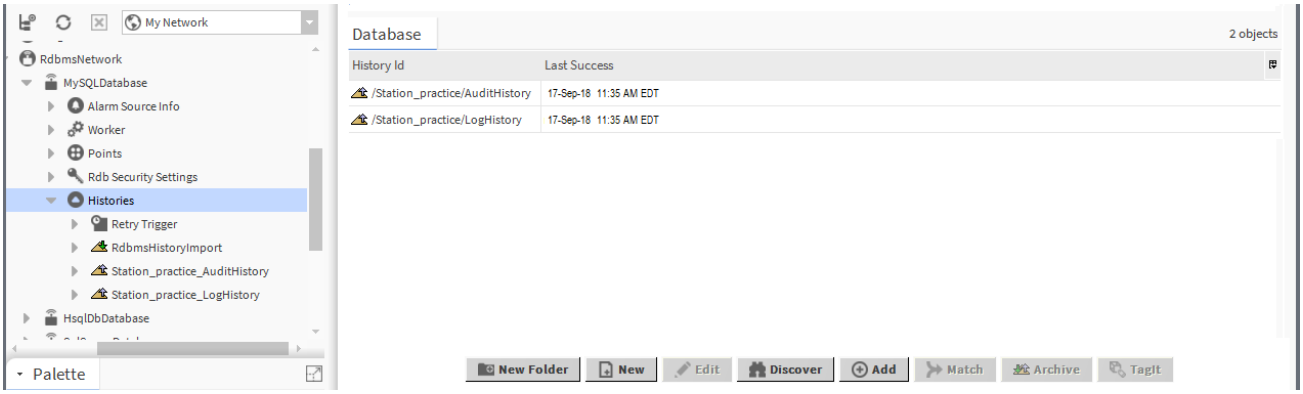
Rdbms implementation example

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, archiving store records for three types of fuel to the database every 15 minutes. To graphically display updated information over the Internet, they use 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.

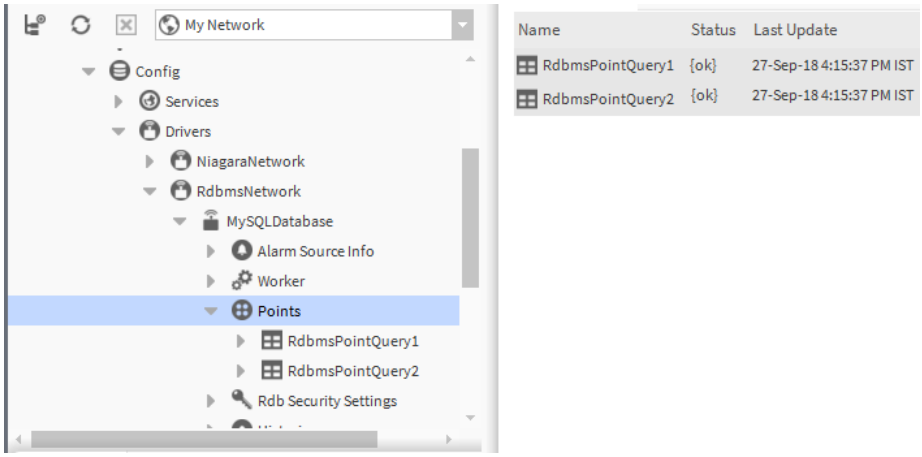
Figure 4 Rdbms histories



NOTE: Embedded controllers support only the HsqlDbDatabase.

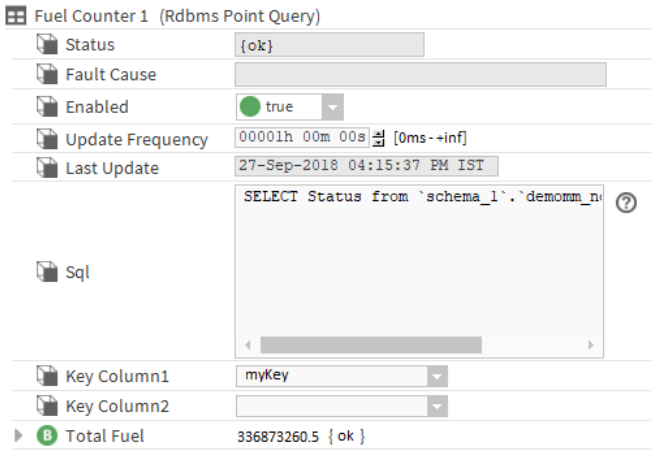
2. For each store, a Supervisor station creates proxy points for each fuel type, by creating and configuring an Rdbms Point Query for each fuel type.

Figure 5 Fuel points



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

Figure 6 Fuel counter query



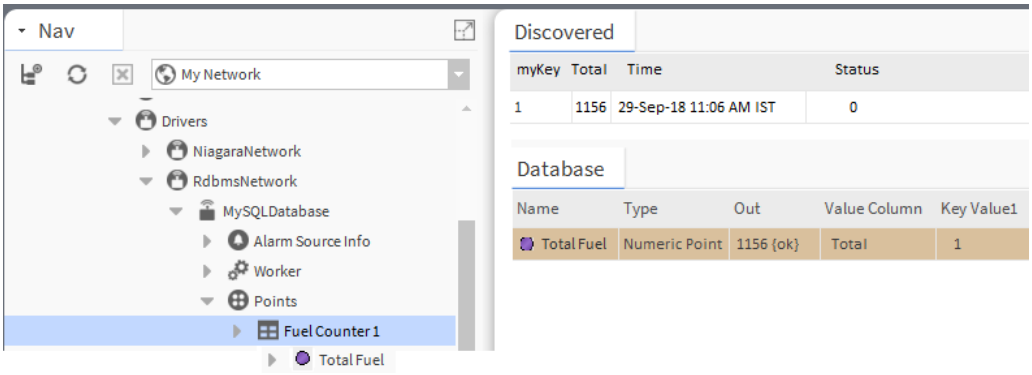
This counter is scheduled to update every 15 minutes, providing a total that updates every 15 minutes along with the time of the update.

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.

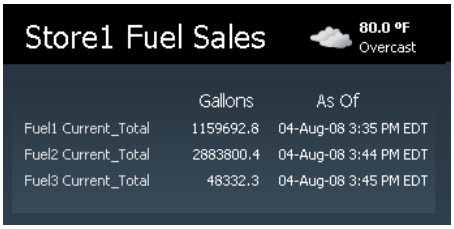
4. Data are discovered and added under the **Rdbms Point Device Extension** node.

Figure 7 Rdbms Point Device Extension Manager view



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.

Figure 8 Graphical display of total gallons sold



Exporting history data to an Rdbms database

Each Rdbms Device extension has an associated History Device Extension, which you can configure and use to export data to an Rdbms database (MySQL, Oracle, and SqlServer) and framework history file.

Prerequisites: You are working in Workbench in a Supervisor station with an established connection to a database.

The driver supports two database schema, which provide data export by ID or by Type. If the Database Administrator (DBA) did not manually create each schema, the RdbmsNetwork driver creates them automatically as it exports each history.

Your DBA 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 Database refers to any of the database types.

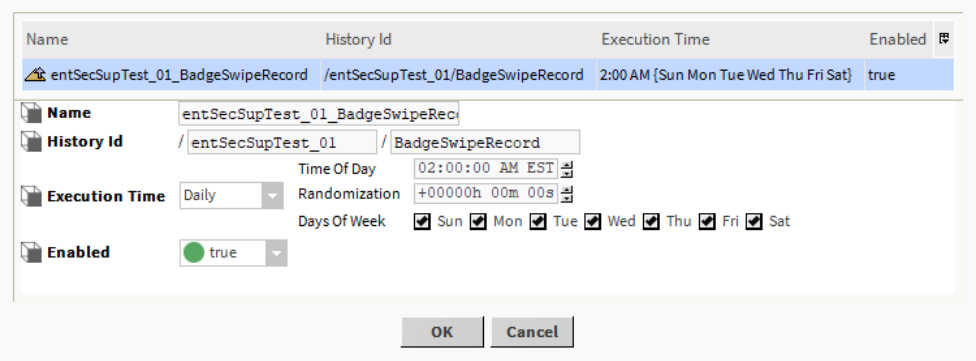
Step 1 Expand **Drivers→RdbmsNetwork→rdb Database**, and double-click the **Histories** node in the Nav tree.

The **History Export Manager** view opens.

Step 2 Click the **New** button

As an alternative to the **New** button and window, you can click the **Discover** button and browse to find and select the history file to export.

The **New** window opens.



Step 3 Edit the **`Name`**, **`History Id`**, **`Execution Time`**, and **`Enabled`** properties, and click **Ok**.

The new history export descriptor(s) appears in the **Database** (lower) pane.

Step 4 Do one of the following to initiate an export action:

- 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 **rdbmsDatabase 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}		

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

Importing history data from an Rdbms database

Each of the following Rdbms device extensions has an associated history device extension, which you can configure and use to import data to a station from an Rdbms database and framework history file: MySQL, Oracle, and SqlServer. In the following steps, the term rdb database refers to any of these database types.

Step 1 Expand **Drivers→RdbmsNetwork**, expand your rdb database, 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 Click the **New** button

As an alternative to the **New** button and window, you can click the **Discover** button and browse to find and select the history file to import.

The **New** window opens.

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

You need to add one unique import descriptor for each history file to import.

Step 4 To continue, click **Ok**.

A second **New** window opens.

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

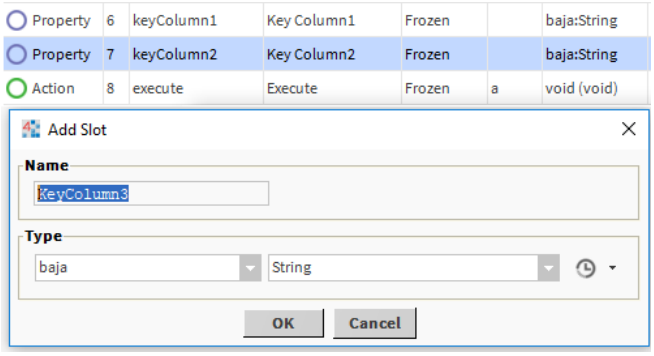
If you leave the key columns blank, the driver automatically uses the first column in the row as the primary key.

