Technical Document

NiagaraAX Rdbms Driver Guide

November 11, 2013

Powered by niagara FRAMEWORK®

NiagaraAX Rdbms Driver Guide

3951 Westerre Pkwy, Suite 350
Richmond, Virginia 23233
U.S.A.

Copyright Notice

Trademark Notices

Copyright and Patent Notice

Contents

Preface

- *Document Change Log*
- *Related documentation*
- *Database licensing*

Document Change Log

Updates (changes/additions) to this *NiagaraAX-3.x Rdbms Driver Guide* document are listed below.

- NiagaraAX-3.8 release revision, November 2013 includes the following changes:
  - In the *Installing the RdbmsNetwork driver and rdbDatabase device* section, the procedure for installing the Rdbms Network and its rdbDatabase device(s) into the Drivers container has been changed. In addition, all variants of the rdbDatabase device are now covered in the same procedure.
  - Reorganized installation information under *Installing the RdbmsNetwork driver and rdbDatabase device* to create a single configuration procedure that includes a summary table of relevant properties that require configuration setting for each database device. Details of each property is included, as before, in the *About the RdbmsNetwork device components and extensions* section.
  - In the *Common Rdbms Connection Problems* section, an extra note on SqlServer RDBMS connections has been added.
  - In the *About the RdbmsNetwork device components and extensions* section, many of the property descriptions have been updated with more description, best practice and other notes. New properties have been added and the section is in line with the new release revision.
  - In the *Types of RdbmsNetwork Driver Devices* section, the SqlServerDatabase list item has been updated with the newer SQL Server releases, notes on authentication and troubleshooting. The MySQLDatabase list item has been updated to include reference to the requirements and availability of the JDBC connector. A reference to this has been added to the *Install the RdbmsNetwork Driver* section.
  - In *Export history data to an Rdbms database*, an additional guidance note has been added to Execution Time.
  - In the *Creating Rdbms proxy points* section, another note has been added about embedded JACE's to the Example Rdbms implementation.
  - In Chapter 3 *Common RdbmsNetwork tasks* two new sections, *Upgrading existing Rdbms databases to support Unicode and UTC* and *Upgrading existing Orion databases to support Unicode* have been added.
  - Chapter 4 *Use Examples* has been added.
  - In the *Components in rdb module* section, two new components have been added along with a procedure describing their use.
  - In the *Components in orion module* section the components that are not visible in the palette have been clearly identified by notes and a new orion-OrionMigrator component has been added.
- Initial release: October 1, 2008

Related documentation

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

- *NiagaraAX-3.x User Guide*
- *NiagaraAX-3.x Drivers Guide*

Database licensing

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 documentation. See *Types of RdbmsNetwork Driver Devices* for links to some database supplier web sites.

Chapter 1 RdbmsNetwork Installation and Configuration

The RdbmsNetwork driver allows a connection to various Relational DataBase Management Systems (RDBMS), for the purposes of importing and exporting historical archive data and populating Control Points with the results of SQL queries against the database. This chapter describes how to use WorkbenchAX to install, configure and test an RdbmsNetwork driver with the following Rdbms database devices:

- Db2
- MySQL
- Oracle
- SqlServer
- HsqlDb

NOTE: For embedded JACEs, the HsqlDbDatabase is the only database that is supported.

To connect to an RDBMS from a NiagaraAX workstation (for example, for importing, exporting, or querying data), refer to *Installing the RdbmsNetwork driver and rdbDatabase device.*

Requirements

The following list includes prerequisites for successfully installing and using the RdbmsNetwork:

- NiagaraAX license feature for the appropriate Rdbms database

  Each database type requires a specific license feature entry in the license file.
- IP Network connection to the database host

  You have to be able to connect to the database over an IP connection.
- User Name and Password for database login

  You must be able to login to the database in order to establish a valid connection.
- Appropriate rights for required data access

  You must have appropriate rights for access to the desired tables and for any actions that you plan on performing with the data.

In addition, there may be specific database-related items that you need to know. For example, if you are trying to connect to a "named instance" of a database, you may need to know the name of the database instance. Also, if the database is using a non-default TCP port number, you need to know the port number so you can configure your RdbmsNetwork database device correctly.

Suggestions for solving problems in making a valid connection to a remote database are included in *Common Rdbms Connection Problems.*

Installing the RdbmsNetwork driver and rdbDatabase device

The following procedures describe how to install, configure, and test the RdbmsNetwork driver:

- *Install the RdbmsNetwork Driver*
- *Configure the RdbmsNetwork driver*
- *Test the RdbmsNetwork connection*

Install the RdbmsNetwork Driver

To install the RdbmsNetwork Driver for use with rdbDb2, rdbHsqlDb, rdbMySQL, rdbOracle, or rdbSqlServer, do the following:

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

Step 2 Click on the **New** button to bring up the **New network** dialog. For more details see “Driver Manager New and Edit” in the *Drivers Guide*.

Step 3 Select `Rdbms Network`, number to add: `1` (or more if multiple networks) and click **Ok**. This brings up a dialog to name the network(s).

Step 4 Click **Ok** to add the Rdbms Network(s) to the station.

You should have an Rdbms Network named “RdbmsNetwork” (or whatever you named it) under your Drivers folder showing a status of "`{Ok}`" and enabled as "`true`".

Step 5 Double-click the Rdbms Network you have just added and click the **New** button at the bottom of the Device Manager view to get the New device dialog. In `Type to Add` select the desired Rdbms Network Driver Device Database.

Step 6 Select number to add: `1` (or more if multiple devices) and click **Ok**. This brings up a dialog to name the database device(s).

Step 7 Click **Ok** to add the Database Device to the Rdbms Network.

You should have an Rdbms Network Driver Device Database under your Rdbms Network Driver.

Configure the RdbmsNetwork driver

To configure the RdbmsNetwork driver in your NiagaraAX station, do the following:

Step 1 In WorkbenchAX, under the RdbmsNetwork node, right-click on the Database Device Component node in the nav tree and select **Property Sheet** from the popup menu. The Database Device Property Sheet view displays.

Step 2 In the Property Sheet view, set values for the all the properties applicable to the Database Device.

A full list of all the properties in each of the RdbmsNetwork driver devices is shown in the table below.

Those properties marked with a ‘-’ are not available in the driver device and some of the properties are provided only for information. Those properties marked with a ‘X’ should be considered for configuration.

Refer to *About the RdbmsNetwork device components and extensions* for a full description of the properties.

Table 1 RdbmsNetwork driver properties

Property	Db2	HsqlDb	MySQL	Oracle	SqlServer
Status (Information)					
Enabled	X	X	X	X	X
Fault Cause (Information)					
Health (Information)					
Alarm Source Info	X	X	X	X	X
Host Address	X	-	X	X	X
User Name	X	X	X	X	X
Password	X	X	X	X	X
Worker	X	X	X	X	X
Export Mode	X	X	X	X	X
Use Unicode Encoding Scheme	X	X	X	X	X
Timestamp Storage	X	X	-	X	X
Points	X	X	X	X	X
Sql Scheme Enabled	X	X	X	X	X
Base Directory	-	X	-	-	-
Database Name	X	X	X	-	-
Instance Name	-	-	-	-	X
Service Name	-	-	-	X	-
Port	X	-	X	X	X
Histories	X	-	X	X	X
Extra Connection Properties	-	-	X	-	X
Version	-	-	-	-	X

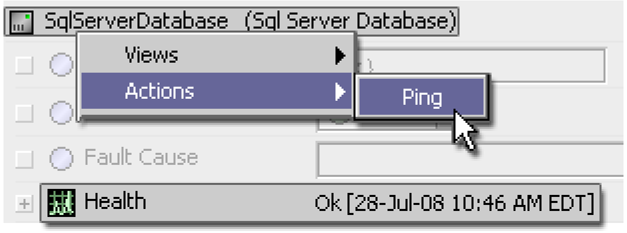
Step 3 Click the **Save** button to save all property settings.

Test the RdbmsNetwork connection

The following procedure applies to any type of rdbDatabase driver.

To test the RdbmsNetwork Connection, do the following:

Step 1 Right-click on the RdbmsDatabase Device Extension in the Property Sheet view (or nav tree) and select **Actions Ping** from the popup menu, as shown below.


SqlServerDatabase (Sql Server Database)
Views
Actions
Ping
Fault Cause
Health
Ok [28-Jul-08 10:46 AM EDT]

If a valid connection to the database is made, the Health property displays an "Ok" value. If the Health property displays a "Fail" value, then a connection is not made and you should examine the Last Fail Cause property for details

Common Rdbms Connection Problems

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

The following list applies to all types of RDBMS connections:

- Make sure that both the RdbmsNetwork and RdbmsDatabaseDevice have their **Enabled** property set to `true`.
- Verify the network connection to the remote database by issuing an ICMP ping from your operating system.
- Check that you have set the correct port number. Default port numbers (listed in Chapter 2 *About the RdbmsNetwork*) may not have been used when the database instance was initially configured.
- 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.

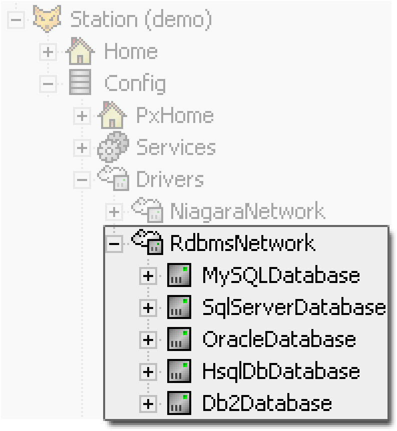
The following list applies to SqlServer RDBMS connections:

- 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 the property value field is empty, it is ignored and the database users default database as assigned to the user in SqlServer will be used.
- In order for a Niagara AX station to connect, the Microsoft SQL Server instance must be configured for Mixed Mode Authentication. Refer to *Types of RdbmsNetwork Driver Devices* for more details.

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

Chapter 2 About the RdbmsNetwork

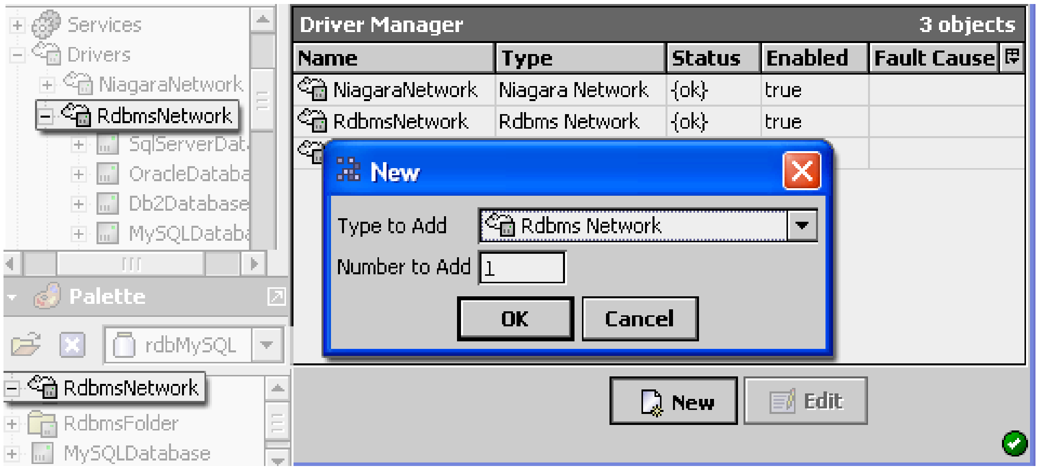
The RdbmsNetwork driver shown below, is a “non field bus” type driver that uses a network architecture similar to other NiagaraAX drivers.



The RdbmsNetwork driver has many properties and extensions in common with other "field bus” type Network Drivers. However, there are several distinctive RdbmsNetwork driver characteristics. The following list briefly notes some of the distinctive points of the RdbmsNetwork driver:

- The RdbmsNetwork Driver Component Location

  There is not a separate RdbmsNetwork driver component palette. In addition to being available from the Driver Manager view shown below, the driver is also located on each of the individual rdb database module palettes.



Driver Manager 3 objects

Name	Type	Status	Enabled	Fault Cause
NiagaraNetwork	Niagara Network	{ok}	true	
RdbmsNetwork	Rdbms Network	{ok}	true	

Type to Add: Rdbms Network

Number to Add: 1

When you add the RdbmsNetwork driver component from the Driver Manager view, using the **New** dialog box, the component is placed under the Station "Drivers" node.

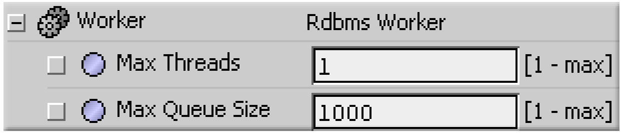
NOTE: Starting in NiagaraAX-3.4, appropriately-licensed NiagaraAX PC based Supervisor stations support SqlServer, Oracle, Db2, MySQL, and HsqlDb databases. For embedded JACEs, the HsqlDbDatabase is the only database that is supported.

- Rdbms Point Device Extension

The 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. Refer to *About the Rdbms Point Device Extension* for details.

- RdbmsNetwork Tuning Policies

  The RdbmsNetwork does not have a "Tuning Policies" component. However, some measure of tuning is provided with the "Rdbms Worker" component shown below and available under individual Rdbms device drivers).

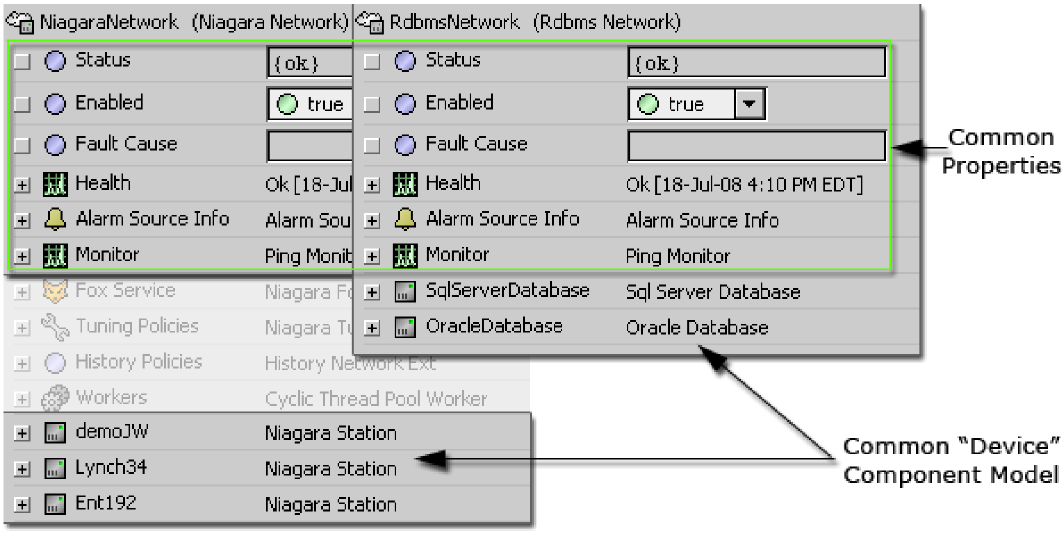


⊟ Worker	Rdbms Worker	
□ Max Threads	1	[1 - max]
□ Max Queue Size	1000	[1 - max]

About the RdbmsNetwork hierarchy

The RdbmsNetwork, like other NiagaraAX networks, provides a top-level component for all NiagaraAX RDBMS drivers. In keeping with the standard NiagaraAX driver architectural model, many of the RdbmsNetwork components, device extensions, and WorkbenchAX views resemble those in other NiagaraAX drivers.

The RdbmsNetwork property sheet view and the NiagaraNetwork property sheet view are compared below. Common network features are described in the *NiagaraAX Drivers Guide*.



