iFIX 6.1

Using SQL

Proprietary Notice

The information contained in this publication is believed to be accurate and reliable. However, General Electric Company assumes no responsibilities for any errors, omissions or inaccuracies. Information contained in the publication is subject to change without notice.

No part of this publication may be reproduced in any form, or stored in a database or retrieval system, or transmitted or distributed in any form by any means, electronic, mechanical photocopying, recording or otherwise, without the prior written permission of General Electric Company. Information contained herein is subject to change without notice.

© 2020, General Electric Company. All rights reserved.

Trademark Notices

GE, the GE Monogram, and Predix are either registered trademarks or trademarks of General Electric Company.

Microsoft® is a registered trademark of Microsoft Corporation, in the United States and/or other countries.

All other trademarks are the property of their respective owners.

We want to hear from you. If you have any comments, questions, or suggestions about our documentation, send them to the following email address:

doc@ge.com

Table of Contents

© 2020 General Electric Company. All rights reserved.

 © 2020 General Electric Company. All rights reserved.

© 2020 General Electric Company. All rights reserved.

© 2020 General Electric Company. All rights reserved.

Using SQL

The Using SQL manual is intended for system administrators responsible for interfacing iFIX® to an ODBC database. This manual assumes an understanding of ODBC relational databases and the SQL language.

Reference Documents

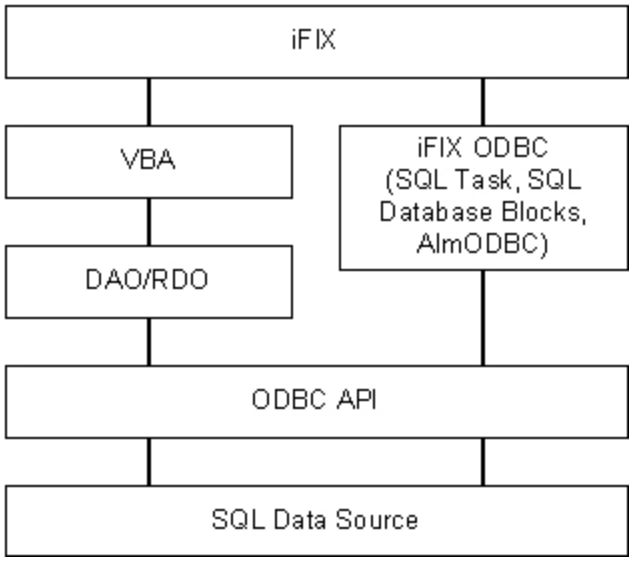
For related information about iFIX, refer to the following manuals:

- Setting up the Environment
- Building a SCADA System
- Configuring Security Features
- Implementing Alarms and Messages
- Using VisiconX

SQL Overview

Structured Query Language (SQL) is a standard language that is used by relational databases to retrieve, update, and manage data. Although it provides the common syntax for applications to use, it does not provide a common application program interface (API). Open Database Connectivity (ODBC) is Microsoft's standard API for accessing, viewing, and modifying data from a variety of relational databases.

To provide access to SQL data sources through the ODBC API, iFIX® allows you to use the following: the iFIX SQL Interface option, called iFIX ODBC, and Microsoft® Visual Basic® for Applications through DAO or RDO.



iFIX Paths to SQL Data Sources

© 2020 General Electric Company. All rights reserved.

Both of these options allow you to systematically:

- Collect and write real-time process data to one or more relational databases.
- Read data stored in the relational database and write it back to the iFIX process database.
- Delete data in relational database tables.
- Back up data and SQL commands to disk if the network fails to maintain a connection to the server or if the server fails.
- Execute backed up SQL commands automatically when the connection to the server is re-established.

Although you can use either VBA or iFIX ODBC to access SQL data sources, each one has specific capabilities that should be considered when making a choice. In many cases, it will be quicker and easier to write scripts in the Visual Basic Editor (VBE) that access and manipulate SQL data sources.

As an alternative, iFIX ODBC requires no knowledge of VBA scripting and allows you to perform all of your SQL tasks. For example, if you want to ensure that your database receives deterministic data, you should use the database blocks available through iFIX ODBC. When using database blocks, the data is sent to the database at scan time with no delays.

ODBC Term Definitions

The following is a list of ODBC concepts and terms you may find helpful while reading this manual.

Client Support – the database client support layer usually contains one or more Dynamic Linked Library (DLL) files in conjunction with configuration files. Database client support is provided by the database vendor. The ODBC driver layer communicates with the client support layer, as illustrated in the first figure. Many database vendors offer TCP/IP network support with their product. The specific files required to communicate using the TCP/IP network protocol is usually part of the client support layer.

Data Source – a data source consists of the data and the information needed to access the data, such as the database management system (DBMS), operating system, and network platform.

Database Layer – the database layer is composed of the database engine and the file or collection of files where the data is actually stored.

Listener Processes – the listener process ties the network protocol to the database engine. This is really the *server* part of a database server. The first figure shows three separate listener processes to demonstrate one possible configuration serving three separate clients. This layer varies widely depending on database vendors and operating systems.

Network Layer – the network layer is completely separate from the ODBC layers and is specific to the operating system. It is usually provided with the operating system or by a network provider. The client computer and the database server computer each contain this layer.

ODBC Administrator Program– the program used to configure ODBC data sources. Typically installed in the Control Panel, but can also be installed as a separate executable (ODBCAD32.EXE).

© 2020 General Electric Company. All rights reserved.

ODBC Application – an application that makes ODBC calls. Since the application communicates to the ODBC layer, it is database independent. This means an ODBC application can be written and, by plugging in various ODBC drivers, can access any database. iFIX ODBC is an ODBC application.

ODBC Driver – an ODBC driver translates an ODBC call issued by the application into a specific call(s) for a particular database. In this module, the application links dynamically to a specific database. ODBC drivers are available from a variety of sources. Some companies specialize in writing database drivers. They provide a package that contains over a dozen ODBC drivers for various databases. ODBC drivers are often available from the database vendor as well. Often times, the user has more than one ODBC driver and manufacturer to choose from. The ODBC driver communicates to the client support layer.

ODBC Driver Manager – a module written by Microsoft that is supplied with most ODBC drivers. It acts as the layer between the application and any ODBC drivers. In fact, it loads the driver when the application requests a connection.

NOTE: There is no ODBC software on the database server computer. The ODBC driver on the client translates the ODBC calls into native database calls that the client support layer can understand. Therefore, by the time the database request leaves the client machine, it has been totally transformed into a native call for that database. The listener and engine on the server computer do not know if the request came from an ODBC application or a native database application.

ODBC Architecture

There are basically two types of ODBC architecture: those that involve *multiple tier* ODBC drivers and *single tier* ODBC drivers.

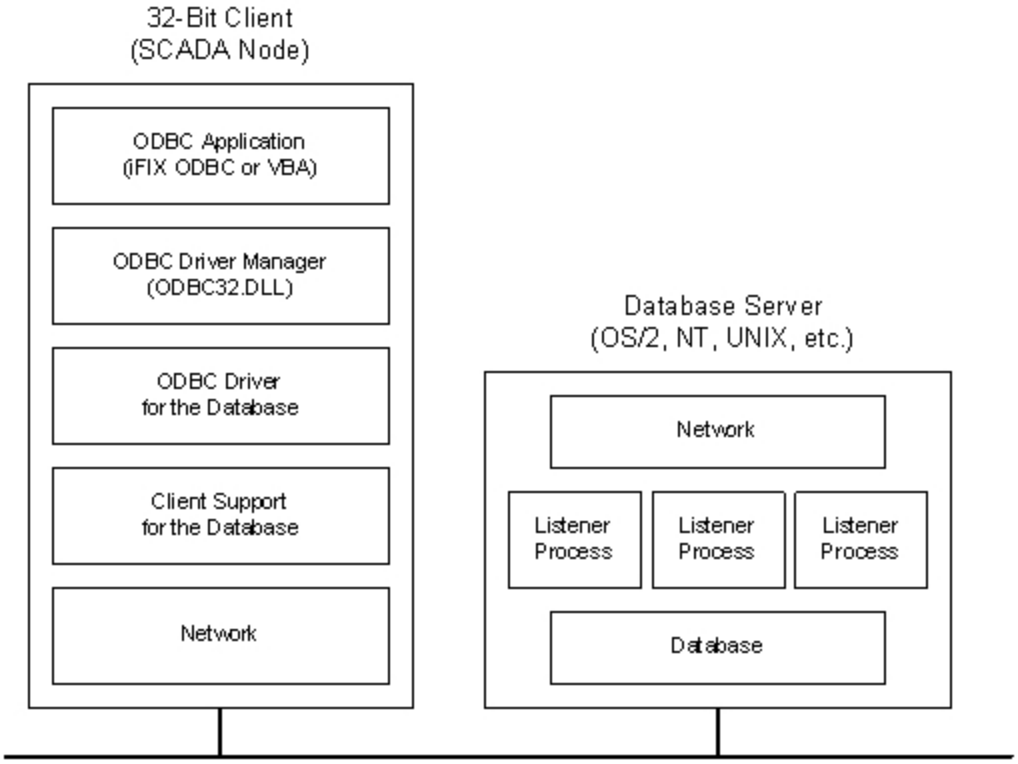
Multiple tier drivers are more common, and are typically used with a remote database server such as Oracle® and SQL Server. Multiple tier ODBC drivers process ODBC calls made by the application, but pass the actual SQL command to the database system.

Single tier drivers, such as the Microsoft® Access® driver, often operate directly on a database file or files. A single tier ODBC driver processes both ODBC calls and the actual SQL commands. In most cases, a configuration using a single tier driver can be contained on just one computer.

Typical ODBC Architecture

The following figure illustrates the typical architecture of a relational database server and an ODBC driver.

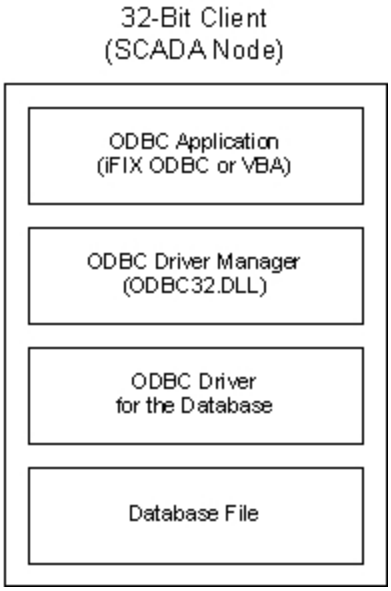
© 2020 General Electric Company. All rights reserved.



Typical ODBC Architecture

Single Tier ODBC Architecture

The ODBC architecture of Microsoft Access is simpler than the typical server architecture. The following figure illustrates the ODBC architecture of Access and other *single tier* ODBC drivers.



