StarSQL™ for UNIX User's Guide

Version 6.4

v64_04.01.2020

Contents

CHAPTER 1 Introduction

StarSQL is available as an ODBC driver for Windows- and UNIX-based computers, and as a JDBC driver for any computer that has the Java Runtime Engine (JRE) or Java Virtual Machine (JVM) installed.

The StarSQL software is not copy protected, rather the usage is limited based upon the maximum number of concurrent connections (“CCs”) licensed. The StarQuest license allows for you to install and run any of the StarSQL drivers on any number of client computers, subject to terms of the license grant. CCs may be made available for clients on a single computer (“Node-locked License”) or any computer on the network (“Floating License”).

The StarSQL for UNIX Driver

The StarSQL for UNIX driver is an Unicode driver that is compliant with the ODBC v3.0 specification. The StarSQL driver resides on the UNIX client computer, and the data resides in a DB2 database on a host computer.

The StarSQL driver performs the following functions to enable ODBC applications to communicate with a DB2 host:

- provides a programmatic interface for executing dynamic SQL functions
- provides transparent translation of character encoding between different client and host operating systems

StarSQL for UNIX supports directly connecting to a DB2 host in a TCP/IP or SSL-encrypted network.

StarSQL for UNIX consists of two versions of the ODBC driver—a 32-bit version and a 64-bit version. The 32-bit version can be installed and run on either a 32-bit or 64-bit computer; the 64-bit version is optimized for use only on a 64-bit computer.

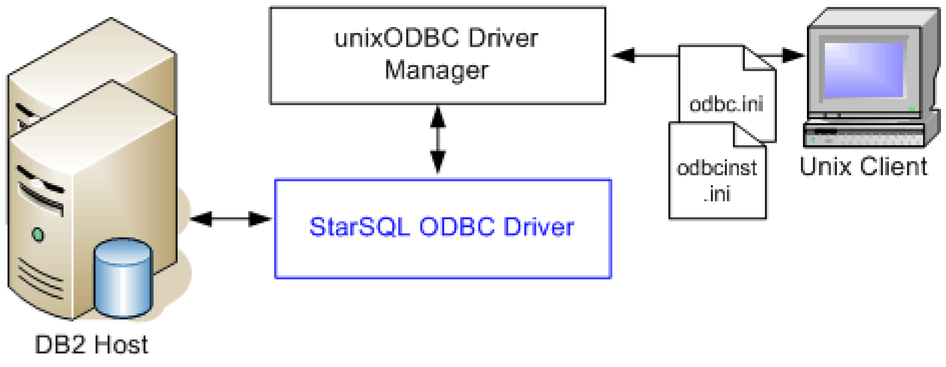
This User's Guide describes how to install, configure, and use both the 32-bit and 64-bit version of the StarSQL for UNIX driver. Refer to the product documentation that is included with the StarSQL for Java or StarSQL for Windows driver to install, configure, and use those versions of the driver.

The ODBC Driver Manager

An ODBC driver manager works with an ODBC driver to provide interoperability among multiple types of databases (such as DB2 and Oracle), and runtime binding to a data source. This allows an application to link only with the driver manager shared object (such as `libodbc.so` on Linux), eliminating dependencies on which ODBC driver will be used. StarSQL for UNIX includes the unixODBC driver manager for use with the StarSQL driver.

The ODBC Driver Manager maintains a repository of the installed ODBC drivers, including the StarSQL driver, in a file named `odbcinst.ini`. The configuration information for system data sources, including those used by the StarSQL driver, are stored in a file named `odbc.ini`. As shown in the following illustration, the ODBC Driver Manager uses the information in the driver repository and the data source configuration to load the appropriate driver. The StarSQL driver manages the data that is sent between the client and host computers.

Figure 1. StarSQL Driver and ODBC Driver Manager Enable UNIX Client to Communicate with DB2 Host



The ODBC driver manager resolves data source names and ensures that the proper driver is loaded and unloaded at the appropriate time. The driver manager also maps calls from applications developed to the ODBC v2 API to the ODBC v3 API so the applications based on v2 can run under the v3 standard without modification.

The StarSQL for UNIX distribution includes the unixODBC driver manager, which is an open source project available under the GNU Library General Public License (LGPL). For more information about unixODBC, see http://www.unixodbc.org.

If an existing driver manager that meets the minimum requirements is not found when you install the StarSQL software, the unixODBC driver manager that is included with StarSQL is installed The StarSQL distribution also includes the driver manager ODBCConfig utility to help you easily configure StarSQL data sources for an ODBC application to use.

Note

The version of unixODBC that is included with StarSQL meets the minimum requirements and has been tested to work with the StarSQL driver. The StarSQL driver requires the following versions of the unixODBC driver manager:

— Release 2.3.4 or later

Network Requirements

The network requirements for using StarSQL and a list of the supported DB2 hosts are provided below. The system requirements for the client computer are provided on page 19 for the 32-bit version of StarSQL, and on page 23 for the 64-bit version. Because the StarSQL configuration involves the client computer, the network, and the host, several individuals may be involved with setting up the StarSQL environment.

Network Protocols

StarSQL for UNIX supports TCP/IP or SSL access to a DB2 host. Connections can also be made through a StarPipes gateway.

Supported DB2 Host Systems

Before using StarSQL on the UNIX computer, the host must be prepared as described in "Preparing Hosts for StarSQL Access" on page 39. You will need information about the network and the host databases, as described in "Preparation Required for All Hosts" on page 39, to configure StarSQL data sources after StarSQL is installed.

StarSQL for UNIX can connect to any of the following host databases:

- DB2 for z/OS v8.1 and later
- DB2 for i (formerly known as DB2/400, DB2 UDB for iSeries, and DB2 for i5/OS) running OS/400 V5R4 and later
- Db2 (formerly known as DB2 for Linux, UNIX, and Windows (DB2 LUW)) v8.2 and later

Data Type Mappings

StarSQL v5 added support for DB2 LOB data types. The DB2 types are mapped to ODBC SQL types as shown in Table 1:

Table 1. Mapping of DB2 Data Types to ODBC SQL Data Types

DB2 Type	StarSQL ODBC SQL Type
BLOB	SQL_LONGVARBINARY
CLOB	SQL_LONGVARCHAR
DBCLOB	SQL_LONGVARCHAR

As shown in Table 2, StarSQL v5 changes the mapping for DB2 long strings—they are no longer differentiated from short strings:

Table 2. Mapping of DB2 Strings

DB2 Type	StarSQL v4.x ODBC SQL Type	StarSQL v5 and v6 ODBC SQL Type
VARCHAR	SQL_VARCHAR	SQL_VARCHAR
LONG VARCHAR	SQL_LONGVARCHAR	SQL_VARCHAR
VARGRAPHIC	SQL_VARCHAR	SQL_VARCHAR
LONG VARGRAPHIC	SQL_LONGVARCHAR	SQL_VARCHAR

DB2 Type	StarSQL v4.x ODBC SQL Type	StarSQL v5 and v6 ODBC SQL Type
VARCHAR FOR BIT DATA	SQL_VARBINARY	SQL_VARBINARY
LONG VARCHAR FOR BIT DATA	SQL_LONGVARBINARY	SQL_VARBINARY

These new mappings affect ODBC SQL catalog query results. For example, SQLColumns for a DB2 LONG VARCHAR returns ODBC type SQL_VARCHAR for a package bound by StarSQL v5. If StarSQL v4.x uses packages bound by StarSQL v5, catalog queries will return the new StarSQL v5 results, but otherwise the driver should run as before.

Parameters for SQL statements are also affected. If an application binds a parameter as SQL type SQL_LONGVARCHAR, it is sent as a DB2 CLOB. StarSQL v4.x sends SQL_LONGVARCHAR as a DB2 VARCHAR string.

StarSQL has an optional data source setting, LongStrParams (see "LongStrParams" on page 91), for applications that require backwards compatibility for SQL_LONGVARCHAR (or SQL_LONGVARBINARY) parameters. This data source setting does not affect the types returned for result set columns, or for types returned by catalog queries.

StarSQL no longer returns type SQL_FLOAT from SQLGetTypeInfo. StarSQL maps DB2 REAL to SQL_REAL, and DB2 DOUBLE to SQL_DOUBLE, and only these two floating point types are returned by SQLGetInfo. StarSQL still accepts SQL_FLOAT, as a synonym for SQL_DOUBLE, as a type, for example, for SQLBindCol.

For an introduction to LOB support in DB2 for z/OS, see the IBM Redbook "Large Objects with DB2 for z/OS and OS/390" (SG24-6571), available at http://www.redbooks.ibm.com.

Documentation

There are many sources of information that can help you install, configure, and use the StarSQL driver. The following sections describe the information available from StarQuest and provides references to other information that may be particularly useful.

Quick Path to Using StarSQL

StarQuest provides StarSQL Quick Start Guides that provide step-by-step instructions for quickly installing and using the StarSQL ODBC and JDBC driver on a particular computing platform. The procedures in the Quick Start Guides are appropriate for the most common environments and describe the fastest way to install and configure the software you need to begin using the driver. If you need to customize the StarSQL driver settings or have an environment for which the default values are not appropriate you can refer to the product documentation for details.

All the Quick Start Guides are listed at http://www.starquest.com/docs/Supportdocs/browseQuickStarts.shtml.

StarSQL Product Documentation

The StarSQL for UNIX product documentation consists of the following components:

- this User's Guide
- Release Notes
- Command Manual Pages
- Technical documents in the Info Center (website)

User's Guide

This User's Guide provides information about installing the StarSQL software, and configuring a client license to use the driver. It also describes how to use the StarSQL driver and the utilities and programs that are included with it. Licensing is managed by StarLicense server software, which includes separate documentation for installing and configuring a StarLicense server.

Organization

The 32-bit and 64-bit versions of the StarSQL driver differ mainly in how they are installed and configured. Therefore, the information about installing and configuring the 32-bit and 64-bit driver is provided in separate chapters and appendixes.

To install:	Refer to:
StarSQL 32-bit driver	"Installing 32-bit StarSQL for UNIX" on page 19
StarSQL 64-bit driver	"Installing 64-bit StarSQL for UNIX" on page 23, or
both 32- and 64-bit StarSQL drivers	"Installing 32-bit StarSQL for UNIX" on page 19, and "Installing 64-bit StarSQL for UNIX" on page 23

The remaining chapters and appendixes are common to either version of the driver, with any minor differences noted in context.

Conventions

The StarSQL driver can be installed to a user-defined location, and differs if both the 32-bit and 64-bit versions of StarSQL are installed on the same computer. The User's Guide shows **`$STARDIR`** to indicate the location where the StarSQL software is installed. Substitute the $STARDIR variable with the full pathname to the installation directory.

You also can specify and export an environmental variable that defines the location of the StarSQL software rather than typing the path in place of the $STARDIR placeholder. For example, you could specify an environment variable named $STARDIR to specify the location to the 32-bit driver and, if you also install the 64-bit version of StarSQL on the same computer, specify a $STARDIR64 environment variable to provide the path to that version. The $STARDIR placeholder in the documentation refers to the location of whichever version of StarSQL you want to use.

Release Notes

The Release Notes contain important information about using StarSQL for UNIX in specific environments, known limitations, and a history of changes to the driver software. The Release Notes specific to each version of StarSQL for UNIX are located in the respective 32-bit and 64-bit subdirectories.

Command Manual Pages

StarSQL for UNIX provides commands for viewing packages, testing connections, administering licenses, and troubleshooting. These commands are described in *Appendix C* on page 67 of this User's Guide. You also can obtain information about the commands in the form of man pages, which are located in $STARDIR/man. Change to the $STARDIR/man directory and then enter the **man** command along with the name of the StarSQL command you want to display information about. The following example would display the man page for the **starping** command.

```
# cd $STARDIR/man
```

```
# man starping
```

TContacting StarQuest

Please use the following methods to contact StarQuest Ventures if you need to obtain a license key, or have suggestions or need information about StarQuest products.

Support

The easiest method for licensing StarQuest products is to use the online licensing feature of the License Configuration utility (see "Licensing StarQuest Products" on page 30). If you cannot request a license over the Internet you can obtain a license key for a StarQuest product by sending an email to contact@starquest.com with the following information:

- TCP/IP address or Host ID of the computer on which the license will be installed
- Number of connections purchased
- Company Name
- Contact Name
- Phone Number
- Email Address

StarQuest Support will send a reply email that provides the license key for your organization's use of the product. Since the license is unique to the computer on which it will be installed, you must contact StarQuest should you need to move the license from one computer to another.

Additional technical support may be available subject to the prices, terms, and conditions specified in your organization's maintenance contract with StarQuest Ventures, Inc.

Sales and Service

If you have ideas for product enhancements or need more information about StarQuest products, please contact us via any of the following methods.

Address	StarQuest Ventures, Inc. 548 Market St, #22938 San Francisco, CA 94104-5401
Telephone	415-669-9619 Option 1: Sales Option 2: Technical Support
Fax	415-669-9639
Email	support@starquest.com
World Wide Web	www.starquest.com

CHAPTER 2

Installing 32-bit StarSQL for UNIX

This chapter describes how to install and configure the 32-bit version of StarSQL for UNIX. It also discusses issues to consider if you are upgrading from a prior version of StarSQL. Be sure to review the Release Notes included in the distribution for important information about installing or upgrading the StarSQL for UNIX driver.

If you do not already have one or more StarLicense servers set up, you may want to obtain license keys and install and configure the StarLicense software, before you install StarSQL. Licensing for StarSQL for UNIX is further discussed in "Licensing StarQuest Products" on page 27, and the StarLicense server software includes documentation for installing and configuring a license server.

Client Computer Requirements

Refer to "The StarSQL for UNIX Driver" on page 9 for a description of the network and host requirements for using StarSQL. The 32-bit version of StarSQL for UNIX has been tested to work with the following operating systems.

- Red Hat Enterprise, Oracle, or CentOS Linux 6.x and later (x86/x64)
- SUSE Linux Enterprise Desktop or Server (x86/x64)
- Ubuntu Desktop or Server Edition
- Other distributions of Linux may also work with the StarSQL driver.

You also can run 32-bit StarSQL on a 64 bit Linux (x64) system if the calling application is a 32-bit x86 binary and you install the 32-bit runtime compatibility libraries.

Upgrading StarSQL

Versions of the StarSQL software are categorized as major releases, point releases, and hot-fixes. Major releases usually are designated with a new major release number such as v4.1, v5.3, or v5.5. Point releases provide defect corrections to a major release, and are designated with a version number such as v5.21, v5.34, 5.51 and so on. If a particular problem arises between scheduled releases, one or more of the StarSQL components may be provided as a “hot-fix” until the fix is incorporated into a point release or a major release.

If you are running a previous version of StarSQL for UNIX, the installer will automatically uninstall the prior version before it installs the new version of StarSQL. Before you start the installation, review the Release Notes that are included with the StarSQL for UNIX distribution, as they contain important information about upgrading from a prior release.

If you have a customized `$STARDIR/etc/swodbc.ini` file, make a backup copy of it before you update StarSQL, as the file will be replaced during the installation. You must use the new version of the `swodbc.ini` file to take advantage of the support for additional DB2 hosts. Copy any custom settings from the backup copy of the `swodbc.ini` file to the new version of the file after you update the StarSQL driver.

Typically users access the system-wide `swodbc.ini` file on a shared network drive. Having a writeable `.swodbc.ini` file in the home directory allows a user to modify the behavior of global data source settings (see "Global DSN Keywords" on page 76) and run the **trcstart** command to collect DRDA traces (see "trcstart" on page 73). If the user home directory contains an `.swodbc.ini` file, the settings in that file take precedence over any settings in the `swodbc.ini` file in `$STARDIR/etc`.

StarSQL for UNIX 32-bit Driver Installation

You can use the Easy Install method to install StarSQL on all UNIX platforms, or you can manually install the software. Manually installing the software lets you specify where the software is installed, but also requires that you specify environment variables and run a post-installation script. The following steps use the Easy Install method. Refer to "Manually Installing 32-bit StarSQL for UNIX" on page 63 if you prefer to customize the installation.

The StarSQL Easy Install is identical on all supported UNIX platforms. After you have downloaded StarSQL, follow the steps in this section to install the software.

1. Logon to UNIX as `root` user.
2. Change to the directory that contains the StarSQL installer for your version of UNIX.
3. Enter the following command to execute the shell script “setup,” which also runs the post-installation script to create the necessary symbolic links and install conversion tables.

```
# ./setup
```

The setup script installs StarSQL to the default location, which is `/usr/share/starsql` on Linux.

The installation script also either installs the unixODBC driver manager that is included in the StarSQL distribution and creates the `odbcinst.ini` file if it does not find an existing driver manager, or adds a [StarSQL] section to an existing `odbcinst.ini` file to define the driver to the active version of the driver manager.

After you perform the Easy Install procedures you need to configure a connection to a StarLicense server and configure StarSQL data sources for applications to use, as described in "Using StarSQL for UNIX" on page 27.

Removing the StarSQL 32-bit Driver

This section describes how to uninstall the StarSQL for UNIX software from a computer.

UNIX Variant	**Command for Un-installing Software**
Linux	using RPM: `rpm -e starsql` using tar: `rm -rf /usr/share/starsql`

CHAPTER 3

Installing 64-bit StarSQL for UNIX

This chapter describes how to install and configure StarSQL for UNIX on a 64-bit computer. It also discusses issues to consider if you want to install both the 32-bit and 64-bit versions of StarSQL on a 64-bit computer. Be sure to review the Release Notes included in the distribution for important information about installing or upgrading the StarSQL for UNIX driver.