`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 may 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 import properties, and click **Ok**.

The new history import descriptor(s) appears in the **Database** (lower) pane.

Step 8 Do one of the following to initiate an import action:

- 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 import to occur, as scheduled.

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

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
♦ Properties dictionary

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

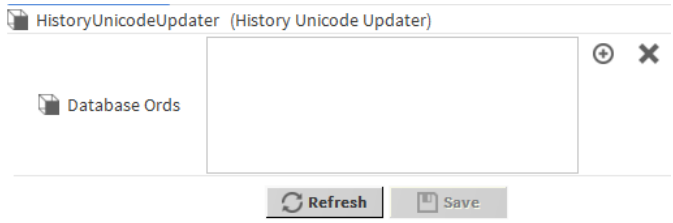
This module provides components for updating RDBMS histories.

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.

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



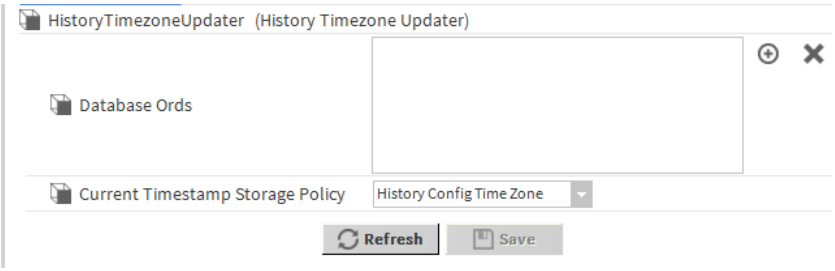
You access this property by double-clicking the **HistoryUnicodeUpdater** node in the Nav tree.

Individual properties are documented in the *Properties Dictionary* (in this guide).

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.

Individual properties are documented in the *Properties Dictionary* (in this guide).

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:

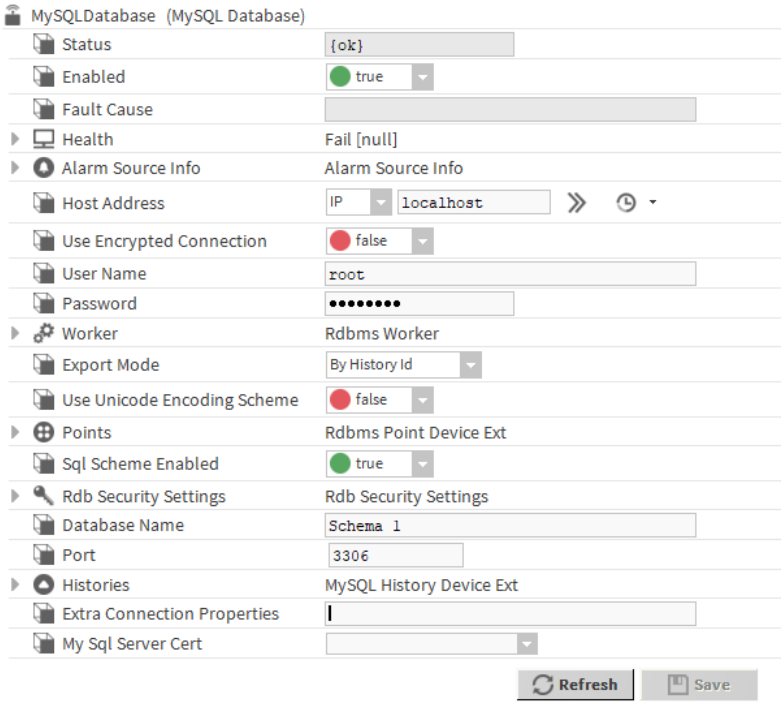
- rdbHsqlDb
- rdbMySQL
- rdbOracle
- rdbSqlDerver

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.

The RdbmsNetwork **Property Sheet** view and the NiagaraNetwork property sheet share common properties described in the *Niagara Drivers Guide*.

Figure 19 Example of an rdb Database Property Sheet



You access this view by expanding the **Drivers→RdbmsNetwork**, right-clicking your rdb Database and clicking **Views→Property Sheet**.

Individual properties are documented in the *Properties Dictionary* (in this guide).

RdbmsFolder

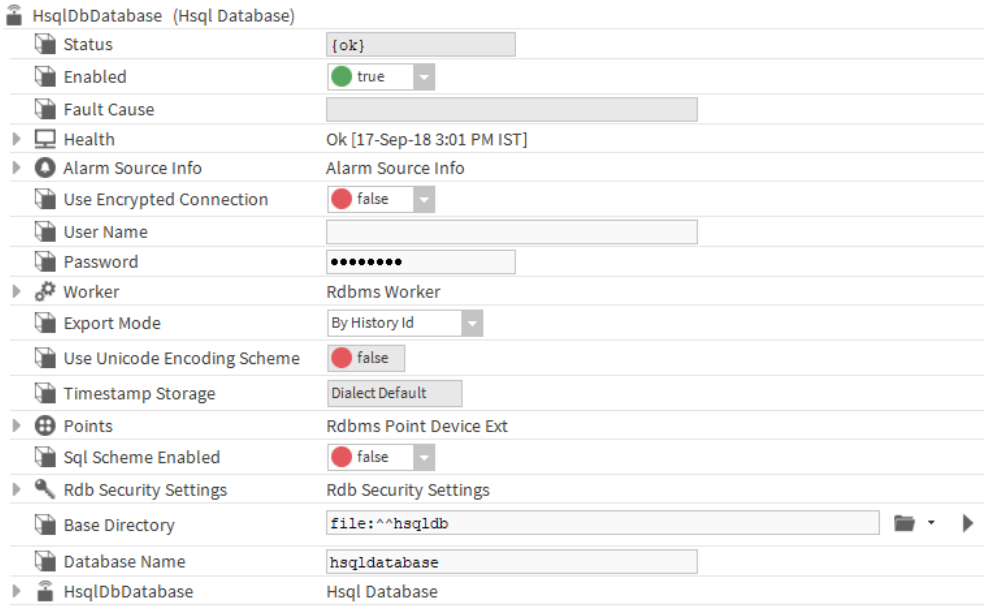
This component is used to organize databases under an **RdbmsNetwork**.

You create this folder using the **New Folder** button in the **Device Manager** view for any **RdbmsNetwork**.

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 framework's HsqlDatabase component.

Figure 20 HsqlDatabase Property Sheet



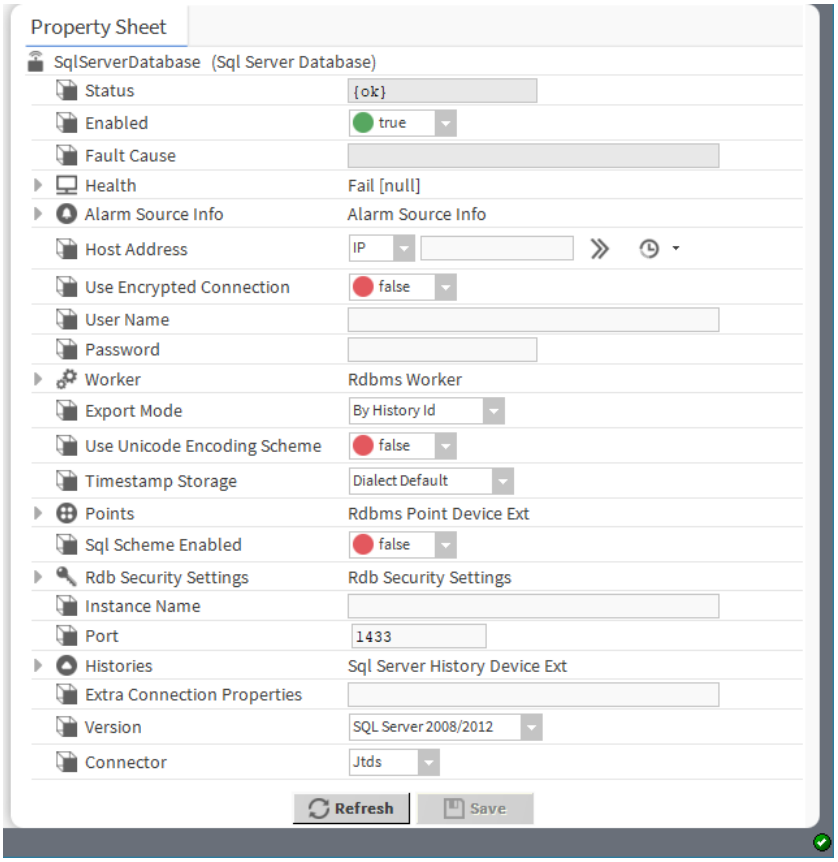
You access the properties to configure the driver for this database by expanding **Drivers→RdbmsNetwork**, and double-clicking the **HsqlDatabase** node in the Nav tree.

Individual properties are documented in the *Properties Dictionary* (in this guide).

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.

Figure 21 MySQLDatabase Property Sheet



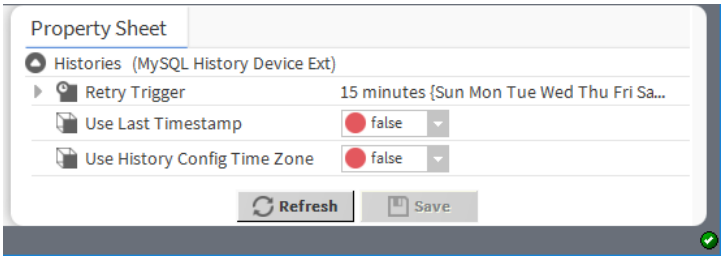
You access the properties to configure the driver for this database by expanding **Drivers→RdbmsNetwork**, and double-clicking the **MySQLDatabase** node in the Nav tree.

Individual properties are documented in the *Properties Dictionary* (in this guide).

MySQL Histories

This component is the MySQL implementation of HistoryDeviceExt. It is a child of a **MySQLDatabase** that configures when to create a history and what time information to add to each history record..