© 2020 General Electric Company. All rights reserved.

Single Tier ODBC Architecture

Note that there is no database server computer, client support layer, or network in the previous figure. The Access ODBC driver works directly on the database file. In this simple configuration, the database file is located on the same computer as the application. By using Microsoft networking, Novell networks, and the like, the database file could be located on another computer just like any other shared file. In this way, an Access database could be shared among several computers and applications.

Accessing SQL Data Sources Through VBA

VBA is embedded directly into iFIX, allowing you to access SQL data sources from any relational database through the ODBC API. To access the Visual Basic Editor (VBE) in the iFIX WorkSpace from the Ribbon View, on the Home tab, click the Visual Basic Editor option. In Classic view, on the WorkSpace menu, click Visual Basic Editor. For complete information on accessing SQL data sources through VBA, refer to the Writing Scripts manual.

VBA supports two Microsoft technologies that allow iFIX to connect to SQL data sources: Data Access Objects (DAO) and Remote Data Objects (RDO). Both of these technologies provide an object-based interface for accessing relational databases. Although either technology can be used, each one has specific capabilities that should be considered when making a choice.

Data Access Objects (DAO)

DAO is an object-based data access interface that provides access to SQL data sources through VBA. Using DAO, you can:

- Retrieve, add, change, or delete data in a database.
- Create a new database or change the design of an existing database.
- Implement security to protect your data.

To use DAO within iFIX, you must first set a reference to the Microsoft DAO object library.

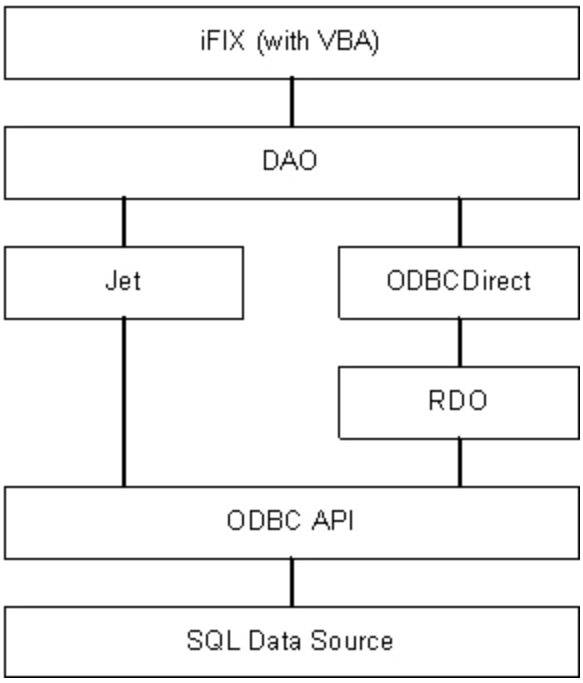
NOTE: Microsoft DAO 3.5 no longer supports some of the objects, properties, and methods that were supported in earlier versions of Microsoft DAO. If necessary, you can set a reference to the Microsoft DAO 2.5/3.0 Compatibility Library, the Microsoft DAO 2.5/3.5 Compatibility Library, or the Microsoft DAO 3.0 Object Library from the VBE.

Once you have set a reference to the Microsoft DAO object library, you can view the DAO objects in the Object Browser by selecting DAO in the Project/Library box.

DAO supports two technologies for accessing SQL data sources: Microsoft Joint Engine Technology (Jet) and ODBCDirect. The DAO technology you use depends on the type of operation you need to perform.

The following figure illustrates how DAO uses Jet and ODBCDirect to access a SQL data source.

© 2020 General Electric Company. All rights reserved.



Accessing SQL Data Sources with DAO Jet versus DAO ODBCDirect

Joint Engine Technology (Jet)

Jet was designed primarily to access native Jet/Access (.MDB) databases and selected ISAM databases, such as dBase, Paradox, FoxPro, and Btrieve. The following is a list of exclusive Jet capabilities not offered by ODBCDirect.

Linking remote tables for form and control binding - allows forms or controls to be bound to data in an ODBC data source.

Heterogeneous data access - allows you to join data stored in different back ends.

Programmatic DDL - provides table definitions and the ability to create or modify tables using Data Definition Language (DDL).

Support for Find and Seek methods - permits the use of the Find and Seek methods.

Although Jet is capable of accessing ODBC data sources, this functionality is limited and, compared to ODBCDirect, has two major disadvantages. First, Jet loads the Microsoft Jet database engine even when it is not a Jet database being accessed. Second, since calls must be passed through the Jet database engine before reaching the ODBC API, Jet is slower than ODBCDirect.

ODBCDirect

As the name suggests, ODBCDirect provides more direct access to ODBC data sources by way of bypassing the Jet engine. ODBCDirect is a thin DAO wrapper around the RDO interface, meaning it

© 2020 General Electric Company. All rights reserved.

routes DAO objects, methods, and properties to equivalent RDO functions. The following is a list of ODBCDirect advantages not provided by Jet.

Direct data access - allows direct access to ODBC data resources.

Reduced resource requirements - eliminates the resources required by Jet to load the Jet database engine.

Asynchronous queries - optimizes performance by allowing alternative functions to perform while an operation completes.

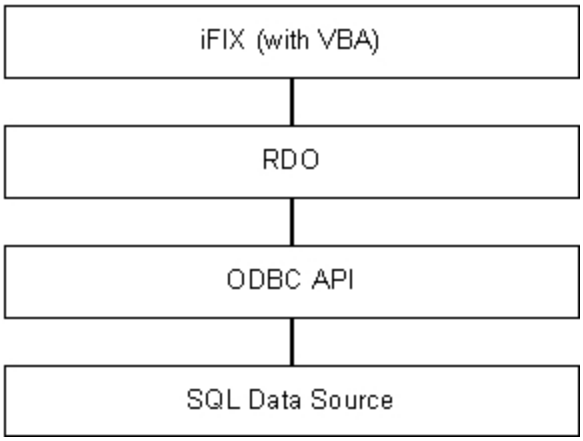
Local batch processing - caches Recordset changes locally and submits the changes to the server in a single batch.

Remote Data Objects (RDO)

RDO is a thin object layer interface to the ODBC API that is optimized for speed and flexibility. It provides the same ease of use as DAO, while exposing the low-level power and flexibility of ODBC. Using RDO, you can:

- Establish a connection to a SQL data source asynchronously.
- Submit queries.
- Perform synchronous or asynchronous operations.
- Create result sets and cursors.
- Process the query results.

The following figure illustrates how RDO accesses a SQL data source.



Accessing SQL Data Sources with RDO

RDO incorporates most of the higher-level functions of ODBCDirect, such as support for asynchronous operations. It also provides additional functionality, such as triggered events for connections and queries, advanced support for stored procedures and multiple-select queries, and enhanced error trapping. Since RDO directly calls the ODBC API, it's speed nearly matches that of calling the ODBC API directly, and it's use of resources is reduced.

To use RDO within iFIX, you must first set a reference to the Microsoft Remote Data Object library.

© 2020 General Electric Company. All rights reserved.

Once you have set a reference to the Microsoft Remote Data Object library, you can view the RDO objects in the Object Browser by selecting RDO in the Project/Library box.

Configuring Data Sources

Before you can use ODBC with DAO or RDO, you must configure the ODBC data source. You can configure a data source through the Windows Control Panel, as described in the Installing and Configuring Data Sources chapter, or with Visual Basic code using DAO or RDO.

To configure a data source with DAO, use the RegisterDatabase method and the following syntax:

```
DBEngine.RegisterDatabase "dbname", "driver", silent, _ attributes
```

To configure a data source with RDO you use the rdoRegisterDataSource method and the following syntax:

```
RdoRegisterDataSource, "dsName", "driver", silent, attributes
```

Accessing ODBC Data Sources

Once you have configured a data source, you are ready to access an ODBC database. DAO and RDO each have their own unique objects, methods, and properties. However, they both allow you to access an ODBC data source with just a few simple lines of code. Refer to the following sections for more details:

- Accessing an ODBC Data Source with DAO
- Accessing an ODBC Data Source with RDO

Accessing an ODBC Data Source with DAO

To access an ODBC data source with DAO you must set up the Workspace object, open the database, and create a record set.

NOTE: All references to the Workspace in conjunction with DAO refer to the Workspace object, not the iFIX WorkSpace. The Workspace object defines a named session and allows you to open multiple databases and connections, and manage transactions. It also controls whether your application interacts with data through Microsoft Jet or ODBCDirect.

To create a Workspace object, use the CreateWorkspace method as shown below.

```
Set workspace = CreateWorkspace ("name", "user", "password", _ type)
```

To open a database, use the OpenDatabase method as shown below:

```
Set database = workspace.OpenDatabase ("dbname", options, _
read-only, connect)
```


© 2020 General Electric Company. All rights reserved.

When using ODBCDirect, you can also use the OpenConnection method to connect to a data source. The OpenConnection method allows you to perform asynchronous operations and use QueryDef objects. To connect to a data source using the OpenConnection method, use the code shown below.

```
Set connection = workspace.OpenConnection ("name", options, _
read-only, connect)
```

To create a record set, use the OpenRecordset method as shown below.

```
Set recordset = object.OpenRecordset (type, options, lockedits)
```

NOTE: To use the MSFlexGrid, you must install it and then set a reference to it by selecting References from the Tools menu in the VBE. You can use the MSFlexGrid or any similar spreadsheet, such as the VideoSoft VSFlexGrid, in your applications. GE does not provide the MSFlexGrid; it is referenced in the documentation for illustration purposes only.

Accessing an ODBC Data Source with RDO

To access an ODBC data source with RDO, you must set up the environment, make a connection, and create a result set.

To set up the environment, use the rdoEnvironments collection as shown below.

```
Set en = rdoEngine.redoEnvironments (n)
```

You can use a variety of methods to connect to a database, including:

- Declaring an rdoConnection object and using the OpenConnection method of the rdoEnvironment object.
- Creating a stand-alone rdoConnection object using the *Dim x As New* syntax, setting its properties, and using the EstablishConnection method.
- Using the EstablishConnection method on an existing rdoConnection object after you have either created a stand-alone rdoConnection object or you have used the Close method on an existing rdoConnection object.

You can also create an asynchronous RDO connection using the rdAsyncEnable option of the EstablishConnection method.

You can use a variety of methods to create an rdoResultset object, including:

- Using the OpenResultset method against an rdoConnection object.
- Using the OpenResultset method against an rdoQuery object.
- Creating a stand-alone rdoQuery object, setting its properties, and associating it with a specific connection using the ActiveConnection property. An rdoResultset object can then be created against the rdoQuery using the OpenResultset method.