NiagaraNetwork	(Niagara Network)
□ Status	{ok}
□ Enabled	true
□ Fault Cause	
⊞ Health	Ok [18-Jul
⊞ Alarm Source Info	Alarm Sou
⊞ Monitor	Ping Monit
⊞ Fox Service	Niagara F
⊞ Tuning Policies	Niagara T
⊞ History Policies	History Network Ext
⊞ Workers	Cyclic Thread Pool Worker
⊞ demoJW	Niagara Station
⊞ Lynch34	Niagara Station
⊞ Ent192	Niagara Station

RdbmsNetwork	(Rdbms Network)
□ Status	{ok}
□ Enabled	true
□ Fault Cause	
⊞ Health	Ok [18-Jul-08 4:10 PM EDT]
⊞ Alarm Source Info	Alarm Source Info
⊞ Monitor	Ping Monitor
⊞ SqlServerDatabase	Sql Server Database
⊞ OracleDatabase	Oracle Database

Types of RdbmsNetwork network-level components and properties

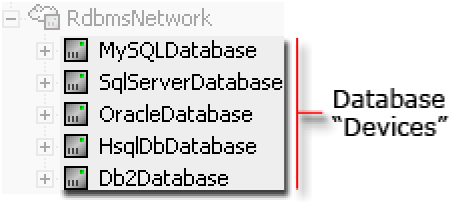
RdbmsNetwork network-level components and properties are listed below:

- Status
- Enabled
- Fault Cause
- Health
- Alarm Source Info
- Ping Monitor

These components and properties are analogous to the components and properties of the same, or similar, name in other network drivers and are described in the *NiagaraAX Drivers Guide*.

About the RdbmsNetwork device components and extensions

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



The following device components and properties are identical to those listed under *Types of RdbmsNetwork network-level components and properties* and described in the *NiagaraAX Drivers Guide.*

- Status
- Enabled
- Fault Cause
- Health
- Alarm Source Info

Properties and components that are common to (or similar among) most Rdbms database devices are visible from the Rdbms driver property sheet view are described in the following list:

- Enabled

  Select **true** from the option list.
- Host Address

  Use this field to set 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.
- User Name

  This property is the user name that is used to login to the database.

  NOTE: Your login credentials must provide sufficient database privileges to allow you to perform one or more (depending on database type) of the following commands:CREATE TABLE, CREATE INDEX, CREATE SEQUENCE

- Password

  This property is the password of the user than is used to login to the database.
- Worker

  This is a child component of all Rdbms driver devices. The "Rdbms Worker" manages the queue and threads for asynchronous operations on a single parent database. The Worker 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. The following two parameters apply to Worker properties:

- Max Threads

  This value allows you to specify the maximum number of multiple concurrently 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.

  This can normally be increased to a value between 20 and 50 to improve performance when handling large volumes of data.

- Max Queue Size

  This value allows you to specify the maximum queue size supported for Rdbms actions specific to the associated database driver (i.e. exports, imports). The default Max Queue Size is 1000.

  `Max Queue Size` needs to be adjusted to reflect the peak number of import or export records per archive that are expected to be processed and will normally be increased when handling large volumes of data.

- Export Mode

  This property allows you to specify how histories will be exported into the specified database. The two possible options are:

  - By History Id

    This choice specifies that histories are exported as one table per History Id. This is the default value setting.

  - By History Type

    This choice specifies that histories are exported as one table per History Type. This option may make the data easier to query once it has been exported.

  NOTE: For good practice, **Export Mode** should normally be set to `By History Type`, as most DBA's would rather manage a few tables instead of thousands of individual tables. When exporting by `By History Type`, histories of the same data type (Boolean, Numeric, Enum, String) are exported to the same table. For example, all numeric type histories are exported to a table named HISTORYNUMERICTRENDRECORD. The table contains an additional column named HISTORY_ID which stores the history ID reference (ORD) from the Niagara station for each record in the table.

- Use Unicode Encoding Scheme

  This boolean property has `true` and `false` (default) options that allow you to create history table schemas with Universal character set Transformation Format (UTF-8) or Unicode data types for string valued columns in order to store Asian character sets. This property is available starting in NiagaraAX-3.8 and it defaults to `false` to maintain backwards compatibility with the existing history export mechanism. When this property is set to `true`, the resulting history tables created in the database will have the 'NVARCHAR' data type for any columns in the tables that expect string data.

NOTE: An **Update Wizard** is available to help you easily perform an upgrade of existing databases to support Unicode as described in *Upgrading existing Rdbms databases to support Unicode*.

- Timestamp Storage

  This enum property allows you to export or update history timestamps to Coordinated Universal Time (UTC). It enables History records to be exported from different timezones into a common database and be chronologically correct and independent of any specific source timezone characteristics. In other words, exported histories will show the timestamp data from where the history is actually stored, making useable histories with a consistent timestamp.

  This property is available starting in NiagaraAX-3.8 and it defaults to `Dialect Default` to maintain backwards compatibility with the existing history export mechanism. This property does not apply to the MySQL device. The possible options are:

  - Dialect Default

    This choice specifies that the timestamp export policy is in legacy mode to maintain backwards compatibility with the existing history export mechanism. This is the default value setting.

    NOTE: If you are selecting the `Dialect Default` option, then you must set both **Use Last Timestamp** and **Use History Config Time Zone** properties in the History Device Extension to `true` to export histories using the History Config Timezone. (If the two properties are set to `false`, the histories will be exported using the station timezone).

  - Local Timestamp

    This choice applies when using Orion databases only and does not apply to history exports.

  - Utc Timestamp

    This choice specifies that all subsequent exported histories will be exported with their timestamps adjusted to UTC time.

    NOTE: Do not switch back and forth between `Dialect Default` and `Utc Timestamp` because the database table will become polluted with inconsistent timestamps and this will affect any query that is run to determine whether or not to export newer records.

    NOTE: If you are selecting the `Utc Timestamp` option, then this takes precedence over any setting of the **Use Last Timestamp** and **Use History Config Time Zone** properties in the History Device Extension, rendering them effectively irrelevant.

  - Utc Millis

    This choice applies when using Orion databases only and does not apply to history exports.

NOTE: Associated with Timestamp Storage and available from NiagaraAX-3.8 is the addition of a new column called 'DB_TIMEZONE' to the HISTORY_CONFIG and HISTORY_TYPE meta tables that are created the first time histories are exported. 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's being stored). When records are being exported in UTC mode, the DB_TIMEZONE column will update accordingly.

NOTE: An **Update Wizard** is available to help you easily perform an upgrade of existing databases to support UTC as described in *Upgrading existing Rdbms databases to support UTC*.

- Points

  This component is available starting in NiagaraAX-3.4 and is a child of all Rdbms driver devices. 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 feature, you can represent relational database cell values as proxy points. No configuration is required for RdbmsNetwork Driver configuration. See “Using the Rdbms Point Device Extension”, and "About the Points Extension" in the *NiagaraAX Drivers Guide*.

- Sql Scheme Enabled

  This property has `true` and `false` (default) options that allow you to specify that Sql Scheme is to be enabled.

  You must set this property to `true` if you are going to execute queries against the relevant database. It also needs to be enabled as a prerequisite of using the Rdbms Point Device Extension.

  You must also set this property to `true` if you are going to chart or load data back in from an Oracle database into Niagara. This is also useful for debugging as you can run limited queries against the database from Niagara.

  If you are not executing queries, using the Rdbms Point Device Extension, or loading data from Oracle, for good security practice you should set the property to `false`.

- Base Directory, Database Name, Instance Name, Service Name

  Type in the name of the database that you are connecting to (see the following notes).

NOTE: For HsqlDb device only:- The “Base Directory” property is 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”, then a path to the folder is specified as “file:^hsqldb”.

NOTE: For Db2, HsqlDb and MySQL Database devices only: The "Database Name" property allows you to configure a connection to the Database.

NOTE: For SqlServerDatabase device only: The "Instance Name" property allows you to configure 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 if you wish to connect to a name which is not the default. If a default name has been set up and you need to connect to it then you can leave this property blank.

NOTE: For OracleDatabase device only: The "Service Name" property 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 if you wish to connect to an SID which is not the default. If a default SID has been set up and you need to connect to it then you can leave this property blank.

- Port

  The value of this property specifies the port number to use when connecting with the database. Default values are:

  - Db2Database - Port 6789
  - HsqlDbDatabase - no port is specified because this rdb is for local database use only.
  - MySQLDatabase - Port 3306
  - OracleDatabase - Port 1521
  - SqlServerDatabase - Port 1433

- Histories

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

  In addition to the normal default “Retry Trigger” component, the History Device Extension contains the following two parameters which are specific to this driver:

  - Use Last Timestamp

    This property has `true` and `false` (default) options.

    When this property is set `false`, every time an export descriptor processes, the Niagara station must query the database to determine the value of the last timestamp in the database. Depending on the size of the database table and the number of export descriptors being processed, this can be a time consuming activity.

    When this property is set `true`, instead of querying the database each time for the last record in the database, the Niagara station stores the last timestamp as a property on the export descriptor each time it processes the export. This is an efficiency related setting that helps with overall system performance when exporting data. You may consider therefore, to set the option `true`.

  - Use History Config Time Zone

    This property has `true` and `false` (default) options.

    This property was originally added to configure how the timestamp is normalized before it is to written to the external database. Prior to NiagaraAX-3.8, timestamps stored in the external database did not include time zone information, rather they were simply date and time.

    When this property is set `false`, timestamp values are normalized to the supervisor station’s time zone before being written to the external database.

When this property is set `true`, the original timestamp value is retained by the supervisor and written to the external database. Unfortunately, there is no timezone information in this data and difficulties have arisen with local timezones especially in relation to daylight savings time. It is for this reason that the NiagaraAX-3.8 release introduced the **Timestamp Storage** property allowing you to export or update history timestamps to Coordinated Universal Time (UTC).

NOTE: The effect of these History properties may be renderered irrelevant depending upon the setting of the **Timestamp Storage** property.

- Extra Connection Properties

  This would normally be left blank but it would typically be used for advanced diagnostics or tuning under the guidance of a support channel. It is not required for a normal connection to the database. Extra connection properties are additional parameters which are passed in plain text by the driver when connecting to a database. It is beyond the scope of this document however, to define the parameters that could be used but they are typically entered in the format: ;parameter1=value1;parameter2=value2;...etc. This property applies to the MySQL and SqlServer devices.

- Version

  This is used to determine whether 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. You must set this property to match the version of the database you are connecting to. This property only applies to the SqlServer device. The possible options are:

  - Sql Server 2008

    This choice specifies that the Version of Sql Server you are connecting to is Sql Server 2008. This is the default value setting.

    NOTE: Use this `Sql Server 2008` option if you are connecting to a Sql version that is more recent than Sql Server 2008, such as Sql Server 2012.

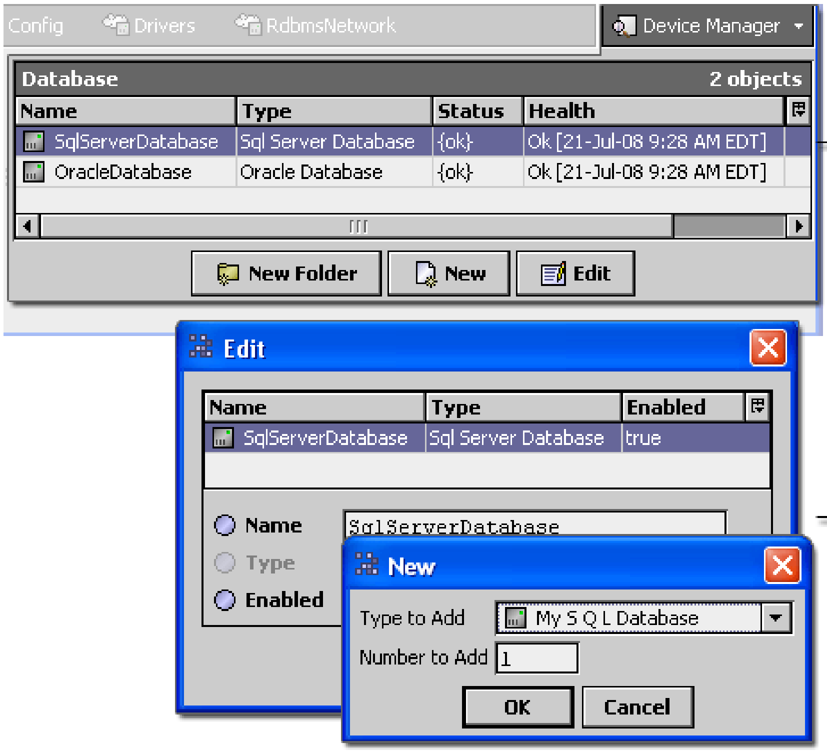
  - Sql Server 2005

    This choice specifies that the Version of Sql Server you are connecting to is Sql Server 2005.

  - Sql Server 2000

    This choice specifies that the Version of Sql Server you are connecting to is Sql Server 2000.

RdbmsNetwork Device Manager View



Database 2 objects

Name	Type	Status	Health
SqlServerDatabase	Sql Server Database	{ok}	Ok [21-Jul-08 9:28 AM EDT]
OracleDatabase	Oracle Database	{ok}	Ok [21-Jul-08 9:28 AM EDT]

Edit

Name	Type	Enabled
SqlServerDatabase	Sql Server Database	true

Name SqlServerDatabase
Type
Enabled

New

Type to Add My S Q L Database
Number to Add 1

RDBMS Device Manager View

NEW and EDIT Dialog Boxes

In addition to the standard views (property sheet, wire sheet, slot sheet, and others) the RdbmsNetwork also uses the Device Manager view. The Device Manager view has **New** and **Edit** dialog boxes that are used to add, configure, and monitor RdbmsNetwork devices similar to the way other network drivers are used.

Individual RDBMS devices have some 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 RDBMS driver devices.

Types of RdbmsNetwork Driver Devices

NOTE: 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 database supplier documentation (links to database supplier web sites are provided below).

The following Rdbms Databases are supported in NiagaraAX-3.4 and later:

NOTE: See *Installing the RdbmsNetwork driver and rdbDatabase device*) for details about installing and configure RdbmsNetwork Driver Devices.

- Db2Database

- The NiagaraAX rdbDb2 module supports the use of data from DB2 databases, version 8.1. DB2 is an IBM relational database management system (RDBMS) that requires a proprietary license.
- For more information about DB2, refer to: `http://www-306.ibm.com/software/data/db2/`

HsqlDbDatabase

- The NiagaraAX rdbHsqlDb module supports the use of data from HSQLDB databases. 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/`

MySQLDatabase

The NiagaraAX rdbMySQL module supports the use of data from MySQL databases. Note the following about the MySQL database and associated driver:

- MySQL is an Oracle Corporation relational database management system (RDBMS) that requires a GPL or proprietary license.
- In order to use MySQL with a supervisor installation, you must also install a Java Data Base Connectivity (JDBC) connector in the following folder (where “C:” represents the NiagaraAX installation location): `$niagara_home\jre\lib\ext`, where niagara_home denotes the AX installation location eg c:\Niagara\Niagara-3.7.106
- The connector that you need to install is “ConnectorJ” with a filename similar to the following: " `mysql-connector-java-5.1.11-bin.jar`". It is available for download from: *http://dev.mysql.com”*

- OracleDatabase
  - The NiagaraAX rdbOracle module supports the use of data from Oracle databases. At the time of writing, the supported version is version 9i but you should contact your support channel for more up to date information. Oracle is an Oracle Corporation relational database management system (RDBMS) that requires a proprietary license.
  - For more information about Oracle, refer to: `http://www.oracle.com/database/index.html`

SqlServerDatabase

- The NiagaraAX rdbSqlServer module supports the use of data from Microsoft SQL Server database versions: SqlServer 2000, SqlServer 2005, SqlServer 2008, SqlServer 2008 R2 and SqlServer 2012.
- SQL Server is a Microsoft Relational Database Management System (RDBMS) that requires a proprietary license.
- For more information about Microsoft SQL Server, refer to: `http://www.microsoft.com/sql/default.mspx`
- Whilst it is beyond the scope of this document to describe the setup of the database, it is worthwhile noting that during SQL Server setup, you must select an authentication mode for the Database Engine. There are two possible modes: Windows Authentication mode and Mixed Mode. The SQL Server must be configured for Mixed Mode Authentication because the NiagaraAX driver does not support Windows Authentication.