Figure 22 MySQL History Property Sheet



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

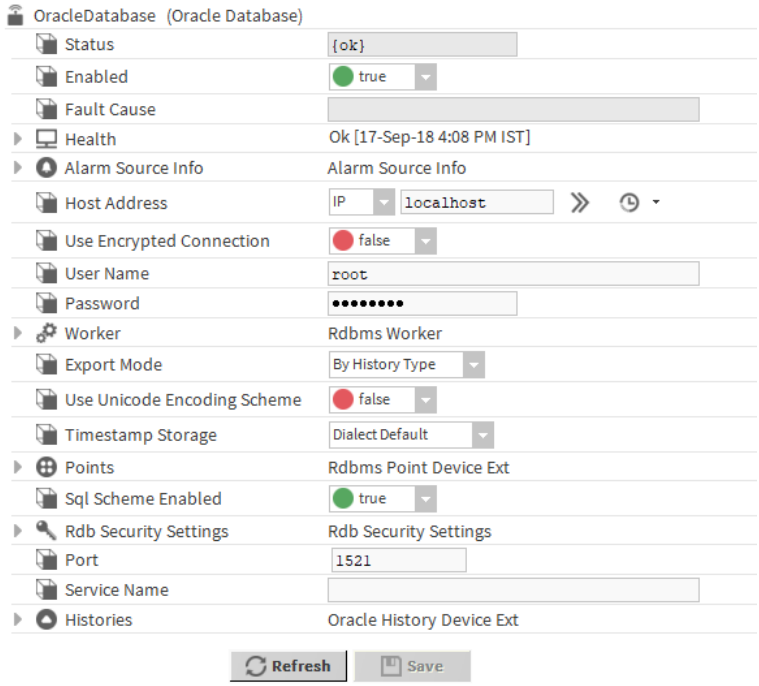
This container itself contains the **`Retry Trigger`** container.

Individual properties are documented in the *Properties Dictionary* (in this guide).

OracleDatabase

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

Figure 23 Oracle Database Property Sheet



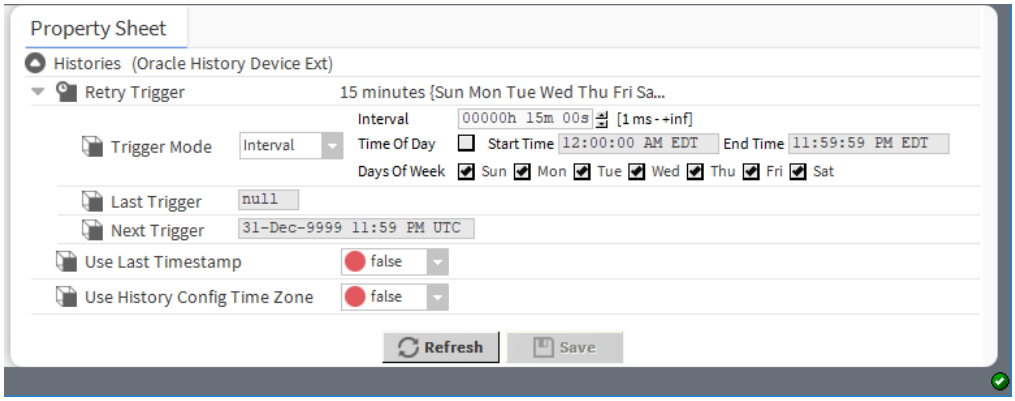
You access the properties to configure the driver for this database by expanding **Drivers→RdbmsNetwork**, and double-clicking the **OracleDatabase** node in the Nav tree.

Individual properties are documented in the *Properties Dictionary* (in this guide).

OracleHistoryDeviceExt

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

Figure 24 Oracle history device extension properties



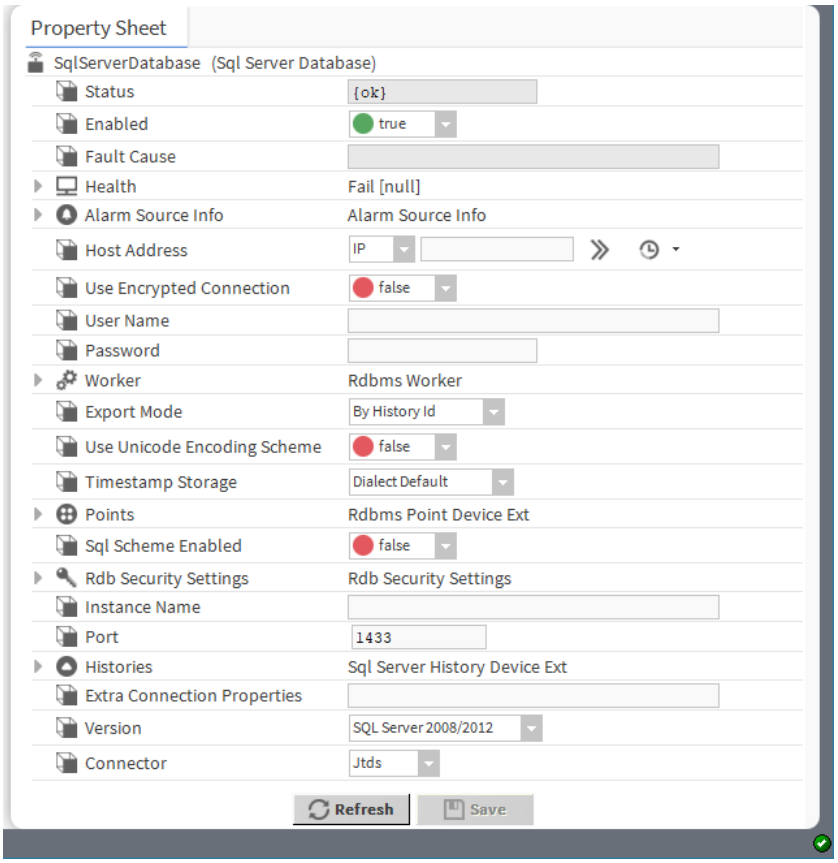
The History Device Extension is fully described in the *Niagara Drivers Guide*. It is not available on the HsqlDbDatabase device since this database may not be used to import or export histories.

Individual properties are documented in the *Properties Dictionary* (in this guide).

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 25 SqlServerDatabase Property Sheet



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

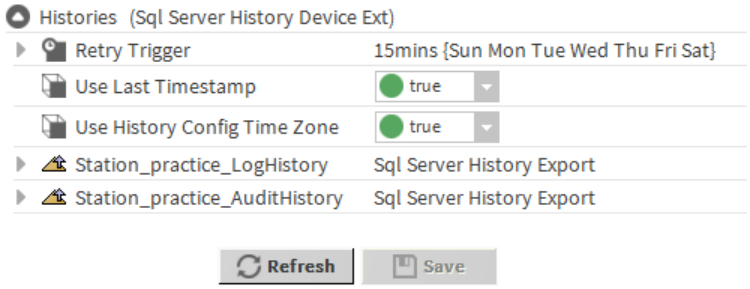
The Rdbms driver does not support Windows Authentication alone. You must have selected SQL Server Authentication (mixed mode) for any user of this database. This is an SQL Server login property, not a Niagara property.

Individual properties are documented in the *Properties Dictionary* (in this guide).

SqlServerHistoryDeviceExt

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

Figure 26 SqlServer History Property Sheet



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

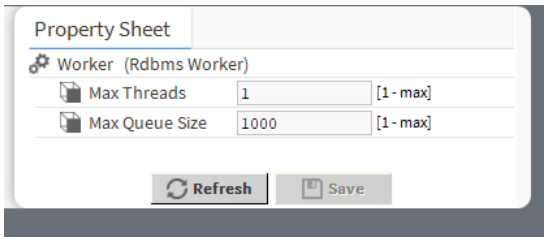
This container itself contains the **`Retry Trigger`** container.

Individual properties are documented in the *Properties Dictionary* (in this guide).

Worker container

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

Figure 27 Worker Property Sheet



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

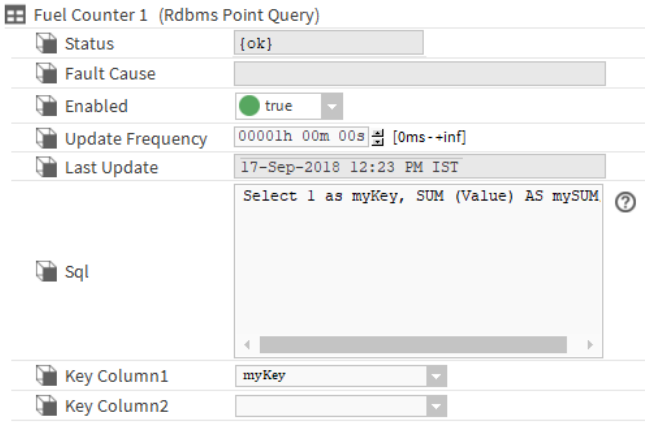
This component provides two properties for setting the maximum number of concurrent threads and for setting the maximum allowable queue size for the database connection. These worker properties are not pooled at the Network level, as with the NiagaraNetwork driver. Each database driver has its own thread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

Individual properties are documented in the *Properties Dictionary* (in this guide).

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.

Figure 28 Rdbms Point Query properties



You access these properties by expanding the **Drivers→RdbmsNetwork** node in the Nav tree, followed by expanding your rdbDatabase and the **Points** folder it contains, right-click the **RdbmsPointQuery** node, and click **Views→Property Sheet**.

The **`Sql`** property is a text editor used to create and display an entry for each Rdbms Point 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.

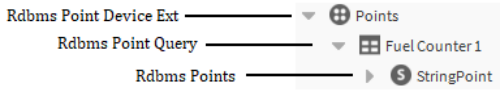
General information

With this component you can represent relational database cell values as proxy points. The RdbmsNetwork Driver requires no configuration. For more information, refer to "Using the Rdbms Point Device Extension", and "About the Points Extension" in the *Niagara Drivers Guide*.

The rdbDatabase 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 in Niagara, as shown below.