StarSQL requires a license key to successfully connect to a DB2 host. You may want to obtain license keys, and install and configure the StarLicense software, before you install StarSQL. Licensing is further discussed in "Licensing StarQuest Products" on page 27.

Client Computer Requirements

StarSQL for UNIX has been tested with the following UNIX operating systems:

- Red Hat Enterprise or CentOS Linux 6.x and later (x64)
- SUSE Linux Enterprise Desktop or Server (x64)
- Ubuntu or Debian Linux (x64)
- Other distributions of Linux may also work with the StarSQL driver.

Determining Application Architecture

Because 32-bit and 64-bit applications can coexist on a 64-bit system, it can sometimes be difficult to determine whether an application is 64-bit or 32-bit. It's important to know the application architecture because 32-bit applications require a 32-bit DSN and ODBC driver manager and 64-bit applications require a 64-bit DSN and ODBC driver manager.

On UNIX systems, you can use the file command to tell whether an application is 32-bit or 64-bit. To do this, locate the application binary and then run **file <application_binary>**. If the file is 64-bit, the command output will contain "ELF 64-bit LSB executable". If the file is 32-bit, the command output will contain "ELF 32-bit LSB executable".

Upgrading StarSQL

Refer to "Installing 32-bit StarSQL for UNIX" on page 19 for upgrade considerations.

StarSQL for UNIX 64-bit Driver Installation

You can use the Easy Install method to install StarSQL, or you can manually install the driver software. Manually installing the software gives you more control over where the software is installed, although the procedures differ depending on which UNIX operating system you are using and are more involved than using the Easy Install. The following steps use the Easy Install method. Refer to "Manually Installing 64-bit StarSQL for Linux" on page 65 if you prefer to use the platform-specific installation instructions.

The StarSQL Easy Install is identical on all supported platforms. After you have downloaded StarSQL, follow the steps in this section to install the software.

1. Logon to UNIX as `root` user.
2. Change to the directory that contains the StarSQL installer for your version of UNIX.
3. Enter the following command to execute the shell script "setup," which also runs the post-installation script to create the necessary symbolic links and install conversion tables.

```
# ./setup
```

The setup script installs StarSQL to the default location, `/usr/share/starsql64 (Linux)`, and either creates the `odbcinst.ini` file, or adds a [StarSQL64] section to an existing `odbcinst.ini` file to define the driver to the active unixODBC driver manager.

Removing StarSQL from a 64-bit Computer (Linux)

If you need to remove the StarSQL for UNIX software you can use either the RPM Package Manager or the UNIX rm command, as shown in the following examples.

To remove StarSQL using the RPM Package Manager, issue the following command:

```
rpm -e starsql64
```

The following command forcibly removes the contents of the StarSQL installation directory and its contents without prompting for confirmation. Use this method if StarSQL was installed using the tar-based installer:

```
rm -rf /usr/share/starsql64
```

CHAPTER 4

Using StarSQL for UNIX

StarSQL requires a license to successfully connect to a DB2 host. This chapter describes how to license and use StarSQL for UNIX.

Licensing StarQuest Products

All StarQuest products are licensed for use. Each product setup contains a client module used to configure the specific license option used to enforce the use of the product. The licensing options allow you to use a node-locked license or a floating license.

Node-Locked License

A node-locked license allows you to use the Product on a single computer: Node-locked licenses are only available for computers using Microsoft Windows Operating Systems.

Floating License

A floating license allows multiple computers using a StarQuest product to share use of the software license. The software license can be used on any computer within a network provided that the number of concurrent requests does not exceed the limit allowed by the license. All StarQuest products for UNIX and Mac OS X ***must*** use a floating license. StarQuest products for Windows ***may*** use a floating license. Generally, only one license server need be installed on a network, to service any number of clients.

For a floating license, in addition to the setup of StarQuest product you will be provided with a StarLicense Server setup and a unique registration code to activate the license server. The *StarLicense Server for UNIX User's Guide* contains details about installing and configuring the server software, but in general:

- The StarLicense Server software should be installed on a network server.

- The network server is usually identified by a unique, static IP address.
- The StarLicense Server should be activated online or via e-mail.
- The StarLicense Server controls the total number of concurrent connections within the network.

For a client to obtain a license from a StarLicense server the parameters of the network server where the StarLicense Server is installed are specified in the appropriate client license module on any computer using the StarQuest product.

Configuring a StarSQL License

StarSQL for UNIX requires that a floating license be used, although the StarLicense Server software can be located on a remote host or co-located on the same computer as the StarSQL driver. To configure StarSQL for UNIX to check out a license from a StarLicense server, you will need to specify:

- the hostname or IP address of the StarLicense Server
- the port on which the StarLicense Server is listening for license requests
- the product ID of the license installed on the StarLicense Server

The hostname, port, and product ID specified for the client must match how that server is configured—contact the administrator responsible for configuring the StarLicense server(s) if you need details about how the server is configured.

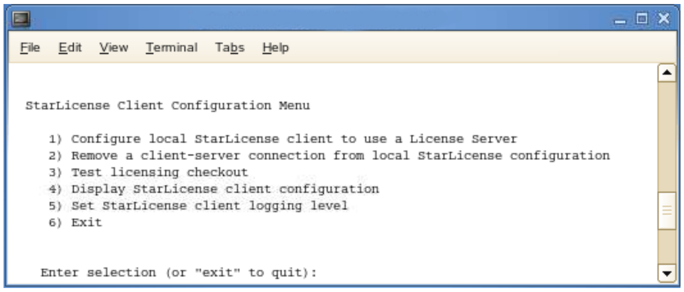
For UNIX-based computers the necessary StarLicense client software is installed when you install the StarSQL driver. Licenses configured with either the 32-bit or 64-bit version of StarLicense are available to both 32-bit and 64-bit applications that use StarSQL to access the host.

A StarLicense Client Configuration utility allows you to easily specify which StarLicense server to use when the UNIX client computer needs a connection.

1. Log into the computer as `root` user.
2. Change to the directory where StarSQL is installed.
3. Enter the following command to start the configuration utility.

```
# ./config-lic
```

Figure 2. StarLicense Client Configuration Menu



```
StarLicense Client Configuration Menu

   1) Configure local StarLicense client to use a License Server
   2) Remove a client-server connection from local StarLicense configuration
   3) Test licensing checkout
   4) Display StarLicense client configuration
   5) Set StarLicense client logging level
   6) Exit


  Enter selection (or "exit" to quit):
```

4. From the StarLicense Client Configuration Menu, select option 1, “Configure local StarLicense client to use a License Server.”
5. Enter the hostname or IP address of the StarLicense server. (If the StarLicense server software is installed on the local computer you can specify the loopback address 127.0.0.1 or “localhost”.)
6. Enter the number of the port on which the StarLicense server is listening for license requests.
7. Enter the product ID of the license that is configured on the StarLicense server.
8. After the connection to the StarLicense server has been added, select Option 3, “Test licensing checkout,” to verify that a license can be checked out.

Refer to "StarLicense Client Configuration Utility" on page 99 for information about all the options available from the StarLicense Client Configuration Menu.

Customizing the unixODBC Driver Manager Configuration

Most distributions of Linux include a version of the unixODBC driver manager. The StarSQL driver has been tested to work with the unixODBC driver manager that is included with StarSQL. You can use either the unixODBC binaries that are supplied with your operating system, or the unixODBC included with StarSQL, but do not use both.

Note

The version of unixODBC that is included with StarSQL meets the minimum requirements and has been tested to work with the StarSQL driver. The StarSQL driver requires the following versions of the unixODBC driver manager:

— Release 2.3.3pre

To determine which version of the unixODBC driver manager is in use, issue the ldd command to show the shared libraries that your ODBC application requires access to. The example output below shows that the isql program is using the driver manager (`libodbc.so.1`) from the $STARDIR directory. The unixODBC driver manager that is included with operating system distributions is typically located in `/usr/lib`. To control which version of unixODBC is used you can either set the LD_LIBRARY_PATH environment variable or edit `/etc/ld.so.conf` and run **ldconfig** to update the configuration.

```
$ ldd /usr/share/starsql/odbc/bin/isql
libodbc.so.1 => /usr/share/starsql/odbc/lib/libodbc.so.1
(0x40014000)
libpthread.so.0 => /lib/libpthread.so.0 (0x40081000)
libc.so.6 => /lib/libc.so.6 (0x400d2000)
libdl.so.2 => /lib/libdl.so.2 (0x401f9000)
/lib/ld-linux.so.2 +> /lib/ld-linus.so.2 (0x400000000).
```

The unixODBC driver manager uses the file `odbcinst.ini` to keep track of which ODBC drivers are installed. The location of this file can vary. You can locate the `odbcinst.ini` file using the odbcinst program provided with the unixODBC driver manager, as shown below.

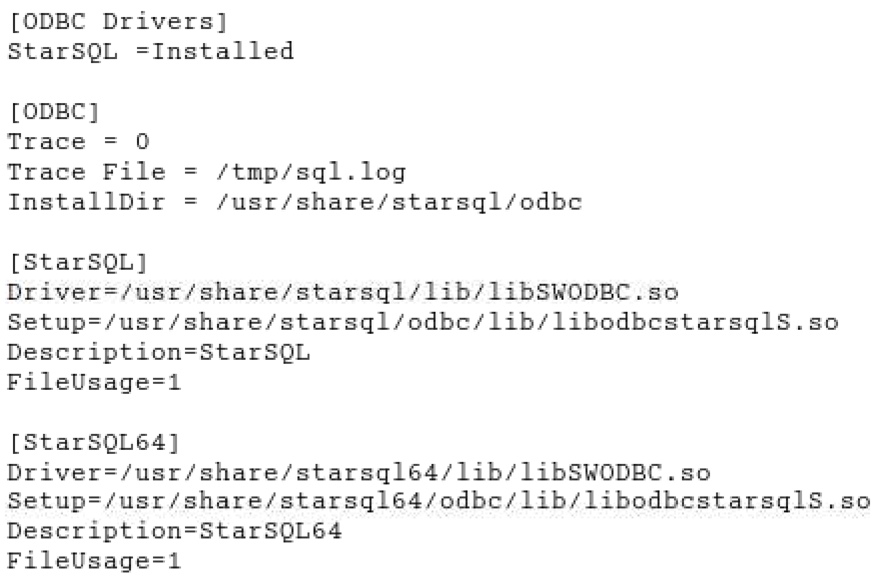
```
$ odbcinst -j
```

The odbcinst program shows the version of unixODBC that is installed, and the location of the files that specify the ODBC drivers (`odbcinst.ini`), the system DSNs (`odbc.ini`), and the user DSNs (`.odbc.ini`).

The unixODBC driver manager that is supplied with StarSQL looks for the global configuration files— `odbcinst.ini` and `odbc.ini`— in `/usr/local/etc`. If these files are located in a different directory you can define and export the `ODBCSYSINI` environment variable to specify a different location on the local computer. Note that both of the configuration files must be in the location specified by the ODBCSYSINI variable. Copy the configuration files from the /usr/local/etc directory as these versions of the files contain the DSN and StarSQL driver information.

The default unixODBC driver manager configuration works with most environments. If you need to modify the behavior, such as to turn on ODBC tracing to troubleshoot a problem, use a text editor to modify the `odbcinst.ini` file. The below listing shows sample contents of an `odbcinst.ini` file after installing both the 32-bit and 64-bit versions of the StarSQL driver.

Figure 3. Sample odbcinst.ini file with StarSQL Driver Defined



```
[ODBC Drivers]
StarSQL =Installed

[ODBC]
Trace = 0
Trace File = /tmp/sql.log
InstallDir = /usr/share/starsql/odbc

[StarSQL]
Driver=/usr/share/starsql/lib/libSWODBC.so
Setup=/usr/share/starsql/odbc/lib/libodbcstarsqlS.so
Description=StarSQL
FileUsage=1

[StarSQL64]
Driver=/usr/share/starsql64/lib/libSWODBC.so
Setup=/usr/share/starsql64/odbc/lib/libodbcstarsqlS.so
Description=StarSQL64
FileUsage=1
```

Refer to the unixODBC Readme file, included in the StarSQL distribution, for more information about using the driver manager.

Configuring Data Sources

A Data Source Name (DSN) definition controls how the StarSQL driver interacts with a specific data source. Each data source to which you want to connect using StarSQL must be defined in a text file named `odbc.ini` for system DSNs, and `.odbc.ini` for user DSNs. If there is a user DSN and system DSN with the same name, the user DSN is used.

Where possible, StarSQL supplies default configuration values in the example `odbc.ini` file that is distributed in the $STARDIR/etc directory. However, there are some values, such as specific names used in your environment, which you must configure.

The unixODBC driver manager includes a graphical ODBC Data Source Administrator program, ODBCConfig, to make it easy to define the minimum values necessary for a StarSQL DSN. Creating and modifying DSNs using the ODBCConfig interface modifies the `odbc.ini` or `.odbc.ini` file, and the `odbcinst.ini` file, as appropriate, with the values you specify using the graphical interface.

Both 32-bit and 64-bit data sources are visible from either version of the ODBC Data Source Administrator, but it is important to use the 32-bit version of the Administrator to create DSNs for 32-bit applications and the 64-bit version of the Administrator to create 64-bit DSNs. You cannot run both the 32-bit and 64-bit versions of the ODBCConfig program at the same time.

To run the ODBCConfig program you need to log in to the UNIX system from a GUI environment.

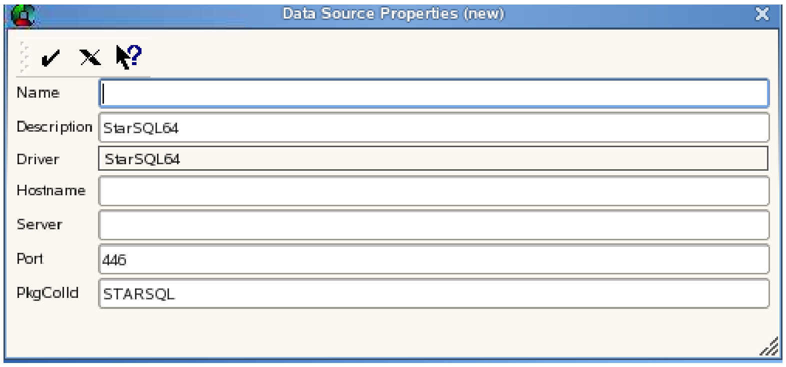
1. If you plan to create system DSNs, log in to the UNIX computer as `root` user. You do not need to be `root` user to create a user DSN.
2. Change to the directory in which the desired version—either the 32-bit or 64-bit—of ODBC Data Source Administrator is installed.
3. Type **ODBCConfig** and press Enter to start the ODBC Data Source Administrator.
4. From the ODBC Data Source Administrator, click on the User DSN or System DSN tab to configure the desired type of data source. The following table describes each type of DSN you can create. Note that, although there is a File tab in the Administrator window, ODBCConfig does not currently support creating a File DSN. You can still use a File DSN with StarSQL if the file is created manually or produced using the SAVEFILE connection attribute to SQLDriverConnect.

DSN Type	**Description**
User	A User DSN is local to the computer on which it is defined and can be used only by the current user.
System	A System DSN is local to the computer on which it is defined but can be used by the system or any user with privileges for the computer. Creating a System DSN requires sufficient authority to create and write the odbc.ini file in the system directory used by the ODBC Driver Manager (e.g. /usr/local/etc)
File	A File DSN contains information for a single data source that is stored in a file that can be shared by multiple users, or multiple computers if the file is stored on a network file server.

5. To modify an existing data source, select a Data Source Name and click Configure.
To add a new data source, click the Add button.

6. If you are creating a new data source, a pane entitled Select a Driver appears for you to select which driver you want to use. Select StarSQL or StarSQL64, as appropriate, and click the OK button to display the properties you can configure for the DSN.

7. Specify the connection and host properties that are appropriate for your environment. The following illustration shows example values for a StarSQL DSN.



Name	
Description	StarSQL64
Driver	StarSQL64
Hostname	
Server	
Port	446
PkgColId	STARSQL

Provide a unique name for the data source in the Name field. Do not specify a data source name that includes parentheses [( )]. The ODBC driver manager does not allow parentheses in data source names. If you configure a data source with parentheses in the name, the DSN will not be usable and can be removed only by manually editing the `odbc.ini` or `.odbc.ini` file. System DSN information is written to the `odbc.ini` file in the ODBC system directory (such as /usr/local/etc or $ODBCSYSINI), and user DSN information is written to the `.odbc.ini` file in the user's home directory.

You must specify a valid host name, and you can optionally specify a different port number or SQL package collection for the data source to use. The data source configuration parameters are further described in the following table.

Table 3. Minimum Data Source Configuration Parameters

Keyword	**Description**
HostName=	The host name refers to the name of the DB2 host system. Set the HostName field to either a TCP/IP host name (such as host5.mydomain.com) or a static IP address in dotted decimal notation (such as 198.147.235.1).
PkgColID=	*optional* The SQL Package Collection ID indicates the location on the DB2 host of the packages required by StarSQL to execute Dynamic SQL. On DB2 for i, the SQL Package Collection ID is the name of the collection or library that contains the packages. On all other platforms, the Package Collection ID is the name of the virtual collection associated with these packages. If necessary, obtain the package collection ID from your Database Administrator. The default package collection ID is STARSQL.
Port=	*optional* The Port designates the IP port on which the DB2 host is listening for incoming TCP/IP connection requests. The default port for DRDA communications is 446. For SSL communications, the typical port is 448.
Server=	Enter the name of the database server that will be accessed through this data source. You may need to obtain the database server name from the Database Administrator. The database server name is known by different names depending on the DB2 host implementation. On DB2 for z/OS it is called the location name, on DB2 for i it is called the relational database name (RDB), and on Db2 (DB2 LUW), it is the name of the database.
SSL=	*Optional* Enable SSL (encrypted) communications. Values **Yes** or **No (**Default**).** Also confirm that you are using the correct port for your DB2 host; port 448 is typical for SSL.