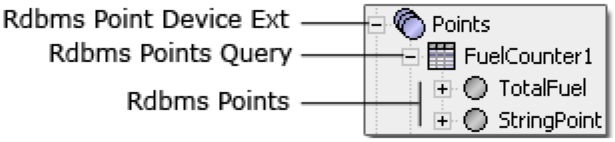
- If you wish to monitor the performance of an instance of SQL Server or troubleshoot problems with the queries being submitted to the database by the Niagara station, then alongside the usual Platform-Application Director output it is often useful to use an SQL Profiler.

About the Rdbms Point Device Extension

Starting with NiagaraAX-3.4, the Rdbms Point Device Extension is available as part of most RdbmsNetwork driver devices. As described in Chapter 2 *About the RdbmsNetwork*, the RdbmsNetwork and RdbmsNetwork device drivers resemble the typical "field bus" type drivers in that they are represented in WorkbenchAX 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.

NOTE: To use this feature, you must set the **Sql Scheme** property to `true` (this property is located on the RdbmsNetwork Driver Device property sheet.

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 NiagaraAX, as shown below.



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.

Characteristics of the Rdbms Point Device Extension

Starting in NiagaraAX-3.4, the Rdbms Point Device Extension is present as a child component of all Rdbms Database Device Components. The Rdbms Point Device Extension is similar to other Device Extensions in some ways but is unique in several ways.

The following is a list of similarities. 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.

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 is always coming from a "secondary" source - the database, not a control point.

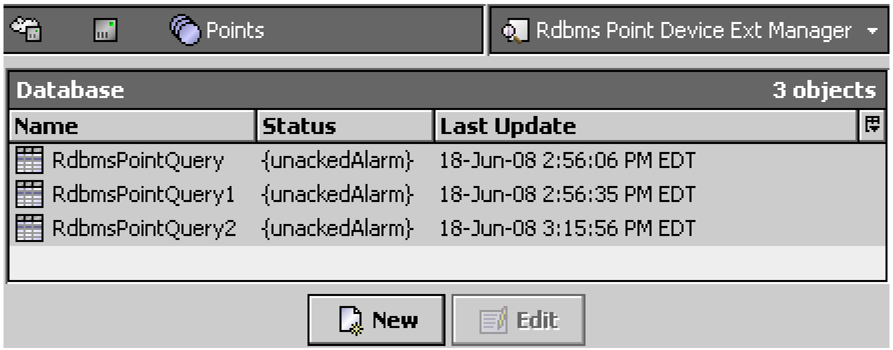
Types of Rdbms Point Device Ext Views

In addition to standard representations of the Wire Sheet, Category Sheet, Slot Sheet, and Link Sheet views, the Rdbms Point Device Extension has the following two views worth noting here:

- *About the Rdbms Point Device Ext Manager View*
- *About the Rdbms Point Device Property Sheet View*

About the Rdbms Point Device Ext Manager View

The Rdbms Point Device Ext Manager View shown below, is the default view of the Rdbms Point Device Extension and the only "manager" view for the Rdbms Points Device Extension. It has a single Database pane that displays any Rdbms Point Queries that are present.



Name	Status	Last Update
RdbmsPointQuery	{unackedAlarm}	18-Jun-08 2:56:06 PM EDT
RdbmsPointQuery1	{unackedAlarm}	18-Jun-08 2:56:35 PM EDT
RdbmsPointQuery2	{unackedAlarm}	18-Jun-08 3:15:56 PM EDT

Using this manager view, you can do the following:

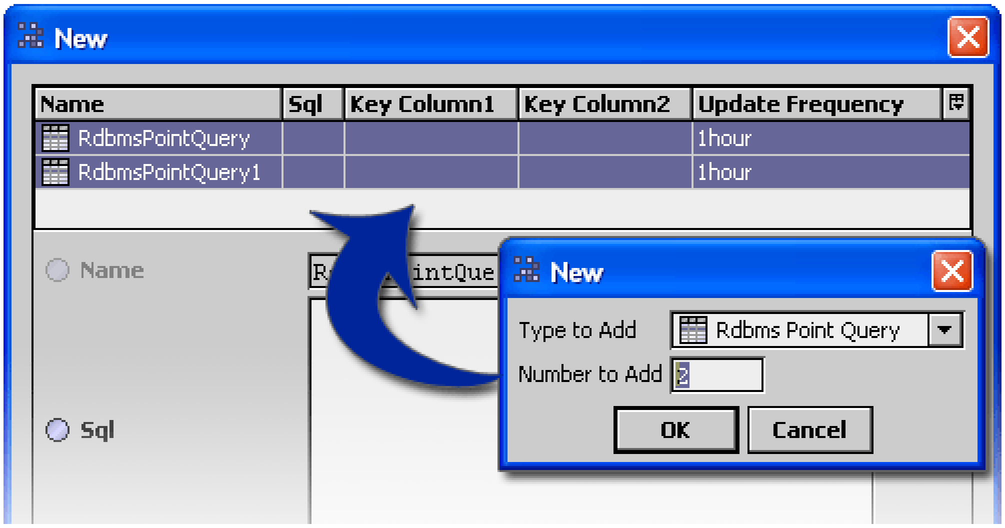
- Select one or more entries in the Database pane and use the popup (right-click) menu:

  Use the Cut, Copy, Duplicate, Rename, or other menu items to edit the rows. In addition, you can select individual rows and go to other views of a selected entry.

- Add new Rdbms Point Queries:

  Using the **New** button and "New" dialog box, you 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.



Name	Sql	Key Column1	Key Column2	Update Frequency
RdbmsPointQuery				1hour
RdbmsPointQuery1				1hour

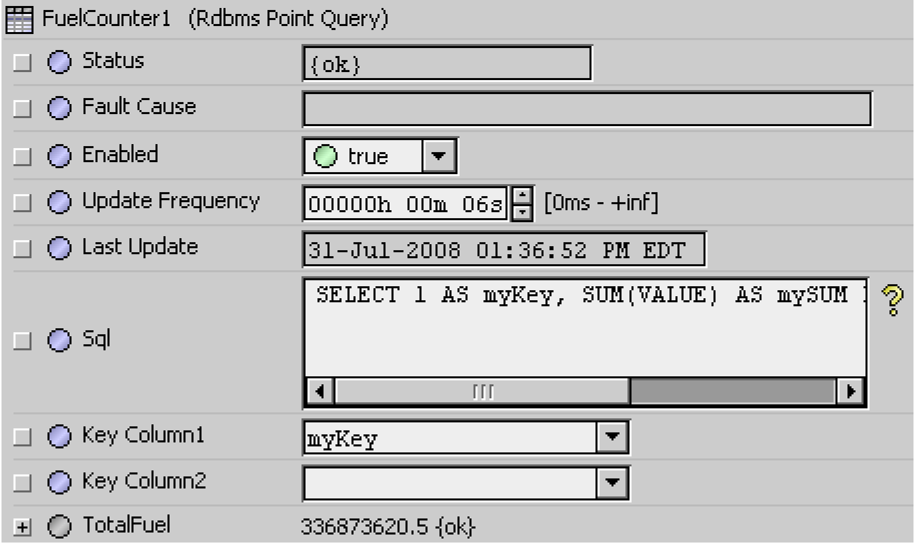
Type to Add	Rdbms Point Query
Number to Add	2

About the Rdbms Point Device Property Sheet View

The Rdbms Point Device Property Sheet view displays an entry for each Rdbms Point Query, as shown below and described in the following list.

NOTE: Note the following points about the Sql Query in the example:

- 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 first column is used.


FuelCounter1 (Rdbms Point Query)

Status	{ok}
Fault Cause	
Enabled	true
Update Frequency	00000h 00m 06s [0ms - +inf]
Last Update	31-Jul-2008 01:36:52 PM EDT
Sql	SELECT 1 AS myKey, SUM(VALUE) AS mySUM
Key Column1	myKey
Key Column2	
TotalFuel	336873620.5 {ok}

- Rdbms Point Query

A single entry for each query is listed in the property sheet view. Each valid query returns a set of data that can be added, as desired using the Rdbms Query Point Manager View. Each entry has the following two properties associated with it.

- SQL Query field

  This property is a large field that displays the text of the SQL Query (if any). You can edit and save the query from this view. You can also execute the query by right-clicking on the query and selecting **Actions** > **Execute** from the popup menu.

- Update Frequency field

  This property displays a time (in hours, minutes, and seconds) that indicates how often the associated query is automatically executed and the control points are updated.

About the Rdbms Point Query

The Rdbms Point Query is a container component and a child of the Rdbms Point Device Extension. You can use this component to query data in any database that you have access to and sufficient privileges on. The Sql property provides a field for you to write an Sql query statement in. You can execute this query manually and also set a regular interval time for updates using the **Update Frequency** property.

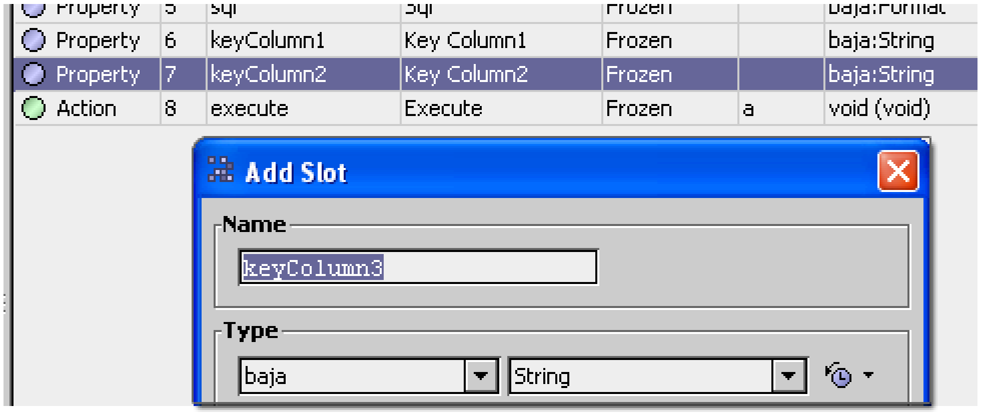
About Key Columns and Primary Keys

- Primary Key

  A primary key is used to uniquely identify each row in a database table. This key might be part of the data record itself (for example, a unique user id) or it can be an extra field that is not really related to the actual data record. A primary key can consist of one or more fields on a table. When multiple fields are used as a primary key, they create a "composite" key. The Key Column properties are provided in the Rdbms Point Query component for designation of a primary (or composite primary) key.

- Key Columns

  The key columns you define (using the Key Column1 and Key Column2 fields) may not actually be primary keys, although they often will be. If you do not define a key column, the first column in the row is automatically used as the key column. In some situations you may need two or more key columns to specify a unique key. If you need more than the two key column fields provided, you can add another key column slot from the RdbmsPointQuery Slot Sheet view, as shown below.



○ Property	5	sql	Sql	Frozen		baja:Format
○ Property	6	keyColumn1	Key Column1	Frozen		baja:String
○ Property	7	keyColumn2	Key Column2	Frozen		baja:String
○ Action	8	execute	Execute	Frozen	a	void (void)

Add Slot

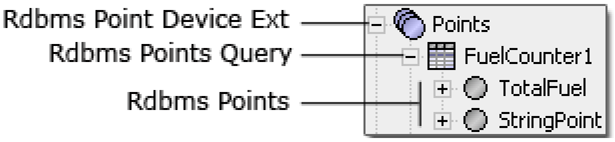
Name: keyColumn3

Type: baja String

An example situation where a single column cannot uniquely identify each row in a table might be a table of fan motor types where there is a column for "manufacturer", "model" and "maximum speed". In this case, in order 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.

Types of Rdbms Point Query Properties

An example Rdbms Point Device Ext used with a SqlServer database is shown below.



The Rdbms Point Query component properties are described in the following list:

- Status

  This is a read-only indication of the component's state.

- Fault Cause

  This is a read-only indication of the reason for a fault.

- Enabled

  This property allows you to enable (`true`) or disable (`false`) the execution of the query associated with this component.

- Update Frequency

  This value dictates how often the query is automatically executed.

- Last Update

  This value indicates the last time that the query was executed.

- Sql

  This property provides a field (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: The SQL field is actually BFormat. This means that additional syntax can be added to the SQL string that will be processed before the SQL is sent to the database.

- Key Column1

  It often (but not always) defines the primary key. Use this property together with Key Column2, if necessary, to establish the unique "composite key". See *About Key Columns and Primary Keys*.

- Key Column2

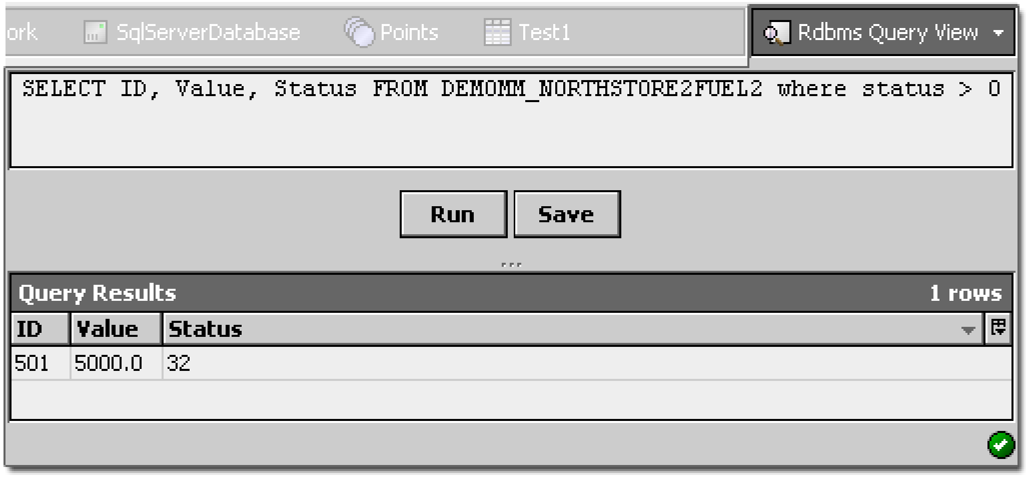
  Use this field if Key Column1 is not enough to establish a unique Key for the imported data. See *About Key Columns and Primary Keys*.

About the Rdbms Query View

This view 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 dialog box displays an error message.

NOTE: This view executes on display. So anytime that you view this screen the query executes the saved query and displays results in the Query Results Pane.

The Rdbms Point Query View has two panes and two buttons, as shown and described below:



```
SELECT ID, Value, Status FROM DEMOMM_NORTHSTORE2FUEL2 where status > 0
```

ID	Value	Status
501	5000.0	32

- Query Pane

  This top pane is a text editor field that allows you to type a query directly into the field. Any "saved" changes that you make in this field are also reflected in the Sql field of the Rdbms Point Query component. Saved changes in the Rdbms Point Query component property sheet are also reflected here.

- Query Results Pane

  This pane displays any data returned by the query (as typed in the Query Pane) any time that you click the **Run** button.

- Run button

  This button executes the query (as typed in the Query Pane) whenever it is clicked.