Figure 29 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)

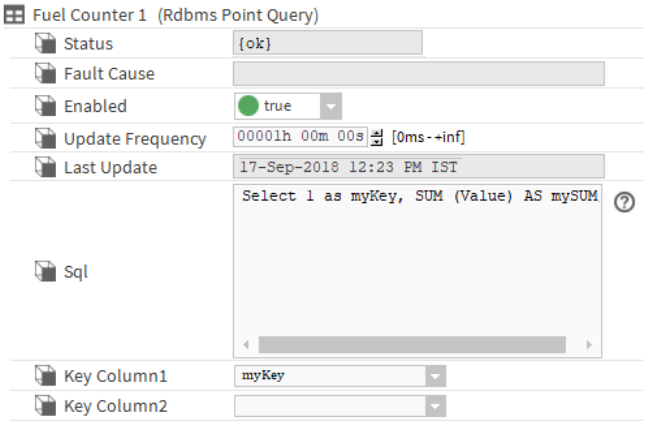
Unlike other Point Device Extensions, the Rdbms Point Device Extension:

- 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.

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 30 RdbmsPointQuery Property Sheet



You access these properties by expanding **Drivers→RdbmsNetwork** followed by expanding your rdb Database node, expanding the **Points** folder, right-clicking an RdbmsPointQuery folder and clicking **Views→Property Sheet**.

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.

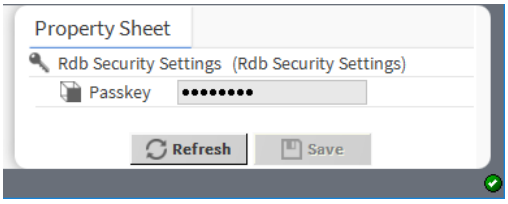
The **`Sql`** property contains the query statement. You can execute this query manually and also set a regular interval time for updates using the **`Update Frequency`** property.

Individual properties are documented in the *Properties Dictionary* (in this guide).

Rdb Security Settings

This property configures security settings for the relational database.

Figure 31 Rdb Security Settings properties



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

Individual properties are documented in the *Properties Dictionary* (in this guide).

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 (before Niagara 4.6) 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

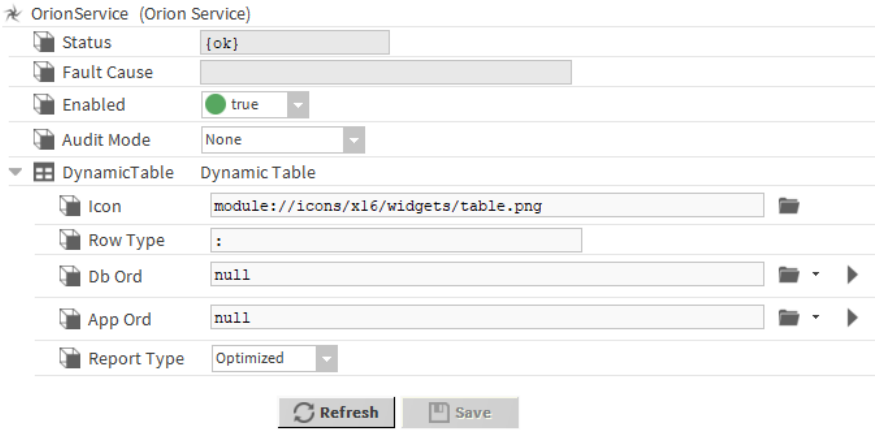
This is a CRUD (Create, Read, Update and Delete) interface for interacting with the Orion database and managing object persistence.

OrionService

This component under the Service 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.

Figure 32 Orion Service Property Sheet



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

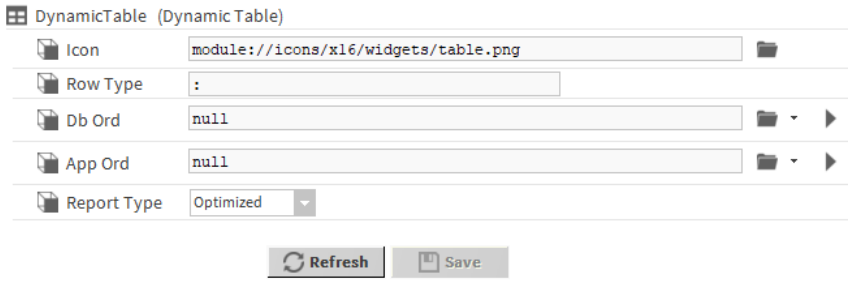
Orion is an Object-Relational (O/R) mapping architecture provided to support distributed-applications, large systems, and other applications that may benefit from having relational data modeled as framework objects. Object-Relational Mapping does not replace the config.bog (station) file, but provides 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.

Individual properties are documented in the *Properties Dictionary* (in this guide).

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.

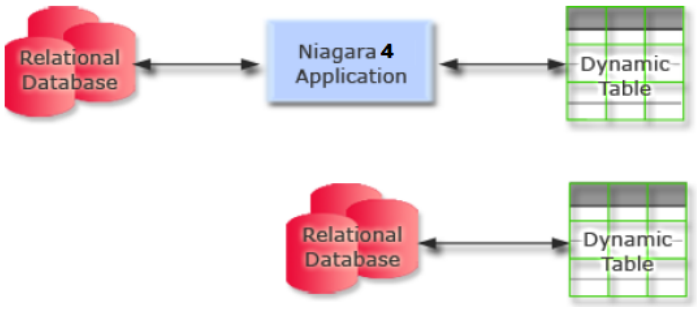
Figure 33 Dynamic Table properties



You drag a **DynamicTable** component from the **orion** palette to a location 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.

These properties are documented in the *Properties Dictionary* in this guide.

Figure 34 Rdbms model with dynamic tables



Individual properties are documented in the *Properties Dictionary* (in this guide).

FoxOrionDatabase

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

OrionModule

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

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.

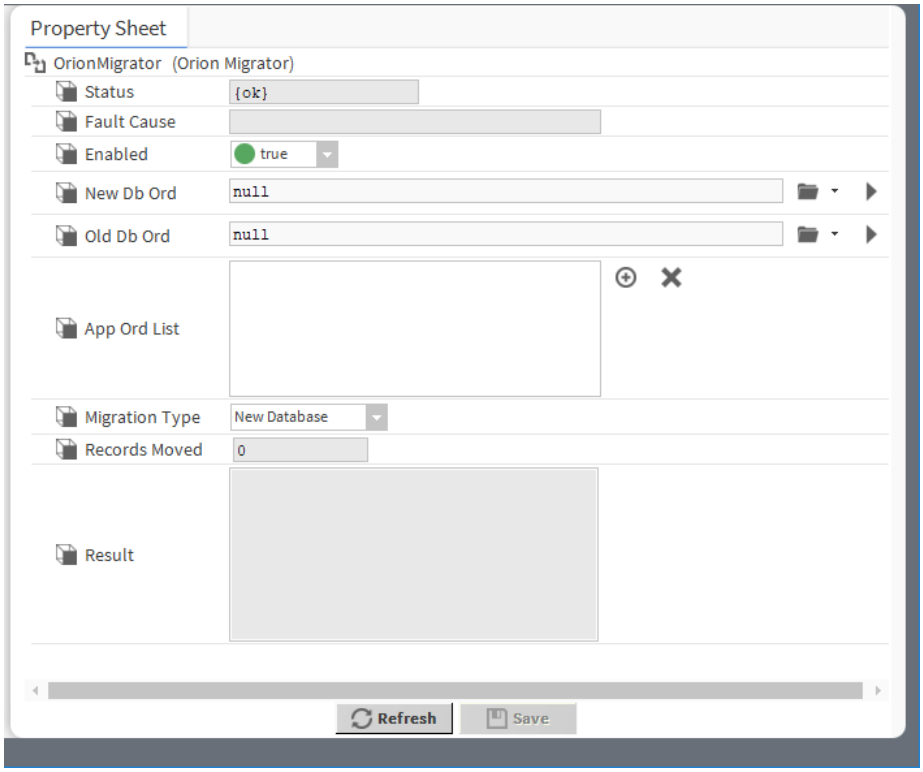
OrionType

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.

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.

Figure 35 Orion migrator properties



Individual properties are documented in the *Properties Dictionary* (in this guide).

Properties dictionary

This topic documents the properties used to configure database and other components.

Property and where to find it	Value	Description
App Ord (right-click DynamicTable in the Nav tree and click **Views→AX Property Sheet**)	ord	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.
App Ord List (Orion Migrator)	text	Lists application locations.

Audit Mode	drop-down list (defaults to `None`)	Configures the type of audit.
		`Audit Event`
		`Database Record`
		`Both`

Base Directory (HsqlDb device only)	file path	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”, a path to the folder is specified as `file:^hsqldb`.

