# Rocket U2 Clients and APIs

## U2 ODBC Developer Guide

*Version 7.241.02*

January 2020
UCC-724102-ODBC-UG-01

# Notices

## Edition

**Publication date**: January 2020
**Book number**: UCC-724102-ODBC-UG-01
**Product version**: Version 7.241.02

## Copyright

## Trademarks

## Examples

## License agreement

# Corporate information

Rocket Software, Inc. develops enterprise infrastructure products in four key areas: storage, networks, and compliance; database servers and tools; business information and analytics; and application development, integration, and modernization.

Website: www.rocketsoftware.com

Rocket Global Headquarters
77 4th Avenue, Suite 100
Waltham, MA 02451-1468
USA

To contact Rocket Software by telephone for any reason, including obtaining pre-sales information and technical support, use one of the following telephone numbers.

| Country | Toll-free telephone number |
|---|---|
| United States | 1-855-577-4323 |
| Australia | 1-800-823-405 |
| Belgium | 0800-266-65 |
| Canada | 1-855-577-4323 |
| China | 400-120-9242 |
| France | 08-05-08-05-62 |
| Germany | 0800-180-0882 |
| Italy | 800-878-295 |
| Japan | 0800-170-5464 |
| Netherlands | 0-800-022-2961 |
| New Zealand | 0800-003210 |
| South Africa | 0-800-980-818 |
| United Kingdom | 0800-520-0439 |

## Contacting Technical Support

The Rocket Community is the primary method of obtaining support. If you have current support and maintenance agreements with Rocket Software, you can access the Rocket Community and report a problem, download an update, or read answers to FAQs. To log in to the Rocket Community or to request a Rocket Community account, go to www.rocketsoftware.com/support.

In addition to using the Rocket Community to obtain support, you can use one of the telephone numbers that are listed above or send an email to support@rocketsoftware.com.

# Contents

Notices.......... 2

Corporate information.......... 3

Part I: Getting Started.......... 8

- Introduction.......... 8
- How U2 ODBC works.......... 8
    - U2 ODBC driver.......... 10
    - U2 ODBC client applications.......... 10
        - UCI Config Editor tool and Microsoft ODBC Data Source Administration tool.......... 10
    - Client/server connections.......... 11
        - Connection and user licenses.......... 11
        - Client connections on AIX.......... 11
    - U2 ODBC driver conformance levels.......... 12
        - ODBC API conformance levels.......... 12
        - Unsupported level 2 functions.......... 12
    - ODBC SQL grammar conformance levels.......... 13
        - Supported extended ODBC SQL grammar.......... 13
        - Supported SQL data types.......... 13
    - Tested ODBC Client applications.......... 14
- Chapter 1: Installing U2 ODBC.......... 15
    - System requirements.......... 15
    - Installing U2 ODBC driver software.......... 15
    - Configuring UCI for the U2 ODBC driver.......... 16
        - UniCall interface and the uci.config file.......... 16
        - Changing the MAXFETCHCOLS parameter.......... 17
        - Changing the MAXFETCHBUFF parameter.......... 17
        - Example.......... 17
        - Changing the COLUMN_DISPLAY_LENGTH parameter.......... 18
        - Changing the UCI connection timeout.......... 18
        - Configuring IPv6 support.......... 19
    - Configuring U2 ODBC driver software.......... 19
        - Strict statement compliance options (UniVerse only).......... 20
        - U2 ODBC parameters (UniVerse only).......... 21
        - U2 ODBC parameters (UniData only).......... 21
        - Performance options (UniVerse only).......... 22
        - NLS options (UniVerse only).......... 22
    - Miscellaneous options (UniVerse only).......... 22

Part II: Administering user accounts.......... 24

- Forward-dated password.......... 24
- Chapter 2: Administering UniData accounts.......... 25
    - Making UniData accounts accessible.......... 25
        - Creating a login paragraph for U2 ODBC connections.......... 25
    - Presenting UniData data in ODBC–accessible format.......... 26
        - Data types.......... 26
        - Multivalued and multi-subvalued data.......... 27
- Chapter 3: Administering UniVerse accounts.......... 28
    - Making UniVerse files accessible to ODBC clients.......... 28
    - Removing ODBC access to UniVerse files.......... 28
    - UniVerse Server Administration menu.......... 29
        - Menu Options.......... 29

Part III: Accessing files.......... 30

Chapter 4: Making UniVerse data accessible to U2 ODBC applications.......31
 Overview.......31
  Making UniVerse accounts accessible.......31
  Making data in a UniVerse account accessible.......31
  Tables, views, and UniVerse files.......31
  Columns and fields.......32
  Presenting UniVerse data in ODBC format.......32
  Data and data types.......32
  Multivalued data.......32
 Accessing UniVerse schemas and accounts.......32
  Using ODBC qualifiers.......33
  Resolving unqualified ODBC table names.......33
 Accessing UniVerse tables, views, and files.......34
  Updating a UniVerse account for ODBC access.......34
  File privileges and permissions.......35
  Enabling access to specific files in an account.......35
  What the ODBC file access utility does.......36
  File information cache.......36
 Accessing columns and fields.......36
  ODBC-accessible columns and fields.......37
  Multivalued columns and fields.......37
  Example.......38
 Modifying UniVerse data and data definitions for ODBC.......39
  Validating tables and files for ODBC clients.......39
  SQL data types.......40
  Length of character data.......41
  Empty or unconvertible data.......42
  Empty-null mapping.......42
  Dirty data.......42
  Conversion codes.......42
 Association keys.......43
  Defining association keys.......43
 Using encryption wallets.......43
 ODBC name mapping.......44
  U2 tools available for creating ODBC-accessible files.......45
  U2 Metadata Manager.......45
  Schema API.......45
 Special considerations for use with the SQL Server Import and Export wizard.......45
Chapter 5: Making UniData data accessible to U2 ODBC applications.......46
 Mapping UniData files for access to U2 ODBC.......46
 Preparing files for U2 ODBC.......46
  U2 tools available for creating ODBC-accessible files.......46
  U2 Metadata Manager.......46
  Schema API.......47
  Creating views or subtables.......47
  Creating schema.......47
  Granting privileges.......47
  Defining stored procedures.......47
  Naming conventions.......48
  Making your non-compliant UniData table or column names accessible.......48
  Addition of system tables.......49
 Using encryption wallets.......49

Part IV: U2 ODBC.......51
Chapter 6: U2 ODBC conformance levels.......52
 API conformance levels.......52
  Core API.......52

Level 1 API\Q....52
Level 2 API....53
U2 ODBC conformance levels....53
U2 ODBC-supported API functions....53
Connecting to a data source....53
Processing an SQL statement....55
Prepared execution....55
Direct execution....55
Supporting parameters....55
Diagrams of processing SQL statements....55
Retrieving results and information about results....56
Retrieving data source and driver information....58
Error handling....58
SQL conformance levels....58
Minimum SQL grammar....58
Core SQL grammar....59
Extended SQL grammar....59
U2 ODBC SQL conformance....59
Returning data....59
U2 ODBC-supported SQL capabilities in UniData....60
Data types....60
Data definition language....60
Data manipulation language....60
Scalar functions....62
Chapter 7: UniVerse U2 ODBC Tester....63
U2 ODBC Tester....63
Using the U2 ODBC Tester....63
Testing a connection....63
Testing SQL statements....64
Testing ODBC-compliant SQL SELECT statements....64
Testing Native SQL statements....66
Sample functions....68
Viewing the subroutine sample....68
Viewing the CALL TCL sample....70
Viewing the Native SQL sample....71
Viewing the SQL SELECT sample....73
Viewing the TOXML sample....74
Viewing the UniVerse Native Syntax sample....76
Accessing the ODBC Data Source Administrator....78
Chapter 8: Tracing and logging....79
U2 ODBC log files....79
Log file names....79
Creating the U2 ODBC log file....79
Locating the U2 ODBC log file....79
Defining a trace file in the environment variables....80

Part V: Supplementary information for UniVerse....81
Chapter 9: Working with Microsoft Excel Power Query....82
Chapter 10: ODBC usage notes....83
Data types in SQL....83
Support for core ODBC SQL grammar....83
Support for extended ODBC SQL grammar....84
SELECT…FOR UPDATE statement....84
Outer joins....84
UNION clause....85
ESCAPE clause in the LIKE predicate....85
Date and time types....85

Scalar functions........................................ 85
UniVerse extensions to ODBC SQL grammar........................................88
NOWAIT keyword........................................ 89
EXPLAIN keyword........................................90
SQL usage notes and restrictions........................................90
INSERT statements........................................ 90
Parameter markers........................................90
No DDL on files........................................ 90
Tolerance of ODBC-compliant names........................................90
Column aliases........................................90
Procedures........................................ 91
Syntax........................................ 91
Parameters........................................91
Result sets........................................ 92
Behavior of SQLRowCount........................................92
Limitations........................................92
Interoperability with generic ODBC applications........................................92
Executing UniVerse SQL........................................ 93
The {NATIVE} Syntax extension........................................93
Retrying statements as UniVerse SQL........................................93
SQL-related connection options........................................94
Transaction support........................................ 94
Autocommit mode........................................94
Multiple statement handles........................................94
Manual mode........................................ 94
ODBC table types........................................95
Chapter 11: UniVerse Error Messages........................................ 96
Message descriptions........................................ 96
Messages........................................96
Chapter 12: Sample UniVerse accounts........................................116
Locating the sample UniVerse accounts........................................116
Database definitions........................................116
HS.SALES account........................................116
HS.SERVICE account........................................117
Locating the sample UniVerse accounts........................................117
Database definitions........................................117
HS.SALES account........................................118
HS.SERVICE account........................................118
Sample data........................................119
Chapter 13: Troubleshooting........................................124
Isolating a problem........................................ 124
Before reporting a problem........................................124
Getting information about your system........................................ 124
Manual communications verification........................................126
Verifying a TCP/IP connection........................................ 127
Ping........................................127
Telnet........................................127
Verifying other types of network connection........................................ 127
Connection problems........................................127
Windows server-related problems........................................ 128
UNIX server-related problems........................................ 128
Windows and UNIX server-related problems........................................128
Client-related problems........................................ 128

# Part I: Getting Started

The chapters in Part I of the manual introduce you to the U2 ODBC driver. They also describe how to install and configure the U2 ODBC driver.

# Introduction

This chapter provides an overview of how a U2 ODBC works. It also describes the U2 ODBC driver conformance levels.

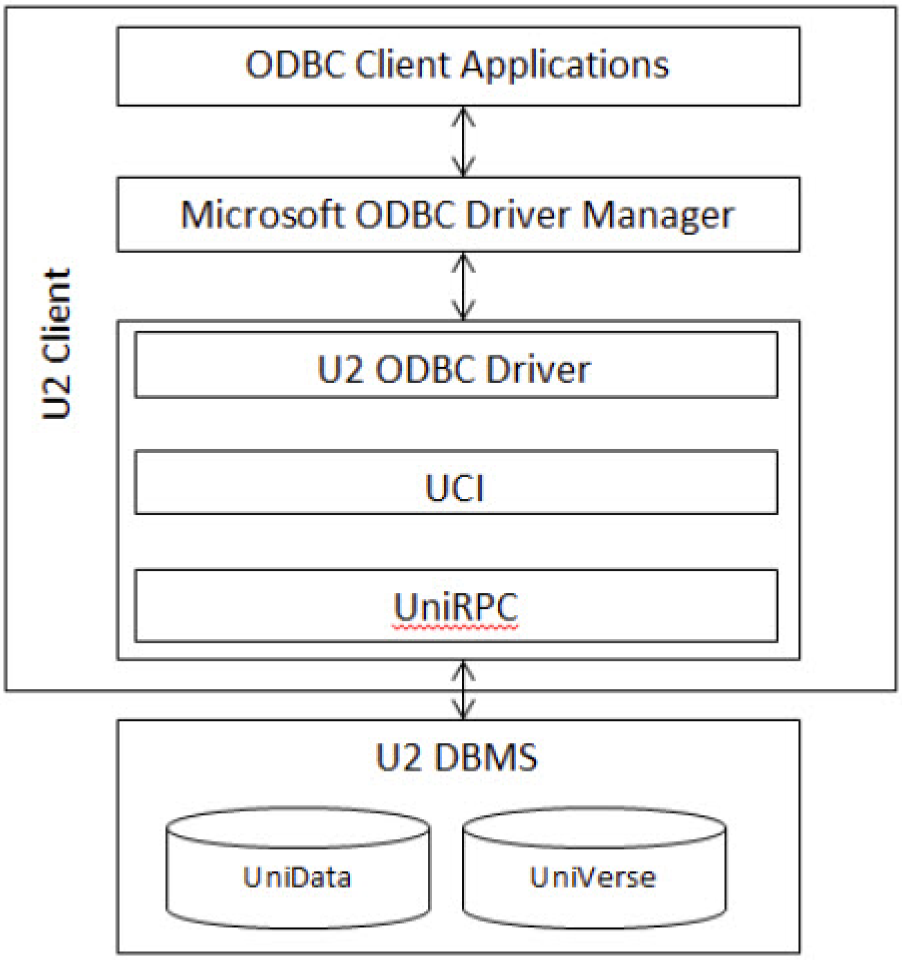
## How U2 ODBC works

The U2 ODBC driver enables ODBC applications to connect to the UniData or UniVerse (U2) database management system (DBMS).

The U2 ODBC driver is an implementation of the Microsoft ODBC Version 3.0 specification, with limitations. Some of the functionality of the ODBC 3.0 specification are not supported due to server-side restrictions.

An ODBC application sends a connection request for a data source name (DSN) definition to the U2 ODBC driver. The driver receives the request and then establishes a connection to the U2 DBMS.

The following figure shows the U2 ODBC architecture.



Refer to the following additional information regarding how U2 ODBC works:

- U2 ODBC driver
  The U2 ODBC driver requests a connection to the server through the UniCall Interface (UCI) and UniRPC.
- U2 ODBC client applications
  A U2 ODBC client application, such as a custom-written ODBC application or a commercial application (for example, Microsoft Access or SAP Business Objects), connects to the U2 ODBC driver. The U2 ODBC driver establishes connections to remote data sources and sends database requests to one or more UniData or UniVerse database servers.
- Client/server connections
  Client/server interactions require either device licenses or connection and user licenses.
- U2 ODBC driver conformance levels
  Conformance levels help application and driver developers by establishing standards of functionality.
- ODBC SQL grammar conformance levels
  The U2 ODBC driver conforms to almost all core SQL grammar.
- Tested ODBC Client applications
  The U2 ODBC Driver has been tested with Windows ODBC Client applications.

## U2 ODBC driver

The U2 ODBC driver requests a connection to the server through the UniCall Interface (UCI) and UniRPC.

**Note:** The U2 ODBC driver has a limitation on concurrent requests. The U2ODBC drivers serialize multiple concurrent requests on the ODBC client platform. If multiple queries are run in parallel, then second and subsequent queries queue until the immediately preceding query has been completed. In environments where multiple concurrent queries are run in parallel and processing time is a factor, then the U2 JDBC and ADO.NET drivers are the recommended alternative, as they do not have this restriction.

**Parent topic:** How U2 ODBC works

## U2 ODBC client applications

A U2 ODBC client application, such as a custom-written ODBC application or a commercial application (for example, Microsoft Access or SAP Business Objects), connects to the U2 ODBC driver. The U2 ODBC driver establishes connections to remote data sources and sends database requests to one or more UniData or UniVerse database servers.

Applications access ODBC data sources that are mapped to UniData or UniVerse accounts through entries in the *uci.config* file and related entries in the ODBC section of the registry. The *uci.config* file contains connection parameters necessary to route requests to the appropriate UniData or UniVerse database server.

When an application connects to a data source, it reads the *uci.config* file and related entries in the ODBC section of the registry on the client system to determine the host system, user name, and password to use when accessing a particular UniData or UniVerse database server. A U2 ODBC application accesses the UniData or UniVerse database across various operating systems. Each configuration entry in the *uci.config* file describes the physical attributes of a database in sufficient detail to perform three tasks:

- Establish communications
- Launch a UniData or UniVerse database server process
- Route query and update requests

**Parent topic:** How U2 ODBC works

## UCI Config Editor tool and Microsoft ODBC Data Source Administration tool

The UCI Config Editor lets you define and configure data sources.