NOTE: To use the MSFlexGrid, you must install it and then set a reference to it by selecting References from the Tools menu in the VBE. You can use the MSFlexGrid or any similar spreadsheet, such as the VideoSoft VSFlexGrid, in your applications. GE does not provide the MSFlexGrid; it is referenced in the documentation for illustration purposes only.

© 2020 General Electric Company. All rights reserved.

iFIX ODBC

iFIX ODBC provides communication between relational databases and the iFIX database. The iFIX database can be configured to communicate based on an event, a time, or a combination of both.

iFIX ODBC consists of the:

- SQL task.
- SQL Trigger (SQT) database block.
- SQL Data (SQD) database block.

The SQL task performs the following functions:

- Executes the SQT blocks that trigger.
- Retrieves process data from the SQD blocks and inserts the data into the relational database.
- Selects data from the relational database and writes the data back to the iFIX database.
- Backs up data in the event of a network failure (backup continues until the primary and secondary paths are full).
- Automatically restores data to the relational database once network communications are established.

The SQL Trigger block defines:

- Which SQL commands in the SQL Library Table are used to manipulate data.
- Whether the SQL commands are backed up in the event that the application loses a connection with the server.
- The time or event that triggers the data transfer.

The SQL Data block defines:

- The data that is collected and transferred.
- The direction of the data transfer.

Both database blocks communicate with the SQL task, WSQLODC.EXE. This task runs on a SCADA node and handles communication with the ODBC driver.

If the connection to the relational database is lost, SQL commands and data can be backed up. When the connection is re-established, SQL commands and data are executed in the order in which they were backed up. Refer to the Backing Up and Restoring Data section for a more detailed description of the back-up and restore process.

NOTE: If you copy and paste the SQT and SQD blocks, they will not fire. You must create these blocks in full.

Understanding the Communication Process

The process for collecting and inserting process data into a relational database using iFIX ODBC is outlined below.

 © 2020 General Electric Company. All rights reserved.

To collect and insert process data into a relational database using iFIX ODBC:

NOTE: In order for this process to work, iFIX must be running. Otherwise, you will receive an error message.

1. The SQL Trigger block (SQT) triggers; the SQL task reads the SQL command name from the SQT block.
2. The SQL task retrieves the associated command from the SQL Library Table.
3. The SQL task reads the tags specified in the SQL command from the SQL Data block (SQD) and reads the values associated with these tags from the iFIX database.
4. The SQL task executes the SQL command and inserts data into or selects data from the relational database.
5. If the SQL command is a SELECT command, the retrieved data is written to the iFIX tags defined in the SQD block.

Understanding Multiple Relational Database Support

iFIX ODBC supports communication to multiple relational databases simultaneously, which allows you to:

- Use multiple accounts within one relational database.
- Read and write data to several different databases.
- Centrally store commands that can be run in different databases.

Refer to the Using Multiple Relational Database Support section for more information on these scenarios and how to configure iFIX ODBC.

Setting Up iFIX ODBC

Complete the following steps before running iFIX ODBC.

To prepare for using iFIX ODBC:

1. Set up your relational database on the server along with user accounts and passwords. Consult your relational database manuals for specific instructions.
2. Set up the ODBC driver to communicate with iFIX. To do this, use the relational database tools that come with your SQL software.
3. Define the SQL commands. To do this, create a library table that contains the SQL commands you want executed. The examples in this manual use SQLLIB as the SQL command table name.
4. Create the error log table that stores all SQL runtime error messages that are recorded by the system.
5. Set up or upgrade your network to support communications.
6. Configure iFIX ODBC.
7. Confirm that the SCADA node can establish network communications with the server by using an ODBC test program supplied by your ODBC vendor (such as MSQUERY).

© 2020 General Electric Company. All rights reserved.

NOTE: When logging alarms to an Oracle database, you must use the Microsoft Oracle driver or data loss could occur.

To configure iFIX ODBC:

1. Ensure that the SQL task is added to the Configured Task List in the SCU Task Configuration dialog box. Refer to the Configuring the SQL Task section for more information on configuring the SQL task.
2. Add an account for each relational database you need to communicate with to the Configured Accounts list in the SCU SQL Configuration dialog box. Refer to the Installing and Configuring Data Sources section for more information on defining specific accounts.
3. Configure the SQL task using the Configure SQL Task dialog box in the SCU. Refer to the Configuring the SQL Task section for more information on configuring the SQL task.
4. Define the SQT and SQD blocks in the iFIX database. Refer to the iFIX Database Reference for more information on blocks.

Installing an ODBC Driver

ODBC drivers are almost always shipped with the following components:

- The ODBC driver. There may be more than one driver included in the set. This is the case with Intersolv's ODBC Data Direct pack and Microsoft Desktop Driver pack.
- The ODBC Driver manager, *ODBC32.DLL*.
- The ODBC Administrator, *ODBCAD32.EXE*, which is used to add, delete, and configure data sources.

To install an ODBC driver, run the Setup program from the CD-ROM or the diskette. If given the option of installing the Driver Manager and Administrator with version checking, do so.

After the driver is installed, an ODBC Administration icon should appear in the Control Panel. Depending on the driver and the installed ODBC components, the icon may have the *32* insignia. If the ODBC icon does not appear in the Control Panel, the driver setup program may have installed it in a separate group.

iFIX ODBC only supports 32-bit ODBC drivers. To run iFIX ODBC as a service under Windows, you must have an ODBC driver built with ODBC 2.5 or later. Refer to the Running iFIX ODBC as a Service section for more information on running iFIX ODBC as a service.

NOTE: The iFIX ODBC driver truncates the A_EGUDESC field to 4 characters, even though you are allowed to type in up to 33. Also, only uppercase characters are supported.

Configuring an ODBC Data Source

Once the ODBC driver and administrator have been installed, you can configure ODBC data sources. Each driver for each database requires different information to make a connection. The one thing they all have in common is the data source name.

Each data source that you configure to connect to a particular database is identified by the data source name. When an application needs to connect to a database, the application passes the data source name, which maps to the information that was configured for that database.

 © 2020 General Electric Company. All rights reserved.

System and User Data Sources

ODBC (specifically the Administrator and Driver Manager) provides the concept of *system* and *user* data sources. Any user can create a *system* data source and that data source becomes available to any user. A *user* data source, on the other hand, is only available to the user who created it.

For example, assume your system has three user accounts: Account1, Account2, and Account3. Account1 creates a system data source *sysAccount1*. Account2 creates a user data source *user-Account2*. Account3 creates a user data source *userAccount3*. Account1 can only use *sysAccount1*. Account2 can use *sysAccount1* and *userAccount2*. Account3 can use *sysAccount1* and *userAccount3*.

The ODBC Data Source Administrator allows you to create a system data source simply by clicking the System DSN button and adding the data source. System data sources appear in the System DSN list. However, when you call up the list of available data sources in the SCU, both system *and* user (for the user currently logged in) data sources are listed. If your ODBC driver installs an earlier version of the driver manager, you can copy the driver manager located in the iFIX Base path to your operating system's SYSTEM directory.

Running iFIX ODBC as a Service

You can run iFIX and various tasks, such as the SQL task, as services under Windows. When you do this, you can log out of the operating system without shutting down iFIX ODBC. This provides a much higher level of security to your process, because operators can log in and log out of a node before and after their shift without affecting the process.

In order to run the SQL task (WSQLODC.EXE) as a service, you *must* use system data sources. If you attempt to use a user data source, you will get an error similar to the following:

```
Data source name not found and no default driver specified.
```

The procedure for configuring a system data source is similar to configuring other data sources. The only difference is that you must click the System DSN button in the ODBC Data Sources dialog box and add the data source you want. Refer to the System and User Data Sources section for more information on system data sources.

Refer to the Installing and Configuring Data Sources section for more specific information on configuring data sources for each database supported by iFIX ODBC.

Installing and Configuring Data Sources

This chapter provides instructions and tips on installing and configuring Microsoft Access, Microsoft SQL Server, and Oracle data sources for use with iFIX ODBC. It includes the following sections:

- Configuring an ODBC Data Source
- Verifying and Editing an ODBC Data Source
- Configuring the SCU for an ODBC Data Source

© 2020 General Electric Company. All rights reserved.

- Microsoft Access
- Microsoft SQL Server
- Oracle
- Network Problems and Workarounds

Configuring an ODBC Data Source

To configure an ODBC data source, you must have an ODBC driver installed for the relational database you want to use. ODBC drivers are often automatically installed and set up when you install the relational database application.

For more information on whether an ODBC driver has been installed for a particular relational database, refer to the documentation supplied with the application.

To configure an ODBC data source:

1. Click the Start button and point to Programs, Administrative Tools, and then Data Sources (ODBC). You can also access the Administrative Tools folder from the Control Panel.

   The ODBC Data Source Administrator program opens.
2. On the User DSN tab, click Add. The Create New Data Source dialog box appears.
3. Select the ODBC driver for the relational database you want to access from the list.
4. Click Finish. An ODBC Data Source Setup dialog box appears for the ODBC driver you selected.

   NOTE: If you do not have the correct ODBC driver installed on your system, an error message appears instead of the ODBC Data Source Setup dialog box.
5. In the ODBC Data Source Setup dialog box, enter the required information.

   NOTE: Skip steps 6-9 if you are not accessing a Microsoft Access data source and not running iFIX as a service.
6. In the System Database area, select the Database option.
7. Click the System Database button. The Select System Database dialog box appears.
8. Select the system database (usually C:\ACCESS\SYSTEM.MDA).
9. Make sure Exclusive and Read Only are unchecked in the Options group.
10. Click OK in the ODBC Data Source Setup dialog box. The new data source appears in the Data Sources dialog box.
11. Click Close.

Verifying and Editing an ODBC Data Source

Once you have configured the settings for an ODBC data source, you can verify that the information is correct and make changes at any time.

© 2020 General Electric Company. All rights reserved.

To verify information and edit the settings for an ODBC data source:

1. Click the Start button and point to Programs, Administrative Tools, and then Data Sources (ODBC). You can also access the Administrative Tools folder from the Control Panel.

   The ODBC Data Source Administrator program opens.
2. On the User DSN tab, select the appropriate data source from the list and click Configure. The ODBC Data Source Setup dialog box appears for the data source you selected.
3. Check the settings and make any necessary changes.
4. In the ODBC Data Source Setup dialog box, click OK.
5. In the Data Source Administrator dialog box, click OK.

Configuring the SCU for an ODBC Data Source

In order for iFIX to connect to an ODBC data source, an entry must be added to the SQL connection list in the SCU.