- Save button

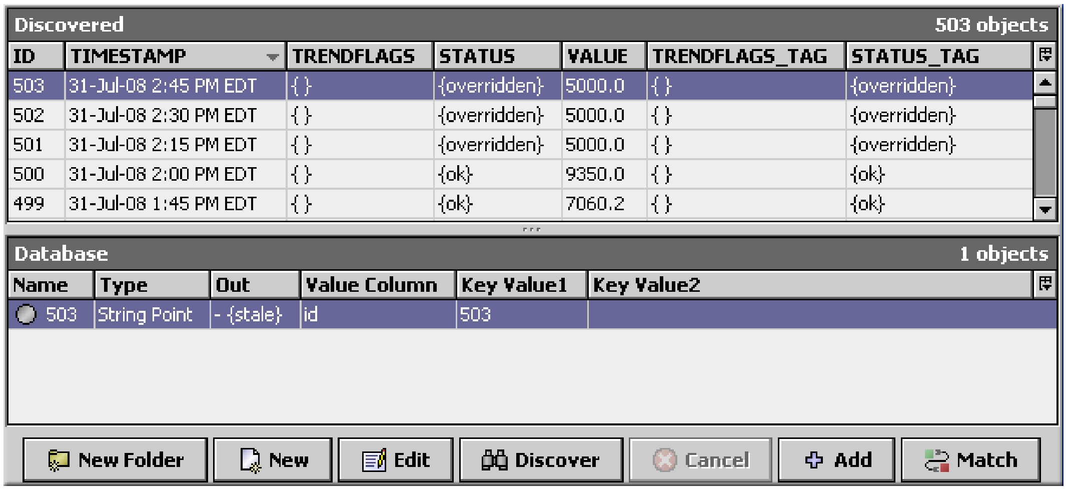
  This button saves the query (as typed in the Query Pane).

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

About the Rdbms Point Query Manager View

The Rdbms Point Query Manager is the default view for the Rdbms Point Query component under the Rdbms Point Device Extension. 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. It is shown and described below.



Discovered 503 objects

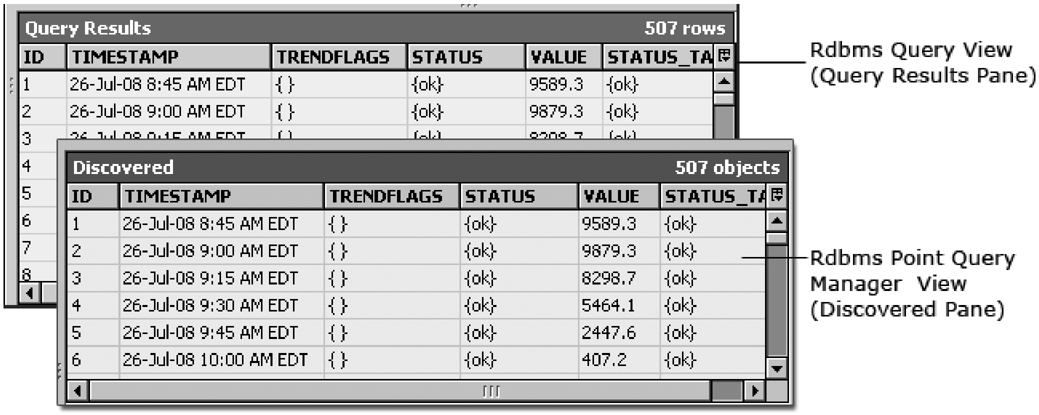
ID	TIMESTAMP	TRENDFLAGS	STATUS	VALUE	TRENDFLAGS_TAG	STATUS_TAG
503	31-Jul-08 2:45 PM EDT	{ }	{overridden}	5000.0	{ }	{overridden}
502	31-Jul-08 2:30 PM EDT	{ }	{overridden}	5000.0	{ }	{overridden}
501	31-Jul-08 2:15 PM EDT	{ }	{overridden}	5000.0	{ }	{overridden}
500	31-Jul-08 2:00 PM EDT	{ }	{ok}	9350.0	{ }	{ok}
499	31-Jul-08 1:45 PM EDT	{ }	{ok}	7060.2	{ }	{ok}

Database 1 objects

Name	Type	Out	Value Column	Key Value1	Key Value2
503	String Point	- {stale}	id	503	

- Discovered Pane

  This is the top pane in view. It displays all the results of the "Discovery" action that is executed when the Discover button is clicked. Discovered points are presented in the top "Discovered" pane and may be added to a "Database" pane for inclusion under the "Points" node of the appropriate RdbmsNetwork database. Discovered points represent the results of the Rdbms Point Query execution. 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. An example of this is shown below.



Query Results 507 rows

ID	TIMESTAMP	TRENDFLAGS	STATUS	VALUE	STATUS_TA
1	26-Jul-08 8:45 AM EDT	{ }	{ok}	9589.3	{ok}
2	26-Jul-08 9:00 AM EDT	{ }	{ok}	9879.3	{ok}
3					
4					
5					
6					
7					
8					

Discovered 507 objects

ID	TIMESTAMP	TRENDFLAGS	STATUS	VALUE	STATUS_TA
1	26-Jul-08 8:45 AM EDT	{ }	{ok}	9589.3	{ok}
2	26-Jul-08 9:00 AM EDT	{ }	{ok}	9879.3	{ok}
3	26-Jul-08 9:15 AM EDT	{ }	{ok}	8298.7	{ok}
4	26-Jul-08 9:30 AM EDT	{ }	{ok}	5464.1	{ok}
5	26-Jul-08 9:45 AM EDT	{ }	{ok}	2447.6	{ok}
6	26-Jul-08 10:00 AM EDT	{ }	{ok}	407.2	{ok}

- Discover button

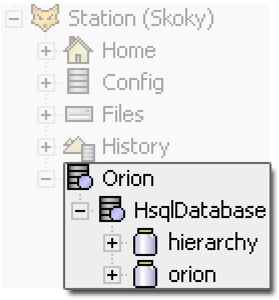
  This button executes the query that is saved as a the value of the Rdbms Point Query "Sql" property. Only data points that meet the parameters of the query are displayed in the Discovered pane. As with the standard Point Manager view, points that appear in the Discovered pane may be added to the Database pane using the Add or Match buttons.

- New Folder, New, Edit, Cancel, Add, Match

  These buttons behave the same way as in the standard Points Manager view, as described in the *NiagaraAX Drivers Guide.*

About Orion

Available starting in NiagaraAX-3.4, 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 Niagara objects. As a new mapping architecture for Niagara components, Object-Relational Mapping does not replace the "config.bog" (station) file but provides for the creation of a new Niagara "space" (the Orion space) that is stored outside the station database file (like histories, files, and modules). This new functionality includes a means for alternative and multiple system hierarchy displays that can be used for data presentation, system identification, and navigation.



As part of this module (orion), the Orion API includes the following:

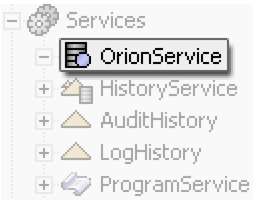
- BRdbms

  this is the RDBMS "driver" that existed in earlier (before NiagaraAX-3.4) versions of NiagaraAX to model an Rdbms database for supporting SQLServer and Oracle database implementations. Applications (including stations running on smaller JACEs) are able to implement O/R mapping and maintain database independence by running a local instance of any of the following RDBMS types:

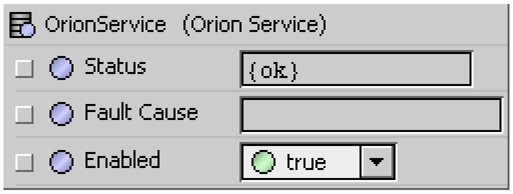
  - HSQLDB
  - DB2
  - MySQL
  - Oracle
  - SQL Server

- BOrionService

  this is the Niagara service, shown below that enables the Orion database in a station and allows the station and its applications to use the Orion database.



The Orion service has a **Status**, **Fault Cause**, and **Enabled** properties, as shown below.



OrionService (Orion Service)

□ ○ Status	{ok}
□ ○ Fault Cause	
□ ○ Enabled	○ true

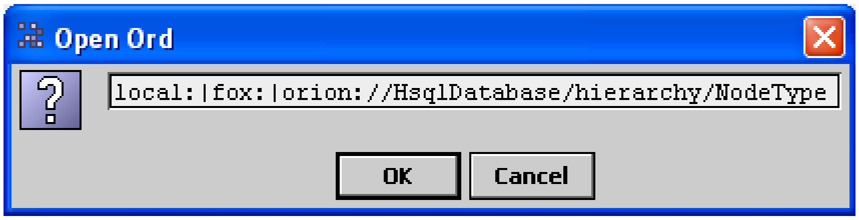
This service allows a station and its applications to use a local relational database running in controllers (including embedded JACEs) to manage and display distributed-system (or distributed-application) information. This managed and configurable information includes certain types of Niagara component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

- BIOrionApp

  this is an installable application that has a set of Orion object types that allow you to manage data in the Orion database.

- BOrionSpace

  this is the component space for storing Orion objects. Other types of "space" include "History", "File", and "Virtual" space. An example ORD using the orion space is shown below.



```
local:|fox:|orion://HsqlDatabase/hierarchy/NodeType
```

- BOrionObject

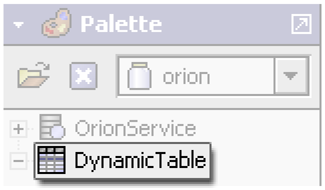
  this is a common subclass for all objects stored in the Orion database. It represents an object associated with Orion that also knows what database it is associated with.

- OrionSession

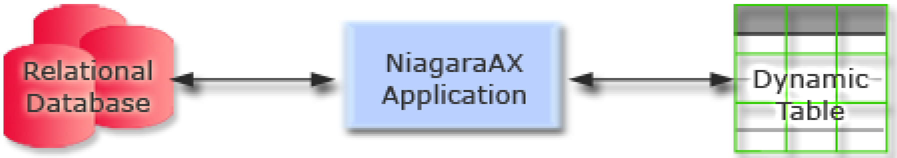
  this is a CRUD (Create, Read, Update and Delete) interface for interacting with the Orion Database and managing object persistence

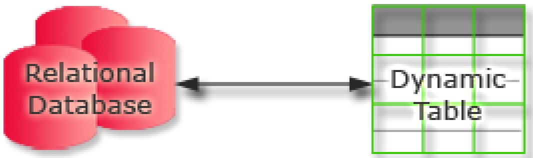
About the Dynamic Table

The DynamicTable component allows you to present relational data in a table format in NiagaraAX-3.4 or later. The DynamicTable component shown below, is located in the Orion module and available on the orion palette.



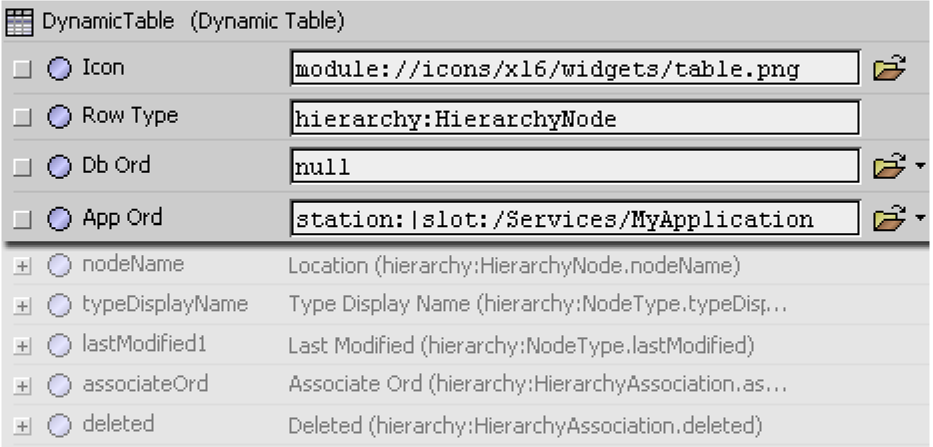
You can drag a DynamicTable component from the orion palette to a location in your station and then configure the component to display data from either an application that is using relational data or directly from a relational database, as shown below.





Types of Dynamic Table properties

The DynamicTable component has the following properties, visible from the Property Sheet view, as shown below.



DynamicTable (Dynamic Table)

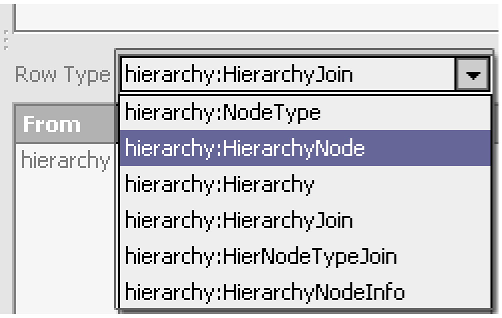
□ Icon	module://icons/x16/widgets/table.png
□ Row Type	hierarchy:HierarchyNode
□ Db Ord	null
□ App Ord	station:|slot:/Services/MyApplication
⊞ nodeName	Location (hierarchy:HierarchyNode.nodeName)
⊞ typeDisplayName	Type Display Name (hierarchy:NodeType.typeDisp...
⊞ lastModified1	Last Modified (hierarchy:NodeType.lastModified)
⊞ associateOrd	Associate Ord (hierarchy:HierarchyAssociation.as...
⊞ deleted	Deleted (hierarchy:HierarchyAssociation.deleted)

- Icon

  This property allows you to designate a graphic icon to associate with the Dynamic Table.

- Row Type

  This property specifies the row in terms of module and type (module:type). You can use the Row Type option list in the Dynamic Table Config view to set the row information in this field as shown below.



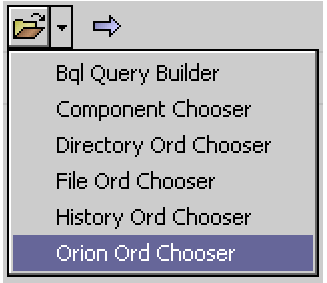
Row Type hierarchy:HierarchyJoin

From
hierarchy

hierarchy:NodeType
hierarchy:HierarchyNode
hierarchy:Hierarchy
hierarchy:HierarchyJoin
hierarchy:HierNodeTypeJoin
hierarchy:HierarchyNodeInfo

- Db Ord

This property allows you to specify an ord path to the relational database. Either this field or the App Ord field (but not both) are required for linking the DynamicTable to a database. With the Orion service installed, you can use the Orion chooser to choose the database under the Orion node, as shown below.



- App Ord

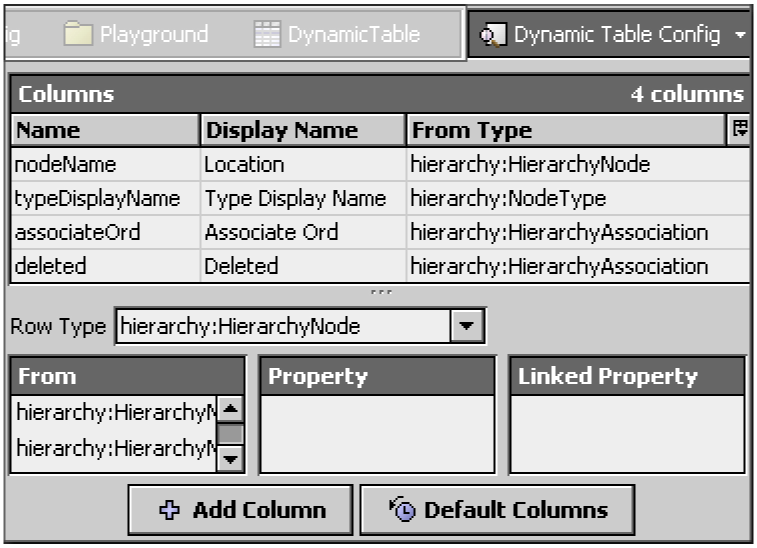
  This property allows you to specify an ord path to an application. Either this field or the Db Ord field (but not both) are required for linking the DynamicTable to a database.