When an application requests a connection to a data source, UCI uses the information in the UCI configuration file (*uci.config* or another UCI configuration file you create) to connect to the data source.

Every U2 ODBC data source definition must reference an entry in the *uci.config* file.

Use the UCI Config Editor tool to do the following:

- Create or edit a data source.

  The UCI Config Editor tool can be used to change an entry in the configuration file if the data source is relocated or the network is altered. This shields application programs from changes in the client/server environment.
- Test U2 ODBC data sources by running the Microsoft ODBC Data Source Administrator tool.

For more information about the UCI Config Editor tool, see *Administrative Supplement for Client APIs* or *Installing and Licensing UniData.*

# Client/server connections

Client/server interactions require either device licenses or connection and user licenses.

Additional information regarding client/server connections:

- Connection and user licenses
  If your organization does not use device licensing, each connection from an ODBC client application to the UniData or UniVerse database server requires one UniData or UniVerse license.
- Client connections on AIX
  When client connections are made on AIX, after the maximum retry count for entering a valid user password has been exceeded, the count of attempted accesses will not be incremented above this limit.

**Parent topic:** How U2 ODBC works

## Connection and user licenses

If your organization does not use device licensing, each connection from an ODBC client application to the UniData or UniVerse database server requires one UniData or UniVerse license.

If your application makes multiple simultaneous connections to UniData or UniVerse, each one uses one UniData or UniVerse license.

For information about device licensing, see *Administering UniVerse* or *Installing and Licensing UniData*.

**Parent topic:** Client/server connections

## Client connections on AIX

When client connections are made on AIX, after the maximum retry count for entering a valid user password has been exceeded, the count of attempted accesses will not be incremented above this limit.

This is an internal restriction in the U2 ODBC and UniObjects implementations and affects AIX platforms only. Use of the user ID will still be blocked by the usual operating system policies and is unaffected by this limitation.

**Parent topic:** Client/server connections

## U2 ODBC driver conformance levels

Conformance levels help application and driver developers by establishing standards of functionality.

This section describes how U2 ODBC conforms to the *Microsoft ODBC 3.0 Programmer's Reference and SDK Guide*. ODBC defines conformance levels for an ODBC driver in two areas:

- ODBC API
- ODBC SQL grammar, including ODBC SQL data types

The following table summarizes ODBC compliance levels for the U2 ODBC driver. For more information about ODBC conformance levels, see *Microsoft ODBC 3.0 Programmer's Reference and SDK Guide*.

| Driver | ODBC API | ODBC SQL Grammar |
|---|---|---|
| U2 ODBC | Core, Level 1, and Level 2 features as described in ODBC API conformance levels, on page 12. | Minimum, core, and extended features described in ODBC SQL grammar conformance levels, on page 13. |

The U2 ODBC driver also provides connection and statement options to control additional option values specific to U2 ODBC.

**Parent topic:** How U2 ODBC works

## ODBC API conformance levels

The U2 ODBC driver conforms to the ODBC Core API and Level 1 API functionality. In addition, the U2 ODBC driver supports the following Level 2 functions:

- SQLExtendedFetch
- SQLPrimaryKeys
- SQLMoreResults
- SQLNativeSql*
- SQLNumParams

**Note:** The SQLNative Sql function is supported in UniVerse only.

## Unsupported level 2 functions

The U2 ODBC driver does not support the following Level 2 functions:

- SQLColumnPrivileges
- SQLDescribeParam
- SQLParamOptions
- SQLProcedureColumns
- SQLProcedures
- SQLSetPos

- SQLSetScrollOptions
- SQLTablePrivileges

## ODBC SQL grammar conformance levels

The U2 ODBC driver conforms to almost all core SQL grammar.

For a list of exceptions, see Support for core ODBC SQL grammar, on page 83.

For detailed information about ODBC SQL grammar conformance in U2 ODBC, see Support for extended ODBC SQL grammar, on page 84.

**Parent topic:** How U2 ODBC works

### Supported extended ODBC SQL grammar

The U2 ODBC driver supports some of the extended ODBC SQL grammar features. These include:

- Outer joins
- UNION clause
- *SELECT…FOR UPDATE statement
- Scalar functions such as SUBSTRING and CONCAT
- DATE and TIME data types
- Procedure calls

**Note:** The SELECT...FOR UPDATE function is supported in UniVerse only.

### Supported SQL data types

Supported SQL data typesU2 ODBC supports the following SQL data types:

- CHAR
- DATE
- *DECIMAL
- DOUBLE PRECISION
- FLOAT
- INTEGER
- *NUMERIC
- REAL
- SMALLINT
- TIME
- VARCHAR

**Note:** The DECIMAL and NUMERIC functions are supported in UniVerse only.

## Tested ODBC Client applications

The U2 ODBC Driver has been tested with Windows ODBC Client applications.

U2 ODBC is certified to be used with the following:

- 32-bit and 64-bit variants
- Microsoft Office 2013 and 2016
- Microsoft SQL Server 2012, 2014, and 2016
- SAP Business Objects 3.1XI (now obsoleted by SAP, but still in use), 4.0, 4.1, and 4.2
- Visual Studio 2010, 2012, 2013, 2015, and 2017

**Parent topic:** How U2 ODBC works

# Chapter 1: Installing U2 ODBC

This chapter describes how to install and configure U2 ODBC driver software.

Ensure that the driver software is installed on a Windows platform.

Review the following sections for details regarding U2 ODBC installation and configuration:

- System requirements
  System requirements for U2 ODBC include supported UniVerse and UniData version information.
- Installing U2 ODBC driver software
  Complete the following steps to install U2 ODBC on Windows:
- Configuring UCI for the U2 ODBC driver
- Configuring U2 ODBC driver software
  You must run the Microsoft ODBC Data Source Administrator tool to define a configuration.
- Miscellaneous options (UniVerse only)

## System requirements

System requirements for U2 ODBC include supported UniVerse and UniData version information.

- Microsoft Windows 8.1 (or later)
- UniVerse version 11.1 or later
- UniData version 7.2 or later

**Parent topic:** Installing U2 ODBC

## Installing U2 ODBC driver software

Complete the following steps to install U2 ODBC on Windows:

1. From the Client installation screen, select the version of U2 ODBC that you want to install.
   - U2ODBC Client 32-bit
   - U2ODBC Client 64-bit
2. After the installation wizard opens, click **Next** to continue with the installation process.
3. After accepting the licensing agreement, click **Next**.
4. Click **Next** to install U2 ODBC in the default folder, or click **Browse** to search for a different folder. By default, the installation process installs the driver in one of the following directories:
   - C:\U2\UniDK\U2ODBC_32bit
   - C:\U2\UniDK\U2ODBC_64bit
5. Follow the remaining prompts to complete the installation process. Click **Close** to exit.

**Note:** The SSL Configuration Editor tool elevates the program privilege to Administrator mode on UAC active machine.

**Parent topic:** Installing U2 ODBC

## Configuring UCI for the U2 ODBC driver

Before you install and configure U2 ODBC driver software, you must configure the UniCall Interface (UCI) on the U2 ODBC client.

For more information about the UCI Config Editor tool, see *Administrative Supplement for Client APIs*.

Review the following sections for details regarding UCI configuration:

- UniCall interface and the uci.config file
- Changing the MAXFETCHCOLS parameter
  The default MAXFETCHCOLS setting is 400 column values.
- Changing the MAXFETCHBUFF parameter
- Changing the COLUMN_DISPLAY_LENGTH parameter
  The default COLUMN_DISPLAY_LENGTH setting is 254 characters. Increase this setting if you want more than 254 characters to display in a column.
- Changing the UCI connection timeout
- Configuring IPv6 support
  For IPv6 support, the uci.config file needs to be configured as shown:

**Parent topic:** Installing U2 ODBC

## UniCall interface and the uci.config file

The U2 ODBC driver uses UCI to access UniData or UniVerse data. The UCI interface uses the uci.config file to communicate to the UniData or UniVerse database server.

The following is an example of a uci.config file:

```
[ODBC DATA SOURCES]
<localuv> DBMSTYPE = UNIVERSE
network = TCP/IP
service = uvserver
host = localhost

<localud>
DBMSTYPE = UNIDATA
network = TCP/IP
service = udserver
host = localhost
```

To access a remote database on a different platform, you must add an entry to the configuration file.

For example, if the remote system you want to access is named hq1 and the account path is /usr/myacct, make up a data source name, such as corp1, and add the data source definition to the uci.config file. The following is an example for UniData:

```
<corp1>
DBMSTYPE = UNIDATA
NETWORK = TCP/IP
SERVICE = udserver
HOST = hq1
ACCOUNT = /usr/myacct
USERNAME = myloginname
```

The ACCOUNT parameter can be set to any one of the following:

- The full path to a UniData or UniVerse account.
- A valid UniData or UniVerse database name.

**Warning:** Entries in the `uci.config` file are order dependent. Changing the order will cause an error.

A UniData database name is valid if it appears as an entry in the ud_database file:

- For UNIX systems, this file is located in /usr/ud7x/include.
- For Windows platforms, it is located in `\udthome\include`.

To access your data from U2 ODBC, you might need to increase the values of two configuration parameters for the UNIDATA or UNIVERSE DBMS type. These parameters are defined in the `uci.config` file. You might also need to increase the UCI connection timeout.

**Parent topic:** Configuring UCI for the U2 ODBC driver

## Changing the MAXFETCHCOLS parameter

The default MAXFETCHCOLS setting is 400 column values.

Increase the value of the MAXFETCHCOLS configuration parameter if either of the following conditions applies:

- Any of your UniVerse files, tables, or views have more than 400 fields defined in the dictionary (field definitions include D-, A-, I-, and S-descriptors)
- Any queries executed from ODBC client applications through U2 ODBC contain more than 400 columns in the result set

**Parent topic:** Configuring UCI for the U2 ODBC driver

## Changing the MAXFETCHBUFF parameter

The default MAXFETCHBUFF setting is 8192 bytes. Increase the value of the MAXFETCHBUFF configuration parameter if either of the following conditions applies:

- The record length of any table or file exceeds 8192 bytes
- Any queries executed through U2 ODBC yield a result set with rows longer than 8192 bytes

**Parent topic:** Configuring UCI for the U2 ODBC driver

### Example

The following UniVerse UCI configuration file definition sets MAXFETCHCOLS to 1000 and MAXFETCHBUFF to 32000 for all UNIVERSE data sources:

```
<localuv>
DBMSTYPE = UNIVERSE
network = TCP/IP
service = uvserver
host = localhost
```

```
 [UNIVERSE]
 MAXFETCHCOLS = 1000
 MAXFETCHBUFF = 32000
```

If any query you try to execute has more than MAXFETCHCOLS result columns or has a result row length greater than MAXFETCHBUFF bytes, the query fails with a UCI error code of 930122:

```
 [U2][SQL Client][UNIVERSE]UniVerse/SQL: Row length
 exceeds buffer size
```

In UniData, the same example would read:

```
 <localud>
 DBMSTYPE = UNIDATA
 network = TCP/IP
 service = udserver
 host = localhost
 [UNIDATA]
 MAXFETCHCOLS = 1000
 MAXFETCHBUFF = 32000
```

## Changing the COLUMN_DISPLAY_LENGTH parameter

The default COLUMN_DISPLAY_LENGTH setting is 254 characters. Increase this setting if you want more than 254 characters to display in a column.

For example:

```
COLUMN_DISPLAY_LENGTH=350
```

**Parent topic:** Configuring UCI for the U2 ODBC driver

## Changing the UCI connection timeout

The one-hour default inactivity timeout value for UCI connections can be too short. If users leave client ODBC connections open but inactive for longer than one hour, a timeout error can occur. To increase this timeout value, log on as an Administrator and edit the unirpcservices file in the unishared/unirpc directory. In the line starting with uvserver or udserver, change the rightmost number to 86400 (the number of "seconds" in 24 hours).

On UniVerse, the line should appear as follows:

```
 uvserver <uvhome>/bin/uvsrvd * TCP/IP 86400
```

where <uvhome> is the path of the UV account directory (for example, /usr/uv on UNIX systems or \U2\UV on Windows platforms).

On UniData, the line should appear as follows:

```
 udserver <uvdhome>/bin/udsrvd * TCP/IP 86400
```

where <udhome> is the path of the UD account directory (for example, /usr/ud on UNIX systems or \U2\UD on Windows platforms).

**Note:** To edit the unirpcservices file, you must have root privileges on UNIX or Administrator privileges on Windows platforms.

**Parent topic:** Configuring UCI for the U2 ODBC driver

## Configuring IPv6 support

For IPv6 support, the uci.config file needs to be configured as shown:

```
<u2server_uv>
DBMSTYPE = UNIVERSE
network = TCP/IP
service = uvserver
host = server_ipv6_ipv4
IPVERSION = IPV6_IPV4
```

The IPV options are:

- IPV4
- IPV6
- IPV4_IPV6
- IPV6_IPV4
- IPVANY

**Parent topic:** Configuring UCI for the U2 ODBC driver

# Configuring U2 ODBC driver software

You must run the Microsoft ODBC Data Source Administrator tool to define a configuration.

### Prerequisites

Before you can configure the U2 ODBC driver, you need to determine which type of application you will be running: 32-bit or 64-bit.

### About this task

A U2 ODBC configuration defines the following:

- Server identification (system name and system type)
- Server authorization (user name and password)
- UniData account to access using U2 ODBC
- UniVerse account to access using U2 ODBC

Complete the following steps to configure a data source:

### Procedure

1. To access the Microsoft ODBC Data Source Administrator tool, choose **Start > Control Panel > Administrative Tools > Data Sources (ODBC)**. The ODBC Data Source Administrator dialog box opens.

   Because both the 32-bit U2 ODBC driver and the 64-bit U2 ODBC driver can be installed on the same system, the Microsoft ODBC Data Source Administrator can be accessed differently in different situations.

   - In general, if the 32-bit driver is installed on a 32-bit machine, the Microsoft ODBC Data Source Administrator can be accessed directly from the Windows control panel, as described.
   - If the 64-bit driver is installed on a 64-bit system and will be connecting to a 64-bit application, the Microsoft ODBC Data Source Administrator can be accessed directly from the Windows control panel, as described.
   - If the 32-bit driver is installed on a 64-bit machine, the Microsoft Data Source Administrator cannot be accessed from the Windows control panel. Instead, access the tool by running the odbcad32.exe executable file, which is located in the `c:\Windows\SysWOW64` directory.

     For 64-bit applications, the odbcad32.exe file is located in the c:\Windows\System32 directory.
2. Click **Add** to add a new data source.
3. In the Create New Data Source dialog box, scroll the Name list and select the correct U2 ODBC driver.
   - For 32-bit applications, select U2 ODBC Driver
   - For 64-bit applications, select U2 64-Bit ODBC Driver.
4. Click **Finish**.
5. In the **ODBC Data Source Name** box, enter the data source name.

   ODBC applications use this name to connect to the data source.
6. In the **Description** box, you can enter a description of the data source if preferred.
7. In the **UCI Data Source** field, select the UCI data source to which you want to connect.

   Choose the UCI data source that you configured in the uci.config file.
8. In the **Database** box, enter the full path to the database to which you want to connect.

   For example, `c:\U2\uv\HS.SALES` or `c:\U2\ud\demo`.
9. In the **User** box, enter the user name.
10. In the **Password** box, enter the password for the user.
11. Click **Test Connection** to ensure U2 ODBC accepts the connection to the server and database you defined.
12. Click **Options** to define optional configuration information.

    **Note:** The optional configuration information is for UniVerse only.

**Parent topic:** Installing U2 ODBC

## Strict statement compliance options (UniVerse only)

UniVerse users can select the following Strict Statement Compliance Options:

- Enforce ODBC Dates/Times – If you select the **Enforce ODBC Dates/Times** check box, U2 ODBC checks all date and time literals for strict adherence to ODBC specifications. You must specify years with four digits, and months, days, hours, minutes, and seconds with two digits.

If you do not select the **Enforce ODBC Dates/Times** check box, U2 ODBC accepts two-digit years, and single-digit months, days, hours, minutes, and seconds, and passes these shorter values to UniVerse.

- Enforce ODBC Function Argument Types – If you select the **Enforce ODBC Function Argument Types** check box, U2 ODBC type-checks arguments to scalar functions, where possible.