Microsoft Access

This section provides instructions and tips specific to installing and configuring Microsoft Access data sources for use with iFIX ODBC. It includes the following topics:

- Installing an Access Driver
- Creating the Library and Error Tables for Access
- Supported Column Data Types for Access
- Handling Errors for Access

Installing an Access Driver

iFIX ODBC supports Microsoft Access version 2.x. Various sources for this driver include Microsoft Word™, Visual Basic™, Excel™, and the Microsoft Desktop Driver Pack 2.0. This package includes drivers for Access, Excel, FoxPro®, and others. It can be obtained from Microsoft at a very modest cost (enough to cover Microsoft's manufacturing cost).

To obtain the driver from Word, Excel, or Visual Basic, install the driver as part of the setup procedure for these products. Refer to the documentation for each of these products for setup instructions. If you need to install a driver from the ODBC driver pack, use the instructions provided in the driver pack's documentation.

Creating the Library and Error Tables for Access

© 2020 General Electric Company. All rights reserved.

If you are using iFIX ODBC to communicate with the iFIX process database, you must create tables in Microsoft Access to store the SQL commands and errors.

To create a table in Microsoft Access:

1. Start Access.
2. On the File menu, click Open. Select a database to open, such as C:\ACCESS\EXAMPLE.MDB.
3. Click the Tables icon.
4. Click New to add a new table.

The spreadsheet displayed has entries for the field (column) name, its datatype, and an optional description. As you move from row to row, the attributes for that field are displayed below the spreadsheet. Use the following tables in this section as guides for the two tables.

Microsoft Access SQLLIB Table

Field name	Data type	Notes
sqlname	Text, size 8	Select Yes for the Indexed attribute. Also, click the Key icon in the toolbar to make this the *Primary Key* for the table.
sqlcmd	Text, size 100 - 255	If any of your SQL commands are longer than 255 characters, make this field a Memo field. This allows up to 64,000 characters. You must also modify the /CL*n* parameter to allow for greater than 255 characters. Refer to the Modifying the Startup Options section for more information on modifying the /CL*n* parameter.

Microsoft Access SQLERR Table

Field name	Datatype
td	Date/Time
node	Text, size 8
tag	Text, size 10-30
sqlname	Text, size 8
fix_err	Text, size 100
sql_err	Text, size 250
prog_err	Text, size 100

The SQLERR table does not need a key, but you can create a counter field to use as a key if you wish.

NOTE: SQLLIB and SQLERR are the default names for the tables. You can name them anything you want as long as those names are reflected in the SCU. However, the field (column) names must be entered ***exactly*** as shown. (Access column names are not case sensitive.)

Supported Column Data Types for Access

You can transfer data out of or into the iFIX process database using iFIX ODBC. The following table lists iFIX and Access field types. It indicates the available transfer direction(s).

© 2020 General Electric Company. All rights reserved.

The transfer direction (In, Out) is determined by your entry in the direction field of the SQL Data block when you define the iFIX process database. An I indicates a direction of In (for select commands) and is supported for Access and iFIX field types. An O indicates a direction of Out (for parameter markers) and is also supported. For more information on configuring SQL Data blocks, refer to the Building a SCADA System manual.

Microsoft Access and iFIX Field Types

Access field type	DATE key-word	TIME key-word	TIMEDT key-word	F_CV	A_
Date/time	O	O	O		I
Number (single, double, integer, long integer, byte)				I, O	I
Counter				I, O	I
Text					I, O

NOTE: iFIX does not support the use of data types that accept large amounts of data. For example, the MEMO field in a Microsoft Access database or the LONG data type in an Oracle database. Do not select a column with either data type from Run. Doing so may cause unexpected results.

Handling Errors for Access

iFIX provides the SQLERR.TXT text file, located in the C:\Program Files (x86)\GE\iFIX\App directory, to help you troubleshoot communication errors from your relational database. This file contains communication error numbers. When a communication error occurs, iFIX displays @ symbols (by default) in a data link to indicate the problem.

If you are using Access, some of the same error numbers listed in the SQLERR.TXT file are used to indicate relational database errors. As a result, you may see @ symbols in the run-time environment when no communication error has occurred. To correct this problem, delete the SQLERR.TXT file.

Microsoft SQL Server

This section provides instructions and tips specific to installing and configuring Microsoft SQL Server data sources for use with iFIX ODBC. It includes the following topics:

- Installing Microsoft SQL Server Database
- Installing and Configuring the SQL Server Client
- Installing a SQL Server Driver

© 2020 General Electric Company. All rights reserved.

- Creating the Library and Error Tables for SQL Server
- Supported Column Data Types for SQL Server

Installing Microsoft SQL Server Database

We strongly recommend that you install SQL Server with the case-insensitive, accent-insensitive option, which is not the default. The Binary Sort order default causes all table and column names to be case-sensitive. Therefore, the following SQL command:

```
`Select ValveSetting1, ValveSetting2 from T1 where index = 23'
```

will not work if the table T1 is actually t1, or if the ValveSetting1 column is actually Valvesetting1, and so on.

Installing and Configuring the SQL Server Client

SQL Server client installs from the same setup program as the database server itself. To install the client software, select the Install SQL Server Components option from the main install screen. After a few screens, select to install the Client Tools. Support for a variety of network protocols, including TCP/IP, Named Pipes, and Novel IPX/SPX, will also be installed. Once the SQL Server client is installed, run the Client Network Utility and select the network protocol you want to use.

Installing a SQL Server Driver

Install the SQL Server driver by adding it in the Control Panel from the Administrative Tools, Data Sources (ODBC), System DSN tab.

Creating the Library and Error Tables for SQL Server

Library and Error tables can be created from the client or the server using Enterprise Manager or ISQL. Use the following tables in this section as guides for the SQLLIB and SQLERR tables.

Microsoft SQL Server SQLLIB Table

Field Name	Datatype	Notes
sqlname	varchar, size 8	Create a unique index on this field.
sqlcmd	varchar, size 100 - 255	If any of your SQL commands are longer than 255 characters, make this field a Text field. This allows over 2 billion characters.

Microsoft SQL Server SQLERR Table

Field Name	Datatype
td	datetime

© 2020 General Electric Company. All rights reserved.

node	varchar, size 8
tag	varchar, size 10-30
sqlname	varchar, size 8
sql_err	varchar, size 250
fix_err	varchar, size 100
prog_err	varchar, size 100

NOTE: SQLLIB and SQLERR are the default names for the tables. You can name them anything you want as long as those names are reflected in the SCU. However, the field (column) names must be entered ***exactly*** as shown.

Use the following commands to create the Library table, Index, and Error Log table if using ISQL:

```
create table sqllib (sqlname varchar(8) NOT NULL,
sqlcmd varchar(250) NOT NULL)

create unique index sqllib_index on sqllib (sqlname)
EXEC sp_primarykey 'sqllib', sqlname

create table sqlerr ( td datetime NOT NULL,
node varchar(8) NOT NULL, tag varchar(10) NOT NULL,
sqlname varchar(8) NOT NULL, sql_err varchar(250),
fix_err varchar(100), prog_err varchar(100) )
```

Supported Column Data Types for SQL Server

Use the following table as a guide to iFIX and SQL Server field types. The direction (I, O) refers to the direction field in the SQL Data (SQD) block. An I indicates a direction of IN (for selects) and is supported for the SQL Server and iFIX field types. An O indicates a direction of OUT (for parameter markers) and is also supported.

Microsoft SQL Server and iFIX Field Types

SQL Server type	DATE keyword	TIME keyword	TMDT keyword	F_CV	A_
datetime	O	O			I
smalldatetime	O	O			I
char/varchar		O			I, O
int				I, O	I
real				I, O	I
float				I, O	I

© 2020 General Electric Company. All rights reserved.

tinyint	I, O	I
smallint	I, O	I
text		I

Oracle

This section provides instructions and tips specific to installing and configuring Oracle 10*g* for use with iFIX. It includes the following topics:

- Installing and Configuring Client Support
- Installing the Oracle Driver
- Creating the Library and Error Tables for Oracle
- Supported Column Data Types for Oracle

Installing and Configuring Client Support

Follow these steps to install and configure 10*g* client support.

To install and configure Oracle 10g client support:

1. Install the Oracle 10*g* Release 2 Client. When prompted for the installation type, select "Runtime" and then complete the install and exit.
2. From the Oracle program group, open the Oracle Net Manager application.
3. In the Oracle Net Manager, expand the tree and select the Service Naming item.
4. On the Edit menu, click Create.
5. In the Net Service Name field, enter a name and click Next.
6. Select the TCP/IP (Internet Protocol) option and click Next.
7. In the Host Name field, enter the computer name where the Oracle database resides, and click Next.
8. In the Service Name field, enter the global database name that you want. This name was set on the Oracle database computer in Oracle 8i or later. Click Next to continue.
9. Optionally, test the connection.
10. Click Finish.
11. Exit and save the configuration.

Refer to your Oracle documentation for more information.

Installing the Oracle Driver

© 2020 General Electric Company. All rights reserved.

The steps that follow describe how to install the Oracle 10*g* Release 2 Client, so that you can use it with iFIX.

To install the Oracle driver:

1. Run the setup program for the Oracle 10*g* Release 2 Client.
2. On the Welcome screen, click Next.
3. When prompted for an installation type, select Custom Installation and click Next.
4. On the Destination screen, in the Name field, change the default setting by selecting the destination you used when installing the "Runtime" Client. The path should automatically change to the path associated with the name. Click Next.
5. On the Available Product Components screen, select the Oracle Windows Interfaces 10.2.0.1.0 and click Next.
6. On the Product-Specific Prerequisite Check screen, click Next.
7. On the Summary screen, click Install. When the installation completes, a final screen appears.
8. Click Exit to finish the install.

Creating the Library and Error Tables for Oracle

Library and Error tables can be created from the client or from the server using SQL*DBA, SQL*Plus, or any other Oracle utility. Use the following tables in this section as guides for the SQLLIB and SQLERR tables.

Oracle SQLLIB Table

Field Name	Datatype	Notes
sqlname	varchar2, size 8	Create a primary key on this field.
sqlcmd	varchar2, size 100 - 2000	If any of your SQL commands are longer than 2000 characters, make this field a long field. This allows up to 64,000 characters.

Oracle SQLERR Table

Field Name	Datatype
td	date
node	varchar2, size 8
tag	varchar2, size 30
sqlname	varchar2, size 8
fix_err	varchar2, size 100
sql_err	varchar2, size 250
prog_err	varchar2, size 100

© 2020 General Electric Company. All rights reserved.

NOTE: SQLLIB and SQLERR are the default names for the tables. You may name them anything you want as long as those names are reflected in the SCU. However, the names of the columns ***must*** be spelled exactly as shown (Oracle is not case-sensitive).