Hover over a field or click the Help icon and click within a field to display information about the data source property.

Click the Save icon ✓ to save the DSN definition.

Click the Cancel icon ✕ to exit the dialog without saving changes.

Refer to "Customizing the StarSQL for UNIX Configuration" on page 75 for information about configuring global data source settings and a complete list of the DSN settings that are available for customizing the behavior of the StarSQL driver.

Testing the StarSQL/DB2 Connection

After you install and configure StarSQL for UNIX, perform the procedures in this section to test that the client can connect to the DB2 host using StarSQL. To establish connectivity to a DB2 database, the host must be configured to accept connections from StarSQL. Refer to "Preparing Hosts for StarSQL Access" on page 39 if your host is not already configured for StarSQL connections.

Testing the Network Connection

You can use the **starping** command to test the network connection between StarSQL and the DB2 host. The **starping** command tests only the ability to connect to the DB2 host—it does not bind packages on the host or test that you have a valid license. Use the simpleconn command, described in the next section, to test for connectivity and proper configuration of StarLicense and the ODBC data source.

To run the starping command, enter the command and respond to the prompts for the data source name, user name, and password:

```
$ starping
Enter Data Source: data source name
Enter User Name: userid
Enter Password: password
```

A successful connection returns the following message:

```
Connection Succeeded!
```

See page 72 for more information about the starping command. If the connection does not succeed, make sure that the host has been properly configured for StarSQL connections, as described in "Preparing Hosts for StarSQL Access" on page 39. Contact Technical Support if you need further assistance.

Testing the Connection

StarSQL for UNIX includes a program named **simpleconn** that allows you to test that StarSQL can connect to a data source using the unixODBC driver manager. You must specify a user name and password of an account that is valid on the host and has permission to access the database. (See "Preparation Required for All Hosts" on page 39 for more information about user accounts and permissions.)

Enter the following command, specifying a Data Source Name (DSN) and user account that is valid in your environment:

`simpleconn` *`DSN UserId Password`*

For example:

```
$ simpleconn DSN1 staruser starpass
Connection #1
          Driver Name: libswodbc.a
          Driver Version: 5.30.1903
          Database Name: DB2 Universal DataBase
          Database Version: 08.021.0007
Connection #2
          Driver Name: libswodbc.a
          Driver Version: 5.30.1903
          Database Name: DB2 Universal DataBase
          Database Version: 08.021.0007
```

Upon successful connection, the **simpleconn** command returns the database name, platform, and version, and the driver name and version. Refer to page 68 for more information about using the **simpleconn** command.

Using the isql Utility to Test Connections

The version of unixODBC that is included with StarSQL includes the isql command line utility. You can use isql to test a connection but it is designed to be used by those experienced with the Structured Query Language (SQL). The isql utility allows you to:

1. connect to a Data Source (using a DSN)
2. send SQL commands to the Data Source
3. receive results from the Data Source

To determine which copy of isql is active, issue the following command.

```
$ which isql
/usr/share/starsql/odbc/bin/isql
```

The example output shows that the version of isql in the $STARDIR directory is active. The version of isql that is included with operating system distributions is typically located in /usr/bin. Set your PATH environment variable to determine which version of isql is used.

In addition to supplying ad-hoc query commands such as `SELECT * FROM MYTABLE`, the "help" command can be used to make catalog calls. Type "help help" to see a list of available commands.

A Unicode version of the utility "iusql" is also supplied.

Using StarSQL with ODBC Applications

After you have configured data sources and tested that the StarSQL driver can use the unixODBC driver manager and check out a license, you can use StarSQL with any ODBC application to connect to a DB2 host. The method for selecting which data source to use from a specific application varies with each application; refer to the documentation for your application if you need details about how to configure the application to use a particular data source.

Developing ODBC Applications for StarSQL

This section describes how to integrate an ODBC application to use the StarSQL driver and unixODBC driver manager.

When developing applications that will be used with the unixODBC driver manager supplied with StarSQL, use the header files in $STARDIR/odbc/include, and link with the libraries located in $STARDIR/odbc/lib. Include the library directive -lodbc to include `libodbc.so`.

Following is an example:

```
ODBCHOME=${STARSQL}/odbc;export ODBCHOME
cc program.c -I${ODBCHOME}/include \
-L${ODBCHOME}/lib -lodbc -lc -o $program
```

See the shell script "build" in $STARDIR/samples/simpconn for examples of building a simple ODBC application that uses StarSQL to connect to a host.

Supporting Two-Phase Commit

In a distributed processing environment where a unit of work spans more than one database, using the two-phase commit protocol can help ensure the integrity of the transaction and databases. When the two-phase commit protocol is in effect, the transaction must complete successfully before it is committed. If the transaction fails, all of the updates are rolled back and the databases remain in the state they were before the transaction was begun.

StarSQL for UNIX 6.40 and later support for the Open Group XA (eXtended Architecure). The interface for this support is based upon Microsoft's ODBC XA Implementation as described here:

https://docs.microsoft.com/en-us/sql/connect/odbc/use-xa-with-dtc?view=sql-server-ver15

Using XA Transactions

This implementation may be further "wrapped" in a struct xa_switch_t style interface in the future.

Contact StarQuest for an sample application that illustrates the use of the XA interface.

StarSQL configuration:

Add the property 2PC=XA to your ODBC datasource by editing ~/.odbc.ini or adding it to the connection string,

Values of 2PC are:

- No
- XA
- Note that the value TCP/IP - "classic" 2PC support for Microsoft DTC is avialable only for StarSQL for Windows

CHAPTER 5 Preparing Hosts for StarSQL Access

This chapter describes how to prepare host systems to enable StarSQL to provide access to the host databases.

It covers:

- Preparation required for all hosts
- Preparing a DB2 for z/OS host
- Preparing a DB2 for i host
- Preparing a DB2 for Linux, UNIX, and Windows host
- Preparing a Derby Network Server Host

These sections cover details of the DB2 environment that are pertinent to StarSQL. For complete documentation of installation and configuration on the host, consult IBM's DB2 documentation, especially IBM's *DRDA Connectivity Guide* (see "TContacting StarQuest" on page 16 for details).

Preparation Required for All Hosts

Regardless of the host platform, you will need the information described in Table 4 to configure an ODBC data source that will use StarSQL to access the DB2 host. You may need to obtain this information from the DB2 administrator. The section "Sample DSN File" on page 79 contains details about configuring data sources.

Table 4. Host Information Required for Data Source Configuration

Data Source Information Item	Information Needed
Package Collection Name	The location of the SQL packages that StarSQL requires. The Package Collection Name on a DB2 for i host is the name of the library that contains the StarSQL packages. For DB2 for z/OS, it is the name of the virtual collection associated with these packages.
Database Server Name	The relational database name. On different DB2 hosts this may be referred to as Location Name, Global Resource Name, RDB name, Database Name, or dbname.
User ID and Password	A valid user id and password for logging into the database. The user account information is not stored in the DSN.
TCP/IP Connection Information	The host name and the port used for DRDA communications.

In addition to properly preparing the host, each StarSQL user must have an account on the host and have permission to access the necessary packages.

User Accounts

To connect to a DB2 database, each StarSQL user needs an account on the host database. An account consists of a user ID and password.

You need to provide the user account information to each StarSQL user who needs to connect to the database.

Permissions

Usually a database administrator (DBA) is responsible for packages on the host, including binding them and granting permissions to use them. Depending on the host platform and the type of package used by the ODBC application, the DBA may need to grant StarSQL users explicit permissions to access data used by the application.

Preparing DB2 on an z/OS Host

Preparing DB2 on a z/OS host for access with StarSQL primarily involves configuring the Distributed Data Facility (DDF), which is a component of DB2 for z/OS. Its primary task is to process DRDA requests. DDF must be active for a desktop to connect to DB2 using StarSQL or any other DRDA requestor or client.

Configuring DDF

If your organization has not implemented distributed database capabilities, DDF may not be configured and activated. The DSNTINST CLIST provides two panels—DSNTIPR and DSNTIP5 for customizing a DB2 for z/OS subsystem to use native DRDA TCP/IP support. The DSNTIP5 panel is specific for TCP/IP.

The values specified on these panels are used to generate the JCL that stores them in the DB2 bootstrap data set (BSDS) communication record.

If you are installing DB2, use the DDF panel DSNTIPR and DSNTIP5 to provide the following parameters. To change the DDF parameters after installation, run a customized configuration job DSNTIJUZ to update the BSDS.

- DDF Location Name. This name must be specified for the Database Server Name of the ODBC data source that StarSQL uses to connect to the host.
- Password used when connecting DB2 to VTAM, if a password is required.
- IP port to use for TCP/IP access. To enable support for TCP/IP, set the DRDA port in the DDF to 446.
- IP port to use for two-phase commit. The RESYNC PORT parameter in the DSNTIP5 panel specifies a TCP/IP port number for processing requests for two-phase commit re synchronization. The RESYNC PORT must be different than the DRDA PORT, and it must match the port number specified for two-phase-commit recovery operations in the StarSQL Resource Manager. The StarSQL Resource Manager uses a default port value of 5020.

For more information about establishing connectivity between your desktop and DB2 for z/OS with TCP/IP, consult the Installation Guide for your version of DB2 for z/OS (see "Documentation" on page 13 for document details).

For more information about configuring DDF, consult IBM's DB2 for z/OS installation documentation and the IBM Redbook, *Distributed Functions of DB2 for z/OS and OS/390*. For more information about establishing connectivity between client computers and DB2 for z/OS over a TCP/IP network, the IBM Redbook, *WOW! DRDA Supports TCP/IP: DB2 Server for OS/390 and DB2 Universal Database*, may be particularly useful.

Starting DDF

Use the following command, which requires authority of SYSOPR or higher, to start DDF:

```
-START DDF
```

When DDF starts successfully, the following messages are displayed:

```
DSNL003I - DDF IS STARTING
DSNL004I - DDF START COMPLETE LOCATION locname LU
netname.luname
```

If DDF has not been properly installed, the START DDF command fails and displays the following message:

```
DSN9032I - REQUESTED FUNCTION IS NOT AVAILABLE
```

If DDF has already been started, the START DDF command fails and displays the following message:

```
DSNL001I - DDF IS ALREADY STARTED
```

The following command shows whether DDF is running and, if so, the parameters that it is using:

```
-DIS DDF
```

Supporting Password Management Using DRDA Flows

Password Management using DRDA flows is supported for a network using the TCP/IP protocol. Users can change their host passwords through StarSQL if DB2 is configured as follows:

Set Extended Security to YES (EXTSEC=YES). .This can be done using either

- the DSNTIPR (DDF) panel on the DB2 installation dialog.
- a customized configuration job DSNTIJUZ, with the option EXTSEC=YES specified.

Using StarSQL with Stored Procedures

Stored procedures are application programs that reside on the host and are invoked via DB2. They are usually written in a traditional programming language like COBOL, RPG, or C. They may contain SQL statements for accessing the DB2 database or they may be used to access non-DB2 resources.

You invoke a stored procedure using the SQL Call statement and receive output data in a result set or in output parameters. The Call statement is executed as any other SQL, using SQLExecute or SQLExecDirect.

Registering Stored Procedures

The stored procedure should be registered on the host so that calling application can obtain information using the SQLProcedures and SQLProcedureColumns functions.

Use the CREATE PROCEDURE command to register the stored procedure in the system. The CREATE PROCEDURE command automatically updates the SYSIBM.SYSROUTINES catalog table.

```
CREATE PROCEDURE SYSPROC.STARPING (
IN REGION CHAR(8) CCSID EBCDIC,
IN PROGRAM CHAR(8) CCSID EBCDIC,
IN TRANSID CHAR(4) CCSID EBCDIC,
IN COMMLEN SMALLINT,
INOUT COMMAREA VARCHAR(32700) FOR BIT DATA,
OUT RC INTEGER,
OUT ABCODE CHAR(4) CCSID EBCDIC
)
PARAMETER STYLE GENERAL
LANGUAGE C
EXTERNAL NAME 'STARPING'
RESULT SETS 0
DETERMINISTIC
NO SQL
NO DBINFO
NO COLLID
ASUTIME NO LIMIT
NO WLM ENVIRONMENT
STAY RESIDENT NO
PROGRAM TYPE MAIN
SECURITY DB2
COMMIT ON RETURN YES
```

Calling Stored Procedures

Each of the following SQL statements for calling the sample stored procedure is valid:

```
Call MyProc (1, 'A', ?, ?)
Call MyProc( parm1=1, parm2='A', parm3=?, parm4=?)
Call MyProc( parm1=1, parm2='A', ?, ?)
Call MyProc( ?, ?, ?, ?)
Call MyProc( 1, 'A', parm3=?, parm4=?)
```

Configuring SSL for DB2 for z/OS

To configure a DB2 for z/OS host system for SSL communications, you must be using DB2 for z/OS 9.1 or later. Refer to the section *Encrypting your data with Secure Socket Layer support* in the DB2 for z/OS documentation.

You must configure AT-TLS (z/OS® Communications Server IP Application Transparent Transport Layer Security) as well as configuring the DB2 server.

The listening port number (typically 448) is specified in the DRDA SECURE PORT field of the Distributed Data Facility Panel 2 (DSNTIP5) during DB2 installation. After initial installation, you can update the SECPORT parameter of the DDF statement in the BSDS with the change log inventory (DSNJU003) stand-alone utility.

The following IBM Redbooks may also be of assistance:

- *Securing and Auditing Data on DB2 for z/OS* (SG24-7720)

  DB2 9 for z/OS: Distributed Functions (SG24-6952-01)

Preparing a DB2 for i Host

This section discusses setting up a DB2 for i host for supporting a connection through StarSQL

It covers:

- creating a library/collection for SQL packages
- determining the RDB name
- enabling DRDA over TCP/IP for IBM i
- supporting password encryption, LOB data types, and two-phase commit transactions

For complete information about setting up DB2 for i, consult IBM’s DB2 for i installation documentation.

Creating a Library for SQL Packages

On DB2 for i, required SQL packages are stored in a collection, or library. You may need to create the collection or library on the host.

Use the CRTLIB command to create a new library for the SQL packages used by StarSQL. You can do this from a 5250 terminal session with a user ID that has QSECOFR privileges. The library does not have to be a SQL collection, but it must be accessible to all StarSQL users.

For example, the following command creates a new library named STARSQL:

```
CRTLIB STARSQL
```

Record the name of the library as it must be specified as the SQL Package Collection ID in the data source configuration.

Determining the RDB Name

Determine the Relational Database (RDB) name of the DB2 for i host. From the DB2 for i command line, enter:

```
WRKRDBDIRE
```

Look for an entry with a Remote Location value of *LOCAL. If such an entry does not exist, create it with the 1=ADD option. A common convention is to use the same name as the DB2 for i system name for the RDB name.

Make a note of the RDB name as you need to specify it as the Server in the data source configuration.

Enabling DRDA Over TCP/IP

The Distributed Data Management (DDM) server allows client computers to access the DB2 functions. The DDM server supports remote SQL access, record level access, and remote journals. To initiate a DDM server job using TCP/IP communications a DRDA application or DDM source system connects to the well-known port number for TCP/IP, port 446 or 447. The DDM listener program, upon accepting the connection request, issues an internal request to attach the client's connection to a DDM server job.

The DDM listener program runs in a batch job in the QSYSWRK subsystem. There is one listener program that serves potentially many DDM server jobs. If you have access to iSeries Navigator you can verify whether DDM is configured by selecting TCP/IP from the Network–>Servers menu.

Follow the steps below if you need to configure an IBM i host to accept DRDA requests over TCP/IP:

1. Log on to the IBM i host.
2. Change the DDM TCP/IP Attributes to automatically start the listener program by entering:
   ```
   CHGDDMTCPA AUTOSTART(*YES)
   ```
3. Start the TCPIP DDM Server by entering:
   ```
   STRTCPSVR SERVER(*DDM)
   ```

When you are logged on to the DB2 for i host, you can examine which port DB2 for i is using to listen for DRDA requests using either the WRKSRVTBLE or the WRKTCPSTS command.

To use the WRKSRVTBLE command:

1. Enter WRKSRVTBLE.
2. Look for the DRDA entry with the port number.

To use the WRKTCPSTS command:

1. Enter WRKTCPSTS.
2. Choose option 3, "Work with TCP/IP connection status."
3. Find the entry with port "drda" and press "F14=Display port numbers." The default port number for DRDA is 446.

Configuring SSL on DB2 for i

To configure a System i host system to use the Secure Sockets Layer (SSL) protocol you must have the following components:

- Digital Certificate Manager - 5722-SS1 (v5rx), 5761-SS1 (v6r1), or 5770-SS1(v7r1) option 34
- TCP/IP Connectivity Utilities - 5722-TC1(v5r4), 5761-TC1 (v6r1), or 5770-SS1 (v7r1)
- IBM HTTP Server - 5722-DG1 (v5rx), 5761-DG1 (v6r1) or 5770-DG1 (v7r1)

Following are general procedures for configuring SSL on the iSeries host. Refer to your IBM documentation for details, especially the AS/400 documentation and the IBM Redbook *IBM iSeries Wired Network Security OS/400 V5R1 DCM and Cryptography Enhancements* (GSG24-6168).