  If you do not select the **Enforce ODBC Function Argument Types** check box, U2 ODBC passes all arguments to scalar functions to UniVerse without performing any type checking.
- Reject non-ODBC SQL Syntax – If you select the **Reject non-ODBC SQL Syntax** check box, U2 ODBC reports an error if it encounters an SQL construct that is not in the ODBC SQL grammar. Select this option if you are writing an application that needs to be portable to other ODBC drivers or databases, or both.

  If you do not select the **Reject non-ODBC SQL Syntax** check box, U2 ODBC accepts certain non-ODBC SQL constructs.
- Reject Constructs Unsupported by UniVerse – If you select the **Reject Constructs Unsupported by UniVerse** check box, U2 ODBC returns an error if it encounters an SQL construct that is part of the ODBC SQL grammar but which UniVerse does not support.

  If you do not select the **Reject Constructs Unsupported by UniVerse** check box, U2 ODBC passes unsupported constructs to UniVerse, which will then issue its own error message.

## U2 ODBC parameters (UniVerse only)

ODBC parameters can no longer be specified on the server.

- UniVerse Name Mapping – Name mapping controls whether U2 ODBC maps the names of tables, columns, and indexes to ODBC-compliant equivalents. If you select **Yes**, name mapping is enabled. If you select **No**, name mapping is disabled.

  **Note:** The HS_NAME_MAPPING environment variable has no effect at UniVerse 10.2.

- Retry as UniVerse SQL – When you send an SQL statement to U2 ODBC, it tries to interpret the statement as if it were written in the ODBC dialect of SQL. If you do not select Retry as UniVerse SQL, U2 ODBC considers only mapped names to be valid in this dialect. If the interpretation fails, U2 ODBC acts according to the Retry as UniVerse SQL option. If you select **Yes**, U2 ODBC assumes that the statement was written in the UniVerse dialect of SQL, and passes it through to UniVerse without modification. It does not apply name mapping nor file access control to the statement.

  **Note:** The HS_RETRY_AS_UV_SQL environment variable has no effect at UniVerse 10.2.

## U2 ODBC parameters (UniData only)

ODBC parameters can no longer be specified on the server.

SQL Column Name – column headings can show the field description or column name. The USE_SQL_COLUMN_NAME parameter in the uci.config file toggles this setting.

By setting the parameter to **0** (the default), U2 ODBC returns the table column header. A setting of **1** returns the table column name.

### Performance options (UniVerse only)

UniVerse users can select any of the following performance options:

- Prefetch Size – This field specifies the number of rows of a result set that are fetched (or prefetched) from the database when one or more unfetched result set rows are requested. Each block of result set rows fetched from the database when result set rows are needed is called a result set chunk. Each result set chunk contains Size rows of the result set. Valid values for this field range from 1 to 32,767.
- Threshold – This field specifies the number of result set rows that must be consumed (via calls to SQLFetch) from the current result set chunk before the next result set chunk is prefetched from the server. The Threshold must be less than or equal to the Size. If a threshold that is greater than the size is entered, U2 ODBC internally adjusts the threshold to be equal to the size when it reads in the values specified. Valid values for this field range from 0 to 32,767.
- Max LONG Type Length – This field specifies the maximum number of bytes of data per data item that are fetched from the database for any SQL_LONGVARCHAR or SQL_LONGVARBINARY type columns. This option is ignored as U2 ODBC currently does not support the SQL_LONGVARCHAR and SQL_LONGVARBINARY data types.
- Fast Connect (Old OTL) – This option applies only to UniVerse files that are not tables. When you select this option, the ODBC table list is not refreshed in memory, and the File Information Cache is read from disk and used instead. This option may result in faster connect times, but possibly in a less up-to-date list of available tables.

  **Note:** The HS_USE_FILEINFO environment variable has no effect at UniVerse 10.2.

- Refresh OTL on Connect – This option applies only to UniVerse files that are not tables. When you select this option, the ODBC table list is queried. This operation may impact initial connect time. The File Information Cache is not updated by the operation. Update the cache by running HS.UPDATE.FILEINFO in the UniVerse account you are accessing on the server system.

### NLS options (UniVerse only)

You can define any of the following NLS options:

- NLS Mapname – The name of the map that defines how a character set is mapped between the internal and external character sets.
- Locale Name – The name of a specific set of conventions in various categories that represent the language, character set, and data formatting conventions used by a group of people.

## Miscellaneous options (UniVerse only)

- Save Password - Select this option if you want your password to be saved.A code page value that corresponds to the preferred encoding.
- First Normal Form – If this option is selected, U2 ODBC always presents multivalued data to ODBC applications in first normal form.

  If this option is not selected, you can use an SQL UNNEST keyword. For example:

  ```
  SELECT SURNAME,FORENAME,TITLE_ID,QTY,PRICE FROM UNNEST
  BOOK_SALES ON SALE_ITEMS;
  ```

- Create Debug Log – Select the **Create Debug Log** check box if you want to create a debug log.

  After selecting this option, you must define a trace file in the environment variables. For more information, see Defining a trace file in the environment variables, on page 80.
- Use Cache Info – This option is selected by default to create the table schema using the cached information found in the .hs_fileinfo file.

  Users with large accounts are encouraged to keep this option to reduce the time required to create the .hs_fileinfo file.
- Use U2 ODBC as 2.0 – Select this option to run U2 ODBC version 2.0.

  By default, ODBC runs in version 3.0. Sometimes, the SQL syntax generated by Crystal Reports or Microsoft SQL Server 2008 is not recognized by the UniVerse SQL engine. In those instances, users running Crystal Reports or Microsoft SQL Server 2008 will need to use U2 ODBC 2.0.
- Config File Name – Displays the full path of the uci.config file, defined in the Registry in read-only mode.
- Code Page – The code page value that corresponds to the preferred encoding.

  For example, the value for US English is 1252 and the value for Western Europe is 850.

**Parent topic:** Installing U2 ODBC

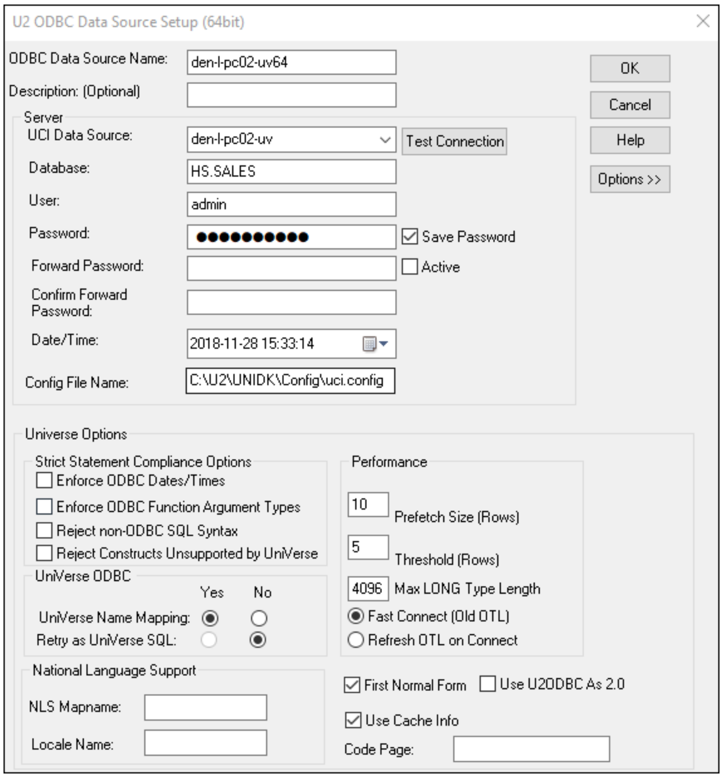
# Part II: Administering user accounts

The sections in Part II describe how to administer UniData and UniVerse user pre-set passwords.

# Forward-dated password

If you have deployed a service that requires a password change at set timed intervals and you have not logged in after the password has been changed, your service will be down until you do so. This option allows you to assign a new password before the change is initiated so that your service continues to run after the old password expires.

The administrator is alerted of a pending password change requirement and they see an option to set a forward-dated password with a date and time.



U2 ODBC Data Source Setup (64bit)

| | |
|---|---|
| ODBC Data Source Name: | den-l-pc02-uv64 |
| Description: (Optional) | |

Server

| | |
|---|---|
| UCI Data Source: | den-l-pc02-uv |
| Database: | HS.SALES |
| User: | admin |
| Password: | ●●●●●●●●●●● ☑ Save Password |
| Forward Password: | ☐ Active |
| Confirm Forward Password: | |
| Date/Time: | 2018-11-28 15:33:14 |
| Config File Name: | C:\U2\UNIDK\Config\uci.config |

Universe Options

Strict Statement Compliance Options
- ☐ Enforce ODBC Dates/Times
- ☐ Enforce ODBC Function Argument Types
- ☐ Reject non-ODBC SQL Syntax
- ☐ Reject Constructs Unsupported by UniVerse

UniVerse ODBC

| | Yes | No |
|---|---|---|
| UniVerse Name Mapping: | ● | ○ |
| Retry as UniVerse SQL: | ○ | ● |

Performance
- 10 Prefetch Size (Rows)
- 5 Threshold (Rows)
- 4096 Max LONG Type Length
- ● Fast Connect (Old OTL)
- ○ Refresh OTL on Connect

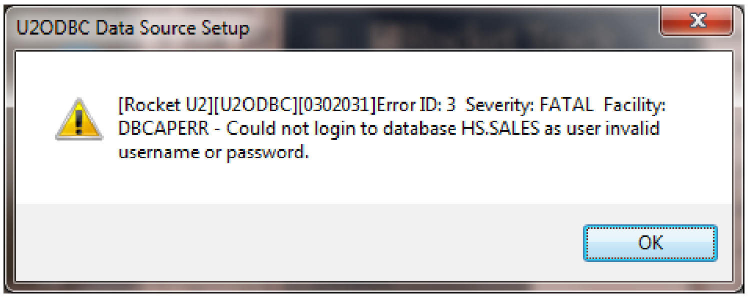
National Language Support
- NLS Mapname:
- Locale Name:

☑ First Normal Form ☐ Use U2ODBC As 2.0

☑ Use Cache Info

Code Page:

The user is made aware of this password change when they receive the following message:



U2ODBC Data Source Setup

[Rocket U2][U2ODBC][0302031]Error ID: 3 Severity: FATAL Facility: DBCAPERR - Could not login to database HS.SALES as user invalid username or password.

# Chapter 2: Administering UniData accounts

This chapter describes the following administrative tasks:

- Making UniData accounts accessible
- Creating a login paragraph for U2 ODBC connections
- Tracing events in UniData
- Presenting UniData data in an ODBC-accessible format

## Making UniData accounts accessible

UniData databases are organized into accounts. U2 ODBC connects to a UniData account and can access the files there. You optionally can define the account as a database in the ud_database file on the server machine. You also can include the account path or database name in the UCI data source definition in the UCI configuration file (uci.config). For information about setting up the UCI configuration file, see Configuring UCI for the U2 ODBC driver, on page 16.

**Note:** You also can specify the account path or database name each time you attempt to connect to the account. In this case, you would not need to include the account path or database name in the UCI configuration file. When you attempt to connect, you are prompted to specify the full path to the account or the database name.

If you want to access an account that has a UDTHOME directory different than the default UDTHOME directory, you must include a definition for that account in the ud_database file on the server machine. For UNIX systems, this file is located in /usr/ud7x/include. For Windows platforms, it is located in *udthome*\include. You can find the path for *udthome* by looking in the system registry under \HKEY_LOCAL_MACHINE\SOFTWARE\Rocket Software\UniData\7.x. Use any text editor to modify the ud_database file.

**Note:** To determine your default UniData home directory, use the UNIX env command.

The following Windows example shows an entry in the ud_database file for a database named UniData:

```
DATABASE=UniData
UDTHOME=d:\disk2\test7x
UDTACCT=d:\disk2\test7x\testacct.
```

In the ud_database file entry, the UDTHOME parameter is optional. You should include it only when the UDTHOME directory is different than the default UDTHOME directory.

## Creating a login paragraph for U2 ODBC connections

ODBCLOGIN is a UniBasic subroutine you can create to initialize the UniData environment for U2 ODBC connections. You must catalog the subroutine.

**Note:** When a connection is made through ODBC, the standard LOGIN paragraph for an account is not executed. You must create an ODBCLOGIN subroutine to initialize environments when accessing through U2 ODBC.

The syntax for ODBCLOGIN is:

```
SUBROUTINE ODBCLOGIN(RTNVAL,USERNAME)
```

When a U2 ODBC connection is made, UniData attempts to execute the ODBCLOGIN paragraph during the verification phase of a connection, in the database you specify in the connection information.

**Note:** If there is no cataloged ODBCLOGIN, the connection is allowed.

The following table describes the parameters of the subroutine.

| Parameter | Description |
|---|---|
| RTNVAL | If RTNVAL is a nonzero value, the connection is allowed. If it is zero, the connection is disallowed. |
| USERNAME | The user name that is being used to establish the connection. |

**Tip:** You can use ODBCLOGIN to define COMMON variables and other environment settings for use during a U2 ODBC connection.

In the following example, the ODBCLOGIN subroutine returns zero and does not allow a connection unless the user name is “root.”

```
SUBROUTINE OEDBCLOGIN(RTNVAL,USERNAME)
IF USERNAME=”root” THEN
RTNVAL = 1
END ELSE
RTNVAL = 0
END
RETURN
```

For more information about login paragraphs, refer to the *Administering UniData on UNIX* or *Administering UniData on Windows* manuals.

## Presenting UniData data in ODBC–accessible format

Data in UniData is organized differently from the way U2 ODBC expects it to be organized. Two areas in which the data differs from what U2 ODBC expects are:

- Data types
- Multivalued and multi-subvalued data

### Data types

UniData does not define data types for data contained in its files. On the other hand, U2 ODBC expects data types for all data. In addition, data in UniData can be of variable length, but U2 ODBC expects data to have either a fixed or a maximum length. To make the data look more like what U2 ODBC expects, you must use either the U2 Metadata Manager, the Visual Schema Generator (VSG) or the Schema API.

**Note:** While VSG is still supported, users are encouraged to use the U2 Metadata Manager tool rather than VSG. For more information, see the U2 Metadata Manager Help. Alternatively, you can use VSG or the Schema API. For more information, see *Using VSG and the Schema API*.

## Multivalued and multi-subvalued data

Multivalued and multi-subvalued dataU2 ODBC expects data to be organized in first normal form (1NF) format. Although some files could be in 1NF format, which means that only one value is stored in each column of each row, many UniData files have columns that store multiple values in the columns of a row (NF2 format). To instruct UniData SQL to present data in 1NF format, you must use the U2 Metadata Manager. For more information, see the *U2 Metadata Manager Help*. Alternatively, you can use VSG or the Schema API. For more information, see *Using VSG and the Schema API*.

# Chapter 3: Administering UniVerse accounts

This chapter describes the following administrative tasks:

- Making UniVerse files accessible to ODBC clients
- Removing ODBC access to UniVerse files
- Using the UniVerse Server Administration MenuMaking UniData accounts accessible

## Making UniVerse files accessible to ODBC clients

UniVerse tables and views are always accessible to ODBC clients. To use UniVerse files that are not tables, however, you must make them accessible—that is, visible—to ODBC clients. Complete the following steps to make such files visible to ODBC clients:

1. Log on as UniVerse Administrator.
2. Change directory to the U2 ODBC administration account directory (HS.ADMIN). For example, if UniVerse is installed in */usr/uv*, enter:

   ```
   # cd/usr/uv/HS.ADMIN
   ```
3. Invoke UniVerse and enter HS.ADMIN to access the UniVerse Server Administration menu.
4. Enter 3 to choose Activate Access to Files in an Account from the UniVerse Server Administration menu. For information about UniVerse Server Administration menu options, see*Administering UniVerse.*
5. Choose one of the following options:
   - Enter the full path of the UniVerse account directory (on Windows platforms, start with the drive letter, for example D:).
   - Enter the UniVerse account name as listed in the UV.ACCOUNT file.
   - Press Enter to see a list of UniVerse accounts in which file access has already been activated.
6. Repeat steps 4 and 5 for all accounts whose files you want to make accessible to U2 ODBC.
7. When finished, press Enter to exit the UniVerse Server Administration menu.