Use the following commands to create the Library table, Index, and Error Log table:

```
create table SQLLIB ( sqlname varchar2(8) NOT NULL PRIMARY KEY, sqlcmd varchar2(1000) NOT NULL )
create table SQLERR ( td date NOT NULL, node varchar2(8) NOT NULL, tag varchar2(30) NOT NULL, sqlname varchar2(8) NOT N
```

Supported Column Data Types for Oracle

Use the following table as a guide to iFIX and Oracle field types. The direction (I, O) refers to the direction field in the SQL Data (SQD) block. An I indicates a direction of IN (for selects) and is supported for the Oracle and iFIX field types. An O indicates a direction of OUT (for parameter markers) and is also supported.

Oracle and iFIX Field Types

Oracle Type	DATE Keyword	TIME Keyword	TMDT Keyword	F_CV	A_CV
date	O		O		I
char, varchar2					I, O
number (p,s) float				I, O	I

NOTE: iFIX does not support the use of data types that accept large amounts of data. For example, the LONG data type in an Oracle database or the MEMO field in a Microsoft Access database are not supported. Do not select either data type from View. Doing so may cause unexpected results.

Network Problems and Workarounds

Each time an ODBC function is called, there are two error codes that can be returned: a native database error code and an ODBC-defined SQL state error code. The database error code is database-specific, but the SQL state is a generic error code that is common between different ODBC drivers. For example, when the database server is unavailable because of a network problem, SQL Server may return a 105 error code. In the same situation, Oracle may return a 6034 or a 3140 error code. But in all cases, the SQL state should be 08S01. By using these pre-defined SQL states, the SQL task can detect when a connection is lost and shift into backup mode.

However, in many cases, ODBC drivers do not set the SQL state correctly. To work around this problem, iFIX provides a text file called SQLERR.TXT. You can list all of your database system errors that correspond to a communication failure in this file. If the ODBC driver does not set the SQL state correctly, the SQL task scans the SQLERR.TXT file for a match. If it finds a match, it knows a communication failure has occurred. If it doesn't, the error is processed as a non-communication error.

NOTE: You may receive an wsqlodc.exe application error when using Mission Control to stop an SQL query or to shut down iFIX. If this happens, you can still start this task and log SQL data without error.

Use the following format when editing the SQLERR text file:

```
! This is a comment
! General SQL*net errors for Oracle
2034
6000,6499
!
```


© 2020 General Electric Company. All rights reserved.

```
! Two-task interface errors for Oracle
3100,3199
!
! SQL Server SPX Cable break errors
237
238
```

You can define a single error number or an error range. In the preceding example, if Oracle returned error code 2034, or any error between 6000-6499 or 3100-3199, the SQL task would interpret the error as a communication failure.

The SQLERR.TXT file installs to the Application path, usually C:\Program Files (x86)\GE\iFIX\App. Do not rename this file.

Configuring the SQL Task

The iFIX ODBC SQL task, WSQLODC.EXE, is responsible for:

- Executing the SQT blocks that trigger.

  NOTE: When using the confirm tag for an SQT block that is triggered by an event, make sure that you define the security so that users do not have access to Mission Control to shut down the SQL task.
- Retrieving process data from the SQD blocks and inserting the data in the relational database.
- Selecting data from the relational database and writing the data back to the FIX® database.
- Backing up data in the event of a network failure (backup continues until the primary and secondary paths are full).
- Automatically restoring data to the relational database once network communications are re-established.

Refer to the following sections for information on:

- Modifying the Startup Options
- Using the SQL Task Dialog Box

Modifying the Startup Options

When SQL support is enabled in the SCU, the SQL task is automatically added to the FIX SCU Task Configuration dialog box. The SQL task can be customized for your application by adding command parameters.

For example, if you want to define a login delay of five minutes (300 seconds), a maximum SQL command length of 1000 and a command cache of 10, enter the following parameters in each SQL task's Parameters field:

```
/LD300 /CL1000 /CA10
```

The following table describes the command parameters available to this task.

SQL Task Command Line Parameters

Parameter Description	Valid Entries

© 2020 General Electric Company. All rights reserved.

/LD*n*	Defines how often the task attempts to log onto the server after it has lost a connection. This parameter is optional. The system defaults to 300 seconds if this parameter is not defined.	60 seconds (minimum). No maximum limit set. However, most applications do not need more than one hour (3600 seconds).
/CA*n*	Stores the specified number of SQL commands in memory (caches) to increase performance. This parameter is optional. Performance benefits vary depending on your relational database. Refer to the Using Command Caching section for more information on using caching.	User specified. No maximum limit set. However, most applications do not need more than 10 to 100.
/CL*n*	Increases the maximum allowable SQL command length from the default of 255. For example, entering /CL1000 allows a command length of 1,000 characters. **NOTE:** Make sure that the relational database field can handle the number of characters you define for this parameter.	User specified.
/LM	Includes ODBC driver information along with relational database error messages. When using this parameter, keep in mind that driver information can be lengthy, and there may not be enough space for the actual error message in enabled alarm destinations. This parameter is optional and if omitted, no driver information is included.	None required.
/D*n*	Defines a startup delay before attempting to log into the SQL database. Use this parameter if you encounter problems connecting to the database during startup.	0 to 65535 seconds.

Using Command Caching

Depending upon the relational database you are using, you can improve communication performance by adding the command caching, /CA*n*, parameter to the SQL task. Command caching allows each command that is retrieved and executed to be stored in memory. Performance improves the next time a command is to be executed. Because the command is in memory, it does not have to be retrieved again.

A command can be executed without actually having to access the database that stores the command. Thus, the connection to the command table database can be broken and the task can still use the command.

Using command caching, it is possible to read a command from one database, execute it in another database specified in the SQL Trigger block, and save the handle. Refer to the Using Multiple Relational Database Support section for more information on using multiple relational databases with iFIX ODBC.

Using the SQL Task Dialog Box

The iFIX ODBC SQL Task must be configured to meet your application requirements. This is done through the SQL Task dialog box.

 © 2020 General Electric Company. All rights reserved.

The SQL Task Configuration dialog box defines how the SQL task executes SQL commands and allows you to define how data is handled when iFIX ODBC is operating. Each dialog box field is described below:

SQL Support - enables or disables the FIX ODBC option. Select an option button to enable or disable this application.

Error/Debug Message Routing - displays the Configure Alarm Areas - SQL Error Messages or SQL Debug Messages dialog box. Using this dialog box, you can enable the alarm areas that receive error or debug messages generated by iFIX ODBC. Error messages indicate that the SQL system task encountered an error. Debug messages provide information to help you troubleshoot the connection to the relational database. Refer to the Implementing Alarms and Messages manual for more information on how iFIX performs alarming.

Error/Debug message to screen - determines if error or debug messages are sent to the SQL task. Select each check box to enable this function.

Primary and Secondary Backup Files - defines the primary and secondary back-up paths and file names that iFIX ODBC uses when it cannot write to the relational database. If iFIX ODBC cannot connect to the server, or loses a connection with the relational database, it backs-up data to the file identified in the Primary path field. If iFIX ODBC fails to write to this file, it backs-up data to the file identified in the Secondary path field.

If you set the primary path to a file server, consider setting the secondary path to a local drive. With this configuration, if iFIX ODBC cannot connect to the server because of a bad cable connection, the secondary path can back-up data to the local drive. Once iFIX ODBC re-establishes a connection to the relational database, it first processes any backed up SQL commands and data. The back-up file is deleted when the back-up operation completes.

NOTE: Backed-up SQL commands are processed on a first in, first out (FIFO) basis.

You can enter any valid iFIX path in these fields along with a back-up file name. Enter the path and file name in the following format:

```
drive:path\filename.SQL
```

If your path entry does not exist at runtime, iFIX ODBC generates an error message. This is because it tries to send back-up data to a file that is assigned no destination path. In this case, if backed-up SQL commands and data are stored in a file, it cannot be used in restoring data. The SQL system task does not support the following characters in backup file names:

```
, + * = | < > [ ] " : ; ?
```

Refer to the Backing Up and Restoring Data section for more information on how iFIX ODBC performs backups.

NOTE: For SQT blocks to log to the primary or secondary backup files, you must select the Enable BackUp checkbox found on the Advanced tab. You must do this for each SQT block you want to utilize backup files.

Database ID - defines the default location for both SQL commands and the data for SQL Trigger blocks. The Database ID is the ODBC data source name specified during ODBC setup for the relational database.

If you leave this field blank, you can define a Database ID in each SQL Trigger block instead. Refer to the Using Multiple Relational Database Support section for more information on

© 2020 General Electric Company. All rights reserved.

connecting to several relational databases simultaneously.

SQL cmd table - identifies the name of the SQL Library Table that contains the SQL commands. The default name is SQLLIB. However, the table name can have between 1 and 31 characters (inclusive). Refer to the Installing and Configuring Data Sources section for more information on the SQL Library Table.

Error log table - identifies the name of the error log table to which the SQL task sends error messages. If an SQL transaction fails, an entry is made to this table, providing useful information for debugging SQL transactions.

The default name for the error log table is SQLERR. However, the table can have between 1 and 31 characters (inclusive).

NOTE: Completing the error log table field is optional. If no table name is entered, the application does not record error messages to the relational database.

Task Sleep Interval - determines how often the SQL task processes the SQT blocks in the database. Be sure to enter a time that is sufficient to monitor your application. Valid entries are 0 to 99 seconds. The default is 5 seconds.

Once you have configured the SQL task for your application, the next step is to configure the SQL chains in the iFIX database.

Using SQL Commands

All SQL commands are supported by iFIX ODBC. However, this manual focuses on the most frequently-used SQL commands, shown below:

- INSERT
- UPDATE
- SELECT
- DELETE

SQL commands are stored in the SQL Library table. The SQL Trigger block defines which SQL command to use in this table.

INSERT Command

The INSERT command adds data from iFIX tags into a new row in the relational database. INSERT statements can have only one record associated with them.

SQT1 - the SQL Trigger block defines the SQL name and command to use. In this example, the following INSERT command in the SQLLIB table executes when the SQT block triggers:

```
Insert into TBL1 (COL1, COL2, COL3) values (?, ?, ?);
```

SQD1 - the SQL Data block references three tag and field name combinations in the database, and sets the direction for the data transfer to OUT.

```
T01-. AI1.A_DESC
T02-. AO1.F_CV
T03-. DO1.A_ADI
```


© 2020 General Electric Company. All rights reserved.

TBL1 - is the table referenced by the INSERT command that is used in the SQT block.