Types of Dynamic Table views

In addition to the standard Slot Sheet, Property Sheet, and other views, the following views are associated with the DynamicTable component:

- Dynamic Table Config view

  This view, shown below 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 lower pane contains three selection windows and a Row Type option list that you can use to add specific columns to the table pane using the **Add Column** button. The **Default Columns** button adds columns to the upper pane based on Row Type selection.



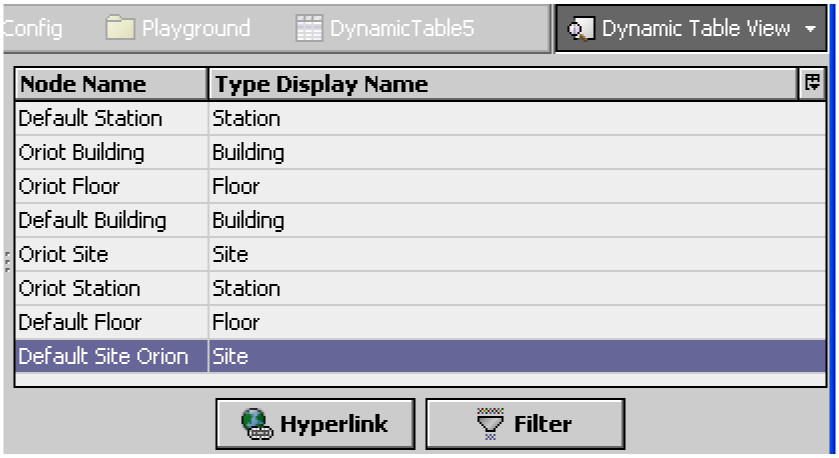
Name	Display Name	From Type
nodeName	Location	hierarchy:HierarchyNode
typeDisplayName	Type Display Name	hierarchy:NodeType
associateOrd	Associate Ord	hierarchy:HierarchyAssociation
deleted	Deleted	hierarchy:HierarchyAssociation

Row Type hierarchy:HierarchyNode

From	Property	Linked Property
hierarchy:HierarchyN hierarchy:HierarchyN		

- Dynamic Table view

  This view shown below, displays the table of data according to row and column specifications made in the Dynamic Table Config view. 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** dialog box. The **Hyperlink** button, when clicked changes the view to the Property Sheet view of the (single) node that you have selected.



Node Name	Type Display Name
Default Station	Station
Oriot Building	Building
Oriot Floor	Floor
Default Building	Building
Oriot Site	Site
Oriot Station	Station
Default Floor	Floor
Default Site Orion	Site

Chapter 3 Common RdbmsNetwork tasks

The following main sections provide information related to some of the common uses of the RdbmsNetwork:

- *Prerequisites*
- *Creating Rdbms proxy points*
- *Using an Rdbms for NiagaraAX alarm database*
- *Importing and exporting data using the RdbmsNetwork*
- *Upgrading existing Rdbms databases to support Unicode and UTC*
- *Upgrading existing Orion databases to support Unicode*

Prerequisites

The following list includes the prerequisites for successfully installing and using the RdbmsNetwork driver and specific supported rdb database types:

- NiagaraAX license feature for the appropriate Rdbms database. See *Prerequisites*.
- RdbmsNetwork installed in the station (under the "Drivers" node). See *Installing the RdbmsNetwork driver and rdbDatabase device*.
- RdbmsNetwork database device installed under the RdbmsNetwork. See *Installing the RdbmsNetwork driver and rdbDatabase device*.
- IP Network connection to the database host
- User Name and Password for database login
- Appropriate rights for required data access

In addition, there may be specific database-related items that you need to know. For example, if you are trying to connect to a "named instance" of a database, you may need to know the name of the database instance. Also, if the the database is using a non-default port number, you need to know the port number so you can configure your RdbmsNetwork database device correctly.

Creating Rdbms proxy points

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

NOTE: The Rdbms Point Query Manager works differently than in 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 view, as well.

The process of creating Rdbms proxy points can be divided into three procedures, as listed below:

- *Add and Configure the Rdbms Point Query*
- *Discover Rdbms points*
- *Add Rdbms points*

Add and Configure the Rdbms Point Query

NOTE: In the following procedure, the term "rdb*Database"* represents any valid Rdbms Database Device Extension

To add and configure the Rdbms Point Query component, do the following:

Step 1 Make sure that you have reviewed and accomplished all items listed under *Prerequisites*

Step 2 In the WorkbenchAX nav tree, under the station Driver's node , expand the RdbmsNetwork node and the desired rdb*Database* node to expose the Rdbms Point Device Ext node

Step 3 In the nav tree, double-click on the Rdbms Point Device Ext node. The Rdbms Point Device Ext Manager view appears.

Step 4 At the bottom of the Rdbms Point Device Ext Manager view, click the **New** button. The New dialog box displays with the following two properties:

- Type to Add: The only choice here is the default (Rdbms Point Query) - that is what you are adding.
- Number to Add: Type in the number of Rdbms Point Query components that you want to add.

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

Step 5 Click the **OK** button at the bottom of the dialog box. A second New dialog box appears with the following properties:

- Rdbms Point Query pane

  This pane lists a number of Rdbms Point Query components (equal to the number you chose to add in the previous step). You can edit the properties for each entry separately or batch-edit most of the properties (if desired) by selecting one or more rows in the pane and editing them in the lower part of the dialog box.

- Name

  A single default name appears in this field. You can accept the default name(s) or edit them one at a time by selecting a single entry in the pane and editing it using this field. If more than one entry in the pane is selected, the Name field is not available.

- Sql

  Enter a query in this field, or leave it blank for later editing in the Rdbms Point Query Property Sheet view. If the query is not valid, the Point Query view does not display.

- Key Column1

  Identify a key column for the returned data (see About the Rdbms Point Device Extension).

- Key Column2

  If necessary, identify a second key column (usually this is not necessary, see About the Rdbms Point Device Extension).

- Update Frequency

  Set the time for how often you want the query to execute automatically. You can always manually execute an update.

Step 6 Click the **OK** button. The new Rdbms Point Query component(s) are added under the Points Device Extension node in the nav tree and also appear in the Rdbms Point Device Ext Manager view.

Discover Rdbms 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.

To discover Rdbms points, do the following:

Step 1 Double-click on the desired Rdbms Point Query component. The Rdbms Point Query Manager view displays.

Step 2 In the Rdbms Point Query Manager, 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.

Add Rdbms points

To add Rdbms points, do the following:

Step 1 In the Discovered pane, select one or more points to add as proxy points.

NOTE: The Point Manager's **Add** button is available when you have one or more data items selected (highlighted) in the top Discovered pane. Also, 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** dialog box.

Step 2 At the bottom of the view, click the Add button. The Add dialog box appears, with all selected points in the top pane of the dialog box.

Step 3 In the **Add** dialog box, edit the following properties, as desired:

- Name

  Individually edit the Name property, if desired, by selecting a single point and typing in this field. This is the display name of the proxy point, when added.

- Type

  Individually edit (or batch-edit) this property for all points in the top pane, as desired. Once the Point type is designated, it cannot be changed without deleting and re-adding the point.

- Value Column

Individually edit (or batch-edit) this property for all points in the top pane, as desired. The value column is the table column that holds the value that you want the proxy point to display.

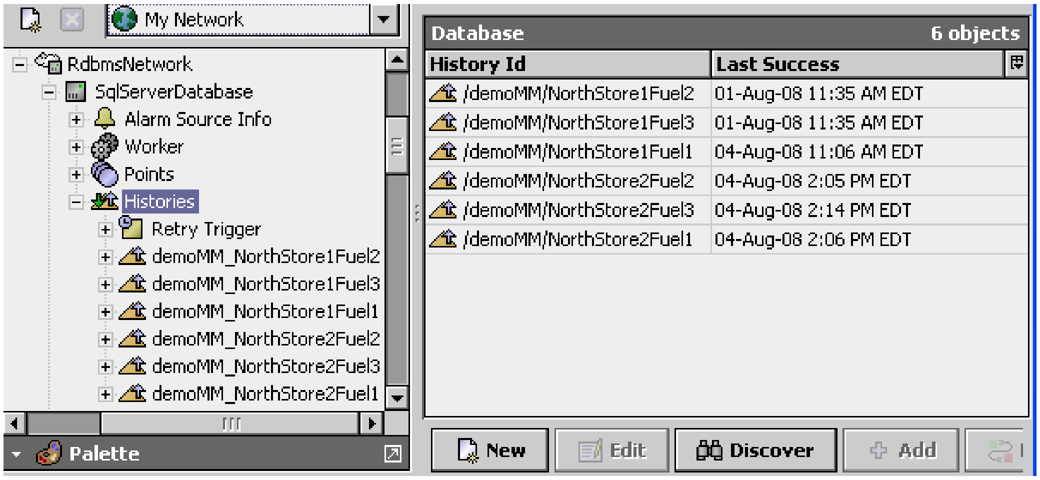
Step 4 Click the **OK** button. The points are added to the Database pane and appear in the nav tree.

Example Rdbms implementation

NOTE: The following is a hypothetical example to illustrate 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. In order to graphically display updated information over the Internet, they use the NiagaraAX RdbmsNetwork and the Point Device Extension, as illustrated in the following list of screen captures:

- Each store exports transaction histories from its JACE via a Supervisor, to a central SqlServer database using the Sql Server History Device Ext and the History Export Manager view, as shown below.

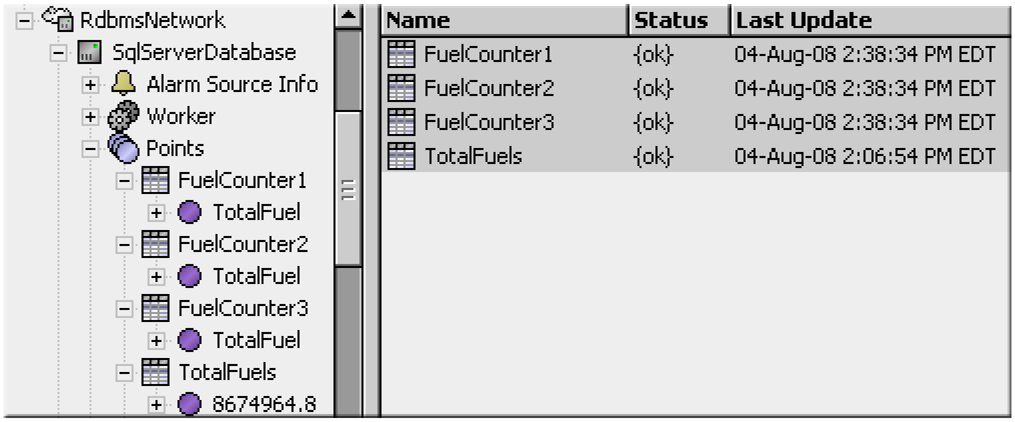


Database 6 objects

History Id	Last Success
/demoMM/NorthStore1Fuel2	01-Aug-08 11:35 AM EDT
/demoMM/NorthStore1Fuel3	01-Aug-08 11:35 AM EDT
/demoMM/NorthStore1Fuel1	04-Aug-08 11:06 AM EDT
/demoMM/NorthStore2Fuel2	04-Aug-08 2:05 PM EDT
/demoMM/NorthStore2Fuel3	04-Aug-08 2:14 PM EDT
/demoMM/NorthStore2Fuel1	04-Aug-08 2:06 PM EDT

NOTE: For embedded JACEs, the HsqlDbDatabase is the only database that is supported.

- For each store, an AXSupervisor station creates proxy points for each fuel type, by creating and configuring an Rdbms Point Query for each fuel type, shown below.

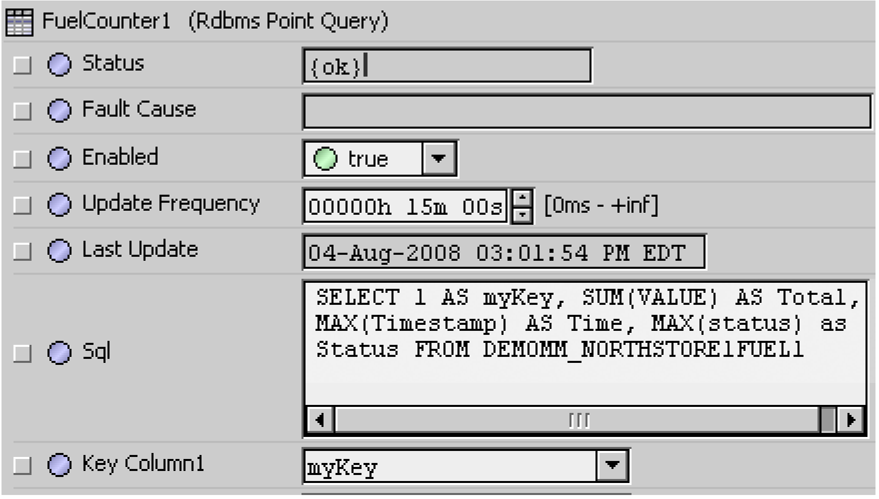


Name	Status	Last Update
FuelCounter1	{ok}	04-Aug-08 2:38:34 PM EDT
FuelCounter2	{ok}	04-Aug-08 2:38:34 PM EDT
FuelCounter3	{ok}	04-Aug-08 2:38:34 PM EDT
TotalFuels	{ok}	04-Aug-08 2:06:54 PM EDT

- An Sql query for total fuel sold as of the latest update is shown below, scheduled to update every 15 minutes. This query provides a total that is updated every 15 minutes, along with the time of the update.

  In this example, the following Sql parameters are optional:

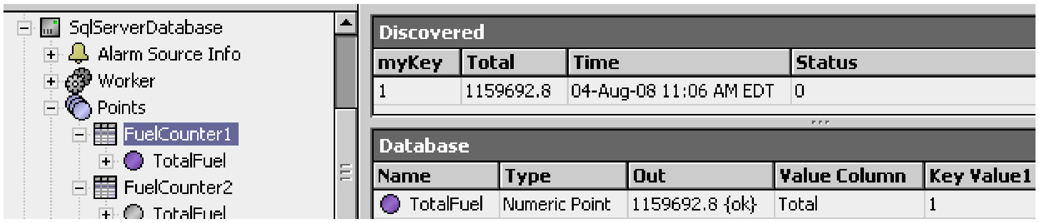
  - “1 AS myKey”: this creates a new column as a key column and is not required
  - “1 AS Total”, “AS Time”, “AS Status”: these create titles for there respective columns. If these optional parameters are not used, columns are titled “column1”, “column2”, “column3”, respectively.



FuelCounter1 (Rdbms Point Query)

□ Status	{ok}
□ Fault Cause	
□ Enabled	true
□ Update Frequency	00000h 15m 00s [0ms - +inf]
□ Last Update	04-Aug-2008 03:01:54 PM EDT
□ Sql	SELECT 1 AS myKey, SUM(VALUE) AS Total, MAX(Timestamp) AS Time, MAX(status) as Status FROM DEMOMM_NORTHSTORE1FUEL1
□ Key Column1	myKey

Data is "Discovered" and "Added" under the Rdbms Point Device Extension node, as shown below:



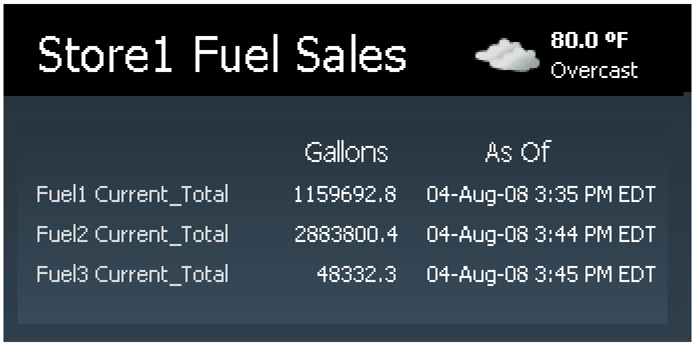
Discovered

myKey	Total	Time	Status
1	1159692.8	04-Aug-08 11:06 AM EDT	0

Database

Name	Type	Out	Value Column	Key Value1
TotalFuel	Numeric Point	1159692.8 {ok}	Total	1

- 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.