Property and where to find it	Value	Description
Capacity (New import histories window)	drop-down list	Configures the number of records to import. `Unlimited` places no limit on the size of the import. `Record Count` opens a property to define the number of records (defaults to 500).
Connector (SqlServerDatabase only)	drop-down list (defaults to `Jtds`)	Selects a JDBC (Java Data Base Connectivity) driver to connect the framework to a Microsoft SQL Server database. `Jtds` selects the jTDS (Java Tabular Data Stream) open-source driver. `Mssql` selects the JDBC driver provided by Microsoft.
Current Timestamp Storage Policy (HistoryTimezoneUpdater)	drop-down list	Selects the policy that defines the time zone: `Station Time Zone` or `History Config Time Zone`.
Database Name (HsqlDb and MySQL Database devices only)	text	Defines the name of the MySQL database.
Database Ords (HistoryUnicodeUpdater, HistoryTimezoneUpdater)	ord	Specifies an ord path to the appropriate updater(s).
Days of the Week	radio buttons	Selects specific days of the week to perform a task, such as sampling data for histories.
Db Ord (right-click DynamicTable in the Nav tree and click **Views→Ax Property Sheet**)	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 **OrionService** installed, you can use a chooser to select the database under the **DynamicTable** node.
Enabled	`true` or `false`	Activates (`true`) and deactivates (`false`) use of the network, device, point and component.
Execution Time (New import histories window)	drop-down list and additional time properties	Configures when to import the history.
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.
Extra Connection Properties (MySQL and SqlServer devices only)	normally left blank; if used takes the format: parameter1=value1;	Additional parameters, which not required for a normal connection to the database. The driver passes this information in plain text to the database. A description of these parameters is beyond the scope of this document, however, they can be

Property and where to find it	Value	Description
	parameter2=value2;...etc.	used for advanced diagnostics or tuning under the guidance of a support channel.
Fault Cause	read-only	Indicates the reason why a system object (network, device, component, extension, etc.) is not working properly (in fault). This property is empty unless a fault exists.
Full Import On Execute (New import histories window)	drop-down list	Specifies to import either the full database (up to the specified limit) or only the new data on each successive import action.
Full Policy (New import histories window)	drop-down list (defaults to `Roll`)	Determines what happens when the station database reaches the maximum number of records it can hold. `Roll` overwrites the oldest records with the new ones. `Stop` terminates the import job.
History Id (New import histories window)	two text boxes	Specifies the station name and history name into these two fields.
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.
Icon (right-click DynamicTable in the Nav tree and click **Views→AX Property Sheet**)	file path	Designates a graphic icon to associate with a dynamic table.
Instance Name (SqlServerDatabase device only)	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.
Interval (New import histories window)	drop-down list (defaults to `irregular`)	Configures when the system creates a history record. `irregular` creates records randomly. `regular` opens an additional property for selecting a regular interval.
Key Column 1 (New Rdbms Point Query window)	drop-down list	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 (New Rdbms Point Query window) (optional)	drop-down list	Augments **`Key Column 1`** when **`Key Column 1`** is not enough to establish a unique key for each imported data record.

Property and where to find it	Value	Description
Last Update	date and time	Indicates the last time the driver executed the query.
Max Queue Size (database device components, Worker container)	number (defaults to 1000)	Specifies the maximum queue size supported for Rdbms actions specific to the associated database driver (that is, exports, imports). You adjust to reflect the peak number of import or export records per archive that are expected to be processed; normally increased to handle large volumes of data.
Max Threads (database device components, Worker container)	number	Specifies the maximum number of multiple, concurrent connections (threads) that the station makes with the external database. The default value is one thread. 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.

Migration Type (Orion Migrator)	drop-down list (defaults to `New Database`)	Controls what the migrator does.
		`New Database` creates a new database.
		`Insert Only`
		`Overwrite`
		`Force Overwrite`

Name (New import histories window, New Rdbms Point Query window)	text	Provides a name for the import task or point query.
My Sql Server Cert	drop-down list	Defines the database server certificate for TLS secure communication.
New Db Ord (Orion Migrator)	ORD	Identifies the location of the database that supports the UTF-8 Unicode Encoding Scheme.
Old Db Ord (Orion Migrator)	ORD	Identifies a location of the database that does not provide UTF-8 support.
Passkey	text	Defines a strong password that controls access to the database.
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.
Port	number, default value depends on the rdb Database type: HsqlDbDatabase, no port specified because this rdb is for local database use only; MySQLDatabase, 3306; OracleDatabase, 1521; and	Specifies the port number to use when connecting with the database.

Property and where to find it	Value	Description
	SqlServerDatabase, 1433	
Query Predicate (New import histories window)	text	Filters the records that are imported
Rdb Catalog Name (New import histories window)	text	Redefines the catalog name on import.
Rdb Schema Name (New import histories window)	text	Redefine the schema name on import.
Rdb Table Name (New import histories window)	text	Redefines the table name on import.
Records Moved (Orion Migrator)	read-only	Reports the number of records migrated from the old to new databases.
Report Type (Orion Dynamic Table)	drop-down list (defaults to Optimized)	Configures the type of dynamic table to display. `Optimized` displays recent records. `Full Report` displays all records.
Result (Orion Migrator)	read-only	Provides a log of the migration process.
Row Type (right-click DynamicTable in the Nav tree and click **Views→Ax Property Sheet**)	`module:type` (on the property sheet); a drop-down list in the **Dynamic Table Config** view.	Specifies the DynamicTable row in terms of `module` and `type`.
Service Name (OracleDatabase device only)	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.
Sql (New Rdbms Point Query window)	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.
Sql Scheme Enabled (database device components)	`true` or `false` (default)	Turns the Sql scheme on and off. This is a standard Points extension.

Property and where to find it	Value	Description
		`true` is required to execute queries against the relevant database, and as a prerequisite for using the Rdbms Point Device Extension. `false` indicates you are not executing queries using the Rdbms Point Device Extension, or loading data from Oracle. Good security practice sets the property to `false` **NOTE:** To use an rdbDatabase, you must set this property to `true` (this property is located on the RdbmsNetwork Driver Device **Property Sheet**.
Station Time Zone (HistoryTimezoneUpdater)	hours	Defines the time zone of the station to update the history times.
Status	read-only	Indicates the condition of the network, device or component at the last check. `{ok}` indicates that the component is licensed and polling successfully. `{down}` indicates that the last check was unsuccessful, perhaps because of an incorrect property, or possibly loss of network connection. `{disabled}` indicates that the **`Enable`** property is set to `false`. `{fault}` indicates another problem. Refer to **`Fault Cause`** for more information.
Status Column (New import histories window)	text	Specifies if a status column is included in the imported data or not.
Time Of Day	hour	Defines the beginning and end times during the day to collect sample data.
Timestamp Column (New import histories window)	text	Specifies the Timestamp column for the imported data.
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.

Property and where to find it	Value	Description
		`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.
Timezone (New import histories window)	drop-down list	Configures the local time zone.
Trigger Mode	drop-down list (defaults to `Interval`)	Defines the method used to sample periodic data. `Manual` requires an action to sample data. `Daily` samples every day at the same time. `Interval` samples at regular times throughout the day.
Update Frequency (New Rdbms Point Query window)	hours, minutes, seconds	Dictates how often the driver automatically executes the query.
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.
Use History Config Time Zone (OracleDatabase 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`**
Use Last Timestamp (OracleDatabase device only)	`true` or `false` (default)	Enables and disables the reporting of the last date and time reported on a history record. `false` queries the database for the last timestamp in the database every time an export descriptor processes. 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.
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

Property and where to find it	Value	Description
		type) of these commands: CREATE TABLE, CREATE INDEX, CREATE SEQUENCE
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.
Value Column (New import histories window)	text	Specifies the Value column for the imported data.
Value Facets (New import histories window)	Config Facets chooser	Configures units.
Version (SqlServer device only)	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.

Chapter 4 Plugins (views)

Topics covered in this chapter

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

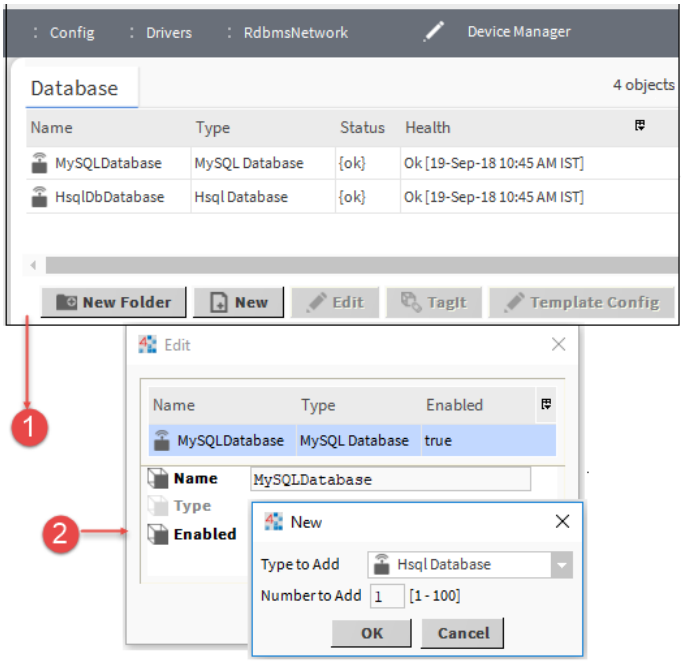
Plugins provide views of components and can be accessed in many ways. For example, double-click a component in the Nav tree to see its default view. In addition, you can right-click on a component and select from its **Views** menu.

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

Device Manager view

This view manages the rdb Database devices.

Figure 36 Device Manager view with Add and Edit windows



1- RDBMS Device manager View
2- Add and Edit window

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

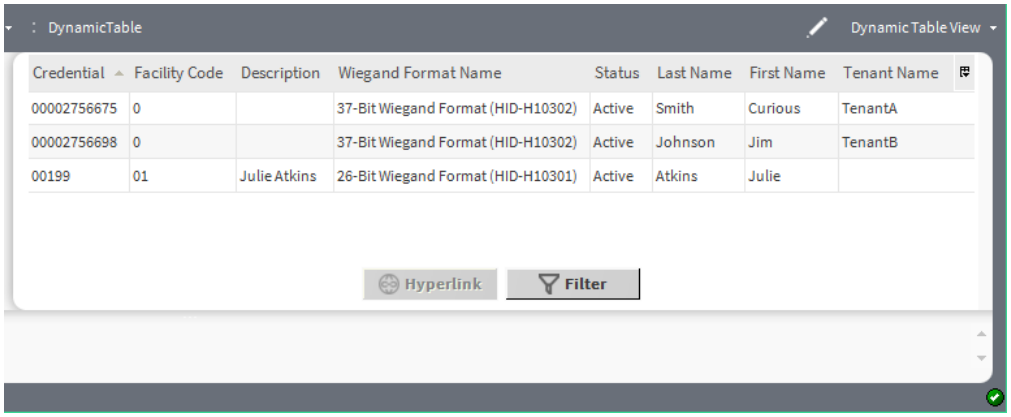
The **New** and **Edit** buttons open **New** and **Edit** windows, which add, configure and monitor rdb Database devices.

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 37 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.