1. Start the Admin HTTP instance and use a browser to configure the Digital Certificate Manager.
2. Create a local Certificate Authority or obtain a certificate from a public Internet Certificate Authority.
3. Create a *SYSTEM certificate store.
4. Use “Manage Applications” to assign a server certificate to the OS/400 DDM/DRDA server.
5. After you assign the certificate, restart the DDM/DRDA server:

   ```
   ENDTCPSVR *DDM
   STRTCPSVR *DDM
   ```
6. If necessary, set the port on which the DDM/DRDA server listens for SSL conversations. Use WRKSRVTBLE to view and modify service table entries; the entry for SSL is ddm-ssl, and the default value is 448.

Registering Stored Procedures on IBM i

This section describes issues regarding stored procedures that are specific to IBM i hosts. Refer to "Using StarSQL with Stored Procedures" on page 43 for general information about using stored procedures.

Following is sample SQL for registering a stored procedure on an AS/400 system. It assumes that the COBOL program MYLIB.MYPRGM already exists on the AS/400. This statement modifies the QSYS2.SYSPROCS and QSYS2.SYSPARMS catalog tables for you.

```
CREATE PROCEDURE MYLIB.MYPROC (INOUT PARM1 CHAR(10))
EXTERNAL NAME MYLIB.MYPGM LANGUAGE COBOL GENERAL
```

In the above example, the procedure name is MYLIB.MYPROC, which references the COBOL program MYLIB.MYPGM. The program takes one input parameter called PARM1 which is a char field of length 10. This procedure does not return a result set.

Refer to the *IBM SQL Reference and SQL Programming Guide* for the appropriate version of AS/400 for more information on registering a stored procedure and the full syntax of the CREATE PROCEDURE statement.

Considerations for Specific IBM i Releases

In general, it is a good idea to stay current on PTF packages and the DB2 Group PTF, and to use the latest version of StarSQL. Refer to the StarSQL Release Notes (readunix.html) for specific issues for particular versions of IBM i.

Preparing a DB2 for LUW Host

This section provides details for setting up a DB2 for Linux, UNIX, and Windows host to support a connection through StarSQL for Windows. It covers:

- Enabling DRDA Support for TCP/IP
- Locating the Database Name

For complete information about setting up DB2 LUW, consult IBM's DB2 LUW installation documentation.

Enabling DRDA Support for TCP/IP

When using TCP/IP to connect to a DB2 LUW host, make sure that the host has a static IP address. You may experience problems installing DB2 LUW on a computer using DHCP. To be recognized, DB2 LUW requires an entry in the DNS (Domain Name Server) or an entry in the HOSTS file.

Although you can configure StarSQL to communicate with DB2 using any available port, port 446 is the standard port used for DRDA communications and is the default value that StarSQL uses if another is not specified. DB2 LUW uses a default value of 50000 for DRDA communications. You can either change the DRDA listening port of the DB2 server or (more commonly) use port 50000 when configuring StarSQL.

If there is a firewall that monitors network traffic to the DB2 host, be sure that it allows DRDA communications to pass through the port that you configure for DRDA requests.

Using db2 Commands to Specify the DRDA Port

Issue the following db2 commands to change the port that DB2 uses for DRDA communications.

1. Enter the following command to determine on which port DB2 is listening to for TCP/IP communications.

   ```
   db2 get dbm configuration
   ```

   In the dbm configuration, look for "TPC/IP Service Name (SVCENAME)." It will have a value similar to "db2c_DB2." This is the symbolic name of the connection port.

2. Find the symbolic name of the connection ports in the services file, which is typically located in /etc/services.

3. Edit the services field and change the value of the TCP/IP connection port to 446 and set the interrupt port to 447.

4. Restart the DB2 instance for the changes to take effect.

To verify that DB2 is listening on the correct port, from either the client or the server enter:

```
telnet <host> 446
```

If DB2 is listening on that port, no error is returned to the telnet window. If DB2 is not listening on that port, you will see an error similar to the following and may need to contact your DB2 administrator for the correct port number to use:

```
Could not open connection to the host, on port 446: Connect
failed.
```

Click the Close (X) icon to close the telnet window.

Configuring SSL for DB2 UDB 9.7 & later:

Refer to the *Configuring Secure Sockets Layer (SSL) support in a DB2 instance* chapter in the DB2 documentation for details.

The following is an example of using self-signed certificates.

Use GSKit to create a keystore file (certificate database) and a certificate. Export the certificate if desired:

```
C> cd C:\Program Files\ibm\gsk8\bin\gsk8capicmd
```

```
C> gsk8capicmd -keydb -create -db "mydbserver.kdb" -pw
"mypassword" -stash
```

```
C> gsk8capicmd -cert -create -db "mydbserver.kdb" -pw "
mypassword " -label "SelfSigned" -dn
"CN=myhost.mydomain.com,O=MyCompany,OU=CustomerSupport,L=C
alifornia,ST=ON,C=CA"
```

```
C> gsk8capicmd -cert -extract -db "mydbserver.kdb" -pw
"mypassword" -label "SelfSigned" -target
"MYHOSTserver.arm" -format ascii -fips
```

On UNIX, make sure that the DB2 instance owner has read access to the keystore file.

Update DB2 and restart it:

```
C> db2 update dbm cfg using SSL_SVR_KEYDB "C:\Program
Files\ibm\gsk8\bin\mydbserver.kdb"
```

```
C> db2 update dbm cfg using SSL_SVR_LABEL SelfSigned
```

```
C> db2 update dbm cfg using SSL_SVR_STASH "C:\Program
Files\ibm\gsk8\bin\mydbserver.sth"
```

```
C> db2 update dbm cfg using SSL_SVCENAME 50029
```

```
C> db2set -i db2 DB2COMM=SSL,TCPIP
```

```
C> db2stop
```

```
C> db2start
```

Note that there is a problem in DB2 UDB 9.7 fp3 (resolved in fp4); see *IC72728: THE PORT NUMBER FOR SSL_SVCENAME IN THE SERVICE FILE IS USED FOR SVCENAME*. The value specified for SSL_SVCENAME is being used for both SSL and non-SSL listeners, causing a conflict. The workaround is to set the SVCENAME parameter using an explicit port number:

```
C> db2 update dbm cfg using SVCENAME 50000
```

Configuring SSL for DB2 UDB 9.1 & 9.5:

The procedure is similar to the above instructions for DB2 UDB 9.7, except:

- The gskit is v7 rather than v8 (e.g. run C:\Program Files\ibm\gsk7\bin\gsk7capicmd instead of C:\Program Files\ibm\gsk8\bin\gsk8capicmd)
- Rather than using DBM configuration, the DB2 SSL parameters are stored in a configuration file sslconfig.ini located in the following directory:

  •Linux and UNIX: INSTHOME/cfg/SSLconfig.ini

  •Windows: INSTHOME/SSLconfig.ini

  where INSTHOME is the home directory of the instance.

Here is a sample sslconfig.ini:

```
DB2_SSL_KEYSTORE_FILE=C:\Program
Files\ibm\gsk7\bin\mydbserver.kdb

DB2_SSL_LISTENER=50448

DB2_SSL_KEYSTORE_PW=mypassword

DB2_SSL_KEYSTORE_LABEL=SelfSigned
```

After configuring gsk7 and creating sslconfig.ini, enter **`db2set -i db2 DB2COMM=SSL,TCPIP`** and restart DB2

Enabling Encryption

If you configure the StarSQL driver to send encrypted user IDs and passwords (see "UseEncryption" on page 95), be sure to enable the database for encryption. On a DB2 for Linux, UNIX, and Windows host you enable encryption by setting the Server Connection Authentication (SRVCON_AUTH) parameter to “Server encrypt” and restarting the instance.

Using db2 Commands to Enable Encryption

Issue the following db2 commands to enable encryption.

1. To enable encryption from a db2 command window, enter the following command:

   ```
   db2 update dbm cfg using SRVCON_AUTH
   SERVER_ENCRYPT
   ```

2. Stop and restart the instance by issuing the following commands:

```
db2stop
db2start
```

Locating the Database Name

When an administrator sets up a DB2 LUW system, they assign names to DB2 databases. You will need to enter the Database Server Name when you configure data sources on the desktop. You can display a list of the databases that have been created by issuing the following command:

```
db2 list database directory
```

Contact your database administrator if you need to determine which databases should be accessible to StarSQL clients.

Preparing a Derby Network Server Host

The Derby Network Server is part of the Derby software distribution and provides a framework for multi-user connectivity to Derby databases across a network. Derby must be started in the Network Server mode (vs. the embedded mode) for client computers to connect to a Derby database using StarSQL for Java.

The default of the Derby Network Server is to start with user authentication disabled. Refer to the Derby documentation for information about enabling user authentication and running under the Java security manager to help avoid security problems.

Setting Network Server Properties

Typically the Java system property `derby.system.home` specifies the system directory, which contains subdirectories that hold the databases that are available to the Derby Network Server. If you do not explicitly set the `derby.system.home` property when starting Derby, the default is the directory in which Derby was started. You need to specify at least the name of the database you want to access with the StarSQL for Java driver and which port to use. Contact your database administrator if you do not know which Derby databases should be accessible to StarSQL for Java clients.

The Network Server properties can be specified three ways:

- on the command line
- in the `.bat` or `.ksh` files, loading the properties by executing **java -D**
- in the `derby.properties` file

Properties in the command line or in the .bat or .ksh files take precedence over the properties in the derby.properties file. Arguments included on commands that are issued on the command line take precedence over property values.

The properties that are particularly of interest for using the StarSQL for Java driver are those that allow remote connections and determine which protocol to use.

```
derby.drda.host=hostname
derby.drda.portNumber=portnumber
derby.drda.sslMode=SSL | TLS || SSLv3 | TLSv1
```

Enabling Remote Connections

The default of the Derby Network Server is to start with user authentication disabled. Before you enable remote connections ensure that you are running under the security manager and that user authorization is enabled. Refer to the Derby documentation for details about enabling user authentication and running under the Java security manager.

The `derby.drda.host` property causes the Network Server to listen on a specific interface, allowing multiple instances of Network Server to run on a single computer with a unique host:port combination. By default the Derby Network Server listens only on the loopback IP address (127.0.0.1) and port 1527, which restricts access to the local computer. To make the Derby Network Server accessible to other computers on the network, specify a particular interface (host name or IP address) and specify a port number other than 1527 on which to listen for connections. The Derby Network Server must listen for connections on the same port that the StarSQL for Java driver is configured to use, which is port 446 by default.

The following shows how to use a java command to start the Derby Network Server to listen on port 446 for connection requests from any host name or IP address.

```
java -jar $DERBY_HOME%\lib\derbyrun.jar server start
          -h 0.0.0.0 -p 446
```

The -h and -p options can be specified regardless of how you choose to start Derby Network Server. Refer to the Derby documentation for details about the additional methods, such as running the startNetworkServer script or using a java command to directly invoke the NetworkServerControl class.

Configuring Support for the SSL Protocol

The Derby Network Server supports network security with Secure Socket Layer/Transport Layer Security (SSL/TLS). With SSL/TLS the client-server communication protocol is encrypted and both the client and the server may, independently of each other, require certificate-based authentication of the other part. You can configure SSL for the Derby Network Server to be Off, or to use SSL encryption with no peer authentication (basic), or to use SSL encryption and peer authentication (peerAuthentication). To enable SSL support, use the `ssl` option on the command line when you start Derby or specify the `derby.drda.sslMode` property in `derby.properties`.

For example, the following command would start Derby Network Server using the "basic" SSL encryption option:

```
java -jar derbyrun.jar server start -ssl basic
```

You also need to use the **keytool** to create a server key pair, and specify the key store and password when starting the server. For SSL operation the server always needs a key pair. If the server runs in peer authentication mode, then each client needs its own key pair. The **keytool** is located in the JRE bin directory. Entering the following command prompts for the information needed to generate a key pair.

```
keytool -genkey myDatabase -keystore serverKeyStore.key
```

The key pair is located in a file which is called a key store and the JDK's SSL provider needs the system properties `javax.net.ssl.keyStore` and `javax.net.ssl.keyStorePassword` to access the key store. Specify the key store and password when starting the server, such as shown below:

```
java -Djavax.net.ssl.keyStore=serverKeyStore.key \
     -Djavax.net.ssl.keyStorePassword=myPassword \
     -jar %DERBY_HOME%\lib\derbyrun.jar server start
                          -h 0.0.0.0 -p 446 -ssl basic
```

Refer to the Derby documentation at http://db.apache.org/derby/docs/10.4/adminguide/cadminsslkeys.html for additional details about key and certificate handling.

CHAPTER 6

Binding Packages

A SQL package is an object that DB2 uses to process a SQL statement. Different packages are required to execute dynamic SQL and ODBC catalog functions.

StarSQL assumes that the dynamic SQL package and the catalog package already exist in the host database. On the initial connection with the host, StarSQL searches for the required packages, and if it does not find them, it creates them automatically.

The names and locations of the packages that are bound vary, depending on how the data source is configured when the initial connection is made.

The StarSQL user requires certain permissions to bind and use packages on the host. Typically, the package will be created by a database administrator and other users will be granted permission to use the packages.

You can use the StarAdmin utility, which is available from StarQuest, to bind packages on a host. Packages that are bound using StarAdmin can be used by any 32-bit or 64-bit version (Java, Linux, UNIX, or Windows) of the StarSQL driver.

There is no need to bind packages when using an Apache Derby database.

Compatibility Among SQL Catalog Packages

Major releases of StarSQL also may require changes to SQL catalog packages on the host. StarSQL uses the SQL packages to provide added functionality. (Point releases and hot-fixes usually do not involve changes to the SQL packages.) It is recommended that the SQL packages be rebound when a major version of StarSQL is deployed throughout an organization, or when directed by the instructions included in the Release Notes included with all StarSQL installation images.

Significant changes were made to catalog packages and dynamic SQL packages with the v5.1 and v5.3 releases of StarSQL. Users may get different results from ODBC catalog functions when using a version of StarSQL prior to v5.1 with packages bound that are bound with StarSQL v5.1 or later.

Table 5 shows the level of functionality supported with SQL packages created by different versions of StarSQL. In general, packages bound by later versions of StarSQL are backward compatible with earlier versions of StarSQL, unless specifically noted. If an earlier version of StarSQL cannot function properly with packages that are bound with a later release of StarSQL, you can maintain separate package collections or upgrade all clients to use the latest version of the driver.

Table 5. Functional Package Differences Among StarSQL Versions

From	To	New Functionality
v4	v5.1	Packages must be rebound to use LOB data types. Upgrade all clients or maintain a separate package collection for clients running a StarSQL release prior to v5.1
v5.1	v5.3	Backwards compatible. Rebind packages to support use of jumbo packages, or to access DB2 for z/OS when the host is configured to use a multi-byte character set. Refer to the descriptions for UseJumboPackages and CustomizePrdid in "Sample DSN File" on page 79 for additional information.
v5.3	v5.5	Rebind packages if you upgrade from a release earlier than v5.38. Client computers that use StarSQL v5.1 or earlier should be upgraded to v5.5 or use a different package collection.

StarSQL for UNIX automatically binds host packages upon the initial connection to the host. To explicitly rebind packages, drop the packages on the host and reconnect with the new version of StarSQL for UNIX.

Binding Packages with the Explain Option Enabled

DB2 provides an explain facility that provides detailed information about the access plan and environment of static or dynamic SQL statements. Beginning with v5.4 the StarSQL driver includes support for a new keyword in the `swodbc.ini` initialization file that binds all the StarSQL packages with the EXPLAIN bind option set to ALL.

Refer to "Explain" on page 77 if you want to enable the EXPLAIN bind option for all packages that are bound by StarSQL.

Catalog Package

The catalog package is needed to execute ODBC functions that query the system catalogs. The value of the CatQual keyword (see page 85) in the data source configuration indicates the name of the catalog package.

Dynamic SQL Packages

There are several packages used for executing dynamic SQL. The default dynamic package varies by host platform; other dynamic packages will be used based on the IsolationLevel, HeldCursors, KeepDynamic, and BindRules keywords in the data source configuration. See "Customizing the StarSQL for UNIX Configuration" on page 75 for more information about these keywords.

Table 6 shows the dynamic packages that can be bound; the default packages vary for different hosts and are shown in **boldface** type.

Table 6. Dynamic Packages

Package Name	IsolationLevel and HeldCursors Setting
SWNC0000	No Commit/Held Cursors=No (default on **DB2 for i)**
SWRC0000	Read Committed / HeldCursors=No (default on **DB2 LUW**)
SWRR0000	Repeatable Read / HeldCursors=No
SWRU0000	Read Uncommitted / HeldCursors=No
SWTS0000	Serializable / HeldCursors=No
SWRC1000	Read Committed / HeldCursors=Yes (default on **DB2 for z/OS**)
SWRR1000	Repeatable Read / HeldCursors=Yes
SWRU1000	Read Uncommitted / HeldCursors=Yes
SWTS1000	Serializable / HeldCursors=Yes

Table 7 through Table 9 show the packages that can be bound when using the KeepDynamic option, or if the BindRules keyword is set to BIND. These tables are only applicable for hosts running DB2 for OS/390 v5.1 and later.