Store1 Fuel Sales

80.0 °F
Overcast

	Gallons	As Of
Fuel1 Current_Total	1159692.8	04-Aug-08 3:35 PM EDT
Fuel2 Current_Total	2883800.4	04-Aug-08 3:44 PM EDT
Fuel3 Current_Total	48332.3	04-Aug-08 3:45 PM EDT

Using an Rdbms for NiagaraAX alarm database

This section describes how to use an RDBMS (SQL Server, Oracle, DB2, or, with NiagaraAX-3.4, MySQL) as the "alarm database" for a NiagaraAX station. This feature has been added (and requires) NiagaraAX-3.2 or higher.

NOTE: Use of an RDBMS to store NiagaraAX alarm records is different than exporting histories to an RDBMS. Exported histories also still 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`).

Setup the RdbAlarmService

NOTE: This procedure describes how to setup the RdbAlarmService using NiagaraAX-3.4. In NiagaraAX-3.2 and 3.3, individual database-specific alarmService components are available in the alarmRdb module and should be copied to the Services folder in Step 2, below.

To setup the RdbAlarmService, do the following:

Step 1 Install and configure the RdbmsNetwork driver and test the connection to your desired database, as described in *Installing the RdbmsNetwork driver and rdbDatabase device.*

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

NOTE: If you need 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 3 Delete the old (standard) AlarmService component from the station's Services folder.

NOTE: Existing alarms are not migrated to the new data store.

Step 4 Right-click on the rdbAlarmService in the nav tree and select **View Property Sheet**.
The rdbAlarmService property sheet view displays.

Step 5 In the rdbAlarmService property sheet, select the desired rdbms database device type from the Driver option list.

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

Step 6 Click the **Save** button at the bottom of the view.

Once you complete this final step, the new RdbAlarmService is available for use.

Importing and exporting data using the RdbmsNetwork

Each of the following Rdbms Device extensions has an associated History Device Extension that you can configure and use to perform exports and imports of data between Rdbms database and NiagaraAX history files:

- Db2
- MySQL
- Oracle
- SqlServer

You can use the History Import Manager or History Export Manager views to specify, configure, and execute the importing or exporting of files. For more detailed information about these views and importing and exporting histories, in general, refer to the "History Import Manager" and "History Export Manager" sections of the *NiagaraAX-3.x Drivers Guide*.

The following procedures describe how to import and export data between an Rdbms database and history files:

- *Export history data to an Rdbms database*
- *Import history data to an Rdbms database*

Export history data to an Rdbms database

This procedure describes how to export history data to any of the following Rdbms database types:

- Db2
- MySQL
- Oracle
- SqlServer

The term *rdbDatabase* in the following steps, refers to any of the database types listed above.

To export history data to an Rdbms database, do the following:

Step 1 In WorkbenchAX, establish a connection to the desired database, as described in *Installing the RdbmsNetwork driver and rdbDatabase device.*

Step 2 In the WorkbenchAX nav tree, under the station **DriversRdbmsNetworkrdbDatabase** node, double-click on the Histories node. The History Export Manager view displays.

Step 3 Click the **New** button at the bottom of the History Export Manager view. The **New** dialog box displays.

NOTE: As an alternative to the **New** button and dialog box, you can click the **Discover** button and browse to find and select the desired history file to export.

Step 4 From the **New** dialog box, select the desired *rdbDatabase* type from the “Type to Add” option list and type in the number of export descriptors to add in the “Number to Add” field.

NOTE: You need to add one unique export descriptor for each history file that you want to export.

Click the **OK** button. A second **New** dialog box appears with one or more export descriptors in the top pane.

Step 5 In the **New** dialog box, edit the following fields and click the **OK** button:

- Name

  Type in a meaningful name for this export descriptor.

- History Id

  Enter the station name and history name into these two fields

- Execution Time

  Select an execution mode from the option list. For Daily or Interval modes, choose the desired execution parameters for the specific mode.

  NOTE: If you are handling large volumes of data it is good practice to make use of the `Randomization` option when setting **Execution Time**, to offset the data exports.

- Enabled

  Select `true` to enable or `false` to leave the export descriptor disabled.

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

Step 6 Initiate an export action by doing any one of the following:

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

The status and time of the last export action is displayed in the Database pane, “Status” and “Last Success” columns, respectively. Also, each export descriptor appears under the **rdbmsDatabase Histories** node in the nav tree.

Import history data to an Rdbms database

This procedure describes how to import history data to any of the following Rdbms database types:

- Db2
- MySQL
- Oracle
- SqlServer

In the following steps, the term *rdbDatabase* refers to any of the database types listed above.

To import history data into an Rdbms database, do the following:

Step 1 In WorkbenchAX, establish a connection to the desired database, as described in *Installing the RdbmsNetwork driver and rdbDatabase device*.

Step 2 In the WorkbenchAX nav tree, under the station **DriversRdbmsNetworkrdbDatabase**, right-click on the Histories node and select **Views Rdbms History Import Manager**. The Rdbms History Import Manager view displays.

Step 3 Click the **New** button at the bottom of the Rdbms History Import Manager view. The **New** dialog box displays.

NOTE: As an alternative to using the **New** button and dialog box, you can click the **Discover** button to populate the Discovered pane with available data tables, then select the desired tables and click the **Add** button at the bottom of the view. In this case an **Add** dialog displays that is similar to the **New** dialog box, described in Step 6, below.

Step 4 From the **New** dialog box, select *Rdb History Import* from the “Type to Add” option list and type in the number of import descriptors to add in the “Number to Add” field.

NOTE: You need to add one unique import descriptor for each history file that you want to import. If you used the **Discover** and **Add** buttons to populate the Discovered pane, then you do not see this dialog box.

Step 5 Click the **OK** button. A second **New** dialog box appears with one or more import descriptors in the top pane.

Step 6 In the **New** dialog box, edit the following fields and click the **OK** button:

NOTE: If you used the **Discover** and **Add** buttons instead of the New button, you are using the **Add** dialog box in this step and the Name and History Id fields are already filled in.

- Name

  Type in a meaningful name for each import descriptor.

- History Id

Enter a station name and history name into these two fields.

- Execution Time

  Select an execution mode from the option list. For Daily or Interval modes, choose the desired execution parameters for the specific mode.

- Enabled

  Select `true` to enable or `false` to leave the import descriptor disabled.

- Capacity

  Set the desired maximum number of records to import.

- Full Policy

  This property allows you to choose what to do when the Capacity number is reached. The `Roll` option drops off the oldest record to make room for the newest record. The `Stop` option simply causes the history to stop recording.

- Interval

  This option allows you to choose between the following two options for setting a data import frequency.

  - irregular

    This option does not specify a particular frequency for imports.

  - regular

    This option allows you to set an import action frequency in terms of hours, minutes, and seconds.

- Value Facets

  This property allows you to set units for the “Value Column” in the imported table.

- Time Zone

  Use this property to set the desired import-location time zone.

- Rdb Table Name

  Type in the name or click on the search and replace icon to use the **Batch Search and Replace** dialog box to redefine the table name on import.

- Rdb Catalog Name

  Type in the name or click on the search and replace icon to use the **Batch Search and Replace** dialog box to redefine the catalog name on import.

- Rdb Schema Name

  Type in the name or click on the search and replace icon to use the **Batch Search and Replace** dialog box to redefine the schema name on import.

- Timestamp Column

  Specify the Timestamp column for the imported data. If you did a Discover, then the table columns are displayed in the option list for you to choose from. Otherwise, type in the column name in the text field.

- Value Column

Specify the Value column for the imported data. If you did a Discover, then the table columns are displayed in the option list for you to choose from. Otherwise, type in the column name in the text field.

- Status Column

  Select or clear the option box to specify if a status column is included in the imported data or not. If the **None** box is cleared, then specify the status column for the imported data. If you did a Discover, then the table columns are displayed in the option list for you to choose from. Otherwise, type in the column name in the text field.

- Query Predicate

  This field allows you to insert a query predicate to filter the records that are imported. For example, you could type in "`where Value > 100`" to import only those records that have a "`Value`" that is greater than 100. Or, you could type in "`where Value between 1 and 100`" to import those records with Values between 1 and 100.

- Full Import On Execute

  Select `Enabled` to import the full database (up to the specified limit) on each successive import action. Select `Disabled` to import (and append) only the new data on each successive import action.

After clicking the **OK** button, the new history import descriptor(s) appears in the Database (lower) pane.

Step 7 Initiate an import action by doing one of the following:

- Select one or more history descriptors in the database pane and click the **Archive** button.
- Right-click on a single history descriptor in the database pane and select **Actions Execute**.
- Using the Daily or Interval settings, as set in the **New** or **Edit** dialog box, allow the import to occur, as scheduled.

The status and time of the last import action is displayed in the Database pane, "Status" and "Last Success" columns, respectively. In addition, the imported histories appear in the nav tree under the History node, organized by device .

Upgrading existing Rdbms databases to support Unicode and UTC

By default, from NiagaraAX-3.8, all Rdbms databases have a property called **Use Unicode Encoding Scheme**. The value of this property defaults to `false` to support existing Rdbms databases. If you want the full UTF-8 Unicode support (NVARCHAR columns instead of VARCHAR columns), set this property to `true` before you connect to your database for the first time. In addition, from NiagaraAX-3.8, most Rdbms databases have a property called **Timestamp Storage** allowing the export or update of history timestamps to Coordinated Universal Time (UTC). The value of this property defaults to `Dialect Default` to maintain backwards compatibility with the existing history export mechanism.

This section describes how to upgrade one or more existing Rdbms databases to support Unicode and/or UTC. An **Update Wizard** is available to help you easily perform each upgrade using the following steps:

NOTE: The upgrade of Rdbms Database's to support either Unicode or UTC using the **Update Wizard** is described in separate procedures below although both upgrades may be carried out in the same wizard session.

Upgrading existing Rdbms databases to support Unicode

Step 1 Right-click on the RdbmsNetwork in the nav tree and select **Update Wizard**. The Database Update Wizard dialog displays.

Step 2 Select **Update text column encodings to UTF-8 (Unicode)** in the Update Procedures dialog. Click the **Next** button at the bottom of the dialog.

Step 3 Select one or more databases to update to UTF-8 (Unicode). Click the **Next** button at the bottom of the dialog.

Step 4 The wizard completes and displays its results for review.

Upgrading existing Rdbms databases to support UTC

Step 1 Right-click on the RdbmsNetwork in the nav tree and select **Update Wizard**. The Database Update Wizard dialog displays.

Step 2 Select **Update timestamp columns to UTC (Coordinated Universal Time)** in the Update Procedures dialog. Click the **Next** button at the bottom of the dialog.

Step 3 Select one or more databases to update to UTC. Click the **Next** button at the bottom of the dialog.

Step 4 Specify the current policy being used to store database timestamps by selecting either the **Timestamps are being stored using the station timezone** or **Timestamps are being stored using the history config timezone** options. Click the **Next** button at the bottom of the dialog.

Step 5 The wizard completes and displays its results for review.

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** will process the database tables and change the data types of string valued columns to the NVARCHAR data type and will adjust the database timestamps to UTC time. Visibility on the progress of individual jobs is provided within the wizard itself and can be viewed from the Job Service at a later stage as well. The update operations are atomic. This implies they either complete entirely or will be rolled back entirely if any error occurs.

CAUTION: The Unicode and UTC upgrade procedures are one-way only and cannot be reversed. It is highly recommended that you back up the database prior to running the upgrades.

NOTE: If either or both upgrade procedures (Unicode and UTC) are run, the wizard automatically sets both the **Use Unicode Encoding Scheme** and **Timestamp Storage** properties on the database device property sheet to `true` and `Utc Timestamp` respectively. The **Timestamp Storage** property will also become read-only and can only be made writable again by right clicking on the database device and selecting **Actions Allow Dialect Modifications**.

Upgrading existing Orion databases to support Unicode

If you have an existing Orion database without UTF-8 support, use the **OrionMigrator**, component in the orion module to migrate your data into a new Rdbms database device. Set the **Use Unicode Encoding Scheme** property to `true` before you setup the **OrionMigrator**.

Follow this procedure to upgrade existing Orion databases to support Unicode:

Step 1 Before using the **OrionMigrator**, first make sure you have 2 working Rdbms devices. One of those Rdbms devices will be the one already in use for your Orion App(s) and the other will be a new database.

Step 2 Open the orion palette, drag and drop the **OrionMigrator** component into the station Services Container.

Step 3 Modify the `Old Db Ord` option to point to the Rdbms Device that you are currently using for your Orion App(s).

Step 4 Modify the `New Db Ord` option to point to the new Rdbms Device that you want to use for your Orion App(s).

Step 5 If you plan to migrate all Orion Apps to the same Rdbms Device, then leave the `App Ord List` option blank. This will also move the 'OrionAudit' and 'OrionAppVersion' tables to the New DB as well.

Step 6 Choose one of these `Migration Type` options based on the following:

- `New Database` — optimized for speed but throws an exception if any record already exists.
- `Insert Only` — if a record exists then skip it.
- `Overwrite` — if a record exists, overwrite it and if the update fails, stop the migration.
- `Force Overwrite` — if a record exists, overwrite it and ignore all errors.

NOTE: If you are using the **OrionMigrator** with an Enterprise Security station and you have an **Intrusion Zone** in the station, then you must use the `Overwrite` option instead of `New Database`.

Step 7 Restart the station to get the **OrionMigrator** into an"`{Ok}`" status. Right click the **OrionMigrator** and select **Actions Migrate**.

If successful, all migrated Orion Apps will have their database Ord changed to point to the new Rdbms device. You will be told how many items were migrated.

If the **OrionMigrator** fails, your Orion Apps will continue to function in the old Rdbms Database. No database Ords will change. There may be some data in the new database. You should repeat the procedure but use the `Force Overwrite` option to complete the transfer.

Step 8 When finished, remove the **OrionMigrator** component from the station Services Container and restart the station to complete the procedure.

Chapter 4 Use Examples

The following main sections describe some use examples of the RdmsNetwork.

- *Schema Description*
- *Exporting by History ID*
- *Exporting by History Type*
- *Understanding Status and Trend Flags*
- *Converting Unix time to readable date format in MySQL*

Schema Description

Two export schema options exist, by ID or by Type.

If they have not been manually created by the Database Administrator (DBA), all the necessary tables are created automatically by the RdbmsNetwork as each history is exported.

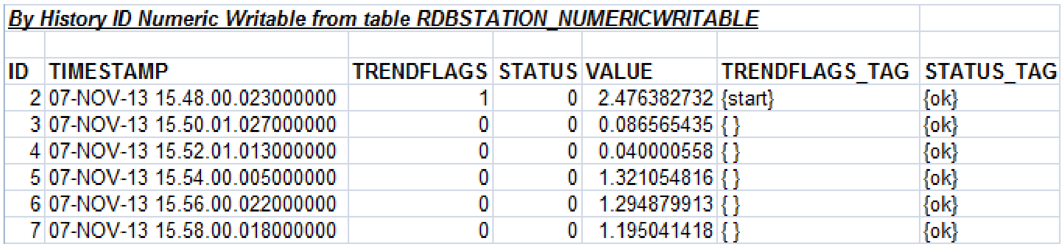
Exporting by history type can be considered by DBA's as more suitable for post process Extracting, Transforming and Loading (ETL) the data beyond the flat file produced by Niagara. This is because there are less tables being created and less permissions required on the database.

Exporting by History ID

Each exported history is stored within a dedicated table with a name corresponding to the history name in NiagaraAX. Duplicate names are suffixed with an incrementing number to ensure uniqueness and they can be traced back to their source via the descriptive HISTORY_ CONFIG table.