## Removing ODBC access to UniVerse files

To remove U2 ODBC access to UniVerse files that are not tables:

1. Log on as UniVerse Administrator.
2. Change directory to the U2 ODBC administration account directory (HS.ADMIN). For example, if UniVerse is installed in */usr/uv*, enter:

   ```
   # cd/usr/uv/HS.ADMIN
   ```
3. Invoke UniVerse and enter HS.ADMIN to display the UniVerse Server Administration menu.
4. From the UniVerse Server Administration menu, enter **4** to choose Deactivate Access to Files in an Account. UniVerse prompts for the name of the account whose files you want to deactivate.
5. Choose one of the following options:
   - Enter the full path of the UniVerse account directory (on Windows platforms, start with the drive letter, for example D:).
   - Enter the UniVerse account name as listed in the UV.ACCOUNT file.
   - Press Enter to see a list of UniVerse accounts in which file access has already been activated.

When you enter an account name, the U2 ODBC-specific VOC entries and the file information cache are removed, and the HS_FILE_ACCESS file is deleted.

**Note:** This does not modify files or their dictionaries in the account, nor does it delete the account.

6. Repeat steps 4 and 5 for all accounts whose files are accessible to U2 ODBC.
7. When finished, press **Enter** to exit the UniVerse Server Administration menu.

## UniVerse Server Administration menu

Use the UniVerse Server Administration menu to perform U2 ODBC system administration tasks. Complete the following steps to access the menu:

1. Log on as UniVerse Administrator.
2. Change directory to the U2 ODBC administration account directory (HS.ADMIN). For example, enter:
   ```
   # cd /usr/uv/HS.ADMIN (UNIX)# cd \U2\UV\HS.ADMIN (Windows NT)
   ```
3. Invoke UniVerse and enter HS.ADMIN.
4. To exit the menu, press **Enter**.

### Menu Options

The UniVerse Server Administration menu includes the options shown in the following table.

| Option | Description |
|---|---|
| 1. List Activated Accounts | List UniVerse accounts whose files are accessible to U2 ODBC clients. |
| 2. Show U2 ODBC Config Configuration for an Account | Display configuration settings that clients need to access the account. This option invokes HS.SHOW.CONFIG. |
| 3. Activate Access to Files in an Account | Make UniVerse files in an account accessible to ODBC clients. This option runs the U2 ODBC file access utility, which does the following:<br>Creates the HS_FILE_ACCESS file<br>Creates the server's file information cache<br>Updates the UV.ACCOUNT file |
| 4. Deactivate Access to Files in an Account | Remove access to UniVerse files in an account. This option deletes the server file information cache and the HS_FILE_ACCESS file, and updates the UV.ACCOUNT file. |
| 5. Run HS.SCRUB on a File/Table | Detect, report, and optionally correct data and dictionary entries that can cause ODBC access problems. This option invokes HS.SCRUB. |
| 6. Update File Information Cache in an Account | Rebuild the file information cache. This option invokes HS.UPDATE.FILEINFO. |

# Part III: Accessing files

The chapter in Part III describe how to make UniData and UniVerse files accessible to U2 ODBC.

# Chapter 4: Making UniVerse data accessible to U2 ODBC applications

This chapter describes how to make data in UniVerse tables and files accessible to ODBC client programs.

You need to do three things:

- Make UniVerse accounts accessible to ODBC applications.
- Specify what data in those accounts ODBC applications can access.
- Present the data in ODBC format.

## Overview

This section gives a general overview of what you need to do to make UniVerse data accessible to ODBC applications. Subsequent sections describe each major step in detail.

### Making UniVerse accounts accessible

ODBC applications connect to ODBC data sources. A U2 ODBC data source is defined as one UniVerse schema or account, although there are ways to make more than one schema or account accessible through a single U2 ODBC connection. For details about how to make UniVerse schemas and accounts accessible, see Accessing UniVerse schemas and accounts, on page 32.

### Making data in a UniVerse account accessible

The following UniVerse data can be made accessible to ODBC applications:

- Tables, views, and UniVerse files
- Columns and fields in tables, views, and UniVerse files
- Saved select lists associated with tables, views, and UniVerse files

#### Tables, views, and UniVerse files

UniVerse tables and views are always accessible to ODBC applications, but UniVerse files that are not tables are not. To make UniVerse files accessible to ODBC applications, you must run the ODBC file access utility in the account. Among other things, this utility creates the file information cache and HS_FILE_ACCESS. The file information cache lists all UniVerse files referenced by F- and Q-pointers in the VOC file. HS_FILE_ACCESS lists all system files which should be invisible to user. You can edit HS_FILE_ACCESS to add user's UniVerse files if you don't want to see them in ODBC application.

You can also make saved select lists that are associated with a table, view, or file visible to ODBC applications.

For detailed information about making tables, views, files, and select lists accessible, see Accessing UniVerse tables, views, and files, on page 34.

### Columns and fields

You can make particular columns and fields accessible to ODBC applications. You can do this by adding or editing an @SELECT phrase in the dictionary of a table, view, or file. If a UniVerse file contains multivalued data, you may also need to edit other entries in the dictionary that control the behavior of multivalued columns and fields. For detailed information about making columns and fields accessible, see Accessing columns and fields, on page 36.

### Presenting UniVerse data in ODBC format

UniVerse data is organized differently from the way ODBC applications expect it to be organized. Two areas where UniVerse data differs from standard ODBC data are:

- Data and data types
- Multivalued data

### Data and data types

Data in UniVerse files has no data type. ODBC applications, on the other hand, expect all data to be one of several data types. In addition, UniVerse data can be of variable length, whereas ODBC expects data to have either a fixed or a maximum length. To make UniVerse data look more like what an ODBC application expects, you may want to do one or more of the following:

- Run the HS.SCRUB utility in the account
- Define SQL data types for data in UniVerse files
- Fix data values that cause SQL and ODBC problems
- Modify certain UniVerse conversion codes

Details about these can be found in Modifying UniVerse data and data definitions for ODBC, on page 39.

### Multivalued data

ODBC applications expect data to be organized relationally in first normal form (1NF). Although some UniVerse tables, views, and files may be in first normal form, with only one value in each column of each row, many UniVerse tables, views, and files have columns that store multiple values in the columns of a row.

U2 ODBC always presents multivalued data to ODBC applications in first normal form. U2 ODBC automatically normalizes UniVerse tables and files in order to present their data to ODBC applications in ODBC format. For detailed information about how UniVerse normalizes UniVerse data files, see Multivalued columns and fields, on page 37 and Association keys, on page 43.

## Accessing UniVerse schemas and accounts

UniVerse databases can be organized into:

- UniVerse SQL schemas
- UniVerse accounts that are not schemas

A U2 ODBC application connects to a UniVerse schema or account and can access the tables, views, and files there. The schema or account must be defined as a data source on the client machine using:

- The Windows ODBC Administrator
- The UCI Config Editor tool

See Administering UniData accounts, on page 25 for how to configure U2 ODBC data sources.

- Use Q-pointers or remote F-pointers. A remote file with a Q-pointer or F-pointer in the VOC of the local account appears as a local file. In U2 ODBC, Q-pointers and remote F-pointers work only with files, not with tables or views. For information about how to use Q-pointers, see *UniVerse System Description*.
- Use ODBC qualifiers. ODBC qualifiers do not require you to set up Q-pointers, but they work only on SQL tables and views.
- Use multiple U2 ODBC connections, each to a different account.

To access other schemas and accounts, you can do three things:

**Note:** You cannot access files through UV/Net using U2 ODBC.

## Using ODBC qualifiers

A qualifier is the ODBC equivalent of the name of a UniVerse schema or account. In ODBC SQL you can refer to a remote table as *qualifier.tablename*. An ODBC connection has a *current qualifier* which implicitly qualifies unqualified table names.For ODBC tables derived from UniVerse tables and views, U2 ODBC reports the UniVerse schema name as the ODBC qualifier.

For ODBC tables derived from UniVerse files, U2 ODBC reports the empty string as the qualifier. To refer explicitly to a UniVerse file in an SQL statement, qualify the ODBC table name with the empty string qualifier. For example:S

```
ELECT * FROM "".MYFILE;
```

U2 ODBC uses two qualifiers:

- The *current* ODBC qualifier
- The *local* ODBC qualifier

Both the current and the local qualifiers are initially set to:

- The name of the schema to which you are connected
- An empty string if the UniVerse account is not a schema

You can change the current qualifier to the name of another schema (or to an empty string) using the SQL_CURRENT_QUALIFIER option of the SQLSetConnectOption function. The local qualifier is the name of the UniVerse account to which the client is connected and remains constant during the ODBC connection.

### Resolving unqualified ODBC table names

U2 ODBC uses the following rules when resolving an unqualified ODBC table name in an SQL statement:

- If the table name occurs in a place in an SQL statement where UniVerse SQL does not allow a qualified table name (such as DROP TABLE), U2 ODBC resolves the table name using the local qualifier. An error occurs if the SQL statement is a DDL statement and the local account is not a schema, because UniVerse SQL does not allow DDL statements to be executed in accounts that are not schemas.

- If the table name occurs in a place in an SQL statement where UniVerse SQL allows a qualified table name, U2 ODBC first looks for a file defined in the local VOC file that matches the ODBC table name.
- If no match is found, and if the current ODBC qualifier is the name of a UniVerse schema (that is, it is not set to the empty string), U2 ODBC looks for an SQL table or view in that schema that matches the ODBC table name.

## Accessing UniVerse tables, views, and files

UniVerse schemas can contain:

- SQL tables and views
- Files that are not tables or views

UniVerse accounts that are not schemas can contain only UniVerse files that are not tables or views.

Tables and views are always accessible to ODBC applications, subject to SQL privileges and operating system permissions. To make files that are not tables accessible to ODBC applications, you must run the ODBC file access utility. To do this, choose the **Activate Access to Files in an Account** option from the U2 ODBC System Administration menu.

When you run the ODBC file access utility, it creates the file information cache and HS_FILE_ACCESS file. File information cache lists all UniVerse files referenced by F- and Q-pointers in the VOC file. HS_FILE_ACCESS lists all system files which should be invisible to user. This makes all UniVerse files in the account, except system files such as &DEVICES&, DICT.DICT, APP.PROGS, etc., accessible to ODBC applications. You can edit the HS_FILE_ACCESS file to define exactly which files should be ODBC-accessible (see File privileges and permissions, on page 35).

For more information about how U2 ODBC modifies an account to make it ODBC-accessible, see What the ODBC file access utility does, on page 36.

By default, U2 ODBC applies name mapping to the names of tables, views, and files to make them ODBC-compliant. For more information about name mapping, see ODBC name mapping, on page 44.

## Updating a UniVerse account for ODBC access

If you make changes to UniVerse files in the account after running the ODBC file access utility and you want the new information to be accessible to U2 ODBC, you should update the U2 ODBC file information cache. The following changes to UniVerse files require you to update U2 ODBC file access in the account:

- Adding, changing, or deleting F- or Q-pointers in the VOC file
- Creating or deleting UniVerse files
- Defining, changing, or deleting association definitions in file dictionaries
- Adding or deleting unassociated multivalued fields to or from a file

To update the U2 ODBC file information cache for an account, choose the **Update File Information Cache in an Account** option from the UniVerse Server Administration menu. You can also use the HS.UPDATE.FILEINFO command to update U2 ODBC file access in an account.

## File privileges and permissions

UniVerse SQL privileges control access to the tables and views in a schema or an account. However, they do not provide any kind of access control over UniVerse files.

Operating system permissions on the files underlying UniVerse tables, views, and files also control access to those tables, views, and files.

U2 ODBC uses the HS_FILE_ACCESS file, created by the ODBC file access utility, to control access to UniVerse files. By adding UniVerse files to HS_FILE_ACCESS, U2 ODBC controls access to those files.

The IDs of records in the HS_FILE_ACCESS file are the names of the files whose access you want to control. Each record has one field, ACCESS, which contains one of the following values:

- READ_WRITE
- READ
- NONE

These values define the type of access to the file referenced by the record ID.

The HS_FILE_ACCESS file contains a record called HS_DEFAULT that controls default access to all files in the account. When you first run the ODBC file access utility, it sets the HS_DEFAULT record to READ_WRITE and sets the records for UniVerse system files (APP.PROGS, BASIC.HELP, ERRMSG, UV.ACCOUNT, DICT.DICT, NEWACC, and so forth) to NONE (no access). To restrict U2 ODBC to read-only access for all files in the account, change the ACCESS field in the HS_DEFAULT record to READ.

If an account does not have an HS_FILE_ACCESS file, U2 ODBC denies all access to UniVerse files in the account.

**Note:** You can circumvent the access control provided by the HS_FILE_ACCESS file by using the native SQL syntax extension to the ODBC SQL grammar or by reparsing. These mechanisms allow SQL statements and UniVerse commands to be passed directly to UniVerse, bypassing U2 ODBC. For more information see *Executing UniVerse SQL*.

## Enabling access to specific files in an account

As of UniVerse 11.3.1, you can enable ODBC access for specific files by issuing the `*HS.WRTFINFO` command at TCL within the required UniVerse account.

The `*HS.WRTFINFO` routine uses a select list of files named `HS_ENABLED_FILES` located in `&SAVEDLISTS&`. This select list needs to have been created prior to running the routine. Upon successful completion, the `HS_FILE_ACCESS` file will be updated with entries for each file specified in the select list. Each entry will have permission set as "READ_WRITE". The UniVerse account's file information cache (.hs_fileinfo) will also be updated to reflect the new permissions.

Following is an example using the `*HS.WRTFINFO` command:

```
>CT &SAVEDLISTS& HS_ENABLED_FILES

HS_ENABLED_FILES
0001 TEST.FILE1
>
>*HS.WRTFINFO

Updating file information cache
from select list HS_ENABLED_FILES.
```

```
HS.FILEINFO errors saved in /usr/uv1131acct/&COMO&/HS_FILE_ERRS
17 AUG 2016 (09:28:10) ---------------------------------
>
>
>CT HS_FILE_ACCESS TEST.FILE1

TEST.FILE1
0001 READ_WRITE
>
```

## What the ODBC file access utility does

When you run the ODBC file access utility in an account, it does the following:

- Creates the HS_FILE_ACCESS file
- Writes an @EMPTY.NULL X-descriptor in all UniVerse file dictionaries
- Writes S or M in field 5 of A- and S-descriptors
- Creates Q-pointers for multiple data files
- Creates the file information cache
- Updates the UV.ACCOUNT file

### File information cache

When you run the ODBC file access utility, U2 ODBC scans the dictionaries of all nonsystem files in the account and stores the information in a file information cache. The U2 ODBC driver uses this cache to provide ODBC applications with a list of accessible ODBC tables without having to construct this information at run time, thus improving performance.

**Warning:** Warning: The file information cache is for U2 ODBC internal use only and should not be modified. Any changes to the cache cause unpredictable behavior and can make all UniVerse files in the account inaccessible to U2 ODBC.

To control how the server uses the file information cache, use the **Fast Connect (Old OTL)** or the **Refresh OTL on Connect** option on the U2 ODBC Data Source Setup menu of the U2 ODBC Administrator.

- Choose **Fast Connect (Old OTL)** (the default setting) if you want the server always to use the file information cache to construct the list of available ODBC tables. If the server cannot access the cache, it constructs the ODBC table information from scratch at connection time.
- Choose **Refresh OTL on Connect** if you want the server to construct the ODBC table list from scratch at connection time, ignoring the cache. The advantage of this mode is that the list of accessible ODBC tables is more current. The disadvantage is that initial connection time can be significantly increased for an account containing many files. Responses to subsequent requests for dictionary information are faster.

## Accessing columns and fields

This section describes what columns and fields are accessible to ODBC, and how to make multivalued columns and fields accessible.

## ODBC-accessible columns and fields

For any table, view, or file, the columns and fields accessible to ODBC are those returned by the following SELECT statement, executed in NF2 mode:

```
SELECT * FROM tablename
```

For tables and views, accessible columns are those listed in the @SELECT phrase, if it exists. If there is no @SELECT phrase, the columns are those listed in the table's SICA, as defined by CREATE TABLE or CREATE VIEW, and as modified by ALTER TABLE.