Table 7. Dynamic Packages on DB2 for z/OS with KeepDynamic=Yes

Package Name	IsolationLevel and HeldCursors Setting
SWNC4000	No Commit/Held Cursors=No
SWRC4000	Read Committed / HeldCursors=No
SWRR4000	Repeatable Read / HeldCursors=No
SWRU4000	Read Uncommitted / HeldCursors=No
SWTS4000	Serializable / HeldCursors=No
SWRC5000	Read Committed / HeldCursors=Yes (default on **DB2 for z/OS**)
SWRR5000	Repeatable Read / HeldCursors=Yes
SWRU5000	Read Uncommitted / HeldCursors=Yes
SWTS5000	Serializable / HeldCursors=Yes

Table 8. Dynamic Packages on DB2 for z/OS with BindRules=BIND and KeepDynamic=No

Package Name	IsolationLevel and HeldCursors Setting
SWRC2000	Read Committed / HeldCursors=No
SWRR2000	Repeatable Read / HeldCursors=No
SWRU2000	Read Uncommitted / HeldCursors=No
SWTS2000	Serializable / HeldCursors=No

Package Name	IsolationLevel and HeldCursors Setting
SWRC3000	Read Committed / HeldCursors=Yes
SWRR3000	Repeatable Read / HeldCursors=Yes
SWRU3000	Read Uncommitted / HeldCursors=Yes
SWTS3000	Serializable / HeldCursors=Yes

Table 9. Dynamic Packages on DB2 for z/OS with BindRules=BIND and KeepDynamic=Yes

Package Name	IsolationLevel and HeldCursors Setting
SWRC6000	Read Committed / HeldCursors=No
SWRR6000	Repeatable Read / HeldCursors=No
SWRU6000	Read Uncommitted / HeldCursors=No
SWTS6000	Serializable / HeldCursors=No
SWRC7000	Read Committed / HeldCursors=Yes
SWRR7000	Repeatable Read / HeldCursors=Yes
SWRU7000	Read Uncommitted / HeldCursors=Yes
SWTS7000	Serializable / HeldCursors=Yes

Permissions for Packages

After the packages have been bound, StarSQL users must be granted permission to execute them. On DB2 for i, users must be granted *USE permission on the packages, which can usually be done by the package owner. On other hosts, the DBA must grant StarSQL users EXECUTE or RUN permissions on the packages.

Static and Catalog Packages

The StarSQL user executes applications that use the catalog package and any static SQL packages under the permissions of the package owner. No additional permissions are required to use these packages.

Dynamic SQL Packages

On all hosts, except for DB2 for z/OS, for applications that use dynamic SQL, the StarSQL user needs explicit permissions to read (SELECT) and write (UPDATE/INSERT/DELETE) the columns and tables accessed by the application. The DBA needs to grant these permissions to "public" or to the various groups.

On DB2 for z/OS, the required permissions depend on the setting of the BindRules keyword. If the BindRules keyword is set to RUN, which is the default, the StarSQL user needs explicit permissions to read and write the columns and tables accessed by the application. If the BindRules option is set to BIND, the user has the permissions of the package owner, except that the following SQL statements cannot be executed regardless of the permissions of the package owner:

- SET CURRENT SQLID
- GRANT
- REVOKE
- ALTER
- CREATE
- DROP
- Any SQL statement that cannot be prepared as a dynamic SQL statement.

Granting Use Permissions

To grant EXECUTE authority on the SQL packages, you can use the StarAdmin application if you have a Windows-based StarSQL computer, or you can execute SQL statements similar to those shown in the following sections.

For DB2 for z/OS

Using SPUFI on the host or the SQL pass through mode of an ODBC-enabled application, execute the following SQL statements:

```
GRANT EXECUTE ON PACKAGE
STARSQL.SYSIBM, STARSQL.SWRC5000
TO PUBLIC
```

For DB2 LUW (Linux, Unix & Windows)

Using the DB2 Command Line Processor or the SQL pass through mode of an ODBC-enabled application, execute the following SQL statements:

```
GRANT EXECUTE ON PACKAGE
STARSQL.SYSCAT, STARSQL.SWRC0000
TO PUBLIC
```

For DB2 for i

Using STRSQL (which is the interactive SQL interpreter, available in the Query Manager and SQL Development Kit for AS/400, Licensed Program 5xxx-ST1), or using the SQL pass through mode of an ODBC-enabled application, execute the following SQL statements:

```
GRANT EXECUTE ON PACKAGE
STARSQL.QSYS2, STARSQL.SWNC0000
TO PUBLIC
```

In addition, you can use AS/400 authority commands to grant *USE authority for all users (*PUBLIC) to the library packages:

1. From a 5250 session, enter **`WRKLIB`** *`<package library name>`*.
2. Select Option 12 (work with objects).
3. Select Option 2 (edit authority on each object one at a time).
4. Change **`PUBLIC *EXCLUDE`** to **`PUBLIC *USE`**.

Another alternative is to use the GRTOBJAUT command from a 5250 session to grant *USE authority for the library and EXECUTE authority for the packages to all users. The following example assumes that the package library is named “STARSQL.”

```
GRTOBJAUT OBJ(STARSQL) OBJTYPE(*LIB) USER(*PUBLIC)
AUT(*USE)
GRTOBJAUT OBJ(STARSQL/*ALL) OBJTYPE(*ALL)
USER(*PUBLIC) AUT(*USE)
```

AutoBind Option

The AutoBind configuration keyword forces SQL packages to be bound at connect time (SQLConnect or SQLDriverConnect). You can set AutoBind in the DSN definition (see page 82) or pass it in the connect string to *SQLDriverConnect()*.

Caution

The use of AutoBind always causes binding to occur and adversely affects connect time performance.

The AutoBind option settings are described in Table 10.

Table 10. AutoBind Keyword Settings

AutoBind Value	Description
0 (Default)	No auto-bind at connect time. The driver still binds packages on-the-fly if it does not find them on the server when it needs them.
1	Binds packages at connect time, but uses NO REPLACE option on bind. If the packages already exist they will not be replaced. In the current version of StarSQL, AutoBind=1 ignores the NO REPLACE option and functions in the same manner as AutoBind=2.
2	Binds packages at connect time and always replaces current packages, if there are any.
Y	Same as 2.
N	Same as 0.

Regardless of the AutoBind setting, the StarSQL driver binds packages dynamically if it does not find them on the host at the time they are needed.

The packages that may be bound by AutoBind=1 and AutoBind=2 are:

- the dynamic package for the current transaction isolation level
- the catalog package

Appendix A

Manually Installing 32-bit StarSQL for UNIX

The procedures for manually installing StarSQL for UNIX vary depending on your operating system and whether you use an X Windows interface or a character-oriented terminal. The procedures in the following sections describe the typical methods for each supported variant of UNIX so you can choose which to use. If you need to remove the StarSQL for UNIX software, refer to "Removing the StarSQL 32-bit Driver" on page 21.

IInstalling StarSQL on Linux

StarSQL for Linux is distributed for installation in RPM format and as a tar file. Follow the procedures in one of the following two subsections, depending on which installation method you prefer to use.

RPM Installation

Some distributions of Linux include the RPM (RPM Package Manager) for installing software packages. The following procedures describe how to install the RPM format of the StarSQL distribution package.

1. Logon to Linux as `root` user.
2. Change to the directory that contains the StarSQL installer, typically `/tmp/starsql`.
3. Enter the following command to start the package manager, replacing the <*RPM file*> variable with the actual name of the StarSQL package (such as `starsql-5.51-1.i386.rpm`).

   ```
   # rpm -i <RPM file>
   ```

4. After the StarSQL software has been installed, edit the $STARDIR/etc/unixodbc.ini file and replace %TARGET with the pathname to the installed StarSQL software (such as usr/share/starsql).
5. Edit the $STARDIR/etc/post_install script and replace %TARGET with the pathname to the installed StarSQL software.
6. Enter the following command to run the “post_install” script, which sets any environment variables that need to be specified and registers the StarSQL driver with the ODBC Driver Manager.

```
$STARDIR/etc/post_install
```

Installation Using tar

The StarSQL for Linux distribution is also available as a tar-based installer that does not require the RPM Package Manager. If your version of Linux does not include the Package Manager, install the tar version of StarSQL.

1. Logon to Linux as `root` user.
2. Create an installation directory and change to that directory, such as:

```
# mkdir /usr/share/starsql
# cd /usr/share/starsql
```

3. Extract the contents of `starsql.tar.`

```
# tar xf /tmp/starsql.tar
```

4. After the tar file is extracted, edit the $STARDIR/etc/unixodbc.ini file and replace %TARGET with the pathname to the installed StarSQL software (such as usr/share/starsql).
5. Edit the $STARDIR/etc/post_install script and replace %TARGET with the pathname to the installed StarSQL software.
6. Enter the following command to run the “post_install” script, which sets any environment variables that need to be specified and registers the StarSQL driver with the ODBC Driver Manager.

```
$STARDIR/etc/post_install
```

Appendix B

Manually Installing 64-bit StarSQL for Linux

The procedures in this appendix describe how to manually install the 64-bit version of StarSQL for UNIX on a Linux-based computer. StarSQL for Linux is distributed in .rpm format for installation by the RPM Package Manager and as a tar file for installation using the tar command. Procedures are provided in this appendix for using either installation method. If you need to remove the StarSQL for UNIX software, refer to "Removing StarSQL from a 64-bit Computer (Linux)" on page 24.

Installing StarSQL on a Linux Computer

The steps in the sections below describe how to install the 64-bit version of StarSQL using either the Package Manager (RPM) or the tar command.

Installation of StarSQL Using RPM

If you are running Red Hat Linux or another distribution that includes the Package Manager, you can use the RPM to install StarSQL.

1. Logon to Linux as `root` user.
2. Change to the directory that contains the StarSQL installer, typically `/tmp/starsql`64.
3. Enter the following command to start the package manager, replacing the <*RPM file*> variable with the actual name of the StarSQL package (such as `starsql64-5.51-1.x86_64.rpm`).

   ```
   # rpm -i <RPM file>
   ```
4. After the StarSQL software has been installed, edit the $STARDIR/etc/unixodbc.ini file and replace %TARGET with the pathname to the installed StarSQL software (such as usr/share/starsql64).

5. Edit the $STARDIR/etc/post_install script and replace %TARGET with the pathname to the installed StarSQL software.
6. Enter the following command to run the “post_install” script, which sets any environment variables that need to be specified and registers the StarSQL driver with the ODBC Driver Manager.

```
$STARDIR/etc/post_install
```

Installation of StarSQL Using tar

The StarSQL for Linux software also is available as a tar file that can be installed from a command line. If your Linux distribution does not include RPM, install the tar version of StarSQL.

1. Logon to Linux as `root` user.
2. Create an installation directory and change to that directory, such as:

```
# mkdir $STARDIR
# cd $STARDIR
```

3. Extract the contents of `starsql.tar.`

```
# tar xf /tmp/starsql64.tar
```

4. After the tar file is extracted, edit the $STARDIR/etc/unixodbc.ini file and replace %TARGET with the pathname to the installed StarSQL software (such as usr/share/starsql64).
5. Edit the $STARDIR/etc/post_install script and replace %TARGET with the pathname to the installed StarSQL software.
6. Enter the following command to run the “post_install” script, which sets any environment variables that need to be specified and registers the StarSQL driver with the ODBC Driver Manager.

```
$STARDIR/etc/post_install
```

Appendix C

StarSQL for UNIX Command Reference

This chapter provides a reference for using the following StarSQL for UNIX commands:

- simpleconn, page 68
- starlic-clientcfg, page 70
- starping, page 72
- trcstart, page 73

simpleconn

Purpose

Tests that StarSQL can use the unixODBC driver manager to connect to a specific data source on the host.

Synopsis

`simpleconn` `DSN UserId Password [#Connections] [-wait]`

| [-pool] | [-nopool] [# *of pooled connections*]

Options

DSN	The data source name to which you want to connect.
UserId	A valid user ID for a user account on the host.
Password	The valid password for the userid.
#Connections	The number of times to connect to the data source.

-wait - The application will establish the connections, then wait until you hit enter to continue before releasing them.

-pool #pooled-connections - enables connection pooling and connects/disconnects N times

-nopool #pooled-connections - connects/disconnects N times without enabling pooling

`Arguments are position-sensitive.`

`-wait, -pool and -nopool are mutually exclusive.`

If you use -pool or -nopool, you should supply a value for # of pooled connections

Description

You can use the simpleconn command to test that StarSQL can use the unixODBC driver manager to successfully connect to a specified data source on the host. Upon successful connection, the simpleconn command returns the database name, platform, and version, and the driver name and version.

You must specify a username and password of an account that is valid on the host and has permission to access the database. (See "Preparation Required for All Hosts" on page 39 for more information about user accounts and permissions.)

The #Connections option lets you specify how many connections you want to establish. The default is 1. If you specify more than one connection, the simpleconn program establishes that number of connections, and then releases them before the command returns.

This command invokes the simpconn example program included with StarSQL in the $STARDIR/samples directory. The simpleconn program is compiled with unixODBC include files and libraries.

Example

The following command example tests that StarSQL can establish two connections to a data source named DSN1 using the unixODBC driver manager.

```
$ simpleconn DSN1 staruser starpass 2

Connection #1
   Driver Name: libswodbc.a
   Driver Version: 5.50.nnnn
   Database Name: DB2 Universal DataBase
   Database Version: 08.021.0007
Connection #2
   Driver Name: libswodbc.a
   Driver Version: 5.50.nnnn
   Database Name: DB2 Universal DataBase
   Database Version: 08.021.0007
```

See Also

starping, page 72

starlic-clientcfg

Purpose

Configure and manage StarLicense client computer.

Synopsis

`starlic-clientcfg` `[options]`

Options

Some of the more commonly-used options for configuring a client computer are shown below. Note that the StarLicense Client Configuration Menu provides an easier method for managing StarLicense client-server connections. Refer to page 99 for information about using the StarLicense Client Configuration Menu.

client-loglevel-set [0 | FFFFFFFF]
: Setting the log level to 0 (zero) disables logging. Set the log level to FFFFFFFF (eight F's) to enable logging of activity between the client and license service. The log file is created in the /tmp directory with the filename format **`starliccli.<`*`processID`*`>.log`**.

client-loglevel-show
: Displays the log level value, which indicates whether logging is enabled (FFFFFFFF) or disabled (0).

client-server-show
: Shows information about the license server(s) the client is configured to use.

client-server-add *<product_ID>* *<hostname>* *<port>* [PRI|SEC]
: Configures the client to check out a license for the specified StarQuest product on the specified host. The product ID must match the product ID specified on the StarLicense server. The *hostname* can be the DNS name or TCP/IP address of the StarLicense server and the *port* variable is the number of the port on which the server is listening for license requests. Specify the option PRI to add the server as a Primary server, or SEC to add it as a Secondary server.

client-server-remove *<product_ID>* *<hostname>* *<port>* [PRI | SEC]
: Removes the specified StarLicense server from the client license configuration.

Description

Use the **client-server** options to configure the client computer to use one or more StarLicense servers. The StarLicense server software can be co-located on the same computer as the StarSQL driver, or it can be installed on a network server.

Example

The following example configures the computer to checkout StarSQL licenses from a StarLicense server named **starlic** that is listening for license requests on port 4999.

```
# ./starlic-clientcfg client-server-add SQ starlic 4999 PRI
```

The following example configures the client computer to use a StarLicense server named **starlic2** as a Secondary server if the Primary license server cannot satisfy the license request. The starlic2 server listens for license requests on port 5001.

```
# ./starlic-clientcfg client-server-add SQ starlic2 5001
SEC
```

See Also

"Licensing StarQuest Products" on page 27

"StarLicense Client Configuration Utility" on page 99

starping

Purpose

Tests the StarSQL connection to a specific data source.

Synopsis

`starping` `[`*`DSN UserID Password`*`]`

Options

`DSN`	The data source to which you want to connect.
`UserID`	A valid user ID for an account on the host.
`Password`	The valid password for the user ID.

Description

The **starping** command connects to the specified DSN using a user ID and password. If you enter the command without the required parameters, the command prompts for the data source name, user ID, and password before it attempts the connection.

After a successful connection, the command returns “Connection Succeeded!” and the data source name (DSN) and database platform version number.

Example

```
starping DSN1 staruser starpass
```

See Also

simpleconn, page 68

trcstart

Purpose

Starts and ends DRDA tracing.

Synopsis

trcstart {**-off**|**-on** *TraceFile.sqd*}

Options

-off	Turns off the DRDA trace recording currently in progress.
-on *TraceFile.sqd*	Turns on DRDA trace recording to the specified trace file.

Description

The **trcstart** command allows a user to capture the DRDA data streams that the StarSQL driver is sending and receiving.

To enable DRDA tracing the `.swodbc.ini` file must be in the user's home directory and the user must have permission to write to the file.

To view the trace file, transfer the file (with a .sqd extension) to a Windows computer in binary mode and use the StarSQL Trace Viewer application installed as a custom selection by StarSQL for Windows.

Examples

To copy swodbc.ini to the user home directory and make it writable:

```
$ cp $STARSQL/etc/swodbc.ini .swodbc.ini
$ ls -l .swodbc.ini
$ chmod 644 .swodbc.ini
```

To enable DRDA tracing:

```
$ trcstart -on /tmp/mytrace.sqd
```

To turn off tracing:

```
$ trcstart -off
```

Appendix D

Customizing the StarSQL for UNIX Configuration

You can control how the StarSQL driver interacts with a data source by modifying the data source configuration settings. There are three primary methods for defining an ODBC data source:

- use the unixODBC ODBC Data Source Administrator, as described in "Customizing the unixODBC Driver Manager Configuration" on page 29
- edit the System DSN file (odbc.ini) or the User DSN file (.odbc.ini)
- create a File DSN

For a StarSQL data source you can specify Global Settings, which determine how the StarSQL driver behaves with all data sources, and you can define specific data source settings, which control how StarSQL interacts with a particular Data Source Name (DSN).

Setting Global DSN Parameters

The Global Settings are defined in a file named `swodbc.ini` (or `.swodbc64.ini` on a 64-bit computer). The `swodbc.ini` file can be located in the program directory ($STARDIR/etc/swodbc.ini), in a shared network location with a symbolic link from the $HOME directory (as .swodbc.ini) for users to find it, or a personal copy can be maintained in the $HOME directory of a 32-bit client computer (`.swodbc.ini`) or a 64-bit client computer (`.swodbc64.ini`).