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

In this example:



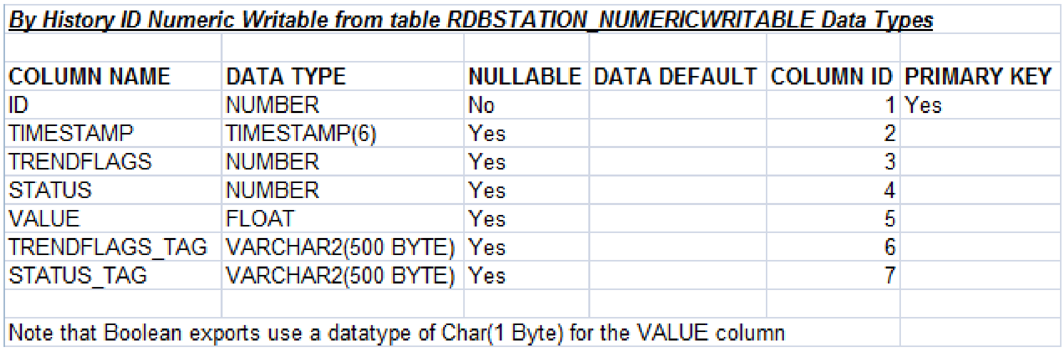
By History ID Numeric Writable from table RDBSTATION_NUMERICWRITABLE

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}

- ID is maintained by the RdbmsNetwork and is used for database indexing but not used by the Niagara 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 will change according to the type of the point exported. e.g. double, float, enum, string or boolean.
- VALUE can be {null}. e.g. where {NaN} is recorded in the source history.
- Point facets (e.g. units) are not exported to the database.

The data types of the HISTORY_CONFIG table are:

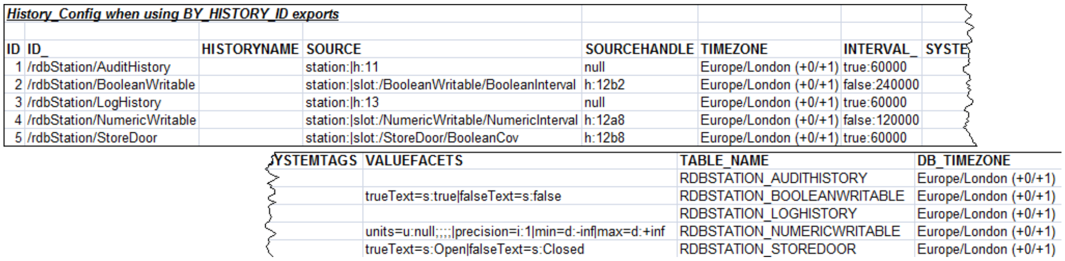
When exporting by ID the relationship between the RDBMS driver and the exported tables is inventoried in the HISTORY_CONFIG table as in this for example:



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	

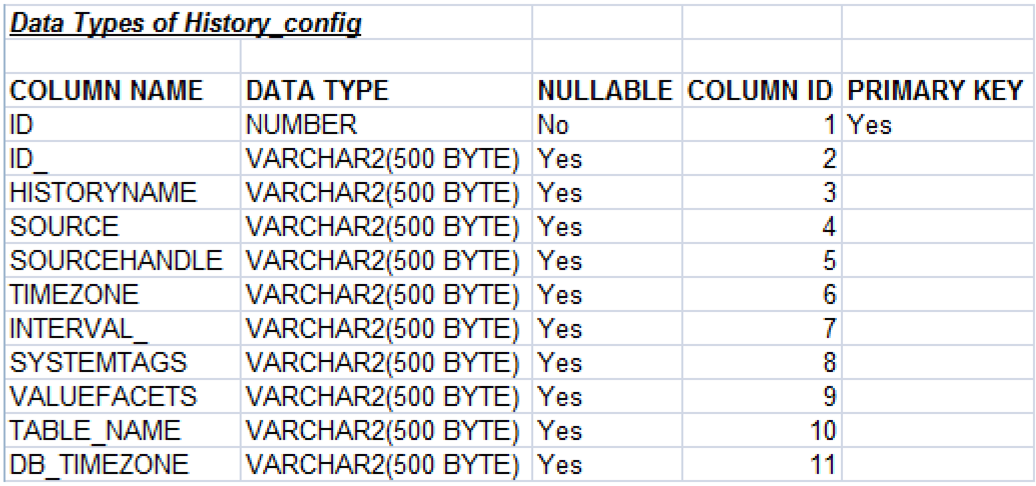
Note that Boolean exports use a datatype of Char(1 Byte) for the VALUE column



History_Config when using BY_HISTORY_ID exports

ID	ID_	HISTORYNAME	SOURCE	SOURCEHANDLE	TIMEZONE	INTERVAL_	SYSTEMTAGS	VALUEFACETS	TABLE_NAME	DB_TIMEZONE
1	/rdbStation/AuditHistory		station:|h:11	null	Europe/London (+0/+1)	true:60000			RDBSTATION_AUDITHISTORY	Europe/London (+0/+1)
2	/rdbStation/BooleanWritable		station:|slot:/BooleanWritable/BooleanInterval	h:12b2	Europe/London (+0/+1)	false:240000		trueText=s:true|falseText=s:false	RDBSTATION_BOOLEANWRITABLE	Europe/London (+0/+1)
3	/rdbStation/LogHistory		station:|h:13	null	Europe/London (+0/+1)	true:60000			RDBSTATION_LOGHISTORY	Europe/London (+0/+1)
4	/rdbStation/NumericWritable		station:|slot:/NumericWritable/NumericInterval	h:12a8	Europe/London (+0/+1)	false:120000		units=u:null;;;;|precision=i:1|min=d:-inf|max=d:+inf	RDBSTATION_NUMERICWRITABLE	Europe/London (+0/+1)
5	/rdbStation/StoreDoor		station:|slot:/StoreDoor/BooleanCov	h:12b8	Europe/London (+0/+1)	true:60000		trueText=s:Open|falseText=s:Closed	RDBSTATION_STOREDOOR	Europe/London (+0/+1)

The data types of this example are:



Data Types of History_config

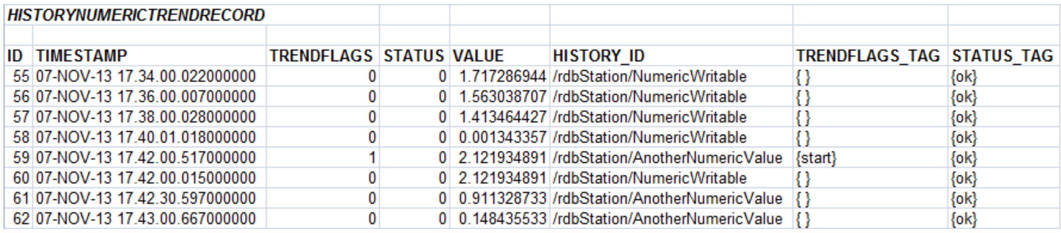
COLUMN NAME	DATA TYPE	NULLABLE	COLUMN ID	PRIMARY KEY
ID	NUMBER	No	1	Yes
ID_	VARCHAR2(500 BYTE)	Yes	2	
HISTORYNAME	VARCHAR2(500 BYTE)	Yes	3	
SOURCE	VARCHAR2(500 BYTE)	Yes	4	
SOURCEHANDLE	VARCHAR2(500 BYTE)	Yes	5	
TIMEZONE	VARCHAR2(500 BYTE)	Yes	6	
INTERVAL_	VARCHAR2(500 BYTE)	Yes	7	
SYSTEMTAGS	VARCHAR2(500 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. The source point ORD and the route taken exported up to supervisor are separated by a space. It is occasionally necessary to extend the length of the SOURCE column when station naming conventions make the ORDs very long. This would be performed by the DBA using SQL or database management tools. This condition would exhibit itself 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.

Exporting by History Type

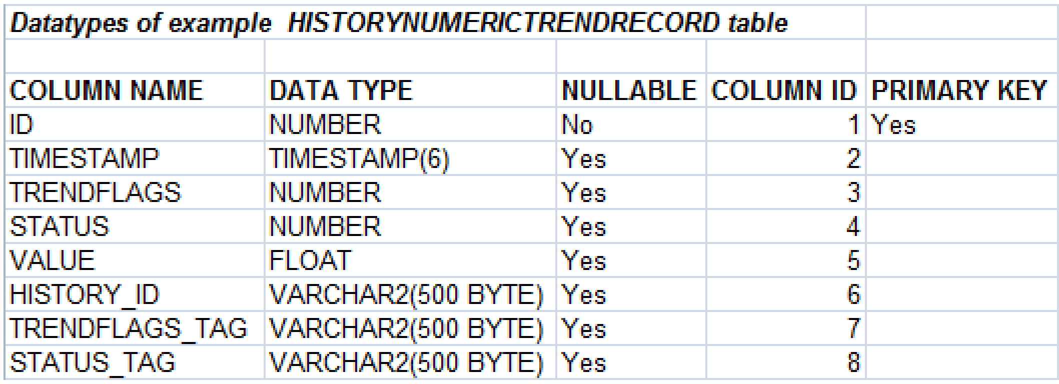
In this method, a single table is created for each type of record exported, recording the source point in that table as in this example of HISTORY_NUMERIC_TREND_RECORD:



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}

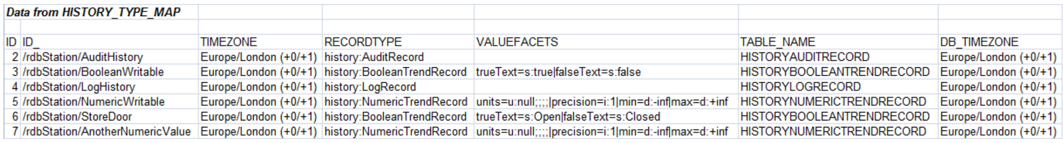
The data types of this descriptive table are:



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	

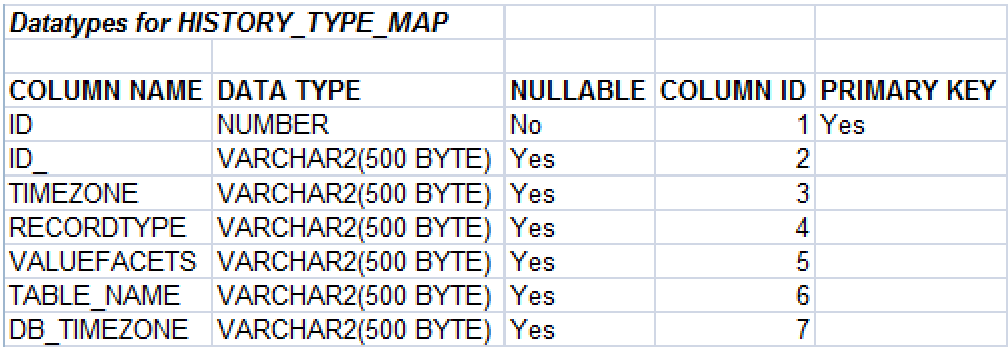
The point type to table mapping required to achieve this type of export are described in the HISTORY_TYPE_MAP table shown below:



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:

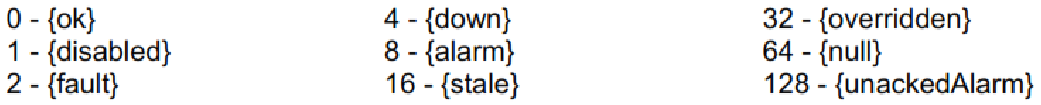


Datatypes for HISTORY_TYPE_MAP

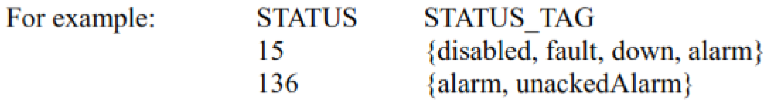
COLUMN NAME	DATA TYPE	NULLABLE	COLUMN ID	PRIMARY KEY
ID	NUMBER	No	1	Yes
ID_	VARCHAR2(500 BYTE)	Yes	2	
TIMEZONE	VARCHAR2(500 BYTE)	Yes	3	
RECORDTYPE	VARCHAR2(500 BYTE)	Yes	4	
VALUEFACETS	VARCHAR2(500 BYTE)	Yes	5	
TABLE_NAME	VARCHAR2(500 BYTE)	Yes	6	
DB_TIMEZONE	VARCHAR2(500 BYTE)	Yes	7	

Understanding 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 listed below:

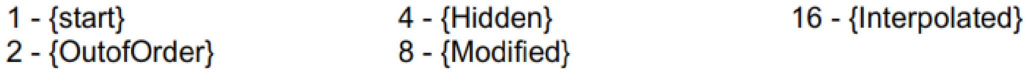


0 - {ok}
1 - {disabled}
2 - {fault}
4 - {down}
8 - {alarm}
16 - {stale}
32 - {overridden}
64 - {null}
128 - {unackedAlarm}



For example:	STATUS	STATUS_TAG
	15	{disabled, fault, down, alarm}
	136	{alarm, unackedAlarm}

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



1 - {start}
2 - {OutofOrder}
4 - {Hidden}
8 - {Modified}
16 - {Interpolated}

Converting Unix time to readable date format in 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 in which Database records are timestamped.

If you wish to convert Unix time into a readable date format in MySQL simply add the following query statement:

FROM_UNIXTIME(TIMESTAMP/1000)

which results in YYYY-MM-DD HH:MM:SS.ssss

Chapter 5 Component Guides

These Component Guides provide summary information about the Rdbms components.

Component Reference Summary

Summary information is provided on components in the following modules:

- *Components in rdb module*
- *Components in rdbDb2 module*
- *Components in rdbHsqlDb module*
- *Components in MySQL module*
- *Components in rdbOracle module*
- *Components in rdbSqlServer module*
- *Components in orion module*

Components in rdb module

- *rdb-HistoryUnicodeUpdater*
- *rdb-HistoryTimezoneUpdater*

rdb-HistoryUnicodeUpdater

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

NOTE: This component's functionality is identical to that achieved by using the **Update Wizard** as described in *Upgrading existing Rdbms databases to support Unicode*. It is available via the palette to provide granular and discrete functionality as an alternative to using the **Update Wizard.** The use of this component is described in the following procedure.

Using rdb-HistoryUnicodeUpdater

Step 1 Drag the HistoryUnicodeUpdater component from the rdb module Palette view and drop it into the RdbmsNetwork in the nav tree.

Step 2 Right-click on the HistoryUnicodeUpdater component in the RdbmsNetwork in the nav tree and select **View Property Sheet**. The HistoryUnicodeUpdater property sheet view displays.

Step 3 Click the ⊕ **Database Ords** add ord button. Navigate to and select a database to update to UTF-8 (Unicode).

Step 4 Repeat the previous step to add more databases to the update as required.

Step 5 Click the **Save** button at the bottom of the view.

Step 6 Right-click on the HistoryUnicodeUpdater component in the RdbmsNetwork in the nav tree and select **Actions Submit Update Job**. The job completes and its results may be inspected in the Job Service in the Services folder in the nav tree.

rdb-HistoryTimezoneUpdater

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

Use the Current Timestamp Storage Policy options to select your desired time zone policy:

- Station Time Zone

  Choose this option to use the time zone of the station to update the history times.

- History Config Time Zone

  Choose this option to use the time zone that is specified in the History Configuration properties for time zone conversion.

NOTE: This component's functionality is identical to that achieved by using the **Update Wizard** as described in *Upgrading existing Rdbms databases to support UTC*. It is available via the palette to provide granular and discrete functionality as an alternative to using the **Update Wizard**. The use of this component is described in the following steps.

Using rdb-HistoryTimezoneUpdater

Step 1 Drag the HistoryTimezoneUpdater component from the rdb module Palette view and drop it into the RdbmsNetwork in the nav tree.