For files that are not tables, accessible fields are those listed in the @SELECT phrase, if it exists. If there is no @SELECT phrase, the fields are those listed in the @ phrase. If neither the @SELECT nor the @ phrase exist, only the record ID is accessible.

**Note:** An @ phrase in the dictionary of a table or view is ignored.

For more information about the @SELECT and @ phrases, see *UniVerse SQL Administration for DBAs*.

By default U2 ODBC applies name mapping to the names of columns and fields to make them ODBC-compliant. For information about name mapping, see ODBC name mapping, on page 44.

## Multivalued columns and fields

UniVerse tables and files can contain any mix of the following:

- Single-valued columns or fields
- Multivalued columns or fields not associated with any other columns or fields
- Multivalued columns or fields associated with other multivalued columns or fields

U2 ODBC shows UniVerse tables, views, and files in *first normal* form. That is, a table, view, or file containing one or more multivalued columns or fields is treated as a set of ODBC tables comprising:

- One table that includes all single-valued columns or fields
- One table for each association of multivalued columns or fields
- One table for each unassociated multivalued column or field

To make multiple values in a column or field accessible to ODBC applications, the column or field must be defined as multivalued in one of the following ways:

- By the CREATE TABLE, ALTER TABLE, or CREATE VIEW statement that defined the column
- By an M in the SM field of the dictionary definition (field 6 in D- and I-descriptors, field 5 in A- and S-descriptors)
- By a C or D in field 4 of the dictionary definition in A- and S-descriptors (Pick-style associations)

**Note:** If U2 ODBC finds multivalued data in a field defined as singlevalued, only the first value in the field is accessible to ODBC.

If an A- or S-descriptor does not have an M or an S in field 5, the U2 ODBC file access utility samples the data in the field, and if it finds multivalued data and can write to the file dictionary, it defines the field as multivalued.

Certain ODBC tables are not visible to the SQLTables function, even though they are accessible through SQL statements. Some ODBC applications provide limited or no access to ODBC tables that are not visible to SQLTables. SQLTables can see the following:

- In UniVerse tables and views, all associations and unassociated multivalued columns defined by the CREATE TABLE, ALTER TABLE, and CREATE VIEW statements
- In UniVerse files, all associations and unassociated multivalued columns
- In UniVerse files, all Pick associations whose controlling field is visible to SQLTables

## Example

Consider a UniVerse file MYFILE comprising the following fields:

| Field Name | Location | Single or Multivalued | Associated |
|---|---|---|---|
| @ID | 0 | S | No |
| CUSTOMER | 1 | S | No |
| ADDRESS | 2 | M | No |
| QTY | 3 | M | Yes |
| DESCRIPTION | 4 | M | Yes |

The dictionary contains a phrase defining the association of fields 3 and 4:

```
LINEITEMS
0001 PH
0002 QTY DESCRIPTION
```

The field descriptors for QTY and DESCRIPTION specify LINEITEMS in field 7.

The dictionary also contains the following @ phrase:

```
@
0001 PH
0002 CUSTOMER ADDRESS QTY DESCRIPTION
```

All fields of this file are accessible to ODBC applications. The @ID field is not included in the @ phrase because it is always accessible unless suppressed by the ID.SUP keyword.

U2 ODBC presents this file as three tables.

| ODBC Table | ODBC Columns |
|---|---|
| MYFILE | @ID CUSTOMER |
| MYFILE_ADDRESS | @ID CUSTOMER ADDRESS @ASSOC_ROW[1] |
| MYFILE_LINEITEMS | @ID QTY DESCRIPTION @ASSOC_ROW[1] |

1. System-generated. @ASSOC_ROW appears automatically for any association or unassociated multivalued column or field, unless the dictionary contains an @ASSOC_KEY.mvname X-descriptor that defines an association key. See Association keys, on page 43.

If name mapping is on, @ID appears as Z_ID and @ASSOC_ROW appears as Z_ASSOC_ROW. For information about name mapping, see ODBC name mapping, on page 44.

# Modifying UniVerse data and data definitions for ODBC

This section describes things you may need to do to make particular kinds of data accessible to ODBC applications:

- Fix data in data files and dictionaries that cause SQL and ODBC problems
- Define SQL data types for data in UniVerse files
- Define the length of character string data in UniVerse files
- Modify certain UniVerse conversion codes

## Validating tables and files for ODBC clients

Use the HS.SCRUB utility to scan data in a table or UniVerse file and fix data file and dictionary problems that cause SQL and ODBC difficulties. The HS.SCRUB utility does the following:

- Reports table and UniVerse file anomalies
- Saves a select list of record IDs of problem records
- Adjusts dictionary entries to accommodate bad data[1]
- Adds an @EMPTY.NULL record to the dictionary
- Adds an @SELECT record to the dictionary
- Fixes the data in the file, optionally saving a copy of the original file

1. Such as nonnumeric values in numeric, date, and time fields, which can lead to adjusting the column type to CHARACTER.

@EMPTY.NULL

HS.SCRUB adds an @EMPTY.NULL record to the table or file dictionary if all the following conditions are met:

- The dictionary does not already include an @EMPTY.NULL record
- The data contains empty values
- Modification of the dictionary is enabled (FIX, AUTOFIX, AUTOFIX DICT)

For information about empty-null mapping, see Empty or unconvertible data, on page 42.

@SELECT

HS.SCRUB adds an @SELECT record to the file dictionary (but not to a table dictionary) if both of the following conditions are met:

- The dictionary does not already include an @ or @SELECT record
- Modification of the dictionary is enabled (FIX, AUTOFIX, AUTOFIX DICT)

For information about @SELECT records, see *UniVerse SQL Administration for DBAs*.

### Running HS.SCRUB

To run the HS.SCRUB utility, enter `5` to choose **Run HS.SCRUB on a File/Table** from the **U2 ODBC System Administration** menu, or use the `HS.SCRUB` command.

If you use the **Run HS.SCRUB on a File/Table** option, UniVerse prompts to enter either the full path of the UniVerse account (for Windows platforms, start with the drive letter, for example D:) or the account name as listed in the UV.ACCOUNT file. Press Enter to see a list of UniVerse accounts in which file access has already been activated.

Next UniVerse prompts to enter the name of the table or file you want to analyze or change. When you enter a file name, UniVerse prompts to enter the mode of operation. Enter one of the following at the Mode prompt:

- Press **Enter** to generate a report without modifying the table or file.
- FIX
- AUTOFIX
- AUTOFIX DICT
- AUTOFIX DATA

See the next section for a description of these modes.

### Using the HS.SCRUB Command

The syntax of the `HS.SCRUB` command is as follows:

```
HS.SCRUB filename [ FIX | AUTOFIX [ DICT | DATA ] ]
```

*filename* is the UniVerse file or table to be analyzed.

FIX puts HS.SCRUB in interactive mode, in which you are prompted to resolve any anomalies following the analysis.

AUTOFIX puts HS.SCRUB in automatic mode; anomalies are corrected with the default action that would have been presented to the user. If you don't specify DICT or DATA with the AUTOFIX option, HS.SCRUB resolves anomalies in both the data file or table and its dictionary.

DICT indicates that HS.SCRUB resolves only those anomalies associated with the file's or table's dictionary.

DATA indicates that HS.SCRUB resolves only those anomalies associated with the file's or table's data.

When neither FIX nor AUTOFIX is specified, HS.SCRUB only reports anomalies. No data or dictionary items are modified.

## SQL data types

To fine-tune or define data type, length, precision, and scale values for fields and I-descriptors in files, you must edit the DATATYPE field of the corresponding dictionary entry (field 8 in D- and I-descriptors, field 6 in A- and S-descriptors). This is especially important for character data, in which the display width defined in the dictionary may be much larger or smaller than the largest data values in the file. The HS.SCRUB utility can automatically make these adjustments based on the data found.

For UniVerse files, UniVerse determines a field's SQL data type by examining its conversion code and format specifications. The UniVerse database server reports this SQL data type to ODBC applications. If the UniVerse-generated SQL data type is inappropriate for the actual data in the field, you can specify the correct SQL data type in the field's dictionary entry. For some data types, the SQL data type syntax is different between dictionary specifications and UniVerse SQL statements (for example, CREATE TABLE) as noted in the following table. Square brackets indicate optional parameters.

| **Dictionary Syntax** | **UniVerse SQL Syntax** | **Notes** |
|---|---|---|
| CHAR [ ACTER ] [,*n*] | CHAR [ ACTER ] [(*n*)] *n* | = number of characters |
| DEC [ IMAL ] [,*p*[,*s*]] | DEC [ IMAL ] [(*p*[,*s*])] *p* | = precision *s* = scale |
| FLOAT[,*p*] | FLOAT[(*p*)] | *p* = precision |
| NUMERIC[,*p*[,*s*]] | NUMERIC[(*p*[,*s*])] *p* | = precision *s* = scale |

| Dictionary Syntax | UniVerse SQL Syntax | Notes |
|---|---|---|
| VARCHAR[, *n*] | VARCHAR[(*n*)] *n* | = number of characters |

Note the syntactic differences regarding the use of parentheses and commas.

The DATE, DOUBLE PRECISION, INT[EGER], REAL, SMALLINT, and TIME data type syntax is identical for dictionaries and UniVerse SQL.

You can specify the SQL data type for any field, real or virtual, in a UniVerse file. You need not specify the SQL data type for any column of a table or view defined by the CREATE TABLE or CREATE VIEW statement, but you may want to specify the SQL data type for other columns in the table or view (such as I-descriptors) that are not defined in the SICA. You cannot modify the SQL data type for columns defined in the SICA, and UniVerse ignores the dictionary definitions for these columns. For more information about the SICA and UniVerse SQL tables and views, see *UniVerse SQL Administration for DBAs* and *UniVerse SQL User Guide*.

## Length of character data

In ODBC, every character column has either a fixed length or a maximum length. This column length is called its *precision*.The precision of a character column is determined from one of the following:

- For a column defined by a CREATE TABLE or ALTER TABLE statement, the precision is defined by the column definition, which is stored in the table's SICA.
- For a column in a view, the precision is defined by the CREATE VIEW statement or by the precision of the column specified by the SELECT statement that creates the view.
- For a column not defined by the CREATE TABLE or ALTER table statement (such as an I-descriptor), or for a field in a UniVerse file, the precision is defined by the data type specified in the DATATYPE field of the field's dictionary definition. If the DATATYPE is not defined, the FORMAT field of the dictionary defines the precision.

The actual number of characters in a UniVerse character column can be greater than its precision. U2 ODBC retrieves such extra characters, up to a limit. The number of bytes of character data that U2 ODBC retrieves from a character column is the smallest of:

- The number of bytes of data the column actually contains.
- 255, or four times the column's precision, whichever is greater.
- The value of the ODBC statement option SQL_MAX_LENGTH, if it has been set.
- The number of bytes of data that UCI can fetch. This is controlled by the ODBC configuration parameters MAXFETCHBUFF and MAXFETCHCOLS, defined in the *uci.config* file.

If your fetch buffer is not big enough to hold all the character data that U2 ODBC retrieves, U2 ODBC fills your buffer and generates a truncation warning.

The actual number of characters in a UniVerse character column can be less than its precision: unlike some DBMSs, UniVerse does not automatically pad CHAR(*n*) columns on the right with spaces. If you insert the value "abc " (with two trailing spaces) into a CHAR(10) column, the column contains only five characters, not 10.

Your application and data source must agree on a consistent way to treat trailing spaces in a CHAR(*n*) column. Generally it is better to treat CHAR(*n*) columns as if they were VARCHAR columns with no space padding.

## Empty or unconvertible data

U2 ODBC provides mechanisms for handing empty values (empty strings) and unconvertible data (dirty data).

### Empty-null mapping

UniVerse files use empty strings in much the same way tables use null values. Unfortunately, empty strings in numeric or date columns cause data conversion errors in ODBC, making these columns almost inaccessible to many ODBC applications. To make files with empty values accessible to ODBC, U2 ODBC provides empty-null mapping, converting empty values in UniVerse files to null values in ODBC application buffers, and vice versa.

To turn on empty-null mapping for a UniVerse table or file, add an X-descriptor named @EMPTY.NULL to the dictionary. To turn off empty-null mapping, delete the @EMPTY.NULL entry from the dictionary.

**Note:** The Activate Access to Files in an Account option enables empty-null mapping in all files in the account by creating @EMPTY.NULL dictionary entries.

If empty-null mapping is not enabled, U2 ODBC does the following:

- Returns empty strings from empty character columns
- Translates empty strings in numeric or date columns to null values and returns them with a warning (SQL_SUCCESS_WITH_INFO)

If empty-null mapping is enabled, U2 ODBC assumes the user wants all empty strings in all columns to be returned as null values and therefore does not generate a warning.

### Dirty data

Dirty data such as nonnumeric or out-of-range values in a numeric column is unconvertible. Dirty data is tolerated without modifying the data or causing a run-time error to ODBC applications. When U2 ODBC encounters a dirty data value, it returns it as the null value along with a warning (SQL_SUCCESS_WITH_INFO).

## Conversion codes

When you insert or update a value in a column whose definition specifies a conversion code, UniVerse inserts the value without converting it, whether or not the value satisfies the conversion code. Subsequent SELECT statements, however, may not find such a value because UniVerse applies the conversion code to it and returns an empty string or some other unexpected value. This can occur with the following conversion codes among others:

- G (Group Extraction)
- L (Length)
- P (Pattern Match)
- R (Range)
- S (Substitution)

To avoid this problem, restrict your conversion codes to those that are essential. To enforce patterns on input values in tables, use a CHECK constraint.

## Association keys

UniVerse tables and views have primary keys. UniVerse files have record IDs. Primary keys and record IDs are unique identifiers for each row (record) of data in a table or file.

Since U2 ODBC shows associations and unassociated multivalued fields as ODBC tables, they too require a set of unique identifiers that serve as primary keys. These keys are called association keys.

You can define one or more association columns as the association key, but you do not have to. If you do not, UniVerse SQL generates a virtual column called @ASSOC_ROW containing unique values that, combined with the primary keys or record IDs of the base table, become the association keys for the ODBC table generated from the association.

The @ASSOC_ROW column can have either *stable* or *unstable* characteristics. A stable key allows an association to be managed as a static array of values without compaction (such as an association row representing a quarter, month, or day). An unstable key allows an association to be managed as a dynamic array with compaction (for example, names of multiple customer contacts and phone numbers).

**Note:** Even if you specify @ASSOC_ROW in the @SELECT or @ phrase, it appears as a column or field only in an ODBC table generated from the base table, view, or file. It does not appear as a column in the base table.

### Defining association keys

If a table has any associations or unassociated multivalued columns (such as I-descriptors) that were not defined by CREATE TABLE or ALTER TABLE, you can define association keys for them in two ways:

- Using the ASSOC clause of the CREATE TABLE or ALTER TABLE statement
- By adding an X-descriptor called @ASSOC_KEY.*mvname* to the table or file dictionary

For detailed information about defining association keys, see *UniVerse SQL Administration for DBAs* and *UniVerse SQL Reference*.

**Note:** If you use the @ASSOC_KEY.mvname dictionary entry to define an association key, UniVerse enforces uniqueness on the combination of this key with the record ID for SQL INSERT and UPDATE statements. This means that attempts to update an association row fail if other rows contain the same record ID and association key values, even if you are not changing the key value as part of the update. To avoid this problem, make sure that any column or field you define as an association key has unique values within each record in the UniVerse file.

## Using encryption wallets

You can create an encryption key wallet, which contains encryption keys and passwords. Instead of activating encrypt keys individually, you can supply a wallet and its corresponding password to UniVerse to activate all the encryption keys contained in the wallet. UniVerse stores the wallet in the key store.

Client applications can use the encryption key wallet to activate keys for the entire session. Call the ACTIVATE_WALLET() and DEACTIVATE_WALLET() subroutines to activate and deactivate encryption keys contained in the wallet. Each of these subroutines have the following parameters, which must be supplied in the following order:

1. Wallet_id – The ID of the encryption key wallet2.

2. Wallet_password – The password for the encryption key wallet3.
3. Status – 0 for success, other codes indicate failure4.
4. Error_message – In case of failure, a detailed error message

For clients that use SQL to access the database, such as U2 ODBC and UniOLEDB, you can add WALLETID and WALLETPASSWORD to the uci.config file. Adding these parameters enables UniVerse to perform encryption key activation automatically.