COL1	COL2	COL3
Pump stage 101	99.7	Hello
Temp Zone 2 (C)	-2.1	ByeBye
Manual Override	.004	what

Explanation

When the software executes this INSERT command, it creates a new row in TBL1 that contains the values of the iFIX tags and field names listed in the SQD1 block. The resulting table is shown below:

COL1	COL2	COL3
Pump stage 101	99.7	Hello
Temp Zone 2 (C)	-2.1	ByeBye
Manual Override	.004	what
AI1 Descriptor fld	21.04	ABDFG

AI1 Descriptor fld - comes from the AI1 block's A_DESC.

21.04 - comes from the AO1 block's F_CV.

ABDFG - comes from the DO1 block using the A_ADI field.

NOTE: If the SQL task cannot read a value from a block (for example, it attempts to read the current value of an Analog Input block while the block is off scan), the SQL task substitutes a null value in place of the block value. If the target column does not accept null values, a new row is not inserted and the SQL task generates an error. The SQT block also generates an alarm.

UPDATE Command

The UPDATE command changes the values in the relational database to reflect the current values of the iFIX tags.

SQT1 - the SQL Trigger block uses the following UPDATE command defined in the SQLLIB table:

```
Update TBL1 set COL1=?, COL2=? where COL3=?;
```

SQD1 - the SQL Data block references three tag and field name combinations in the database, and sets the direction for the data transfer to OUT.

```
T01-. AI1.A_DESC
T02-. AO1.F_CV
T03-. DO1.A_ADI
```

TBL1 - is the table referenced by the UPDATE command used in the SQT block.

COL1	COL2	COL3
Pump stage 101	99.7	Hello
Temp Zone 2 (C)	-2.1	ByeBye

© 2020 General Electric Company. All rights reserved.

Manual Override	.004	what
AI1 Descriptor	21.04	ABDFG

Explanation

When the software executes this command, it looks at each value in COL3 for the value that matches the A_ADI field for the DO1 block. Since the last row in the table matches, the system updates that row. COL1 and COL2 receive new values from AI1's descriptor and AO1's current value, respectively.

The resulting table is shown below:

COL1	COL2	COL3
Pump stage 101	99.7	Hello
Temp Zone 2 (C)	-2.1	ByeBye
Manual Override	.004	what
New Descriptor	-23.09	ABDFG

New Descriptor - comes from the AI1 block's A_DESC field.

-23.09 - comes from the AO1 block's F_CV field.

ABDFG - comes from the DO1 block using the A_ADI field.

iFIX does not support updates to a date column. If you need to update a date, we recommend that you carefully consider the design for your relational database tables. Consider what data needs to be accessed and how. For example, you may want to keep the year, month, and day in separate columns and then update each column individually.

If the SQL task cannot read a value from a block, it substitutes a null value in place of the block value. If the target column does not accept null values, rows are not updated and the SQL task generates an error. The SQT block also generates an alarm.

SELECT Command

The SELECT command retrieves data from the relational database based on the selection criteria.

SQT1 - in this example, the SQL Trigger block uses the following SELECT command defined in the SQLLIB table:

```
Select COL1, COL2 from TBL1 where COL3=?;
```

SQD1 - in this example, the SQL Data block references three tag and field name combinations in the database, and sets the direction for two of them to IN and the other to OUT.

```
T01-. AI1.A_DESC
T02-. AO1.F_CV
T03-. DO1.A_ADI
```

TBL1 — is the table referenced by the SELECT command used by the SQT block.

COL1	COL2	COL3
Pump stage 101	99.7	NONE

© 2020 General Electric Company. All rights reserved.

Temp Zone 2 (C)	-2.1	ByeBye
Manual Override	.004	NONE
Deadband	9	ALL

Explanation

When the software executes this command, the value for DO1.A_ADI is read, since it is an outgoing field. The value for the field is ALL. This command retrieves only the last row since it is the only row that matches the selected criteria. The values from COL1 and COL2 in the last row are selected from TBL1 and are written to AI1.A_DESC and AO1.A_CV.

The new descriptor for AI1 is now Deadband. The new current value for AO1 is now 9.

The process database does not accept null values. If the SQL task reads a null value from the SQL table, no value is written to the target database block field. Additionally, the SQT block generates an alarm and a field write error. However, any other non-null values selected are written to the process database.

Selecting Multiple Rows

iFIX ODBC can be used to access more than one row with a SELECT command and write the values to multiple sets of tags or to multiple offsets of register blocks. This subsection describes how you can use iFIX ODBC to select multiple rows of SQL data.

The selection mode is determined by the SQL Trigger block configuration. Use the following table to configure the SQL Trigger block:

To select....	Define the Select Parameters field as...	And use the Rows field to determine the...
One row from several selected rows (result set)	Single Row	Row used.
Many rows	Multiple Rows	Starting row.
Many rows and columns	Array Mode	Limit of how many rows are used out of the result set.

The SQT block's Column field is used to determine the number of columns to be used. This applies only when using Multiple Rows and Array Mode Select Parameters.

NOTE: When querying iFIX database tags, you cannot retrieve the A_CV value from a tag or group of tags when the tag or tags are off scan. If you try to do so, an error message appears.

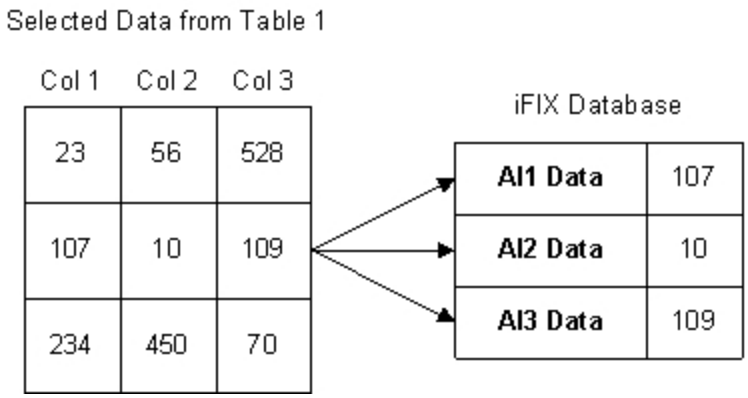
Single Row

If a SELECT command returns multiple rows, you can use the SQT block's Single Row mode and the Rows field to determine which row is written to the SQD block. When the SQT block is configured for Single Row mode, the SQD block accepts only one row of data regardless of the number of rows the SELECT command returns.

The SQT block's Rows field determines the row number within the result set to use. The Rows field normally defaults to zero when you select Single Row mode in the SQT block. If zero is the value in Single Row mode, and more than one row is returned, an error results and no data is written to iFIX tags. If a number other than zero is used, the corresponding row number returned from the selected is used.

© 2020 General Electric Company. All rights reserved.

For example, if the Row field is set to one, the first row of the selected data is used. If it is set to two as shown in the following figure, the second row is used. In this manner you can identify the row to use.



Single Row Mode Example

The Columns field is ignored in this mode. The number of columns is defined by the number of tags in the SQD block that have a direction of IN. If the number of IN tags does not match the number of columns returned, an error results and no data is written.

Multiple Row

When the SQT block is configured for Multiple Row mode, the SQD block accepts more than one row. A set of iFIX tags is used for each row returned. Returned values are written column by column, row by row.

The Columns field must be configured with the correct number of columns in the SELECT command. This information must be defined before the command is executed.

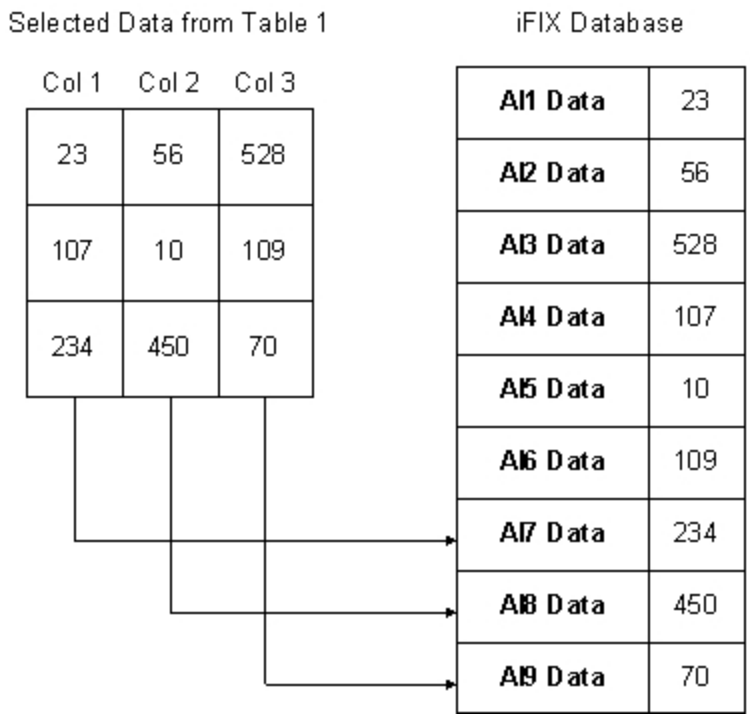
For example, consider the following SELECT statement:

```
Select col1, col2, col3 from table1
```

This command returns three columns from the database. The SQD block must contain a multiple of three iFIX tags. If the SELECT command returns two rows of data, six tags should be specified in the SQD block. If more rows are returned by the SELECT command than tags defined in the SQD block, the additional rows are discarded. If less rows are returned, the extra tags in the SQD block are not written to.

The Rows field determines the starting row to use in the selected data. For example, the Rows field can be defined as 3. If the SELECT command returns 10 rows, the tags in the SQD block receive the values starting with the third row. If the number of IN tags in the SQD block is not an even multiple of the number of columns returned, an error results and no data is written to the tags.

© 2020 General Electric Company. All rights reserved.



Multiple Row Mode Example

You can also use the SELECT command to include parameter markers. For example:

```
Select col1, col2, col3 from table1 where col4 = ?
```

In this case, three columns are returned. The command requires one tag with a direction of OUT for the parameter marker.

Array mode

Array mode is used with register blocks. One register block is specified for each column returned from a SELECT statement. Each register block receives multiple rows from a column. The Rows field is used to set a limit on how many rows to write.