Step 2 Right-click on the HistoryTimezoneUpdater component in the RdbmsNetwork in the nav tree and select **View Property Sheet**. The HistoryTimezoneUpdater property sheet view displays.

Step 3 Click the ⊕ **Database Ords** add ord button. Navigate to and select a database to update to support UTC.

Step 4 Repeat the previous step to add more databases to the update as required.

Step 5 Specify the current policy being used to store database timestamps by selecting either the **Station timezone** or **History config timezone** options in the **Current Timestamp Storage Policy** property.

Step 6 Click the **Save** button at the bottom of the view.

Step 7 Right-click on the HistoryTimezoneUpdater component in the RdbmsNetwork in the nav tree and select **Actions Submit Update Job**. The job completes and its results may be inspected in the Job Service in the Services folder in the nav tree.

Components common to rdb-Database modules

The following components are not in the rdb module but they are common to and available in two or more database components.

- *rdb-RdbmsFolder*
- *rdb-RdbmsNetwork*
- *rdb-RdbmsWorker*
- *rdb-RdbmsPointDeviceExt*
- *rdb-RdbmsPointQuery*

rdb-RdbmsFolder

RdbmsFolder is used to organize databases under an RdbmsNetwork. It is created using the New Folder button in the Device Manager view for any *rdb-RdbmsNetwork*.

rdb-RdbmsNetwork

RdbmsNetwork models a network of BRdbms objects. It is available in the any of the rdb (relational database) palettes, such as rdbDb2, rdbOracle, and rdbSqlServer.

rdb-RdbmsWorker

RdbmsWorker manages the queue and worker for asynchronous operations on a single database. It is available under any of the rdb (relational database) types, such as *rdbDb2-Db2Database*, *rdbOracle-OracleDatabase*, and *rdbSqlServer-SqlServerDatabase*.

rdb-RdbmsPointDeviceExt

RdbmsPointDeviceExt 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, such as *rdbDb2-Db2Database*, *rdbOracle-OracleDatabase*, *rdbSqlServer-SqlServerDatabase*, and *rdbMySQL-MySQLDatabase*.

rdb-RdbmsPointQuery

The Rdbms Point Query component is a container component and a child of the Rdbms Point Device Extension. You can use this component to construct and execute queries against any database that you have access to and sufficient privileges on. The Sql property provides a field for you to write an Sql query statement in. You can execute this query manually and also set a regular interval time for updates using the Update Frequency property. See also, *About the Rdbms Point Device Extension*.

Components in rdbDb2 module

See *Components common to rdb-Database modules* for common components. The following components are unique to the rdbDb2 module.

- *rdbDb2-Db2Database*
- *rdbDb2-Db2HistoryDeviceExt*

rdbDb2-Db2Database

Db2Database models an Db2 relational database. It is available in the rdbDb2 palette.

rdbDb2-Db2HistoryDeviceExt

Db2HistoryDeviceExt is the Db2 implementation of HistoryDeviceExt. It is a child of a *rdbDb2-Db2Database*.

Components in rdbHsqlDb module

See *Components common to rdb-Database modules* for common components. The following component is unique to the rdbHsqlDb module.

- *rdbHsqlDb-HsqlDatabase*

rdbHsqlDb-HsqlDatabase

HsqlDbDatabase models an HsqlDb relational database. It is available in the HsqlDb palette.

Components in MySQL module

See *Components common to rdb-Database modules* for common components. The following components are unique to the rdbMySQL module.

- *rdbMySQL-MySQLDatabase*
- *rdbMySQLHistoryDeviceExt*

rdbMySQL-MySQLDatabase

MySQL models a MySQL relational database. It is available in the rdbMySQL palette.

rdbMySQLHistoryDeviceExt

MySQLHistoryDeviceExt is the MySQL implementation of HistoryDeviceExt. It is a child of a *rdbMySQL-MySQLDatabase*.

Components in rdbOracle module

See *Components common to rdb-Database modules* for common components. The following components are unique to the rdbOracle module.

- *rdbOracle-OracleDatabase*
- *rdbOracle-OracleHistoryDeviceExt*

rdbOracle-OracleDatabase

OracleDatabase models an Oracle relational database. It is available in the rdbOracle palette.

rdbOracle-OracleHistoryDeviceExt

OracleHistoryDeviceExt is the Oracle implementation of HistoryDeviceExt. It is a child of a *rdbOracle-OracleDatabase*.

Components in rdbSqlServer module

See *Components common to rdb-Database modules* for common components. The following components are unique to the rdbSqlServer module.

- *rdbSqlServer-SqlServerDatabase*
- *rdbSqlServer-SqlServerHistoryDeviceExt*

rdbSqlServer-SqlServerDatabase

SqlServerDatabase models an SqlServer relational database. It is available in the rdbSqlServer palette.

rdbSqlServer-SqlServerHistoryDeviceExt

SqlServerHistoryDeviceExt is the SqlServer implementation of HistoryDeviceExt. It is a child of an *rdbSqlServer-SqlServerDatabase*.

Components in orion module

- *orion-DynamicTable*
- *orion-FoxOrionDatabase*
- *orion-OrionModule*
- *orion-OrionRoot*
- *orion-OrionService*
- *orion-OrionType*

orion-DynamicTable

Available starting in NiagaraAX-3.4, this component provides a means for configuring, retrieving and displaying 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. *About Orion*.

orion-FoxOrionDatabase

Available starting in NiagaraAX-3.4, this component represents the Orion database.; It contains the individual Orion Module Types. *About Orion*.

NOTE: It appears in the nav tree directly under the OrionRoot node and is not visible in the Orion palette.

orion-OrionModule

Available starting in NiagaraAX-3.4, the OrionModule represents a module with types registered in an Orion database. The orion-OrionModule is not visible in the Orion palette.

See also, *About Orion*.

orion-OrionRoot

Available starting in NiagaraAX-3.4, this 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.

See also, *About Orion*.

orion-OrionService

Available starting in NiagaraAX-3.4, this is the Niagara service that enables the Orion database in a station. This service allows a station and its applications to use a local relational database running in a controller (including embedded JACEs) to manage and display distributed-system (or distributed-application) information. This managed and configurable information includes certain types of Niagara component data that are well suited for the relational model. It is available in the Orion palette.

Also see *About Orion*.

orion-OrionType

Available starting in NiagaraAX-3.4, 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.

See also, *About Orion*.

orion-OrionMigrator

Available starting in NiagaraAX-3.8 this component, available in the palette is used to migrate existing Orion database data without UTF-8 support into to a new RdbmsDevice which does support the UTF-8 Unicode Encoding Scheme.

Chapter 6 Plugin Guides

There are many ways to view plugins (*views*). One way is directly in the tree. In addition, you can right-click on an item and select one of its views. Plugins provide views of components.

In Workbench, access the following summary descriptions on any plugin by selecting **Help** > **On View**(F1) from the menu, or pressing F1 while the view is open.

Types of plugin modules

Summary information is provided on plugins in the following modules:

- *Plugins in orion module*
- *Plugins in rdb module*
- *Plugins in rdbDb2 module*
- *Plugins in MySQL module*
- *Plugins in rdbOracle module*
- *Plugins in rdbSqlServer module*

Plugins in orion module

- *orion-DynamicTable*
- *orion-DynamicTableConfig*
- *orion-OrionDbManager*
- *orion-OrionModuleTypes*
- *orion-OrionTypeSummary*
- *orion-OrionTypeTableView*

orion-DynamicTable

The Dynamic Table View is the default view on the Dynamic Table component. This view displays data selected from a database that is specified in the Dynamic Table Property Sheet view and according to the parameters setup in the Dynamic Table Config view.

Also see, *About Orion*.

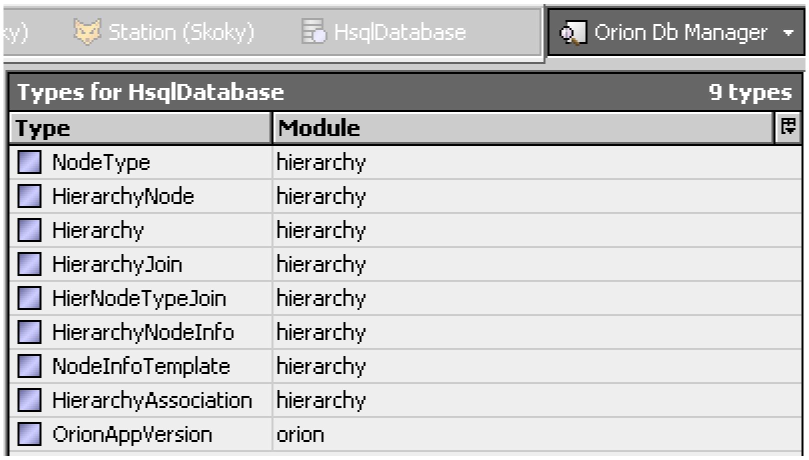
orion-DynamicTableConfig

The Dynamic Table Config View is a view on the Dynamic Table component. This multi-pane view allows you to set the parameters for dynamic table creation. The DynamicTable Property Sheet view allows you to point to a database or application ORD. This database or application is the source for data in the Dynamic Table Config view Row and Column option fields. By choosing a Row Type from the option list 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** button, as desired or by clicking the **Default Columns** button.

Also see, *About Orion*.

orion-OrionDbManager

The OrionDbManager is a view on the FoxOrionDatabase component. This view displays a table of all the orion types and their associated module for the selected database, as shown below.



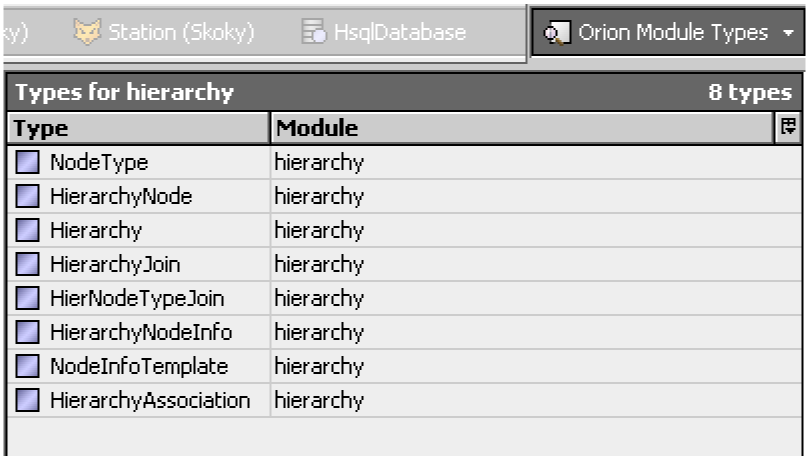
Types for HsqlDatabase 9 types

Type	Module
NodeType	hierarchy
HierarchyNode	hierarchy
Hierarchy	hierarchy
HierarchyJoin	hierarchy
HierNodeTypeJoin	hierarchy
HierarchyNodeInfo	hierarchy
NodeInfoTemplate	hierarchy
HierarchyAssociation	hierarchy
OrionAppVersion	orion

Also see, *About Orion.*

orion-OrionModuleTypes

OrionModuleTypes is a view on the FoxOrionDatabase component. This view displays a table of all the orion types and their associated module for the selected database, as shown below.



Types for hierarchy 8 types

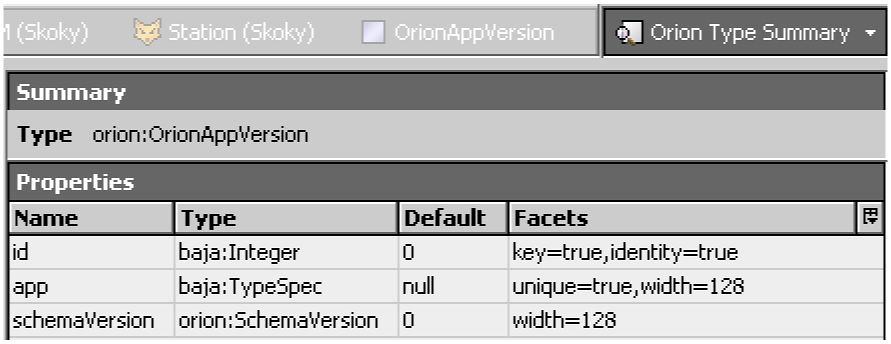
Type	Module
NodeType	hierarchy
HierarchyNode	hierarchy
Hierarchy	hierarchy
HierarchyJoin	hierarchy
HierNodeTypeJoin	hierarchy
HierarchyNodeInfo	hierarchy
NodeInfoTemplate	hierarchy
HierarchyAssociation	hierarchy

Also see, *About Orion.*

orion-OrionTypeSummary

OrionType Summary is a view on the OrionType component that has an upper and lower section, as shown below. 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.



Summary

Type orion:OrionAppVersion

Properties

Name	Type	Default	Facets
id	baja:Integer	0	key=true,identity=true
app	baja:TypeSpec	null	unique=true,width=128
schemaVersion	orion:SchemaVersion	0	width=128

Also see, *About Orion.*

orion-OrionTypeTableView

OrionTypeTableView is a view on the OrionType component. This view shows a table listing of type records.

Also see, *About Orion.*

Plugins in rdb module

- *rdb-RdbmsHistoryImportManager*
- *rdb-Rdbms Point Device Ext Manager*
- *rdb-RdbmsPointQueryManager*
- *rdb-RdbmsQueryView*
- *rdb-RdbmsSessionView*

rdb-RdbmsHistoryImportManager

The RdbmsHistoryImportManager allows you to import records from an Rdbms database as Niagara histories. It is a view on the Histories extension of an Rdbms database, for example *rdbDb2-Db2HistoryDeviceExt*, *rdbOracle-OracleHistoryDeviceExt*, and *rdbSqlServer-SqlServerHistoryDeviceExt*.

rdb-Rdbms Point Device Ext Manager

The Rdbms Point Device Ext Manager View is the default view of the Rdbms Point Device Extension and the only "manager" view for the Rdbms Points Device Extension. This view is used for all Rdbms Device types. It has a single Database pane that displays any Rdbms Point Queries that are present.

Also see, *About the Rdbms Point Device Ext Manager View.*

rdb-RdbmsPointQueryManager

The Rdbms Point Query Manager is the default view for the Rdbms Point Query component under the Rdbms Point Device Extension. 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.

Also see *About the Rdbms Point Device Extension*.

rdb-RdbmsQueryView

This is a view on the Rdbms Point Query component. It is typically a view for working with queries as you are developing them because 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 dialog box displays an error message.

Also see *About the Rdbms Point Device Extension*.

rdb-RdbmsSessionView

The RdbmsSessionView provides a view for a session to a relational database.

Plugins in rdbDb2 module

- *rdb-Db2HistoryExportManager*

rdb-Db2HistoryExportManager

The Db2 History Export Manager view allows you to discover, configure and export history records into a Db2 Rdbms database. It is a view on the rdbDb2 Histories extension.

Also see, *Importing and exporting data using the RdbmsNetwork*.

Plugins in MySQL module

- *rdb-MySQLHistoryExportManager*

rdb-MySQLHistoryExportManager

The MySQL History Export Manager view allows you to discover, configure and export history records into a MySQL Rdbms database. It is a view on the MySQL Histories extension.

Also see, *Importing and exporting data using the RdbmsNetwork*.

Plugins in rdbOracle module

- *rdb-OracleHistoryExportManager*

rdb-OracleHistoryExportManager

The Oracle History Export Manager view allows you to discover, configure and export history records into an Oracle Rdbms database. It is a view on the Oracle Histories extension.

Also see, *Importing and exporting data using the RdbmsNetwork*.

Plugins in rdbSqlServer module

- *rdb-SqlServerHistoryExportManager*

rdb-SqlServerHistoryExportManager

The Sql Server History Export Manager view allows you to discover, configure and export history records into a SqlServer Rdbms database. It is a view on the SqlServer Histories extension.

Also see, *Importing and exporting data using the RdbmsNetwork.*