If you are using UniObjects, you can access these methods through the subroutine method of a session object.

For more information on automatic data encryption, see the *UniVerse Security Features* manual.

## ODBC name mapping

ODBC requires that table and column names begin with a letter, followed by either letters, digits, or underscores (see *Microsoft ODBC 3.0 Programmer's Reference,* Appendix C, "SQL Grammar").

Most UniVerse databases have file names, column names, and field names that do not conform to ODBC naming rules. Periods (.) are the most commonly used illegal characters. Several key ODBC client applications, including PowerBuilder and all Microsoft products (such as Access, Visual Basic, and Visual C++) that use the Jet database engine, reject periods in identifiers.

To make UniVerse data accessible to U2 ODBC, offending file names, column names, and field names are translated to conform to ODBC naming rules. The key features of name mapping are as follows:

- By default, name mapping is in effect to ensure that any UniVerse files, columns, and fields are ODBC-accessible.
- Name mapping is controlled on the client. For more information about the UniVerse Name Mapping option, see U2 ODBC parameters (UniVerse only), on page 21.
- Once a U2 ODBC connection maps a UniVerse file, column, or field name to comply with ODBC, that mapping remains for the duration of the connection, even if the object is deleted and re-created. This prevents U2 ODBC from unexpectedly renaming an object.
- When name mapping is off, you can use non-ODBC compliant identifiers as long as they are delimited by double quotation marks in SQL statements.

The following algorithm translates names to conform to the ODBC 3.0 specification for user-defined names:

- Z_ is prefixed to names beginning with a digit.
- Z is prefixed to names beginning with an underscore or an ODBC-illegal character such as @.
- ODBC-illegal characters are converted to underscores throughout the name.
- Names are truncated to 120 characters.

If these transformations cause two or more names to conflict within a name space, UniVerse appends unique numbers in the following format to the names to distinguish them:

_*n*

The sequence number *n* is a decimal integer, up to seven digits long.

Name mapping is not applied in the following cases:

- When SQL statements or UniVerse commands are executed by the U2 ODBC native syntax extension, as described inThe {NATIVE} Syntax extension, on page 93.
- When retrying as UniVerse SQL. See Retrying statements as UniVerse SQL, on page 93.

## U2 tools available for creating ODBC-accessible files

U2 provides users with three tools that can be used to make UniData data accessible to U2 ODBC applications: U2 Metadata Manager, Schema API, and Visual Schema Generator.

### U2 Metadata Manager

The U2 Metadata Manager (U2 MDM) client is an Eclipse-based tool used to create metadata files and first normal form maps, and to manage schema at an account level. It is the recommended utility for making UniData files accessible to ODBC. UniData users can also use the metadata files for data type enforcement (DTE). U2 MDM generates metadata based on existing U2 dictionaries and their relationship to the data files.

The U2 MDM perspective contains multiple panes, or views. From these views, you can view U2 dictionaries, define metadata to describe the file's data, and create first normal form maps of metadata files.

### Schema API

The Schema API (Application Programming Interface) consists of a series of UniBasic subroutines designed to prepare data stored in an NF2 database for access by the ODBC driver and desktop applications, such as Visual Basic and PowerBuilder. For step-by-step procedures, see *Using VSG and the Schema API*.

### Visual Schema Generator

Visual Schema Generator (VSG) is a Windows-based Graphical User Interface (GUI) tool for making UniData files ODBC/SQL accessible, so you can access them with U2 ODBC. VSG supports GUI-based creation, deletion, and modification of SQL subtables and views, as well as a GUI interface to the ANSI privilege (security) settings for the tables, subtables, and views.

## Special considerations for use with the SQL Server Import and Export wizard

In certain circumstances, U2 ODBC files behave unexpectedly when used with the SQL Server Import and Export wizard. In order for the files to behave as expected, the following considerations apply.

- Users must define the **USE_U2ODBC_AS_20=1** environment variable for schema data to display correctly.
- Users must correctly define the SQL Server data types for the Date and Time variables using the column mapping options. If incorrect data types are defined, the target SQL Server table will not be created.

# Chapter 5: Making UniData data accessible to U2 ODBC applications

U2 provides several mapping tools that can be used to prepare your UniData files for desktop access through U2 ODBC. This “mapping” process translates UniData nested relational data to adhere to ODBC/SQL rules. It allows you to use your UniData files in conjunction with ODBC tools such as Microsoft Visual Studio for .NET and Microsoft Office. This chapter introduces you to the mapping process and helps you decide which tool to use.

## Mapping UniData files for access to U2 ODBC

Mapping prepares UniData data for access through U2 ODBC-compliant applications. It allows you to use your UniData files in conjunction with ODBC tools such as Microsoft Visual Studio for .NET and Microsoft Office.

ODBC-compliant applications assume that the data being passed to them is in 1NF (first normal) form. Since UniData files are in the NF2 (nested relational) format, UniData data files must be translated to the 1NF format. The process of translating UniData nested data to adhere to ODBC/SQL rules is called mapping.

The following section describes the methods you can use to map your data for access from desktop tools using U2 ODBC.

## Preparing files for U2 ODBC

You need to prepare your UniData files for use by U2 ODBC. The steps to take to prepare your data depend on how you will access the data.

- Make UniData files into SQL-accessible tables with compliant attribute and table names.
- Create views or subtables on the tables and their 1NF schema. Views provide read-only access; subtables provide update access.
- Grant privileges to users
- Define any stored procedures you want to use

### U2 tools available for creating ODBC-accessible files

U2 provides users with three tools that can be used to make UniData data accessible to U2 ODBC applications: U2 Metadata Manager, Schema API, and Visual Schema Generator.

#### U2 Metadata Manager

The U2 Metadata Manager (U2 MDM) client is an Eclipse-based tool used to create metadata files and first normal form maps, and to manage schema at an account level. It is the recommended utility for making UniData files accessible to ODBC. UniData users can also use the metadata files for data type enforcement (DTE). U2 MDM generates metadata based on existing U2 dictionaries and their relationship to the data files.

The U2 MDM perspective contains multiple panes, or views. From these views, you can view U2 dictionaries, define metadata to describe the file's data, and create first normal form maps of metadata files.

## Schema API

The Schema API (Application Programming Interface) consists of a series of UniBasic subroutines designed to prepare data stored in an NF2 database for access by the ODBC driver and desktop applications, such as Visual Basic and PowerBuilder. For step-by-step procedures, see *Using VSG and the Schema API*.

### Visual Schema Generator

Visual Schema Generator (VSG) is a Windows-based Graphical User Interface (GUI) tool for making UniData files ODBC/SQL accessible, so you can access them with U2 ODBC. VSG supports GUI-based creation, deletion, and modification of SQL subtables and views, as well as a GUI interface to the ANSI privilege (security) settings for the tables, subtables, and views.

## Creating views or subtables

The decision to create a view or a subtable depends on the type of task you need to accomplish:

- A view is a “virtual table,” representing a stored query. This stored query may be designed to retrieve all or only some of the attributes in the base table. Views retrieve data in the base table as needed, so the data is not replicated. 1NF-mapping views provide read-only access to the tables through U2 ODBC, and are used for reporting purposes only. Currently, views cannot be used in U2 MDM.
- A subtable is an updatable view that presents data stored in a table in such a way that it can be viewed and updated using 1NF SQL data manipulation commands. As a result, ODBC-compliant and other 1NF application can select, insert, update, and delete data stored in UniData nested relational tables. However, subtable structure may be more complex than that of a view; especially when multi-subvalued attributes are involved.

## Creating schema

Creating schema on views or subtables translates attribute format and conversion specifications into the corresponding SQL/ODBC data type and places this information into system tables for use by the ODBC, JDBC, and OLEDB drivers. You can create schema using U2 MDM, VSG, or Schema API.

## Granting privileges

You must grant privileges to give users permission to access tables, views, and subtables. Currently, this can be done in VSG only, as U2 MDM does not provide support for managing privileges.

## Defining stored procedures

ODBC supports the execution of stored procedures. Stored procedures extend the capabilities of an ODBC application beyond the limitations of SQL. Defining a stored procedure identifies to an ODBC or JDBC application information about the existing cataloged UniBasic subroutine. You can define stored procedures using VSG or Schema API.

## Naming conventions

There are a number of restrictions necessary in a UniData SQL environment that are not required in a UniData environment. For 1NF applications, such as reporting tools, to be able to access UniData files, the following restrictions must be addressed:

- The length of the file name or attribute name must not be longer than 30 characters.
- The first character in the file name or attribute name must be an alphabetic character. If the file name is not compliant, you can enclose the file name between quotation marks (“”) to work around this restriction.
- Except for the first character, all characters in the file name or attribute name can be alphanumeric characters or an underscore character (_). An at sign (@), a dollar sign ($), a pound sign (#) and a dot (.) are not allowed. If the file name or attribute name is not compliant, you can enclose the name between quotation marks (“”) to work around this restriction.
- The file name or attribute name must not be a UniData SQL reserved word.
- The file name must not be an existing view name.
- The value specifier in the dictionary file cannot contain a null value. Valid specifications are S, M, MV, or MS. (Although M is a valid attribute type, we recommend you use MV or MS to avoid ambiguity.)
- If an attribute name is part of an association, that association name must exist in the dictionary as one of the PH attributes.
- An association cannot contain a single-valued (S) attribute.

The ECL CONVERT.SQL command addresses UniData SQL requirements to make any UniData file accessible through UniData SQL.

When using U2 MDM, it is not necessary to run CONVERT.SQL, as U2 MDM-generated schemas automatically map file attributes to compliant mapping subtables.

## Making your non-compliant UniData table or column names accessible

Beginning at UniData 7.3, UniData provides a **QUOTED_IDENTIFIER** configuration parameter. The configuration parameters are located in the udtconfig file.

- On a Windows system, this file is located in the udthome\include directory.
- On a UNIX system, this file is located in the /usr/ud73/include directory.

When the value of QUOTED_IDENTIFIER is **1**, any table or column identifier can be delimited by double quotation marks, and literals must then be delimited by single quotation marks. This is the default setting.

When the value of QUOTED_IDENTIFIER is 0, literal data delimited by double quotations marks will be treated as a constant value. This setting is not recommended.

If the table or column name is not SQL-compliant, enclose the name in double quotation marks and it will be read without causing an error.

**Note:** The use of double quotes on constant text data, such as SQL: SELECT * FROM STATES WHERE ID="CO"; is not SQL compliant and will cause an error. If you have constant text data that is enclosed in double quotation marks, change the double quotation marks to single quotations marks to make your data SQL compliant.

## Addition of system tables

U2 ODBC uses a set of system tables to translate UniData information for access by reporting tools. These system tables identify a variety of parameters, including the “unnested” files described previously. These system tables are created and updated by U2 MDM, VSG, and Schema API.The following table lists the system tables U2 ODBC needs and describes the contents of each table.

| System Table | Description |
|---|---|
| SQLColumns | Contains lists of column names in specified tables. Required for ODBC compliance. Located in each user database. |
| SQLTables | Contains the list of table names stored in a specific data source. Required for ODBC compliance. Located in each user database. |
| SQLStatistics | Combines information for a specific table with information about each index. Required for ODBC compliance, but may contain no records. Located in each user database. |
| SQLSpecialColumns | Contains information about columns in a specified table. Required for ODBC compliance, but may contain no records. Located in each user database. |
| _METADATA_REPOSITORY_ | A UniData hash file that is on every account to which the U2 MDM tool has successfully connected. It is a system file containing the U2 Metadata Manager's data regarding metadata and 1NF maps for an account's database files. |

## Using encryption wallets

You can create an encryption key wallet, which contains encryption keys and passwords. Instead of activating encrypt keys individually, you can supply a wallet and its corresponding password to UniVerse to activate all the encryption keys contained in the wallet. UniVerse stores the wallet in the key store.

Client applications can use the encryption key wallet to activate keys for the entire session. Call the ACTIVATE_WALLET() and DEACTIVATE_WALLET() subroutines to activate and deactivate encryption keys contained in the wallet. Each of these subroutines have the following parameters, which must be supplied in the following order:

1. Wallet_id – The ID of the encryption key wallet2.
2. Wallet_password – The password for the encryption key wallet3.
3. Status – 0 for success, other codes indicate failure4.
4. Error_message – In case of failure, a detailed error message

For clients that use SQL to access the database, such as U2 ODBC and UniOLEDB, you can add WALLETID and WALLETPASSWORD to the uci.config file. Adding these parameters enables UniVerse to perform encryption key activation automatically.

If you are using UniObjects, you can access these methods through the subroutine method of a session object.

For more information on automatic data encryption, see the *UniVerse Security Features* manual.

# Part IV: U2 ODBC

The chapters in Part IV describe how to work with U2 ODBC. It includes information about U2 ODBC conformance levels and the U2 ODBC Tester, and discusses the log files created by U2 ODBC.

# Chapter 6: U2 ODBC conformance levels

This chapter introduces you to U2 ODBC and explains how it conforms to version 3.0 of Microsoft's ODBC specification.

For more information on ODBC conformance levels, instructions on developing applications with ODBC, and ODBC API function syntax, see the *Microsoft ODBC 3.0 Programmer's Reference and SDK Guide*.

## API conformance levels

ODBC specifies three levels of API conformance:

- Core API
- Level 1 API
- Level 2 API

The following are the definitions of the ODBC API Conformance Levels as specified in the *Microsoft ODBC 3.0 Programmer's Reference and SDK Guide*.

### Core API

- Allocate and free environment, connection, and statement handles.
- Connect to data sources. Use multiple statements on a connection.
- Prepare and execute SQL statements. Execute SQL statements immediately.
- Assign storage for parameters in an SQL statement and result columns.
- Retrieve data from a result set. Retrieve information about a result set.
- Commit or roll back transactions.
- Retrieve error information.

### Level 1 API\Q

- Core API functionality.
- Connect to data sources with driver-specific dialog boxes.
- Set and inquire values of statement and connection options.
- Send part or all of a parameter value.
- Retrieve part or all of a result column value.
- Retrieve catalog information (tables, columns, primary keys, foreign keys, special columns, statistics, procedures, and procedure columns).
- Retrieve information about driver and data source capabilities (API and SQL levels, data types, scalar functions).

## Level 2 API

- Core and Level 1 functionality.
- Browse available connections and list available data sources.
- Send arrays of parameter values. Retrieve arrays of result column values.
- Retrieve the number of parameters and describe individual parameters.
- Use a scrollable cursor.
- Retrieve the native form of an SQL statement.
- Retrieve enhanced catalog information (privileges, keys, procedures).
- Call a translation DLL.

## U2 ODBC conformance levels

This chapter introduces you to U2 ODBC and explains how it conforms to version 3.0 of Microsoft’s ODBC specification.

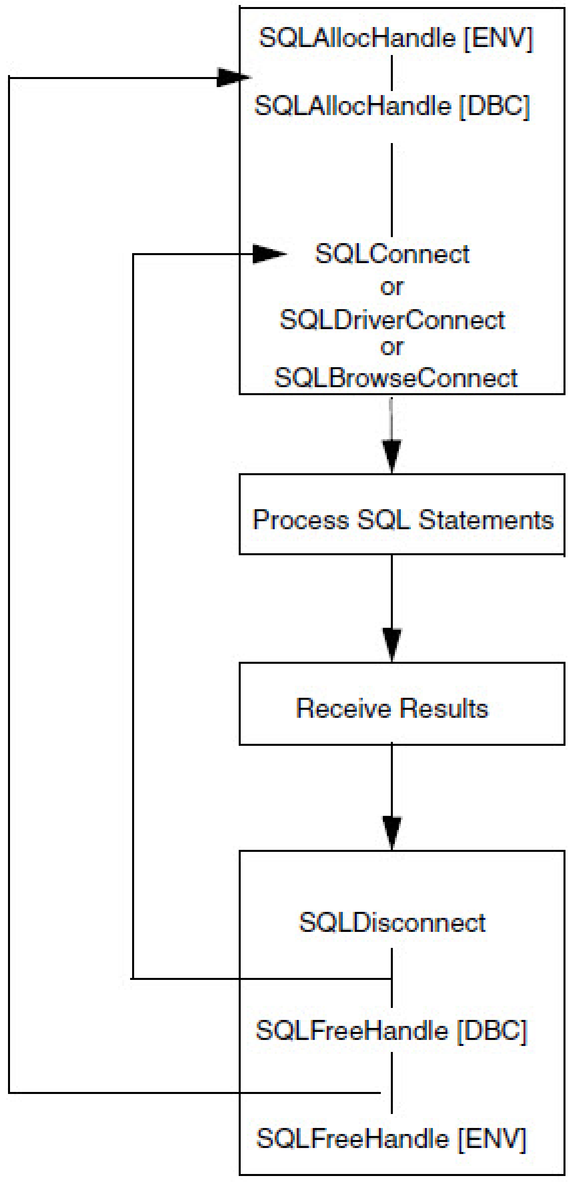
For more information on ODBC conformance levels, instructions on developing applications with ODBC, and ODBC API function syntax, see the *Microsoft ODBC 3.0 Programmer’s Reference and SDK Guide*.