In the following example, the SELECT statement returns three columns and requires one parameter marker:

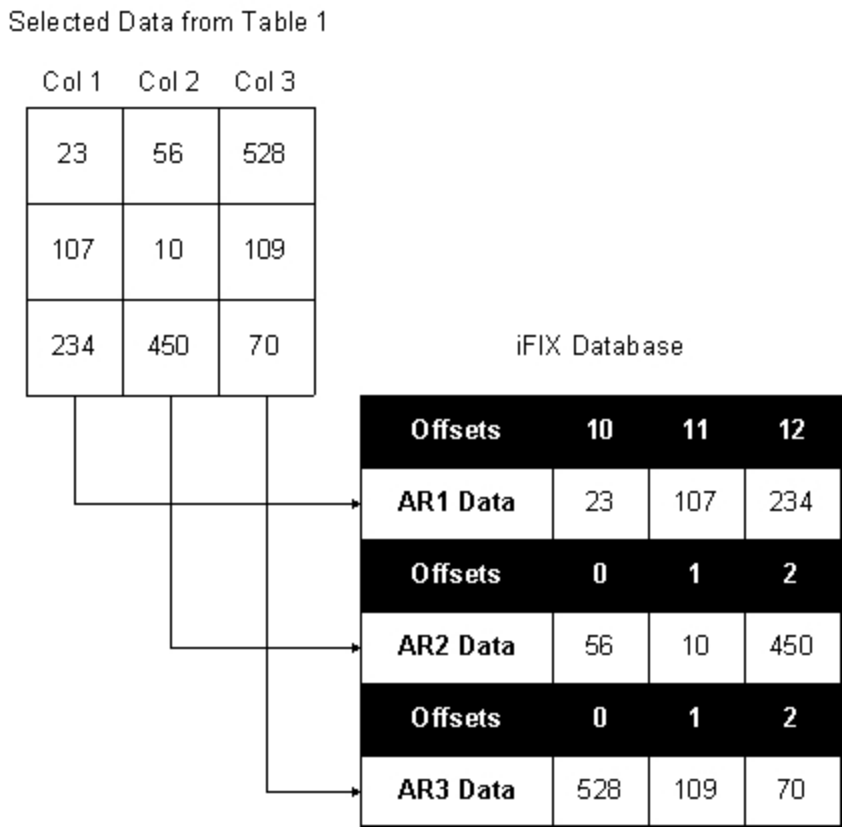
```
Select col1, col2, col3 from table1 where col4 = ?
```

In array mode, the SQD block contains one IN register block for each column returned. A tag with an OUT direction is specified in the SQD block for each parameter marker.

Each Register block receives one row of data starting with the register offset specified in the SQL Data block. Data is written until either the Rows limit is reached (specified by the Rows field) or until the data is exhausted.

The F_*n* field is used for the register block in the SQD block, where *n* is the offset from the base address that the block references. For example, AR1.F_10, AR2.F_0, and AR3.F_0 are shown in the following figure. Values from the selected data are written to the blocks starting at the offset specified, 10,0 and 0 respectively.

© 2020 General Electric Company. All rights reserved.



Array Mode Example

The Columns field is ignored in this mode since the number of columns is the same as the number of IN SQD tags defined.

DELETE Command

The DELETE command removes records from the relational database based on the selection criteria.

SQT1 - the SQL Trigger block uses the DELETE command in the SQLLIB table. The DELETE command executes the following statement when the SQT block triggers:

```
Delete from TBL1 where COL3 = ?;
```

SQD1 - the SQL Data block references a tag and field name combination in the database, and sets the direction for the data transfer to OUT.

```
T01-.DO1.A_ADI
T02-.
T03-.
T04-.
```

TBL1 - is the table referenced by the DELETE command used by the SQT block.

COL1	COL2	COL3

© 2020 General Electric Company. All rights reserved.

Pump stage 101	99.7	NONE
Temp Zone 2 (C)	-2.1	ByeBye
Manual Override	.004	NONE
Deadband	9	ALL

Explanation

The value of DO1.A_ADI is equal to NONE. When the software executes this command, the value is read and substituted into the SQL command. The command is then:

```
Delete from TBL1 where COL3 = `NONE';
```

Since both row one and row three have `NONE' as the value in column three, they are both deleted. The resulting table is shown below:

COL1	COL2	COL3
Temp Zone 2 (C)	-2.1	ByeBye
Deadband	9	ALL

Stored Procedures

Stored procedures are compiled blocks of code in the relational database. They are useful since they can have conditional statements and flow statements. Stored procedures can perform INSERT, UPDATE, DELETE, and SELECT commands. However, stored procedures can also take arguments and return results. The arguments may be values to insert or values to use in where clauses.

Procedures can be much faster than SQL commands for the following reasons:

- Executing a stored procedure requires only one call.

  To execute an SQL command, two calls are made to the relational database: one to retrieve the command and another to execute it. Using a stored procedure, the reference is made by name and the name is passed to the database, including any parameters.

- A stored procedure is already compiled in the database.

  When using an SQL command that is not stored, both calls are *ad-hoc* queries, so the database must compile them at runtime (unless caching is used). Refer to the Using Command Caching section for more information on configuring the SQL task to use command caching.

To configure the SQL blocks to use a stored procedure, use the following information:

- Select Procedure in the Command Type group box in the SQT block's SQL Parameters dialog box.
- Enter the name of the procedure in the SQL Name field, using no more than eight characters.
- Define any input arguments required by the stored procedure using the Direction field in the SQD block.

  For example, if the stored procedure takes two input arguments and returns data from a SELECT statement, configure the SQD block with OUT tags for the parameter markers and IN tags for the

© 2020 General Electric Company. All rights reserved.

results from the SELECT statement.

Refer to the Selecting Multiple Rows section for more information on configuring the database blocks to use multiple rows of data.

NOTE: Microsoft Access does not support the use of stored procedures. If you are using Microsoft Access as your relational database, do not use the PROCEDURE command in your command language scripts and do not select the Stored Procedures option button in the SQL Command Configurator. Doing so can produce unexpected results.

Using Multiple Relational Database Support

Multiple database support allows iFIX ODBC to execute commands in different databases. Implementing multiple relational database support allows you to:

- Use different user name/password combinations in a database.
- Communicate with several different relational databases.
- Store all SQL commands in one database and execute the commands in several different relational databases.

NOTE: The methods of storing SQL commands and using multiple relational databases discussed in this chapter cannot be mixed.

This chapter includes the following sections:

- Managing Multiple SQL Connections
- Using Multiple User Accounts
- Using Multiple Databases
- Storing Commands Centrally

Managing Multiple SQL Connections

The SQL task manages multiple connections using the following:

When the...	The SQL task...
System starts up	Reads the SQL Task Configuration information from the SCU and logs into the Database ID (if defined) in the SQL Task Configuration. This data source is the default login.
SQT Block triggers	Logs into a database using the following search pattern for Database IDs: 1. SQT block's Database ID. 2. SCU Database ID (location for command & Errlog tables). 3. First Database ID listed in the SQL Accounts dialog box.
Connection is established	Maintains the connection until the SQL task is stopped.

© 2020 General Electric Company. All rights reserved.

Connection is lost	Retries the connection at the interval specified by the /LD parameter and continues processing SQL blocks to other data sources. Commands are backed up to the backup file specified in the SCU until the SQL task can re-establish a connection to the database.
Connection is re-established	Searches through the backup file for entries. Entries to databases that are not connected are ignored until the connection can be re-established.

Using Multiple User Accounts

You may want to use the same database for all transactions, but have different user name/password combinations. For example, an Oracle database can be configured for two users. One user has access to one set of tables, the other user to other tables.

To configure iFIX ODBC:

1. Click the Start button and point to Programs, Administrative Tools, and then Data Sources (ODBC). You can also access the Administrative Tools folder from the Control Panel.

   The ODBC Data Source Administrator program opens.
2. Select the System DSN tab.
3. Select the Add button. The Add Data Sources dialog box appears.
4. Select the type of relational database you want to use and click OK. A dialog box appears.
5. In the Data Source Name field, enter the data source name. For example, enter ORA_USER1. Make sure the data source name matches the name defined in your project.
6. In the Server field, enter the name of your relational database server.
7. Click OK or continue through the rest of the configuration.
8. Repeat steps 3-7 to add another DSN. For example, ORA_USER2. Make sure that the information you enter matches the first DSN (since they are for the same database), except for the name.
9. In the SCU, configure two SQL login accounts (one for each data source) and give each account a different user name and password.
10. In the SQT blocks, in the Database ID field, create the iFIX database chains, defining the correct account, ORA_USER1 or ORA_USER2.

The SQL Task Configuration does not include a reference to a database identifier. Instead, you must define the Oracle account in each SQT block using the Database ID field.

When an SQT block triggers to execute a command using ORA_USER1, the SQL task logs into the relational database with the this user name and password. When an SQT block triggers to execute a command using ORA_USER2, the SQL task logs into the same database but with a different user name/password combination.

Using Multiple Databases

© 2020 General Electric Company. All rights reserved.

You can use iFIX ODBC to access several different relational databases you have installed in your configuration. In this mode, the command and error log table can also be located in the same account. However, the tables must use a consistent naming convention. For example, if you name the SQL command table SQLLIB in one relational database, you must use this name in the other relational databases.

You can store recipes to download in an Access database and log plant values in an Oracle database.

To define this configuration:

1. In the Control Panel, select Administrative Tools, and then Data Sources (ODBC). Select the System DSN tab, and click add to create an ODBC data source for each relational database. For example, the data sources can be called ACC_DB and ORA_DB.
2. In the SCU, configure two SQL accounts (one for each data source). The Database ID field should be the data source names defined in the previous step: ACC_DB and ORA_DB.
3. In the SQL Task Configuration dialog box, leave the Database ID field blank.
4. In each SQT block, in the Database ID field, type the appropriate database ID to access the relational database.

The SQL Task Configuration does not include a reference to a database identifier. Instead, define the relational database in the Database ID field of each SQT block.

When SQT1 triggers, the configured command executes in the database, ACC_DB, defined in the block configuration. In this example, the chain triggers a recipe download. The blocks defined in the SQD block accept the downloaded recipe values. The iFIX database chain for SQT2 logs plant values to the Oracle database using ORA_DB as the Database ID.

Storing Commands Centrally

Multiple database support can store all the SQL commands and errors in one location. You can then run these commands to access data in other relational databases. For example, you can store all of your configured commands in one Access database, the recipes to download in another Access database, and log plant values to an Oracle database.

To define this configuration:

1. In the Control Panel, select Administrative Tools, and then Data Sources (ODBC). Select the System DSN tab, and click add to create an ODBC data source for each relational database. For example, the data sources can be called CMD_DB, ACC_DB, and ORA_DB.
2. In the SCU, configure three SQL login accounts (one for each data source). The Database ID field should be the data source names defined in the previous step: CMD_DB, ACC_DB, and ORA_DB.
3. In the SQL Task Configuration dialog box, define CMD_DB as the Database ID field. The list of configured Database IDs in the Accounts dialog box is presented when the browse (...) button is selected.
4. In each SQT block, in the Database ID field, type the appropriate database ID to access the relational database. For example, if the command is executed in the Access database, then for the SQT block, in the Database ID field, type ACC_DB.