The global DSN parameters are described within a stanza of the `swodbc.ini` configuration file and affect all DSNs. Within each stanza you specify keywords and values in the format shown below:

```
Keyword=value
```

The next section provides details about each configuration keyword you can specify in the `.swodbc.ini` global settings file.

Global DSN Keywords

For most users, the default values for the global data source settings, which are stored in the `$STARDIR/swodbc.ini` file, are adequate. However, you can implement special behavior by modifying the global settings.

AlwUpd

All database transactions are read/write by default (AlwUpd=Yes). Setting AlwUpd=No restricts the driver to read-only transactions.

You also can control whether transactions can be updated with the specific DSN configuration ReadOnly keyword (see "ReadOnly" on page 92). If both the AlwUpd global setting and the ReadOnly DSN setting are specified, the StarSQL driver uses the most restrictive setting. A global setting that allows read/write transactions (AlwUpd=Yes) can be overridden by a specific DSN setting that allows just read-only transactions (ReadOnly=Yes). Likewise, a transaction to a data source is restricted to read-only if the global setting restricts the driver to read-only (AlwUpd=No), even if the specific DSN configuration allows read/write (ReadOnly=No).

AutoTypDefOvr

The AutoTypDefOvr setting stores the Coded Character Set Identifiers (CCSIDs) that the DB2 host provides to StarSQL upon connection. After the connection with the host is established, StarSQL uses the CCSID values specified for AutoTypDefOvr to send SQL statement and parameter data to the host unless the TypDefOvr value specifies to use different CCSIDs. The TypDefOvr setting takes precedence over values specified in the AutoTypDefOvr setting. However, if the AutoTypDefOvr CCSID values are appropriate, TypDefOvr values do not need to be sent with each DRDA request.

The AutoTypDefOvr keyword value consists of three comma-separated CCSID values for the single-byte character set (SBCS), the double-byte character set (DBCS), and the mixed-byte character set (MBCS), respectively.

```
<SBCS CCSID>,<DBCS CCSID>,<MBCS CCSID>
```

If the host returns a different CCSID than you want to use for the connection, or you want to override a multi-byte client code page with a MBCS CCSID, set the TypDefOvr to send a particular CCSID value with each DRDA request. (See "TypDefOvr" on page 94 for more information about the TypDefOvr setting.)

CacheUID

If you set CacheUID=Yes (the default), the username to connect using StarSQL is displayed in the connect dialog and the user only needs to enter their password. Set CacheUID=No to require users to enter both their username and their password to connect to the DB2 host using StarSQL.

Explain

DB2 provides an explain facility that provides detailed information about the access plan and environment of static or dynamic SQL statements. The captured information can help administrators understand how individual SQL statements are executed so the statement and database configuration can be tuned for performance. You can use a command-line tool or DB2 Visual Explain to display explain information.

The keyword Explain keyword in the `swodbc.ini` initialization file instructs StarSQL to bind all StarSQL packages with the EXPLAIN bind option set to ALL. This includes dynamic SQL packages that are bound with StarAdmin Classic and packages that are created and bound as needed by the StarSQL driver. If you are using StarAdmin (Java-based), the Explain option can be found in Advanced properties.

To enable the EXPLAIN bind option for all packages that are bound by StarSQL, add the following line to the [Defaults] section of the `.swodbc.ini` file:

```
[Defaults]
Explain=All
```

After you modify the initialization file to include the Explain=All statement, the explain information for packages that are subsequently bound by StarSQL can be displayed.

Setting Up System and User DSN Files

System DSN settings are defined in `odbc.ini`, and user DSN settings are defined in $HOME/`.odbc.ini.` The system DSN file uses the same format and keywords as for a user DSN file. The following procedures describe how to set up and edit the sample `.odbc.ini` file included in the StarSQL distribution to create a system or user DSN file with the required settings.

1. To create a system DSN, edit the odbc.ini file. The location of the file varies depending on which version of UNIX the computer is running, but is typically located in `/etc/odbc` or `/usr/local/etc`.

   To create a user DSN, copy the `odbc.ini` file from $STARDIR/etc to your home directory and rename it `.odbc.ini` (preceded with a period):

   ```
   $ cp $STARDIR/etc/odbc.ini $HOME/.odbc.ini
   ```

2. Make sure the system or user initialization file can be written to. To change the permissions for the system DSN file:

   ```
   $ chmod 644 $STARDIR/etc/odbc.ini
   ```

   To change permissions for the user DSN file:

   ```
   $ chmod 644 $HOME/.odbc.ini
   ```

3. Using a text editor such as vi or emacs, edit the required keywords, described in Table 11, with values appropriate to your environment. You can further customize the DSN configuration with any of the keywords and values described in "Sample DSN File" on page 79.

Table 11. Required Data Source Configuration Parameters

Keyword	**Description**
Driver=	If you install StarSQL to a location other than the default directory, specify the fully qualified path to the driver. You also can specify the name of the driver as it appears in the section head of the `odbcinst.ini` file, such as Driver=StarSQL or Driver=StarSQL64, to automatically use the location specified in the driver manager repository (Figure 3 on page 31 shows an example `odbcinst.ini` file).
HostName=	The host name refers to the name of the DB2 host system. Set the HostName field to either a TCP/IP host name (such as host5.mydomain.com) or a static IP address in dotted decimal notation (such as 198.147.235.1).
PkgColID=	The SQL Package Collection ID indicates the location on the DB2 host of the packages required by StarSQL to execute Dynamic SQL. On DB2 for i, the SQL Package Collection ID is the name of the collection or library that contains the packages. On all other platforms, the Package Collection ID is the name of the virtual collection associated with these packages. If necessary, obtain the package collection ID from your Database Administrator.

Keyword	Description
Port=	The Port designates the IP port on which the DB2 host is listening for incoming TCP/IP connection requests. The default port for DRDA communications is 446 (unencrypted) or 448 (SSL-encrypted).
Server=	Enter the name of the database server that will be accessed through this data source. You may need to obtain the database server name from the Database Administrator. The database server name is known by different names depending on the DB2 host implementation. On DB2 for z/OS it is called the location name, on DB2 for i it is called the relational database name (RDB), and on DB2 LUW for Windows and UNIX, it is the name of the database.

The next section provides details about the configuration keywords you can specify in the `odbc.ini` system DSN file or the `.odbc.ini` user DSN file to customize the behavior of the StarSQL driver. The information for a particular DSN is described within a stanza of the configuration file. Within each stanza you specify keywords and values for the DSN in the format shown below:

```
Keyword=value
```

Sample DSN File

Specific data source settings, which are stored in **$HOME/.odbc.ini** for user DSNs and in **$STARDIR/etc/odbc.ini** for system DSNs, control how StarSQL interacts with a particular data source. Below is a sample `.odbc.ini` file that provides a DSN definition for a data source named SAMPLEDSN. The parameters listed at the beginning of the [SAMPLEDSN] section in **bold** are the minimum required for connectivity and are described in the table "Required Data Source Configuration Parameters" on page 78.

```
[ODBC Data Sources]
SAMPLEDSN=StarSQL

[SAMPLEDSN]
Driver=/usr/share/starsql64/lib/libSWODBC.so
HostName=host.domain.com
PkgColId=STARSQL
Port=446
Server=RDBNAME
```

```
Accounting=""
AllowSynonyms=Yes
AlwaysWide=No
AutoBind=No
AutoSqlSet=D
AutoTypDefOvr=1252, 0, 0
BinaryCCSID=37
BindRules=Run
CacheUID=Yes
CatFilters=’NAME1’,’NAME2’
CatQual=SYSIBM
CharacterSubstitution=Warning
CreateTable=IN DATABASE DSNDB04
CustomizePrdid=
DefaultQualifier=STARUSER
Descriptions=Sample DSN Configuration
FetchAhead=Yes
FoldUIDPWD=No
GetInfoCatalogUsage=
HeldCursors=Default
HideTsScale=No
IncludeSynonyms=Yes
IsolationLevel=<default>
KeepDynamic=Yes
LongStrParams=No
MaxRows=
OverrideCodeset=1252,S,Windows-1252,MYCODESETNAME
PkgOwnID=
QryBufSiz=2097151
ReadOnly=No
SpecialColumns=Yes
SSL=No
SSLOptions=<64-bit hex string>
StrictParsing=Yes
TrustedConnection=No
TwoPC=No
TypDefOvr=1252,0,0
UseDSCRDBTBL=Yes
UseDynamicCatalogSQL=Yes
UseEncryption=Any
UseJumboPackages=No
UseSYSDUMMYAEU=Yes
```

The following sections describe the keywords you can specify in the DSN definition should you need to further customize how the StarSQL driver connects to a data source.

System and User DSN Keywords

For most users, the default values that StarSQL uses for connecting to a data source are adequate. However, you can implement special behavior by adding or modifying the keywords described in this section.

Accounting

z/OS allows you to specify an accounting string for charging back the mainframe resources that particular users consume while connected to DB2 databases through StarSQL. The accounting string is passed in the PRDDTA parameter of the ACCRDB DRDA command. It is stored as an accounting record on the mainframe.

An accounting string contains a system-generated prefix and a user-generated suffix, which together provide the information needed to associate resource usage with a user's access for the purpose of charge-back accounting. The value you specify for the Accounting keyword provides the user-defined suffix of the full accounting string.

The accounting string prefix consists of 56 bytes generated by StarSQL. The fields are right-padded and are described in Table 12.

Table 12. Accounting String Prefix

Field	**Description**
acct_str_len	1 byte, length of the accounting string minus 1 in hexadecimal format
client_prdid	8 bytes, product ID for the driver
client_platform	18 bytes, client operating system
client_appl_name	20 bytes, client application
client_authid	8 bytes, authorization id of the StarSQL user
suffix_len	1 byte, length of the accounting sting suffix in hexadecimal format

The user-defined suffix is used to further refine the accounting record. The user enters the suffix in the accounting string field in the data source setup. In the following example accounting string, the user-supplied suffix is SALES:

```
"x'3C'STARSQL2UNIX mypgm BARNES x'05'SALES"
```

If there is no value specified for the Accounting keyword, the accounting string will have a null suffix, as shown in the following example:

```
"x'3C'STARSQL2UNIX mypgm BARNES"
```

For information about interpreting the accounting record on DB2, see the IBM documentation for your version of DB2 for z/OS.

AllowSynonyms

The default behavior of StarSQL is to retrieve index information for synonyms from DB2 for z/OS. To disable synonym support, enter AllowSynonyms=No in the data source configuration. If you set AllowSynonyms to No, StarSQL uses the default qualifier to qualify unqualified table references in catalog calls, such as with SQLStatistics().

AlwaysWide

Setting this option to Yes will force all SQL parameters and resultset columns of “character-type” to be described as their “WIDE” equivalents.

AutoBind

The AutoBind keyword forces SQL packages to be bound at connect time. Normally the use of AutoBind is not recommended, because it always causes binding to occur and thus adversely affects connect time performance.

The AutoBind option settings are described in Table 13.

Table 13. AutoBind Keyword Settings

AutoBind Value	Description
0 (Default)	No auto-bind at connect time. The driver still binds packages on-the-fly if it does not find them on the server when it needs them.
1	Binds packages at connect time, but uses NO REPLACE option on bind. If the packages already exist they will not be replaced. In the current version of StarSQL, AutoBind=1 ignores the NO REPLACE option and functions in the same manner as AutoBind=2.
2	Binds packages at connect time and always replaces current packages, if there are any.
Y	Same as 2.
N	Same as 0.

Regardless of the AutoBind setting, the driver still binds packages dynamically if it does not find them on the host at the time they are needed. The packages that may be bound by 1 and 2 are: the dynamic package for the current transaction isolation level and the catalog package.

AutoSqlSet

Some DBMS systems accept information that identifies the client connection---user ID, computer name, and application name. Set AutoSqlSet to No if the host does not support this functionality or you do not want to send this client connection information to the DBMS.

If StarSQL detects that the host supports the client connection information, it automatically sets the AutoSqlSet setting to Yes. Set AutoSqlSet to D (for "disabled") if you do not want StarSQL to change the AutoSqlSet setting to Yes for hosts that accept client connection information.

AutoTypDefOvr

You can set a value for AutoTypDefOvr in the global DSN `.swodbc.ini` file, for a system DSN in the `odbc.ini` file, or for a user DSN in the `.odbc.ini` file. The AutoTypDefOvr setting stores the Coded Character Set Identifiers (CCSIDs) that the DB2 host provides to StarSQL upon connection.

After the connection with the host is established, StarSQL uses the CCSID values specified for AutoTypDefOvr to send SQL statement and parameter data to the host unless the TypDefOvr value specifies to use different CCSIDs. The TypDefOvr setting takes precedence over values specified in the AutoTypDefOvr setting. However, if the AutoTypDefOvr CCSID values are appropriate, TypDefOvr values do not need to be sent with each DRDA request.

The AutoTypDefOvr keyword value consists of three comma-separated CCSID values for the single-byte character set (SBCS), the double-byte character set (DBCS), and the mixed-byte character set (MBCS), respectively.

```
<SBCS CCSID>,<DBCS CCSID>,<MBCS CCSID>
```

If the host returns a different CCSID than you want to use for the connection, or you want to override a multi-byte client code page with a MBCS CCSID, set the TypDefOvr to send a particular CCSID value with each DRDA request. (See "TypDefOvr" on page 94 for more information about the TypDefOvr setting.)

BinaryCCSID

If you are experiencing problems with character data being returned as binary data from DB2 for i, specify the CCSID for the character set you want to use in the BinaryCCSID field. This instructs StarSQL to treat binary data as character data using the specified CCSID. For the complete list of supported CCSIDs, refer to the “StarSQL Character Conversion and National Language Support” document in the StarSQL technical documents of the StarQuest Web site.

BindRules

The default value for the BindRules setting is RUN. This setting applies only to applications that use dynamic SQL on a data source on DB2 for z/OS. Set this option to RUN or BIND as desired.

On DB2 for z/OS, the permissions required to run applications that use dynamic SQL depend on the value of the BindRules option. If it is set to RUN, the StarSQL user needs to have explicit permissions granted by the administrator to read and write the columns and tables accessed by the application. If it is set to BIND, the StarSQL user executes with the permissions of the owner of the dynamic SQL package.

If the option is set to BIND, the following SQL statements cannot be executed, regardless of the actual permissions of the package owner:

- SET CURRENT SQLID
- GRANT
- REVOKE
- ALTER
- CREATE
- DROP
- Any SQL statement that cannot be prepared as dynamic SQL

CacheUID

If you set CacheUID=Yes (the default), the username to connect using StarSQL is displayed in the connect dialog and the user only needs to enter their password. Set CacheUID=No to require users to enter both their username and their password to connect to the DB2 host using StarSQL.

CatFilters

The CatFilters setting restricts the amount of data retrieved by ODBC catalog functions when a NULL or `‘%’` is passed as the schema name. Set CatFilters to the name of a particular library or collection to limit the amount of data retrieved from the specified library or collection when a catalog function is executed. You can specify a maximum of 10 names for filtering the data. The names must be enclosed in single quotation marks and separated by a comma (such as `‘name1’,‘name2’,‘name3’,...`).

This feature is supported by the following DB2 hosts:

- DB2 for z/OS
- DB2 for i
- DB2 for Linux, UNIX & Windows (Db2 or DB2 LUXW)

CatQual

Set the catalog qualifier to the location of the DB2 system tables that StarSQL uses when an application calls an ODBC catalog function. Set the catalog qualifier to SYSIBM for DB2 for z/OS, SYSCAT for DB2 LUW, or QSYS2 for DB2 for i (OS/400).

CharacterSubstituion

If one or more characters cannot be converted between the host and client character sets, StarSQL substitutes a question mark, ?, in place of the character(s). The CharacterSubstitution parameter determines whether a warning is generated when characters are replaced (parameter value of Warning), or the characters are substituted without generating a warning (parameter value of Silent).

CreateTable

Set the CreateTable string to contain a directive that you want to append to SQL CREATE TABLE statements. The directive indicates where you want the table to be created.

For example, if **`CreateTable=IN DATABASE DSNTEST`** and an application passes the following statement:

```
create table test (num integer)
```

the StarSQL driver passes the statement to DB2 as:

```
create table test (num integer) IN DATABASE DSNTEST
```

Valid values for the CreateTable keyword depend on the host where the data source is located, as shown in Table 14.

Table 14. Supported CreateTable Directives

DB2 Host	Valid Table Creation Directives	Comment
DB2 for z/OS	**IN DATABASE** *DATABASE_NAME* **IN** *DATABASE_NAME.TABLESPACE*	
DB2 for i	**IN NODE** *GROUP_NAME*	A node group name is a qualified or unqualified name that designates a nodegroup, which is a group of DB2 for i systems across which a table is distributed.
DB2 LUW	**IN** *TABLESPACE_NAME*	A tablespace name is a long qualifier that identifies a distinct table space that is described in the DB2 catalog.

CustomizePrdid

The CustomizePrdid parameter customizes the behavior of a DB2 for z/OS host that is using a double-byte character set (DBCS). DB2 for z/OS does not support the ESCAPE clause in SQL syntax when the LIKE argument uses a mixed-byte character set.

When the CustomizePrdid parameter is set to M (for Mixed), StarSQL will strip the backslash escape character (\) from both the catalog SQL that is used to bind catalog packages and from dynamic catalog SQL calls. The CustomizePrdid parameter must be enabled prior to binding packages, and it must be enabled in the DSN that is used to connect to the host from the StarSQL client.