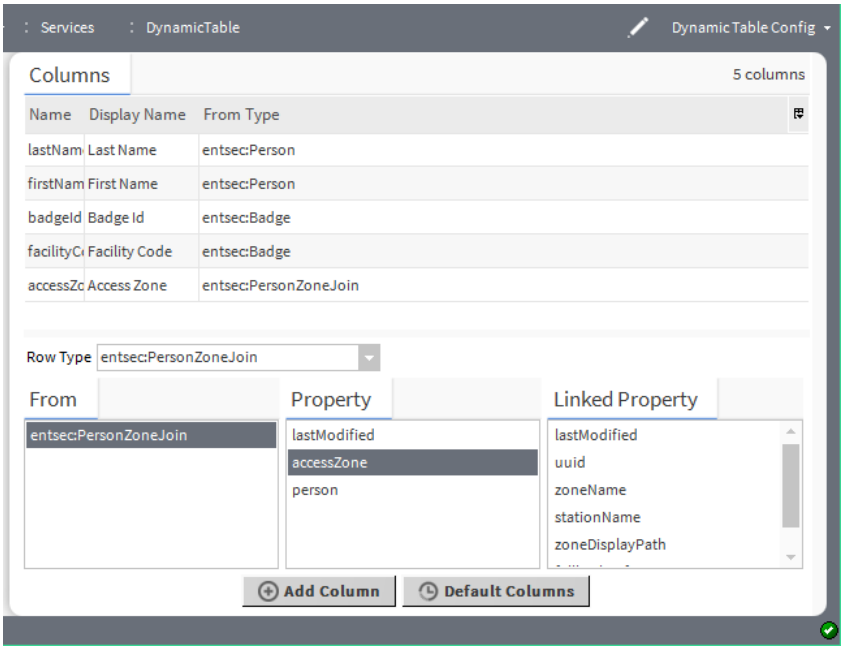
Columns

Column Name	Description
Credential	Displays the required identifier.
Facility Code	Identifies the geographic location.
Description	Report additional information.
Wiegand Format Name	Identifies the wiring standard for the 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 the columns on a dynamic table view.

Figure 38 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** view 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.

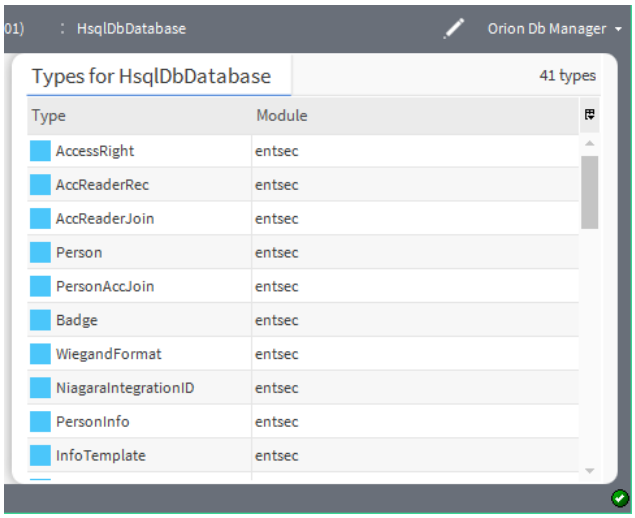
Buttons

Column name	Description
Add Column	Allows you to add new specific column.
Default Column	Adds column based on the selection of row type.

Orion Db Manager view

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 39 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).

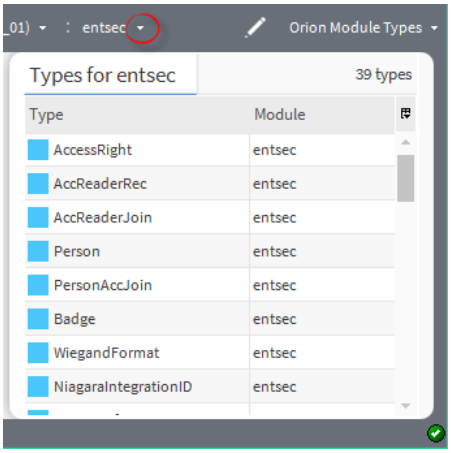
Columns

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 40 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.

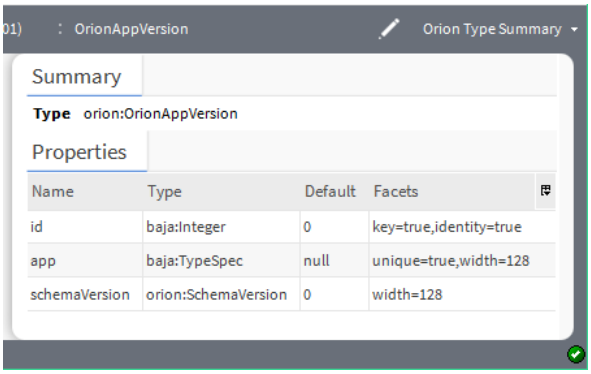
Columns

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 41 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.

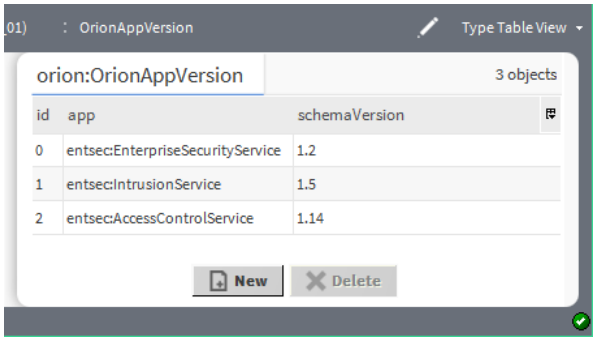
Columns

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

Orion Type Table View

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

Figure 42 Orion Type Table view



This view shows a table of Orion application records.

Columns

Column name	Description
id	Displays the ID assigned to the app.
app	Displays the module name and service used.
Schema Version	Displays the version of the schema used.

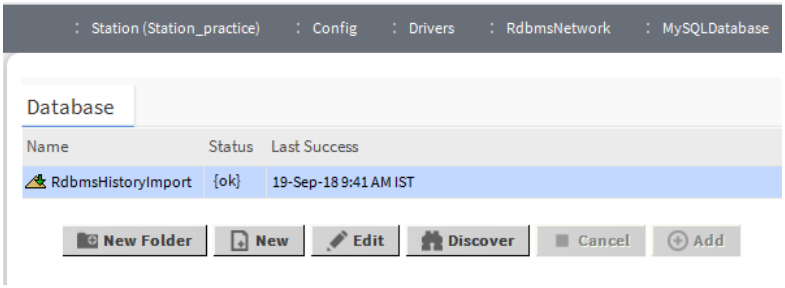
Buttons

Button name	Description
New	Allows to create new app version.
Delete	Deleted the selected app version.

Rdbms History Import Manager view

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

Figure 43 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

- **Name** identifies the type of History.
- **History ID** refers the History Id.
- **Execution Time** displays the last execution time.
- **Enabled** displays the status.
- **Status** displays history exported status.
- **State** displays current state of the database
- **Last Success** Date and time of last history exported.
- **Last failure** displays the last failure for the discovery of the history.
- **Fault Cause** shows the reason of last fail cause of history.
- **Capacity** displays the capacity to log the history
- **Full Policy** allows the policy to roll or stop the history to be logged.
- **Interval** displays the interval in which the history is logged.
- **Value Facets** displays the units.
- **Time Zone** displays the current time zone.

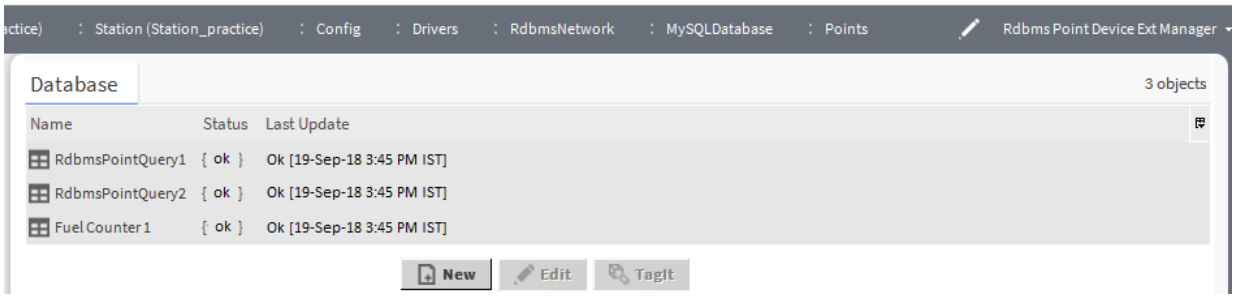
Buttons

- **New Folder** adds a folder for organizing multiple devices in the Database view.
- **New** adds new History type.
- **Edit** edits the history type.
- **Discover** discovers the already added histories.
- **Add** adds the discovered histories.
- **Match** matches the discovered histories and the already present histories.
- **Archive** archives the history types.

Point Device Ext Manager view

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 44 Point Device Ext Manager view



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

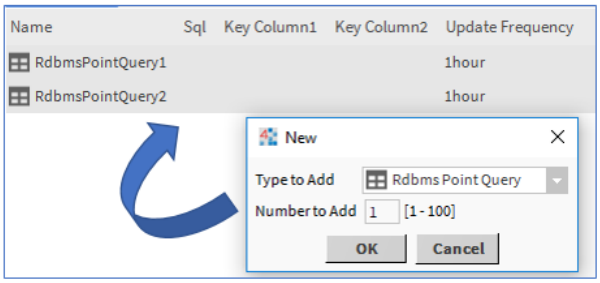
Columns

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:

- Add one or more queries to the Rdbms Points Device Extension. Once added, these Rdbms Point Queries appear in the manager view as well as in the Nav tree.

Figure 45 The New query window



- Click **Edit** to edit the added query. The **Edit** window opens.

Figure 46 The Edit Query Window

Following are the fields you can edit using the**Edit** window.

Column Name	Description
Name	Edits the name of the point.
Sql	Edit the database name.
Key Column 1	Edits the primary key.
Key Column 2	Edits the secondary key.
Update Frequency	Edits the frequency in hours.