Refer to the Configuration for Multiple Databases figure to view the information you need to define in the SCU to use two different relational databases. The Database ID in the SQL Task Configuration refers to

 © 2020 General Electric Company. All rights reserved.

the relational database table that is used to store all the SQL commands that are used with all of the relational databases. The SQT block Database ID field defines where the command is executed. The following figure illustrates the information you need to define in the SCU to use a relational database to store all the SQL commands.

When executing a SQL command, the command is retrieved from the CMD_DB database and executed in the relational database specified in the SQT block.

Monitoring and Controlling Database Communication

There are a few ways you can monitor and control iFIX ODBC during runtime operation. Since you can observe your process from an operator display, the procedures described in this chapter relate to what you can do through display links:

- Changing Block Settings Through Links
- Transaction Tracking

Changing Block Settings Through Links

The real power of the software is in its ability to perform process operations tailored to your needs. You can use the SQL block fields in links to provide greater flexibility in communicating with the relational database. Using these link fields, you can create pictures that allow you to:

- Manually trigger SQT blocks.
- Reset the trigger specifications.
- Reset the SQD block specifications.

The link fields that are available for use with the two SQL blocks can be accessed through the Properties dialog box in the Database Manager.

To access these link fields:

1. Start the Database Manager.
2. In Ribbon view, on the View tab, in the Settings group, click Properties.

   - Or -

   In Classic view, on the View menu, click Properties.

   The Properties dialog box appears.

Manually Triggering the Application

iFIX ODBC allows you to trigger SQT blocks through a time, a date, or a tag-based event. You can also manually trigger the execution of an SQL command through a link, providing direct control over iFIX ODBC while testing a new process or while monitoring the performance of your application.

© 2020 General Electric Company. All rights reserved.

By defining a Data link with the A_TRIP field, you can first change the SQT block's mode to Manual and then enter a 1 to trigger the SQT block manually. This provides a powerful means of executing SQL commands on demand.

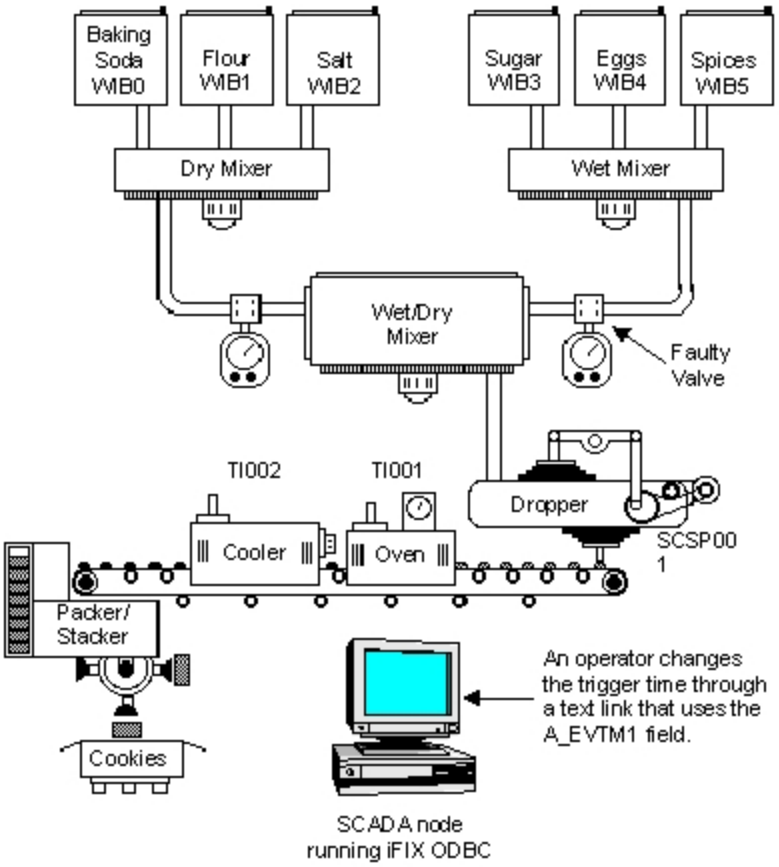
Resetting Time/Date/Events Specifications

Even if you have defined a time, a date, or an event-based triggering scheme, you can change these settings through links. This lets you redefine the operation of your system without modifying the SQT blocks. It provides you with the ability to change the execution of a batch, the upload of process information, or suspend operation until maintenance is performed on critical operating equipment.

For example, if iFIX ODBC downloads batch recipes that are stored in a relational database, operation can be suspended and rescheduled to a future date through a display link.

The following figure shows a cookie production line that has experienced a mechanical failure. The next recipe download has been rescheduled by changing the trigger time through a display link.

NOTE: When a Text block is used to trigger an event, ODBC SQL can only read the first 40 characters from the block's A_CV field. If you include more than 40 characters in this field, the additional text is ignored.



Resetting Execution Times Through Links

Resetting SQD Blocks

© 2020 General Electric Company. All rights reserved.

The SQD blocks are modifiable from an operator display. Through links, you can:

- Redefine tag and field name assignments, either by adding new ones, or deleting existing ones (using A_TF01-20 fields).
- Change the direction of the data transfer (using A_DIR01-20 fields).
- Reset fields in the block (using A_RST01-20 fields).

Transaction Tracking

The following field parameters in the SQT block display information that can help you in debugging your setup or help monitor real-time transactions:

- A_SEQ displays a transaction number indicating that the block has recognized an event. Transaction numbers increment from 1 to 255.
- A_XTIME displays the time of the last transaction.
- A_XDATE displays the date of the last transaction.
- A_STATE displays the current operating state of each SQT block.
- A_DBERR displays the native database error number of the last SQL command executed.
- A_SQLST displays the SQL state of the last SQL command executed.

© 2020 General Electric Company. All rights reserved.

Displaying Alarm and Application Messages

iFIX is capable of sending system-wide error, debug, and system messages relating to iFIX ODBC. These alarm and trace messages can appear in the SQL task's window and in alarm destinations. Refer to the following sections for more information:

- Displaying Alarms
- Displaying Process Messages
- Generating Status Reports

Displaying Alarms

The SQL blocks and the SQL task can be set up for independent alarm and security areas. System messages that are displayed through Alarm Summary Display links provide you with real-time iFIX ODBC information. The SQL blocks can generate unique alarms that you can monitor through an Alarm Summary link or in other alarm destinations.

When building a display to monitor your SQL application, consider adding a Data link that connects to the SQL Trigger block. This block has two alarm link parameters: current alarm (A_CUALM) and latched alarm (A_LAALM). These link parameters display the alarms and block errors listed (by priority) in the following table.

SQL Trigger Block Alarms and Block Errors

Alarm or Block Error	Description
ERROR	An undetermined error occurred.
SQL LOG	An error occurred and the SQL application could not connect to the relational database or you lost a connection.
SQL CMD	An error occurred because the application could not find the SQL command or the command was too long.
FLD READ	An error occurred while reading an iFIX field.
FLD WRIT	An error occurred while writing to an iFIX field.
DAT MATC	An error occurred because the number of values in the SQL command does not match the number of items in the SQL Data block.
OK	No error has occurred.

Refer to the Implementing Alarms and Messages manual for more information on alarming.

Displaying Process Messages

Error, debug, and system messages can be viewed in the following locations:

© 2020 General Electric Company. All rights reserved.

- The SQL task window.
- Alarm destinations.

Messages appear in...	If...
The SQL task window	The Error msg to screen check box or the Debug msg to screen check box is selected.
Alarm destinations	One or more alarm areas are enabled in either the Configure Alarm Areas - SQL Error Messages or SQL Debug Messages dialog box. You also need to enable the required Alarm Services for these selected alarm areas.

Displaying Alarms

You can display SQL block alarms using the Alarm Summary Link. SQL alarms appear in this link if the SQT block generates an alarm.

Generating Status Reports

You can display the setup and status of iFIX ODBC in Mission Control, on the SQL tab. Each status field is described in the following table.

System Task Status Codes

Status Code	Description
Alarm adi	Displays the enabled alarm areas.
Debug adi	Displays the enabled debug areas.
Alarm screen	Indicates whether alarm messages are sent to the SQL task. Yes = send messages; No = send no messages.
Debug screen	Indicates whether debug messages are sent to the SQL task. Yes = send messages; No = send no messages.
Sql cmd tbl	Name of the SQL Library Table.
Sql err tbl	Name of the SQL Error Log Table.
Sql cmd dbid	Name of the Database Identifier specified in the SCU's SQL Task Configuration.
Prim file	Primary back-up file.
Second. file	Secondary back-up file.
Login delay	Number of seconds to wait before attempting to login once a connection is lost. The default is 300 seconds.
Max sql-cmd	Maximum SQL command length.
Nodename	Name of the local node.
Current status	Identifies the status of the SQL task. This field can display whether the SQL task is logged onto one or several relational databases.

© 2020 General Electric Company. All rights reserved.

Caching status	Status of SQL caching.

Backing Up and Restoring Data

During normal operation of iFIX ODBC, data communication takes place successfully across the network as required by your application. Refer to the following sections for more details:

- Backing Up Data
- Restoring Back-up Data

Backing Up Data

If the SCADA node running iFIX ODBC loses a connection to the relational database, every INSERT, UPDATE, and DELETE command that is enabled for backup is automatically sent to either a primary or secondary back-up file.

In situations where the server connection is lost for long periods of time, iFIX ODBC backs up data until the:

- Primary and secondary back-up paths (disks) are full.
- Connection to the server is re-established.
- SQL task is stopped by an operator.

IMPORTANT: iFIX ODBC does not back up a SELECT command's request because there is no means of accurately determining when the connection to the server can be re-established. Since the SELECT command writes values to the process database, the process of selecting must be performed on a controlled and predictable basis (not whenever the connection is re-established).

Restoring Back-up Data

When iFIX ODBC re-establishes a connection with the server, it automatically restores the backed up SQL commands in chronological order, starting with the first backed up SQL command. iFIX ODBC restores backed up data as described below:

1. All backed up SQL commands are restored from both primary and secondary back-up files.
2. iFIX ODBC processes backed up SQL commands in the order in which they were backed up. This means that the backed up SQL commands are processed in a first in, first out (FIFO) basis.
3. Once all SQL commands are restored, the system deletes the back-up files.

NOTE: If large back-up files are created, restoring the SQL commands and data can take a considerable amount of time. However, new SQL commands from the SCADA node are also processed while the files are restored.

© 2020 General Electric Company. All rights reserved.

Index

© 2020 General Electric Company. All rights reserved.

J

L

M

N

O

R

© 2020 General Electric Company. All rights reserved.

© 2020 General Electric Company. All rights reserved.