The escape character precedes a special character, such as the underscore (_) character, to treat the character as a literal. To work with tables that include an underscore in the name, such as MY_TABLE, create an ALIAS, SYNONYM, or VIEW for the table with a name that includes no special characters instead of working directly with the table.

DefaultQualifier

Set DefaultQualifier to a value that you want to use to qualify all unqualified SQL statements. On DB2 for i, this qualifier refers to an AS/400 library. On other DB2 hosts it refers to an Owner or Authorization Identifier.

Description

This is an optional keyword to provide a textual description of the data source.

FetchAhead

FetchAhead is a means of retrieving data from the host before the application needs it. This data is fetched from the host, stored in the driver, and returned to the application on request. This improves performance by reducing delays in receiving data from the host. The default for FetchAhead is Yes.

FoldUIDPWD

Set FoldUIDPWD to Yes if you want StarSQL to force the UserID and Password to uppercase letters if the user is connecting using Microsoft SNA. When FoldUIDPWD is set to No, user IDs and passwords are sent in the case they were entered.

GetInfoCatalogUsage

Applications can call SQLGetInfo with the SQL_CATALOG_USAGE option to determine the classes of SQL statements in which catalog and schema names can be used to qualify objects of SQL statements. Setting the GetInfoCatalogUsage parameter of the StarSQL driver allows you to customize the bitmask that is returned by SQLGetInfo() for SQL_CATALOG_USAGE.

Set GetInfoCatalogUsage to 0 to force StarSQL to report that it does not support three-part naming (no Catalog). This may be necessary in some situations, such as when using StarSQL to perform queries to a SQL Server linked server. If GetInfoCatalogUsage contains an empty string or -1, the normal bitmask is returned.

HeldCursors

You can set the HeldCursors keyword to No, Yes, or Default. Yes enables held cursors. No disables held cursors. The behavior of held cursors when the value is Default depends on the host platform, as shown in Table 15. If the host does not support held cursors, the HeldCursors keyword is ignored.

Table 15. Behavior of HeldCursors=Default by Host

Host	Behavior if HeldCursors=Default
DB2 for z/OS	Enables held cursors (equivalent of HeldCursors=Yes)
DB2 for i and DB2 LUW	Disables held cursors (equivalent of HeldCursors=No)

Normally, a cursor is closed when its transaction commits. If the HeldCursors keyword is enabled, the cursor remains open after the commit. This allows an application to fetch rows from a result set, commit the transaction, and then continue fetching additional rows on the same result set.

Using held cursors, an application can commit immediately after opening the result set to allow locks normally needed to maintain a prepared statement to be freed early. If the application is in auto-commit mode, StarSQL automatically issues a commit at the time the result set is opened and at the time the result set is closed.

An application that uses held cursors should turn off SQL_AUTOCOMMIT_MODE before executing a prepared statement multiple times. Otherwise, StarSQL prepares the SQL statement after each SQLExecute, resulting in slower performance.

Before enabling HeldCursors, manually delete any existing SQL package used for catalog information on DB2. This is the package with the same name as the catalog schema. A new SQL package will be created the next time you establish a connection using the updated StarSQL driver.

If you are upgrading from a previous version of StarSQL, you may need to delete and rebind the SQL catalog package if you change the HeldCursors setting.

HideTsScale

IStarSQL 6.34 & later supports (and properly describes) TIMESTAMP columns with a scale between 0 and 12. We have chosen to expose DB2's default timestamp precision of 6 as a separate data type name in SQLGetTypeInfo(), named "TIMESTAMP(6)", the type name "TIMESTAMP" is now described as having variable scale from 0 to 12.

Setting the data source option HideTsScale=Yes will instruct StarSQL to revert to pre-6.34 behavior and hide the variable scale version of TIMESTAMP in SQLGetTypeInfo().

IncludeSynonyms

IncludeSynonyms support is enabled by default. Set IncludeSynonyms to No if you do not want to support SYNONYMS in catalog calls. When AllowSynonyms or IncludeSynonyms is disabled, synonyms are not returned in results and are not found when passed as input parameters. When AllowSynonyms or IncludeSynonyms is set to Yes, synonyms are supported, as both input and output, in catalog calls.

IsolationLevel

IsolationLevel refers to the degree of concurrency allowed when multiple transactions try to access the same data. You can set the isolation levels for StarSQL using an integer that represents the level, as described in Table 16. The StarSQL driver uses a default isolation level of 0 for the AS/400 host and a default of 2 for the other supported hosts.

Table 16. IsolationLevel Values

IsolationLevel Value	Description
<Default>	The default is dependent on the host platform. On DB2 for i, the default is None. For most host platforms, the default is Read Committed, which allows the StarSQL driver to access non-journaled files.
0	**None**: No isolation of concurrent application processes.
1	**Read_Uncommitted**: For a SELECT INTO, FETCH with a read-only cursor, subquery, or subselect used in an INSERT statement, allows any row read during the unit of work to be changed by other application processes and any row changed by another application process to be read even if the change has not been committed by that application process. For other operations, behaves like Read Committed.
2	**Read_Committed**: Similar to Repeatable Read in that a unit of work cannot be changed by another process until it has completed, and a row changed by another application process cannot be read until it has been committed. However, with Read Committed, an application process that executes the same query more than once may see newly committed rows inserted by another application process that were not present in the original read.
4	**Repeatable_Read**: Ensures that any row read during a unit of work can not be changed by other application processes until the unit of work has completed. Also ensures that any row changed by another application process cannot be read until it has been committed. At this level, a process is completely isolated from the effects of concurrent application processes.
8	**Serializable**: Ensures that any row changed by another application process cannot be read until it is committed by that application process. Ensures only that the current row of every updateable cursor is not changed by other application processes. Rows read during a unit of work can be changed by other application processes.

KeepDynamic

The KeepDynamic keyword enables or disables optimization for prepared statements by tuning the KeepDynamic bind option (DB2 for z/OS and DB2 LUW). The default for KeepDynamic is Yes.

LongStrParams

StarSQL v5 added support for DB2 LOB (large object) data types. The LongStrParams setting provides backwards compatibility to use StarSQL v5 with hosts that do not support LOBs and applications that rely on the previous ODBC data types.

As shown in Table 2 on page 22, StarSQL v5 changes the mapping for DB2 long strings—they are no longer differentiated from short strings:

The new data type mappings affect ODBC catalog query results. For example, SQLColumns for a DB2 LONG VARCHAR now returns ODBC type SQL_VARCHAR instead of SQL_LONGVARCHAR. The new mappings also affect parameters for SQL statements. If an application binds a parameter as SQL type SQL_LONGVARCHAR, it is sent as a DB2 CLOB instead of as a DB2 VARCHAR string.

When LongStrParams is set to Yes, then SQL_LONGVARCHAR is sent as VARCHAR instead of CLOB, and SQL_LONGVARBINARY is sent as VARCHAR FOR BIT DATA instead of BLOB. This LongStrParams setting does not affect the types returned for result set columns, or for types returned by catalog queries.

When StarSQL v5 or later connects to an older version of DB2 that does not support LOB data types, the LongStrParams data source setting is forced to Yes and SQL_LONGVARCHAR and SQL_LONGVARBINARY parameters are sent as strings. Catalog queries, however, will not return SQL_LONGVARCHAR or SQL_LONGVARBINARY for long string types.

MaxRows

Set the MaxRows keyword to the maximum number of rows that can be returned by a result set. Set this value to limit the number of rows returned in queries that do not call SQLSetStmtOption with the SQL_MAX_ROWS parameter.

If you do not want to set an upper limit, enter zero or leave the field blank.

OverrideCodeset

The OverrideCodeset parameter allows you to override the name of the local codeset that StarSQL uses to convert outbound character data. The codeset names are defined in the library that contains the conversion routines.

StarSQL obtains the locale (Linux, UNIX) or code page (Windows) that is set on the client computer and performs a lookup of the corresponding codeset name in the ccsid.csv table to find the corresponding CCSID value to use. For example, if the code page for a Windows client is set to 1252, StarSQL looks for a codeset name of “WINDOWS-1252” in the ccsid.csv file.

The OverrideCodeset parameter can be useful if you want to send data encoded using a different codeset, or to match a local codeset name that is not in the ccsid.csv file to one that is. The value for the OverrideCodeset parameter can be any string.

PkgOwnID

The SQL Package Owner ID (PkgOwnID) specifies the owner ID to associate with the StarSQL driver packages that reside on the host. The target DBMS host uses the PkgOwnID value to validate whether the user has authority to perform the functions represented by the SQL statements in the bound package. This option can be useful in situations where an administrator needs to bind the driver packages but does not want to be the owner of the packages.

The default package owner is the user ID that is used to establish the ODBC connection when a package is bound. Enter up to 8 characters for the PkgOwnID option to specify a different user ID as the package owner. After you set the PkgOwnID option, run StarAdmin to bind the desired StarSQL packages with the package owner that was specified for PkgOwnID.

QryBufSiz

Set the size of the query buffer. The buffer size can be between 512 bytes and 10485760 bytes. The default is 2097151. A large query buffer can improve performance for large queries.

ReadOnly

Set to Yes if the data source is read only. This prevents users from attempting to update the host database. Set to No if you want the StarSQL user to be able to update the host database (to the extent allowed by the host security).

SpecialColumns

The SQLSpecialColumns function retrieves the following information about columns in a specific table:

- the optimal set of columns that uniquely identify a row
- columns that are automatically updated when any value in the row is updated

By default the SpecialColumns function is enabled (set to Yes). To disable the function, set the SpecialColumns keyword to No, which returns an empty results set if the SpecialColumns function is invoked.

SSL

Setting this option to Yes will cause StarSQL to establish a secure connection using SSL/TLS protocols. The following requirements apply:

- SSL support requires certain versions of Linux and OpenSSL Refer to the System Requirements section of the Release Notes for the supported versions of Linux and OpenSSL
- The DB2 host must be enabled for SSL communications. If the host is not enabled for SSL, you can use a StarQuest StarPipes Gateway to provide SSL support.
- Set the Port property to the appropriate value for your DB2 host (typically 448)
- Install the CA certificate for the Certificate Authority that issued the server certificate being used by the DB2 host into the certificate store used by OpenSSL This may not be necessary if the certificate was issued by a public certificate authority that is already known by OpenSSL,

SSLOptions

The value of this parameter is a 64-bit hexadecimal string that can be used to supply advanced options for SSL/TLS communications. Contact StarQuest Support for usage.

StrictParsing

When StrictParsing is enabled, which is the default, the SQL sent by an application must be acceptable to the StarSQL parser. If the SQL does not comply, StarSQL generates an error message and does not send the SQL.

If StrictParsing is disabled, StarSQL passes the SQL to the host database as-is. Some functionality, such as SQLDescribeParam(), may not be available to the application if StrictParsing is disabled.

TrustedConnection

Setting this option to YES will cause StarSQL to attempt to establish a "Trusted Connection". That connection can then be used to switch between active User Ids, optionally using "Identity Propagation".

Currently, this option is supported only for connections to DB2 for z/OS (v9 and later); contact StarQuest support if you plan to use this option for connections to DB2 for LUW.

For further details, see the technical note “Using TrustedConnection and Identity Propagation” on the StarQuest website

.

TwoPC

Set the TwoPC keyword to **XA** to use XA Distributed Transactions. The default value is **No**. The value **TCP/IP**, used for Two-Phase Commit with Microsoft Transaction Server (MTS) and Distributed Transaction Coordinator (DTC), is supported by StarSQL for Windows and is not available for StarSQL for UNIX.

TypDefOvr

Use the TypDefOvr keyword to explicitly set the SBCS CCSID that StarSQL uses to send character data to the DB2 host. When TypDefOvr is set, StarSQL sends the data using the CCSID specified in the TypDefOvr with each DRDA request.

The CCSID specified for TypDefOvr overrides the CCSID values specified for AutoTypDefOvr (see page 76). The TypDefOvr keyword value can be any SBCS CCSID that StarSQL supports. (For the complete list of supported CCSIDs, refer to the “StarSQL Character Conversion and National Language Support” document in the StarSQL technical documents of the StarQuest website.)

If the CCSID specified for TypDefOvr is for a single-byte character set, StarSQL sends all SQL statements and parameters only as single-byte character strings defined by that CCSID. The section "Connecting an MBCS Client to an SBCS Host" on page 107 provides additional information about customizing StarSQL to support specific client and host configurations.

UseDSCRDBTBL

If the UseDSCRDBTBL setting is Yes, the default, StarSQL uses the DRDA DSCRDBTBL command to implement the SQLColumns ODBC function.

For hosts that do not support the DRDA DSCRDBTBL command, such as SQL/DS, DB2/LUW, or any host in a double-byte environment, set UseDSCRDBTBL to No. In this case, StarSQL implements SQLColumns by preparing and executing a SELECT command.

UseDynamicCatalogSQL

Set the UseDynamicCatalogSQL to Yes if you want the StarSQL driver to use dynamic rather than static SQL for catalog functions when connected to DB2/zOS servers. The default value is Yes, causing dynamic SQL to be used when it's available.

UseEncryption

The UseEncryption setting determines whether the login user ID and/or password is sent to the host encrypted or in clear text. When a user logs into the host or changes their host password using a StarSQL connection, StarSQL sends the user ID and password in clear text unless the UseEncryption keyword is set to Yes or Any. To send the user ID and/or password encrypted, you must set the UseEncryption keyword to Yes or Any and the host must support encryption.

DB2 for z/OS or later support or successfully negotiate password encryption. DB2 for i (OS/400) supports password encryption, but does not support encrypted user IDs; therefore the UseEncryption setting must be Any or No. DB2 LUW supports encrypted passwords if you enable the database for encryption as described in "Enabling Encryption" on page 51.

You can specify Any, Yes, and No for the UseEncryption setting, as described in the following table. The default value is Any.

Table 17. UseEncryption Keyword Values

UseEncryption Setting	Description
Any	StarSQL sends the user ID in clear text and the password encrypted. If the login fails, StarSQL then sends both the user ID and password in clear text.

Yes	StarSQL always sends the user ID and password to the host encrypted. If the host does not support encryption, StarSQL returns an error.
No	StarSQL always sends the user ID and password to the host in clear text.

UseJumboPackages

The UseJumboPackages variable affects:

- how dynamic SQL packages are bound, and
- how many SQL statements handles are available when using the StarSQL driver to connect to a DB2 host.

If you configure the StarSQL DSN to use jumbo packages (UseJumboPackages=Yes), StarSQL sets the maximum active statement handle limit to 1,314. Each handle can each be used by one ODBC statement at a time. If you set UseJumboPackages to No, StarSQL binds the packages with a maximum of 64 statement handles.

To take full advantage of this option you must enable the UseJumboPackages variable before you bind StarSQL packages, and it must be enabled in the StarSQL data source that is used to connect to the database.

If a dynamic package that is bound with 64 sections (UseJumboPackages=No) receives 65 or more active statements because the StarSQL data source is configured to use jumbo packages, DB2 may report an error that the sections cannot be found. If packages are bound with the jumbo option enabled (UseJumboPackages=Yes), users can connect to DB2 using a StarSQL data source that is configured to use either the typical (64 statement handles) or jumbo-sized (1,314 statement handles) packages.

UseSYSDUMMYAEU

The UseSYSDUMMYAEU keyword affects how LOB data is processed on a DB2 for z/OS host. The default setting of Yes indicates that the DB2 for z/OS system catalog contains the system tables SYSDUMMYA, SYSDUMMYE, and SYSDUMMYU, and StarSQL will use these tables instead of the standard SYSDUMMY1 table to avoid unnecessary conversion of LOB data that has a CCSID associated with it.

The SYSDUMMYA, SYSDUMMYE, and SYSDUMMYU tables are created by applying PTFs that IBM made available in APAR PQ85495. You can verify whether these tables exist on the host by entering the following statement:

SELECT * FROM SYSIBM.SYSTABLES WHERE NAME LIKE ‘SYSDUMMY%’

Note that only the SYSDUMMYA and SYSDUMMYE tables are created when applying the PTF to DB2 for z/OS v6—the SYSDUMMYU table is not created for that version. If your system has only the standard SYSDUMMY1 table and your database contains LOB data, the following error may be returned at runtime when users try to access LOB data:

```
SQLSTATE 42704 (-204) *[StarSQL][StarSQL ODBC Driver][DB2] IS
AN UNDEFINED NAME*
```

To resolve this error you can:

- set the UseSYSDUMMYAEU keyword to No, or
- apply the PTFs from IBM APAR PQ85495 (http://www-1.ibm.com/support/docview.wss?uid=swg1PQ85495) to create the new SYSDUMMY tables.

If you set the UseSYSDUMMYAEU keyword to No, any double-byte EBCDIC or ASCII GRAPHIC data will be converted to UNICODE on the host before it is sent to StarSQL. If host performance is an issue, apply the PTFs and set UseSYSDUMMYAEU to Yes to avoid the unnecessary conversion.

Appendix E

StarLicense Client Configuration Utility

You can use the StarLicense Client Configuration utility to manage which StarLicense server or servers are used to check out a license. The StarLicense Client Configuration utility allows you to:

- test that a license can be checked out
- configure which license server to use for connections from the local computer to a database
- remove a license server from the local configuration

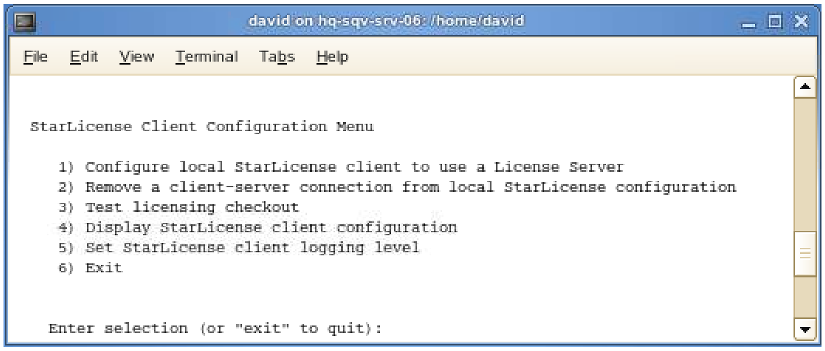
Follow the steps below to start the StarLicense Client Configuration utility.