- Select one or more entries in the **Database** pane and by using (right-click) popup menu following edit can be done to selected row:

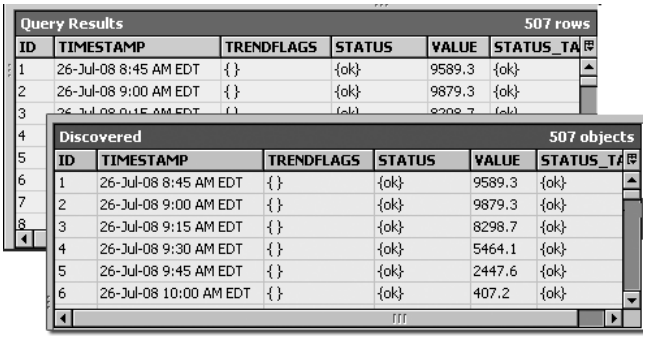
Menu	Description
Views	Display different views for the selected point(for eg, Property sheet, RDBMS Query view and so on).
Action	This allows to execute the query.
New	Allows to add new folder and points.
Rename	Renames the selected point.
Cut	Cuts the selected row.
Copy	Copies the selected row.
Paste	Pastes the copied row.
Delete	Deletes the selected row
Duplicate	Duplicates the selected row.
Reorder	Reorders the query points.

- 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.

Rdbms Point Query Manager view

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 47 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.

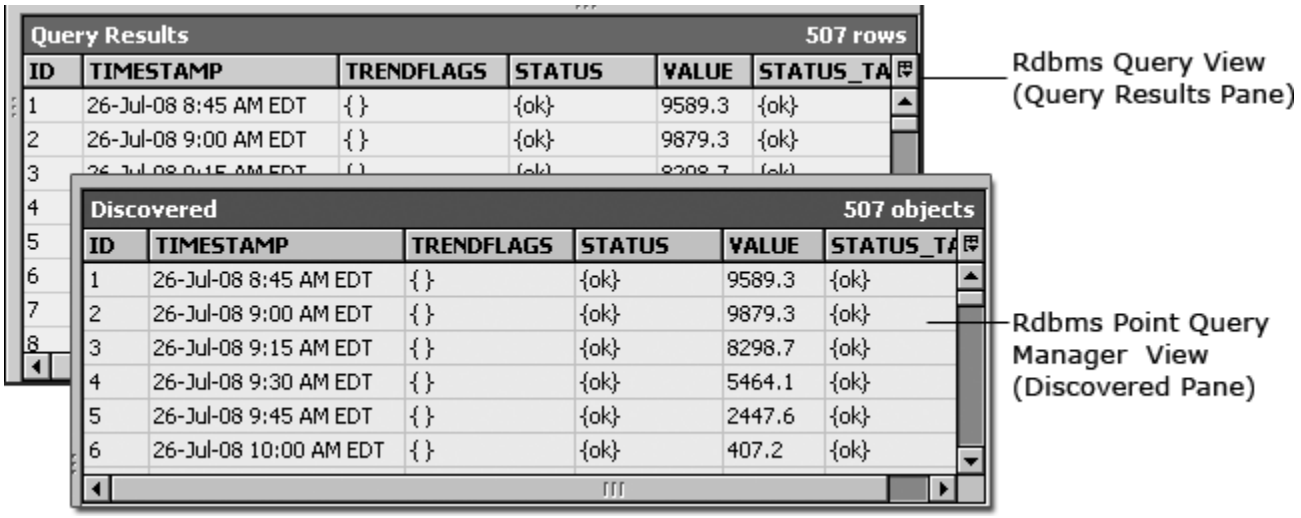
Column name	Description
Key Value 1	Identifies the primary key.
Key Value 2	Identifies the secondary key.

Buttons

Button name	Description
New Folder	Adds a special Device Folder component you can use to organise devices.
New	Adds a new device component to the station.
Edit	Allows you to edit the added device component.
Cancel	A cancel button s enabled during Discovery job to stop discovery process.
Add or **Match**	Moves the discovered and selected points from the **Discovered** pane to the **Database** pane and includes them under the **Points** node of the appropriate database.

Query Results vs Discovery

Figure 48 Example of an Rdbms Query compared to the Discovered pane

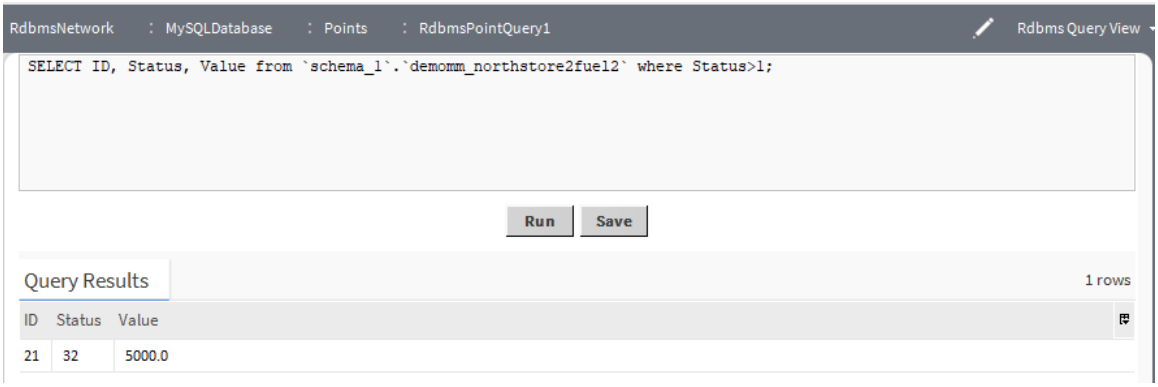


Discovered points represent the results of executing the Rdbms point query. In fact, the results from the data in the **Discovered** pane should match the data presented in the Query Results pane of the **Rdbms Query View**. The **Discover** button executes the query defined by the `Sql` property in 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 49 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. Following are the two buttons used to execute the query:

Button Name	Description
Run	Executes the query written in **Query** pane.
Save	Saves the query typed in **Query** pane.

- The bottom **Query Results** pane displays, the results of executing the Rdbms point query. Click the to do the following actions:

  Select the **Reset Column Widths** to adjust the column width of **Query Results** pane.

  Select **Export** to export the query results. New **Export** window opens
  Select the exporter type from the dropdown list. Select the view from the four options provided. Click `Ok` to get the file exported

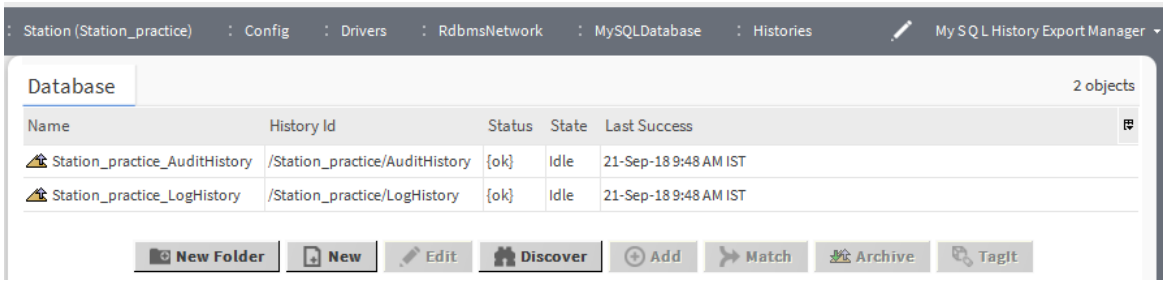
Rdbms Session View

This view provides information about the relational database session.

MySQL History Export Manager view

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

Figure 50 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** identifies the type of History.
- **History ID** refers the History Id.
- **Execution Time** displays the last execution time.
- **Enabled** displays the status.
- **Status** displays history exported status.
- **State** displays current state of the database
- **Last Success** Date and time of last history exported.
- **Last failure** displays the last failure for the discovery of the history.
- **Fault Cause** shows the reason of last fail cause of history.

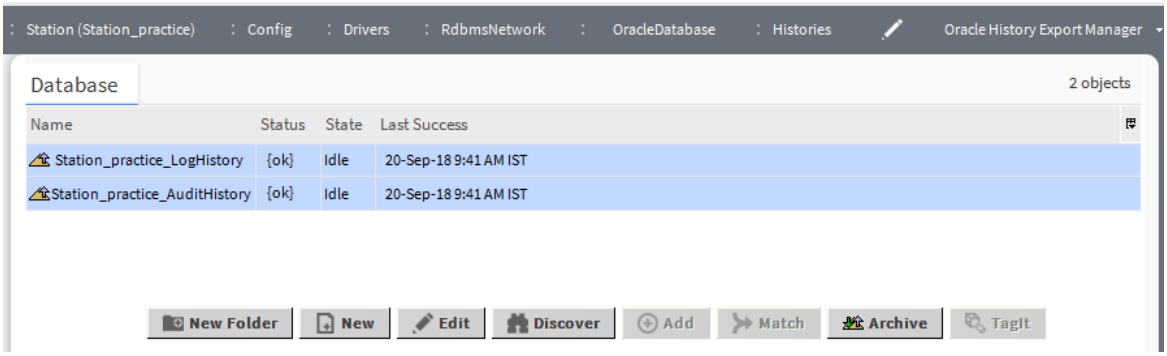
Buttons

- **New Folder** adds a folder for organizing multiple devices in the Database view.
- **New** adds new History type.
- **Edit** edits the history type.
- **Discover** discovers the already added histories.
- **Add** adds the discovered histories.
- **Match** matches the discovered histories and the already present histories.
- **Archive** archives the history types.

Oracle History Export Manager view

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

Figure 51 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** identifies the type of History.
- **History ID** refers the History Id.
- **Execution Time** displays the last execution time.
- **Enabled** displays the status.
- **Status** displays history exported status.
- **State** displays current state of the database
- **Last Success** Date and time of last history exported.
- **Last failure** displays the last failure for the discovery of the history.
- **Fault Cause** shows the reason of last fail cause of history.