# U2 ODBC-supported API functions

U2 ODBC supports the following API functions which are separated into groups according to the tasks they perform. For more information on the functions in this section, see *Microsoft ODBC 3.0 Programmer’s Reference and SDK Guide*.

## Connecting to a data source

The connection information for each data source is stored in the registry. When an application calls SQLConnect, the ODBC Driver Manager uses the information from the registry to load the appropriate driver and passes the SQLConnect arguments to it. The following figure shows the proper sequence of function calls for connecting to a data source and then disconnecting from the data source.



U2 ODBC supports the following functions in this category:

- SQLAllocHandle
- SQLBrowseConnect
- SQLConnect
- SQLConnectAttr
- SQLDriverConnect
- SQLGetConnectAttr
- SQLGetConnectionOption
- SQLDisconnect
- SQLGetEnvAttr
- SQLFreeConnect
- SQLFreeEnv

- SQLSetConnectAttr
- SQLSetConnectionOption
- SQLSetEnvAttr

## Processing an SQL statement

An ODBC application can execute any SQL statement supported by a DBMS. ODBC defines a standard syntax for SQL statements but does not require you to use this syntax. UniData or UniVerse SQL-specific statements can be sent through the same interface, although an ODBC application using UniData or UniVerse SQL's proprietary syntax extensions will not be able to access any other data source.

There are two ways to execute a statement:

- Prepared Execution
- Direct Execution

### Prepared execution

An application uses prepared execution if it needs to execute the same SQL statement more than once, or if it needs information about the result set prior to execution.

### Direct execution

An application uses direct execution if it needs to execute an SQL statement once and does not need information about the result set prior to execution.

### Supporting parameters

An SQL statement can contain parameter markers to indicate values that are supplied at execution time. The following three examples show how an application might use parameter markers in three types of statements: SELECT, non-SELECT, and procedure calls.

Example 1 — SELECT statements with a parameter marker:

```
SELECT CUST, TAPES-RENTED FROM CUSTOMER_TAPES_MV_SUB1 WHERE CUST = ?
```

Example 2 — INSERT statement:

```
INSERT INTO CUSTOMER_TAPES_MV_SUB1 VALUES (?,?)
```

Example 3 — call a procedure in place of an SQL statement:

- For UniData: `CALL MYPROC(?,?)`
- For UniVerse: `{CALL MYPROC(?,?)}`

### Diagrams of processing SQL statements

The following figure shows a simple sequence of ODBC function calls to execute SQL statements. U2 ODBC supports the following functions in this category:

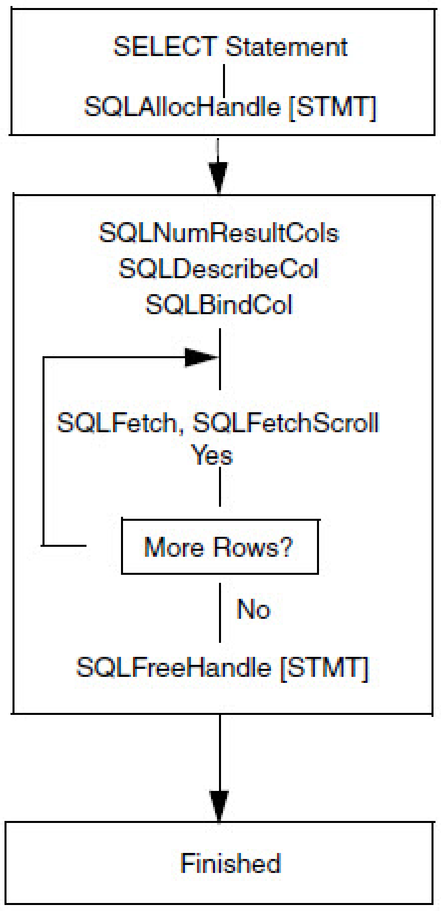
- SQLAllocStmnt
- SQLAllocHandle

- SQLSetStmtOption
- SQLSetStmtAttr
- SQLGetStmtOption
- SQLSetStmtAttr
- SQLPrepare
- SQLBindParameter
- SQLBindParam
- SQLNumParams
- SQLParamData
- SQLParamOptions
- SQLPutData
- SQLExecute
- SQLExecDirect
- SQLTransact
- SQLEndTran
- SQLCancel
- SQLRowCount
- SQLFreeStmt
- SQLFreeHandle
- SQLGetCursorName
- SQLSetCursorName
- SQLCloseCursor
- SQLSetDescRec
- SQLSetDescField
- SQLGetDescField

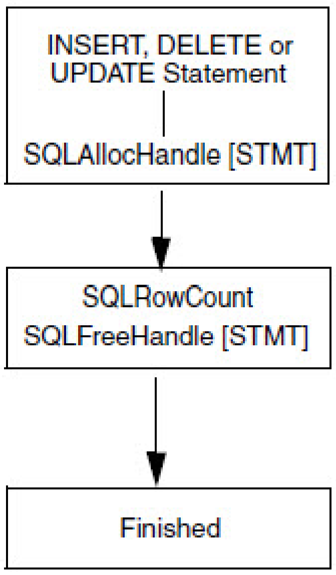
**Note:** The capability of SQLParamOptions to specify multiple values for parameters is not yet supported. The SQLDescribeParam, SQLBulkOperations, SQLCopyDesc, and SQLDescRec functions are not supported. You can process multiple statements simultaneously for a given connection handle.

## Retrieving results and information about results

The data you retrieve as a result of executing an SQL SELECT statement is called a result set. It can contain zero or more rows. Your application can assign storage for results before or after it executes an SQL statement. It then calls SQLFetch to move to the next row of the result set and retrieve the data. The following figure shows a typical data retrieval operation.



SQL UPDATE, INSERT, and DELETE statements do not return result sets. Instead, your application can query the number of rows affected by an INSERT, DELETE, or UPDATE statement. The following figure shows a typical INSERT, DELETE, or UPDATE operation:



U2 ODBC supports the following functions in this category:

- SQLNumResultCols
- SQLDescribeCol
- SQLColAttributes
- SQLBindCol
- SQLGetData
- SQLFetch
- SQLFetchScroll
- SQLRowCount

**Note:** U2 ODBC supports only forward-scrolling capabilities. SQLColAttribute is not supported. U2 ODBC does not support rowset cursors and positioning within a rowset.

## Retrieving data source and driver information

U2 ODBC supports the following functions in this category:

- SQLGetInfo
- SQLGetTypeInfo
- SQLGetFunctions

## Error handling

U2 ODBC supports the following error handling functions in this category:

- SQLGetDiagField
- SQLGetDiagRec

## SQL conformance levels

ODBC specifies three levels of SQL conformance:

- Minimum SQL Grammar
- Core SQL Grammar
- Extended SQL Grammar

The following is the definition of the ODBC conformance levels as specified in the *Microsoft ODBC 3.0 Programmer's Reference and SDK Guide*.

### Minimum SQL grammar

- Data Definition Language: CREATE TABLE and DROP TABLE.
- Data Manipulation Language: simple SELECT, INSERT, searched UPDATE and DELETE.

- Expressions: simple.
- Data Types: VARCHAR, CHAR, LONG VARCHAR

## Core SQL grammar

- Minimum SQL Grammar.
- Data Definition Language: ALTER TABLE, CREATE INDEX, DROP INDEX, CREATE VIEW, DROP VIEW, GRANT and REVOKE.
- Data Manipulation Language: full SELECT.
- Expression: subquery, set functions (like SUM, MIN etc).
- Data Type: DECIMAL, NUMERIC, SMALLINT, REAL, INTEGER, FLOAT, DOUBLE PRECISION.

## Extended SQL grammar

- Minimum and Core SQL Grammar.
- Data Modification Language: outer joins, positioned UPDATE, positioned DELETE, SELECT for UPDATE, unions.
- Expressions: scalar functions, date, time, timestamp literals.
- Data Types: BIT, TINYINT, BIGINT, BINARY, VARBINARY, LONG VARBINARY, TIMESTAMP, DATE,TIME.
- Batch SQL statements.
- Procedure calls.

## U2 ODBC SQL conformance

U2 ODBC is a read/write release supporting the minimal SQL grammar conformance level. Additionally, the U2 ODBC supports selected Core and Extended SQL capabilities.

**Note:** CREATE and DROP statements are not supported in this release of U2 ODBC.

### Returning data

The U2 ODBC driver returns all data residing in a character-type attribute (VARCHAR type column).

For UniData and UniVerse fields where SQLTYPE is not specified:

- If the length specification in FMT field of the dictionary is 254 or less, then the maximum data length is set to 254. For example, if FMT is 15L, then the maximum data length is 254.
- If the length specification in FMT field is greater than 254, then the maximum data length is set to the FMT specification. For example, if FMT is 280L, then set the maximum data length to 280.

For UniVerse fields where SQLTYPE is specified, set the maximum data length to the SQLTYPE specification, ignoring the FMT field.

After the maximum data length is determined, then the driver should return data up to the calculated maximum data length. For example, if the maximum data length is 254 and the actual data received from the server is 500, then the driver truncates the data to 254 before returning it to the application.

As another example, if the maximum data length is 300 and the actual data received from the server is 250, then the driver returns all 250 characters.

## U2 ODBC-supported SQL capabilities in UniData

U2 ODBC supports the following SQL capabilities.

### Data types

U2 ODBC supports the following capabilities in this category:

Minimum SQL support:

- VARCHAR

Core SQL support:

- INTEGER
- FLOAT

Extended SQL support:

- DATE — *yyyy-mm-dd* format
- TIME

### Data definition language

U2 ODBC supports the following capabilities in this category:

Core SQL support:

```
GRANT {ALL | grant-privilege [, grant-privilege]...}
ON table-name
TO {PUBLIC | user-name [, user-name]... }

REVOKE {ALL | revoke-privilege [, revoke-privilege]... }
ON table-name
FROM {PUBLIC | user-name [, user-name]...}
```

Exceptions/Limitations:

- The privilege of REFERENCES is not supported.
- The restrict option of REVOKE is not supported.
- The CASCADE option of REVOKE is supported implicitly (it is the default value but you cannot specify it in you SQL statement).
- Column privileges are not supported.

### Data manipulation language

U2 ODBC supports the following capabilities in this category:

Minimum SQL support:

```
DELETE FROM table-name [WHERE search-condition]

INSERT INTO table-name [( column-identifier [, columnidentifier]...)]
VALUES (insert-value[, insert-value]...)

SELECT [ALL | DISTINCT ] select_list
FROM table_reference_list
[WHERE search_condition]
[order_by_clause]

UPDATE table-name
SET column-identifier = {expression | NULL}
[, column-identifier = {expression | NULL}]...
[WHERE search-condition]
```

Exceptions/Limitations to minimum SQL support:

- The USER keyword in *insert-value* is not supported in the INSERT statement.

Core SQL support:

```
INSERT INTO table-name [( column-identifier [, columnidentifier]...)]
{query-specification | VALUES (insert-value[, insert-value]...)
SELECT [ALL | DISTINCT ] select_list
FROM table_reference_list
[WHERE search_condition]
[GROUP BY column_name [, column_name].. ]
[HAVING search_condition ]
[order_by_clause]
```

Exceptions/limitations to core SQL support in UniVerse:

- The optional AS keyword in *select-list* is not supported
- The USER keyword in *insert-value*, *select-list*, LIKE and IN predicates is not supported

Extended SQL support:

```
ODBC-std-esc-initiator CALL {procedure_name | procedure_name
(procedure_parameter [, procedure_parameter]... ) ODBC-std-escterminator
SELECT [ALL | DISTINCT ] select_list
FROM table_reference_list
[WHERE search_condition]
[GROUP BY column_name [, column_name].. ]
[HAVING search_condition ]
[UNION select_statement]... [order_by_clause]
```

Exceptions/limitations to extended SQL support:

- The optional function specification (?=) is not supported in the procedure call statement
- The USER keyword in *insert-value*, *select-list*, LIKE and IN predicates is not supported (UniVerse only)
- The optional AS keyword in *select-list* is not supported (UniVerse only)
- The ALL keyword in a UNION expression is not supported.

## Scalar functions

ODBC specifies two types of scalar functions which form a part of the extended SQL grammar support. The scalar functions are classified into the following sections:

- String functions
- Numeric functions

**Note:** These functions are supported only when the argument you specify is a column name.

The scalar functions are discussed in more detail in Scalar functions, on page 85.

# Chapter 7: UniVerse U2 ODBC Tester

This chapter describes the U2 ODBC Tester installed with the U2 ODBC driver.

## U2 ODBC Tester

The U2 ODBC Tester is a Windows application developed using the .NET Provider for ODBC. The U2 ODBC Tester allows you to test ODBC connections, ODBC-compliant SQL statements, and UniVerse Native SQL statements. It also allows you to view samples of the following functionality:

- Subroutines
- TOXML statements
- CALL statements

The tester's source code and executable binary code are written in a C# U2ODBC_Tester project, by default, in the following directory:

`C:\U2\UniDK\U2ODBC_64bit\U2ODBC_Tester`

---

**Note:** You must have a 64-bit user Database Source Name (DSN) to use the sample tool. If you have not already created a 64-bit DSN, refer to Configuring UCI for the U2 ODBC driver, on page 16.

---

## Using the U2 ODBC Tester

The U2 ODBC Tester can be used to test SQL and SQL UNNEST statements.

1. Navigate to the U2 ODBC tester, which is located, by default, in the following directory:
   `C:\U2\UniDK\U2ODBC_64bit\U2ODBC_Tester`
2. Double-click the `U2ODBC_Tester64.exe` option.

The U2 ODBC Tester is made up of three views: Input, Output, and Grid. The information you provide in the input view is used to populate the grid view and the output view. The grid view displays data in a grid format, while textual data displays in the output view.

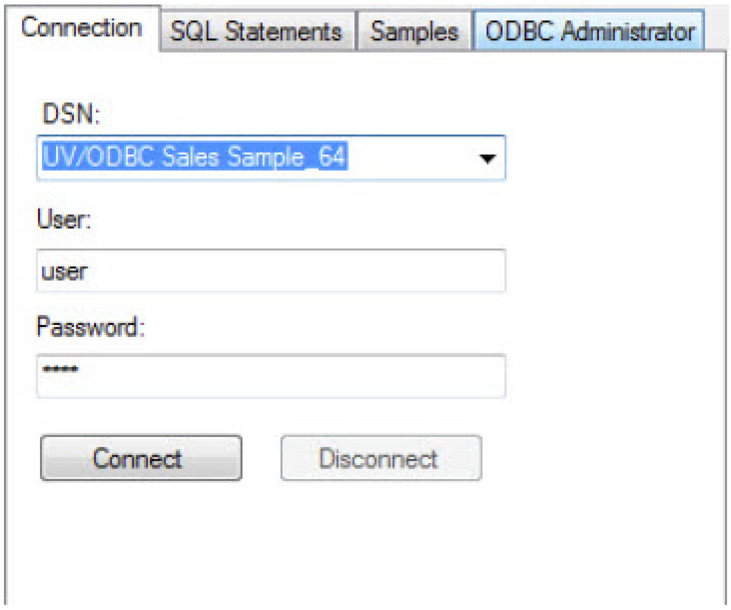
The input view contains four tabs: Connection, SQL Statements, Samples, and ODBC Administrator. The options available from the Connections tab allow a user to test different ODBC connections. You can test the various SQL and Native SQL statements using the options available from the SQL Statements tab. Various sample functions and their result sets are available from the Samples tab. The ODBC Data Source Administrator can also be accessed from the ODBC Administrator tab.