1. Log on to the computer as `root` user.
2. Change to the $STARDIR/bin directory where the StarLicense Client Configuration utility is installed.
3. Enter the following command to execute the startup script and display the StarLicense Client Configuration Menu.

```
# ./config-lic
```

The StarLicense Client Configuration Menu appears.

Figure 4. StarLicense Client Configuration Menu


david on hq-sqv-srv-06: /home/david
File Edit View Terminal Tabs Help

```
StarLicense Client Configuration Menu

   1) Configure local StarLicense client to use a License Server
   2) Remove a client-server connection from local StarLicense configuration
   3) Test licensing checkout
   4) Display StarLicense client configuration
   5) Set StarLicense client logging level
   6) Exit


  Enter selection (or "exit" to quit):
```

Adding a Connection to a StarLicense Server

Option 1 of the StarLicense Client Configuration Menu lets you define a connection to a StarLicense server. The StarLicense Server software can be installed on a remote computer or the local computer. After you select Option 1 a screen appears so you can specify information about the StarLicense server you want the client to use for licensing.

1. Enter the hostname or TCP/IP address of the StarLicense server. (If the StarLicense Server software is installed on the local computer you can specify the loopback address 127.0.0.1 or “localhost”.)
2. Enter the port number that the server is using to listen for license requests. The default port number is 4999.
3. Enter the product ID for the license.

After you provide the StarLicense server information the StarLicense Client Configuration utility shows the client-server connection being added.

Removing a StarLicense Server Definition

Option 2 of the StarLicense Client Configuration Menu lets you remove a StarLicense server from the local computer’s configuration. After you select Option 2 a screen appears with information similar to the following:

```
Hostname=127.0.0.1 Port=4999 ProductID=SQ PRIMARY
Delete this connection (y/n)?
```

If there is more than one connection defined, respond with N until the connection that you want to remove appears and then enter Y to confirm you want to remove the connection.

Testing License Checkout

Option 3 of the StarLicense Configuration Menu lets you test that one or more licenses can be checked out. After you configure a license server it is good practice to execute this option to verify that licenses can be checked out.

After you select Option 3 a prompt appears so you can enter the number of licenses to check out. The default is 1 license, but you can enter any number. After you specify the number of licenses, the utility attempts to check out that number of licenses for the particular product ID and displays the results. Press Enter to check the license(s) back in so they will be available for other connections.

Displaying the StarLicense Configuration

Option 4 of the StarLicense Client Configuration Menu displays information about the current configuration of StarLicense. The information is categorized by Client License Servers, Client License Keys, Server License Keys, and Primary Server Entries, some of which may not be applicable to StarSQL for UNIX licensing. Configuration details include:

- the license keys that have been configured
- which StarQuest product the key is for
- when the license expires
- how many connections the license allows
- the IP address and port that listeners are configured to use
- the primary and secondary servers if more than one remote StarLicense server has been configured
- the logging level configured for the server
- the journaling level configured for the server
- the number of days that the server journal entries are retained

Setting the StarLicense Client Logging Level

Option 5 of the StarLicense Client Configuration Menu allows you to enable logging of the client license activity. Logging information may be needed by StarQuest Customer Support to help troubleshoot a problem with client licensing. If licensing is working fine there is no need to change the logging level from the default value of 0 (zero).

The valid values for the logging level are:

0	no logging
FFFFFFFF	log all license activity

A new log file is created for each process and written to the `/tmp` directory. The filename format for the log file is:

```
starliccli.<processID>.log
```

The log file contains the timestamp, a message class identifier, and a text string that describes the activity of the license service. Be sure to set the logging level back to 0 (zero) when you no longer need to capture license service activity to avoid creating unnecessary log files.

Appendix F

StarSQL National Language Support

StarSQL is a v3.0-compliant Unicode ODBC driver that provides connectivity to any database that is compliant with the Distributed Relational Database Architecture (DRDA). This includes all of IBM's DB2 family of database products running on mainframe, midrange and workstation-based platforms. StarSQL allows connecting to DB2 from all modern Microsoft Windows and most popular UNIX- and Linux-based computers.

To support a wide range of host systems, StarSQL includes conversion support for character encoding based upon Unicode, IBM EBCDIC (including Extended EBCDIC) and ANSI (ASCII) standards. In addition, StarSQL supports legacy UNIX encoding, such as Extended UNIX Coding (EUC) and the International Standards Organization (ISO) encoding.

This appendix provides background of how characters are encoded and data is converted from one system to another, and how StarSQL supports using different languages and character encoding schemes. It is intended for any reader who is interested in character conversion, but is especially pertinent for StarSQL users who are:

- running a client that is configured with a code page for one of the Group 2 languages (Japanese, Chinese, or Korean) that needs to connect to a host that does not support these languages.
- accessing a DB2 z/OS system that is supporting one of the Group 2 languages using extended EBCDIC.

The section "Customizing StarSQL for National Language Support" on page 106 specifically addresses using StarSQL in the above situations.

Character Encoding of Data Across Systems

Most legacy (pre-Unicode) client environments use the ASCII-derived character encoding that was developed by the ANSI standards body to store and manipulate character strings. ASCII encoding allowed each ASCII character to be stored as a byte, with the initial version of ASCII using only 7 of the 8 bits available in a byte. This allowed applications to use 128 different characters. ASCII was subsequently extended to support most European characters by using the eighth bit to expand the total range of characters to 256. Currently ASCII refers to either the 7-bit or 8-bit encoding of characters. The Group 2 languages require many more characters than can be encoded by a single byte. These languages typically employ a variety of extensions to 7-bit ASCII, using the high-order bit to designate the use of an additional byte to encode more characters (approximately 32K). Such encoding schemes are termed "multi-byte" character encoding and several distinct dejure standards exist, including ANSI and ISO, in addition to defacto standards such as EUC and IBM's Extended EBCDIC.

Today most Windows systems provide support for ANSI or Unicode encoding schemes, and most UNIX systems provide support for ISO or Unicode. Most IBM host systems support EBCDIC and Unicode. StarSQL is designed to support connectivity among these different platforms.

To distinguish various encoding schemes, IBM created a standard enumeration of virtually all character encoding using a 16-bit value. This unique value is called the Coded Character Set Identifier (CCSID), and it is used to tag all character data exchanges between StarSQL and the host DBMS. The CCSID is just a number that identifies a particular code page. For example, a host in the U.S. or Canada would typically use the US-English code page denoted by a CCSID of 37, whereas a host in Germany may use a code page represented by CCSID 273.

Character data in most Linux, UNIX, and Windows environments is represented only by a code page, which StarSQL treats as a CCSID. For example, the U.S. form of English using ANSI encoding is CCSID 1252.

StarSQL uses the CCSID to identify client and host encoding schemes in a variety of ways, including for Expert Settings that modify the behavior of the driver.

StarSQL and Unicode

StarSQL uses Unicode internally to parse SQL statements and as the basis for converting unlike data encoding between connected systems. This use of Unicode internally also allows StarSQL to efficiently receive Unicode from modern host systems that use Unicode as the native encoding scheme.

Determining and Setting the Client Code Page

StarSQL determines the local client's code page by a variety of system calls—the value is normally returned as part of the "locale" API support.

On a Windows-based computer the system language is specified in the Regional and Language Options of the Control Panel. The code pages that determine how non-Unicode characters are converted are represented by a numeric string, such as 1252 for US-English code page. StarSQL converts the code page number to a string that the conversion routines can process, such as WINDOWS-1252.

On a Linux- or UNIX-based computer, you specify what code page to use by setting the locale, which is specified by the **LANG** environment variable. You can modify the locale using the **LANG**, **LC-CTYPE**, or **LC_ALL** environment variables.

Determining the Host Code Page

The code pages that are specified for the host to use are detected as part of the initial exchange between StarSQL and DB2. You can determine the CCSIDs that the host is returning to StarSQL by connecting to DB2 using a StarSQL data source. After you connect to the DB2 host, look at the AutoTypDefOvr parameter in the `.odbc.ini` file to see the CCSID values that the host returned to StarSQL. Use the view command, as shown below, or any text editor, to display the contents of the initialization file.

```
> view .odbc.ini
```

The AutoTypDefOvr setting is three comma-separated values of the CCSID for SBCS, DBCS, and MBCS, respectively. For example, the following AutoTypDefOvr value indicates that the DB2 host returned CCSID 37, with no CCSID value set for DBCS or MBCS.

```
AutoTypDefOvr=37,0,0
```

To control the SBCS CCSID that StarSQL uses to send outbound data, you can modify the TypDefOvr value of the StarSQL data source as described in "Customizing StarSQL for National Language Support" on page 106.

StarSQL Implementation Details

The default behavior of the StarSQL driver typically involves the following steps when connecting to a DB2 host and exchanging SQL data.

1. StarSQL connects to DB2 and obtains the CCSID values that the host expects for SBCS, DBCS, and MBCS data. These CCSID values are stored in the AutoTypDefOvr setting of the StarSQL data source for future use.

2. StarSQL sends SQL statements and input data to DB2, converting SBCS data to the host-specified CCSID for single-byte data. Mixed-byte data is sent using the client code page, and the host is responsible for converting it to the host-specified CCSID for DBCS and MBCS data.
3. DB2 converts the data to the host server's code page, if necessary, and then processes the data.
4. DB2 sends the result back to the StarSQL client.
5. StarSQL converts the result to the code page of the client computer.

StarSQL performs inbound data conversion from the host system based upon the mappings that are defined in the ccsid.cvs table that is installed with StarSQL. The **ccsid.csv** table is platform-specific, and is installed to the `\Programs` directory of a Windows-based computer or to the `$STARDIR/etc/conf` subdirectory of a Linux- or UNIX-based computer.

In order for StarSQL to convert data from one character set to another, there must be a mapping defined in the ccsid.csv table for all the CCSIDs used in the exchange. If there is no mapping defined for a specified CCSID, an error is reported. Support for additional character conversions may be requested from StarQuest Customer Support, and you also may be able to customize the StarSQL data source to use a different CCSID that is supported as explained in the next section, "Customizing StarSQL for National Language Support."

If one or more characters cannot be converted between the host and client character sets, StarSQL substitutes a question mark, ?, in place of the character(s). You can configure whether the substitution generates a warning by setting the CharacterSubstitution parameter of the StarSQL data source (see page 86).

Customizing StarSQL for National Language Support

This section addresses situations that may require customizing StarSQL to support a specific client connecting to a specific DB2 host: StarSQL may need to be customized in order to:

- establish a connection between a client that is configured with a code page for one of the Group 2 languages (Japanese, Chinese, or Korean) and a host that does not support these languages.
- access a DB2 z/OS system that is supporting one of the Group 2 languages using extended EBCDIC.

Connecting an MBCS Client to an SBCS Host

IBM encoding of the MBCS support recognizes two distinct CCSIDs for SBCS and DBCS encoded characters. Typically only characters encoded using the SBCS component are used in SQL statements, while both DBCS and MBCS data can be the output of the SQL statements. If a host is not configured to support mixed-byte character sets it can receive only SBCS data.

Typically StarSQL sends SBCS data to the host with characters encoded according to the CCSID value that the host indicates it is using for SBCS data when it returns the AutoTypDefOvr to StarSQL. You can specify a particular CCSID to use for converting SQL data by setting the TypDefOvr parameter of the data source that StarSQL uses to connect to the host.

The TypDefOvr setting overrides the SBCS CCSID value that is specified in the AutoTypDefOvr setting. When converting SQL data to a form the host can use, StarSQL will use the conversion routine that is associated with the CCSID in the TypDefOvr setting, regardless of the client computer's character encoding scheme (SBCS or MBCS) or the AutoTypDefOvr values.

If a host reports to StarSQL that a CCSID being used by the client is not acceptable, useful connectivity may still be achieved by specifying an SBCS CCSID that is supported by the host, which restricts the SQL character data to only single-byte character strings. For example, when connecting a client computer that is using a Japanese character set to a U.S.-based host that does not support Japanese mixed-byte character set, you can force a specific SBCS CCSID to be used by setting the TypDefOvr setting of the StarSQL data source.

Supporting Asian Languages with Extended EBCDIC

Some multi-byte host environments have restrictions regarding the type of SQL phrases which may be used. Specifically, hosts that are running DB2 for z/OS with support for Group-2 languages may not handle the default SQL statements used by StarSQL to fulfill ODBC catalog calls.

When using Extended EBCDIC client encoding, StarSQL packages must be bound with the CustomizePrdid parameter enabled for "Mixed" (see page 87). Setting the CustomizPrdid parameter to M (Mixed) disables use of the backslash character, \., as an escape character in arguments passed to catalog functions. The escape character precedes a special character, such as the underscore (_) character, to treat the character as a literal. To work with tables that include an underscore in the name, such as MY_TABLE, create an ALIAS, SYNONYM, or VIEW for the table with a name that includes no special characters instead of working directly with the table.

StarSQL packages are normally bound using the StarAdmin utility that is included with the Windows version of StarSQL. Be sure that the DSN has CustomizePrdid set to Mixed before binding packages with the StarAdmin utility. The ODBC DSN that StarSQL clients use to connect to the host also must be configured with CustomizePrdid set to Mixed to use the packages that are bound with CustomizePrdid set to Mixed.

Currently Supported CCSIDs

StarSQL supports converting character data between a wide range of CCSID-to-CCSID pairs and CCSID-to-code-page pairs. It supports Group 1, Group 1A, and Group 2 character sets as defined by CDRA.

- Group 1 covers the Roman Alphabet Number 1, which includes Australia, Hong Kong, New Zealand, North and South America, and Western Europe.
- Group 1A covers multilingual scripts Cyrillic, Hebrew, Greek, and Turkish. The Latin 2 character set associated with Central Europe is supported in this group.
- Group 2 covers double-byte coding for Japan, Korea, the People's Republic of China, the Republic of China, and Thailand.

For the complete list of supported CCSIDs, refer to the "StarSQL Character Conversion and National Language Support" document in the StarSQL technical documents of the StarQuest Web site.

Glossary

APAR - Authorized Program Analysis Report A request for correction of a problem caused by a defect in a current unaltered release of a program.

APPL An APPL statement defines the DB2 subsystem to VTAM for the purposes of remote access. It is required for any configuration that involves a DB2 host on an IBM mainframe (z/OS).

authorization identifier On DB2 for z/OS, the authorization identifier is user's login ID or an assigned Authorization Identifier, which corresponds to a group with which the user is associated.

BSDS The BSDS (bootstrap data set) is a VSAM data set that contains name and status information for DB2, as well as RBA range specifications, for all active and archive log data sets, passwords for the DB2 directory and catalog, and lists of conditional restart and checkpoint records.

CCSID CCSID (Coded Character Set Identifier) represents a character set, code page, encoding scheme, and additional coding-related information. Used to support international code sets.

collection A collection is a location on the host for database objects, such as user tables and catalog tables, which are collected together under a single qualifying name.

CoS Class of Service (CoS) is the ability of switches and routers to prioritize traffic into different queues and classes.

data source A data source describes a connection to a database from an ODBC application.

DDF DDF (Distributed Data Facility) is the vehicle that DB2 uses to send and receive remote procedure calls.

DRDA DRDA (Distributed Relational database Architecture) supports access to distributed data by which an application can explicitly connect to another location, using an SQL statement, to execute packages that have been previously bound at that location.

DSN A DSN (Data Source Name) is an ODBC definition that refers to a particular database.

Dynamic SQL Dynamic SQL refers to SQL statements that are prepared and executed within an application program while the program is executing. A dynamic SQL statement can change during program execution. Contrasted with Static SQL.

held cursor A held cursor is a cursor that is not automatically closed when its transaction commits.

installable image An installable image is an image of a StarSQL installation that can be installed on multiple desktops over the network.

isolation level The isolation level refers to the degree of concurrency permitted in a transaction.

ODBC ODBC (Open Database Connectivity) is a call-level interface developed by Microsoft Corporation that allows a single application to access DBMSs from different vendors using a single interface.

Open Edition Open Edition is a component of the z/OS operating system that provides a UNIX-like environment and supports TCP/IP processing.

package A package is an object on the host containing a set of SQL statements that have been bound statically and are available for processing.

PTF Program Temporary Fix - a method used by IBM for distributing fixes quickly.

RDB Name An RDB name is a unique identifier for an RDBMS within a network.

Static SQL Static SQL refers to SQL statements in an application program that are created before the application executes. After being created, a static SQL statement does not change, although values of host variables specified by the statement might change. Contrasted with Dynamic SQL.

system catalogs The system catalogs are tables in the host database that DB2 uses to keep track of the system. On some RDBMSs, they are called the system tables.

TCP/IP TCP/IP (Transmission Control Protocol / Internet Protocol) is a commonly - used network protocol.

user id A user id is a unique identifier that enables a user to logon to a host.

VTAM VTAM (Virtual Telecommunications Access Method) provides network communications on IBM z/OS systems.

VSAM Virtual Storage Access Method - A data storage system used in IBM z/OS. VSAM was designed to organize data more efficiently and to improve access time by searching indexes instead of actual files. .

Index