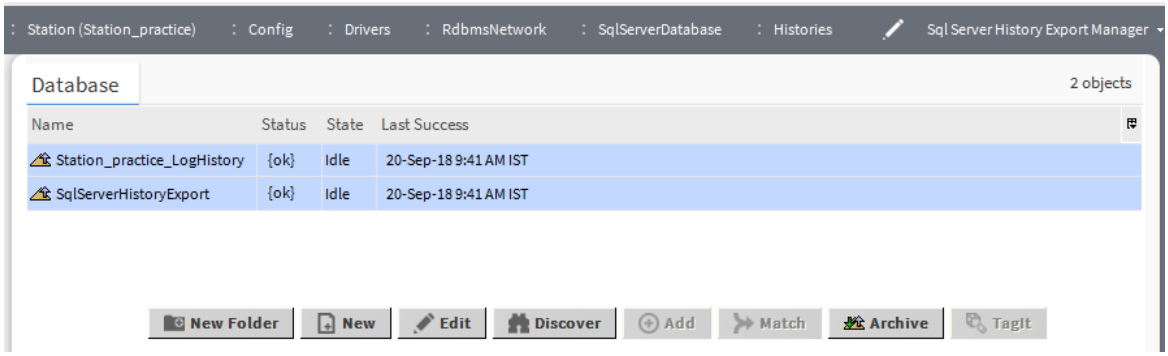
Buttons

- **New Folder** adds a folder for organizing multiple devices in the Database view.
- **New** adds new History type.
- **Edit** edits the history type.
- **Discover** discovers the already added histories.
- **Add** adds the discovered histories.
- **Match** matches the discovered histories and the already present histories.
- **Archive** archives the history types.

SqlServer History Export Manager view

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

Figure 52 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** identifies the type of History.
- **History ID** refers the History Id.
- **Execution Time** displays the last execution time.
- **Enabled** displays the status.
- **Status** displays history exported status.
- **State** displays current state of the database
- **Last Success** Date and time of last history exported.
- **Last failure** displays the last failure for the discovery of the history.
- **Fault Cause** shows the reason of last fail cause of history.

Buttons

- **New Folder** adds a folder for organizing multiple devices in the Database view.
- **New** adds new History type.
- **Edit** edits the history type.
- **Discover** discovers the already added histories.
- **Add** adds the discovered histories.
- **Match** matches the discovered histories and the already present histories.
- **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

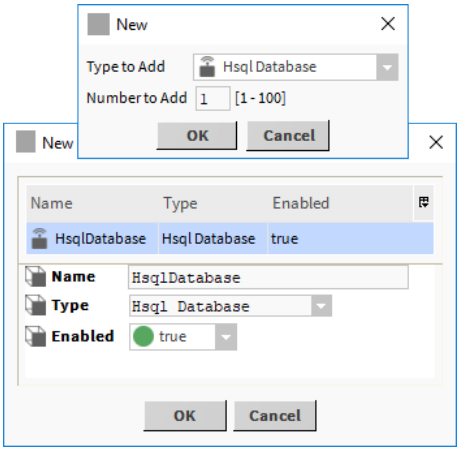
Windows create and edit database records or collect information when accessing a component. You access them by dragging a component from a palette to a nav tree node or by clicking a button.

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 53 MySQL History Device Ext properties



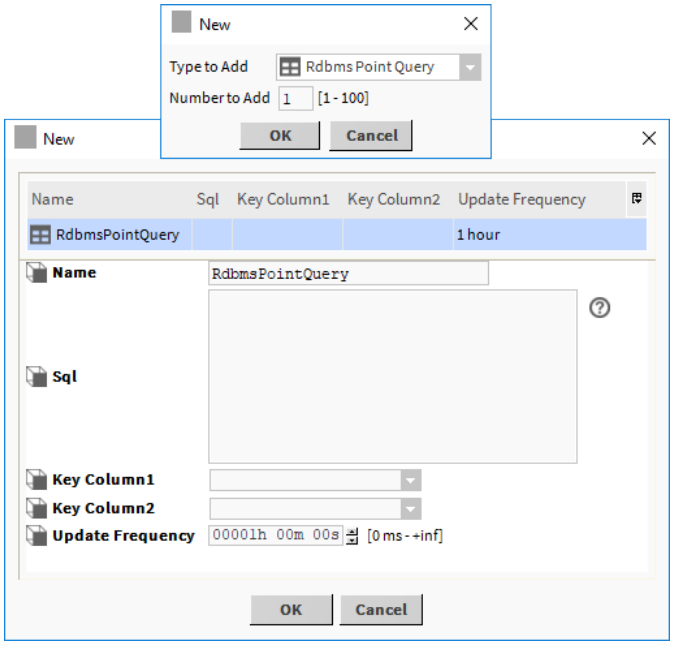
To access these properties, expand **Config→Drivers**, double-click the **RdbmsNetwork** node and click **New**.

Individual properties are documented in the *Properties Dictionary* (in this guide).

New points windows

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

Figure 54 Points window properties



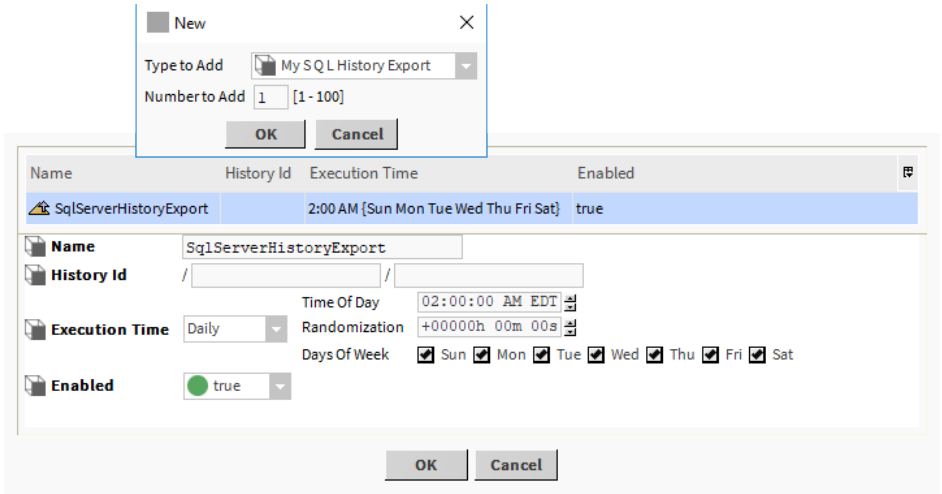
To open these windows, expand **Config→Drivers→RdbmsNetwork**, expand a database node in the network view and click **New**.

Individual properties are documented in the *Properties Dictionary* (in this guide).

New export histories windows

These windows configure when and which histories to export.

Figure 55 New export histories windows



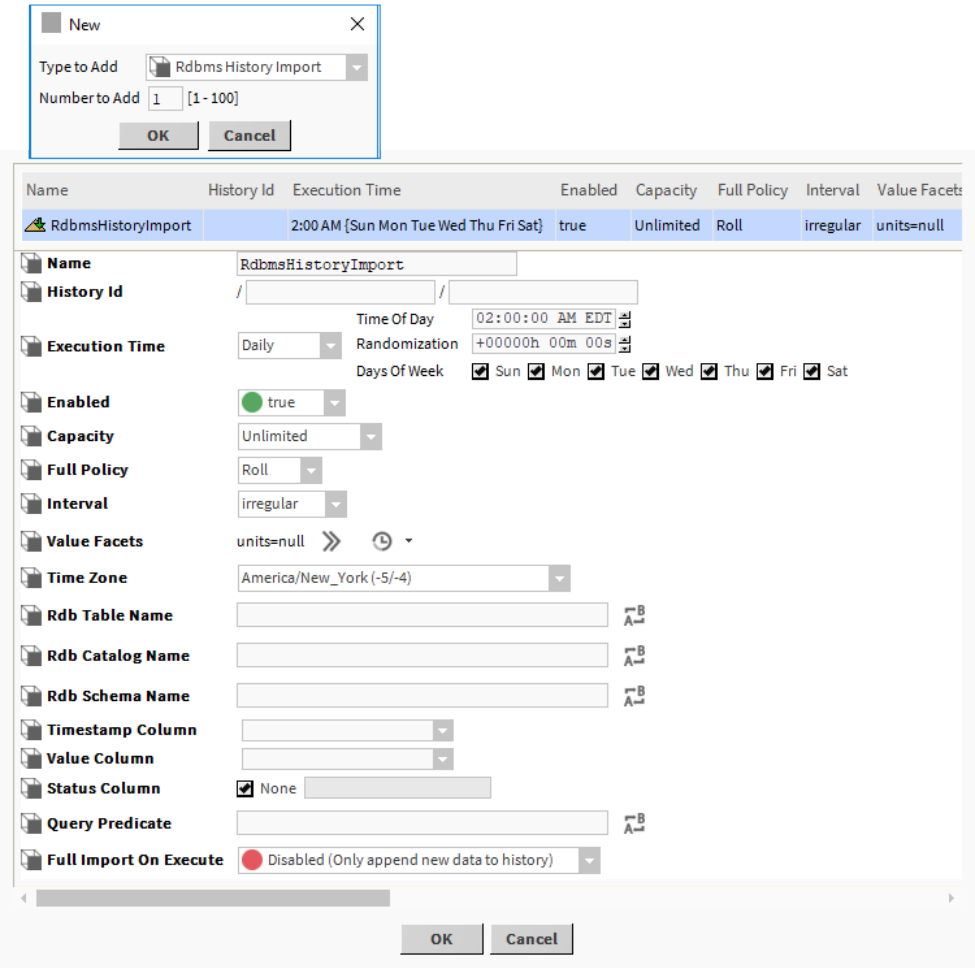
To access these windows, expand **Drivers→RdbmsNetwork**, double-click the **Histories** node in the Nav tree, and click **New**.

Individual properties are documented in the *Properties Dictionary* (in this guide).

New import histories windows

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

Figure 56 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**.

Individual properties are documented in the *Properties Dictionary* (in this guide).

Index

Q

R

S

T

U

V

W