## Testing a connection

Use the U2 ODBC Tester tool to test all ODBC connections.

To test a connection:

1. Enter the connection information as shown:



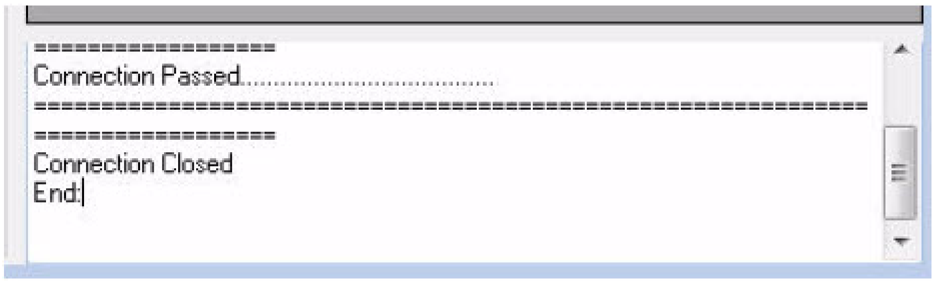
DSN:
UV/ODBC Sales Sample_64

User:
user

Password:
*****

   - In the **DSN** field, select the name of the DSN to use.
   - In the **User** field, enter the appropriate user name.
   - In the **Password** field, enter the appropriate password.

2. Click **Connect**. The tool tests the connection and the results of the connection test appear in the output view, as shown:



```
===================
Connection Passed.....................................
=================================================================
===================
Connection Closed
End:
```

In the example above, the correct connection information was supplied and the connection passed. In instances when the supplied connection information is incorrect, an error message appears in the Output view.

## Testing SQL statements

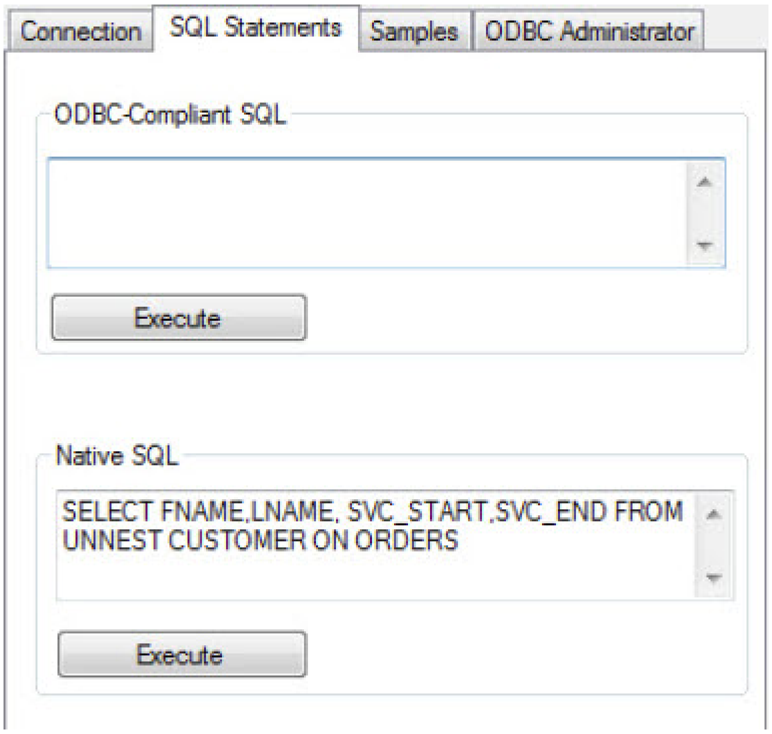
Use the U2 ODBC Tester to test various ODBC SQL SELECT and Native SQL UNNEST statements.

### Testing ODBC-compliant SQL SELECT statements

Use the tool to test your SQL SELECT statements.

To test a SELECT statement:

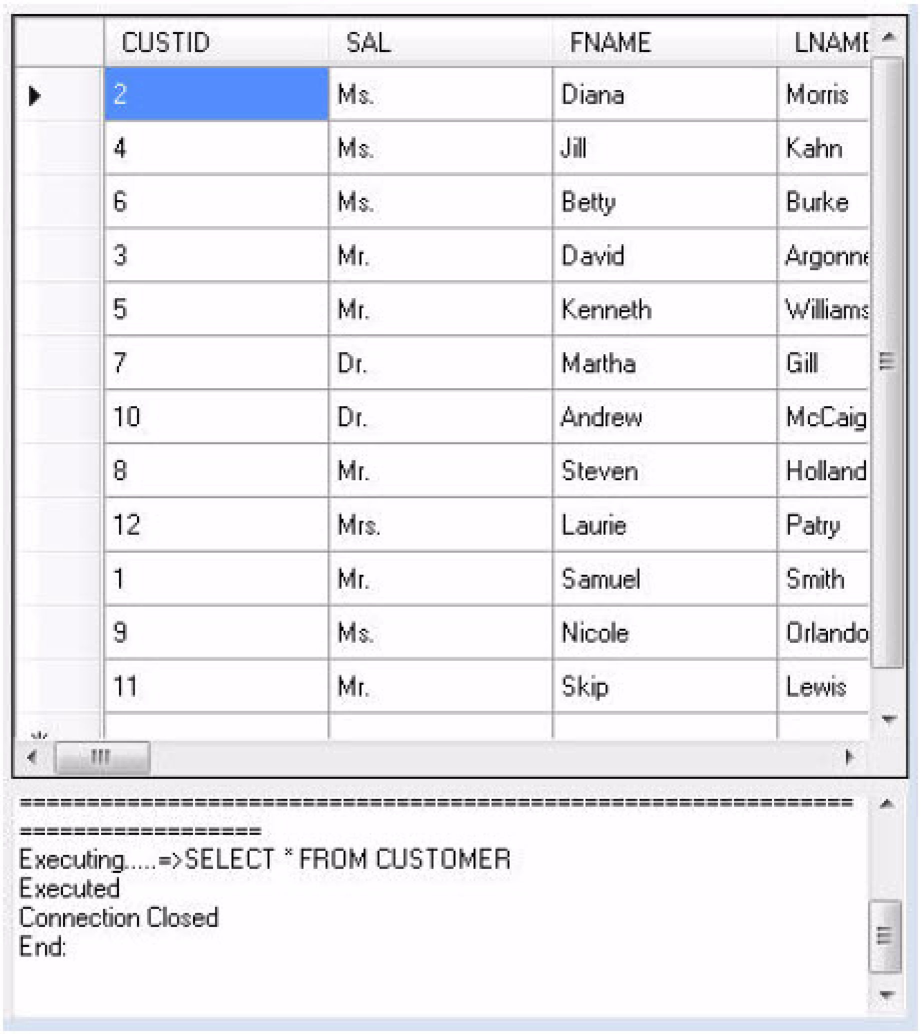
1. Select the **SQL Statements** tab from the menu.
2. In the **ODBC-Compliant SQL** field, enter the SQL SELECT statement that you want to test:



SELECT FNAME,LNAME, SVC_START,SVC_END FROM UNNEST CUSTOMER ON ORDERS

The SQL statement should look similar to the following: SELECT * FROM CUSTOMER

3. Click **Execute**. The tool tests the SELECT statement and the results display in the grid and output views, as shown:



| CUSTID | SAL | FNAME | LNAME |
|---|---|---|---|
| 2 | Ms. | Diana | Morris |
| 4 | Ms. | Jill | Kahn |
| 6 | Ms. | Betty | Burke |
| 3 | Mr. | David | Argonne |
| 5 | Mr. | Kenneth | Williams |
| 7 | Dr. | Martha | Gill |
| 10 | Dr. | Andrew | McCaig |
| 8 | Mr. | Steven | Holland |
| 12 | Mrs. | Laurie | Patry |
| 1 | Mr. | Samuel | Smith |
| 9 | Ms. | Nicole | Orlando |
| 11 | Mr. | Skip | Lewis |

```
=====================================================================
==================
Executing.....=>SELECT * FROM CUSTOMER
Executed
Connection Closed
End:
```

**Note:** The U2 ODBC Tester only tests SELECT statements. Tests for the INSERT, UPDATE, and DELETE functions must be implemented by the developer. Do this by adding an INSERT, UPDATE, or DELETE statement to the Form.cs file found in the U2ODBC_Tester.sln file, which is located, by default, in the following directory: `C:\U2\UniDK\U2ODBC_64bit\U2ODBC_Tester`

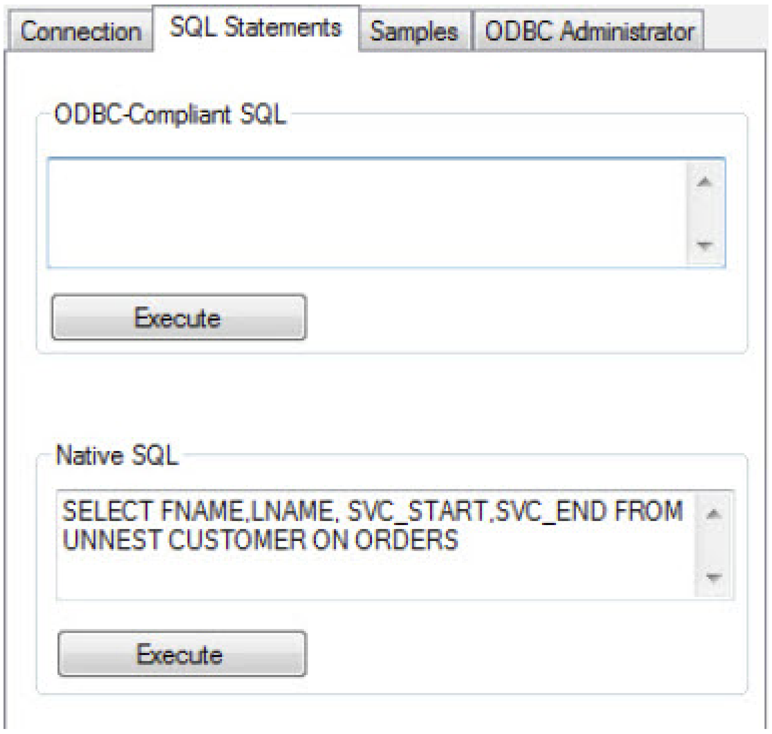
## Testing Native SQL statements

Use the tool to test your SQL SELECT statements.

To test a Native SQL SELECT statement:

1. Select the **SQL Statements** tab from the menu.

2. In the **Native SQL** field, enter the Native SQL SELECT statement to test:

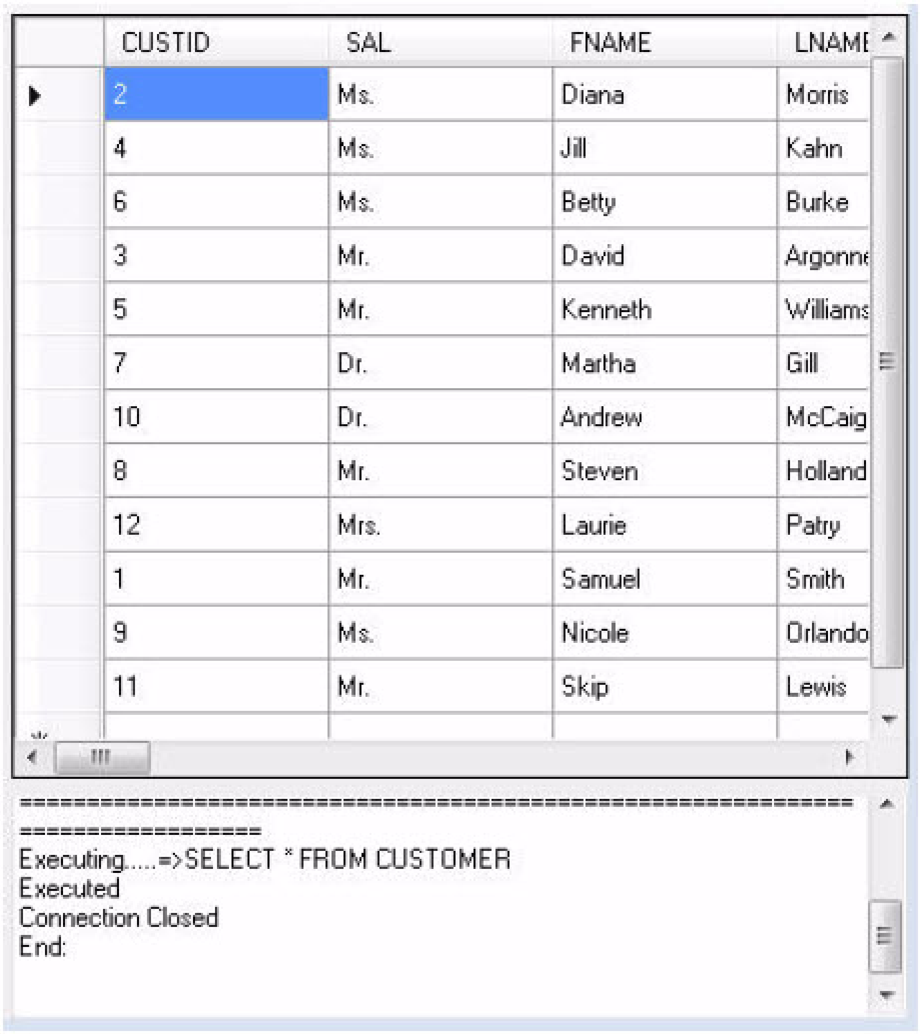


```
SELECT FNAME.LNAME, SVC_START.SVC_END FROM
UNNEST CUSTOMER ON ORDERS
```

The SQL statement should look similar to the following: SELECT FNAME LNAME, SVC_START,SVC_END FROM UNNEST CUSTOMER ON ORDERS

**Note:** Native SQL statements must include an UNNEST clause. If you chose to disable the First Normal Form settings in your DSN definition, you must edit the DSN definition and deselect the First Normal Form checkbox. You can access the DSN definition in the ODBC Data Source Administrator tool. For more information about the UNNEST clause, refer to the UniVerse SQL Reference manual.

3. Click **Execute**. The tool tests the Native SQL statement and the results display in the grid and output views, as shown:



| CUSTID | SAL | FNAME | LNAME |
|---|---|---|---|
| 2 | Ms. | Diana | Morris |
| 4 | Ms. | Jill | Kahn |
| 6 | Ms. | Betty | Burke |
| 3 | Mr. | David | Argonne |
| 5 | Mr. | Kenneth | Williams |
| 7 | Dr. | Martha | Gill |
| 10 | Dr. | Andrew | McCaig |
| 8 | Mr. | Steven | Holland |
| 12 | Mrs. | Laurie | Patry |
| 1 | Mr. | Samuel | Smith |
| 9 | Ms. | Nicole | Orlando |
| 11 | Mr. | Skip | Lewis |

```
=================================================================
==================
Executing.....=>SELECT * FROM CUSTOMER
Executed
Connection Closed
End:
```

## Sample functions

The U2 ODBC Tester provides several sample functions, which a user can execute and then view the output results.

### Viewing the subroutine sample

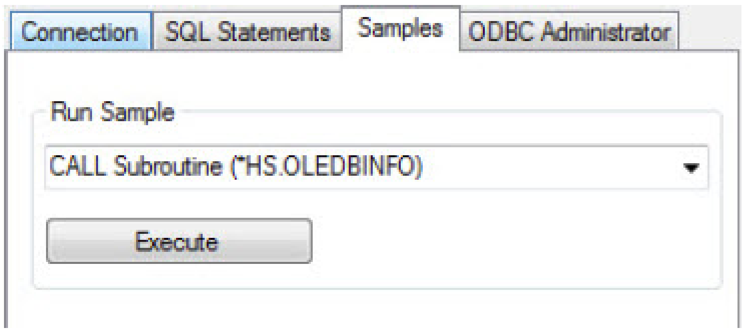
The U2 ODBC Tester provides you with a sample UniVerse subroutine function that has been compiled and cataloged on a UniVerse system. The *HS.OLEDBINFO subroutine fetches metadata. The sample program executes the *HS.OLEDBINFO subroutine and retrieves all of the table metadata.

The subroutines called by the test tool are located in the Form.cs file found in the U2ODBC_Tester.sln file, which is located, by default, in the following directory:

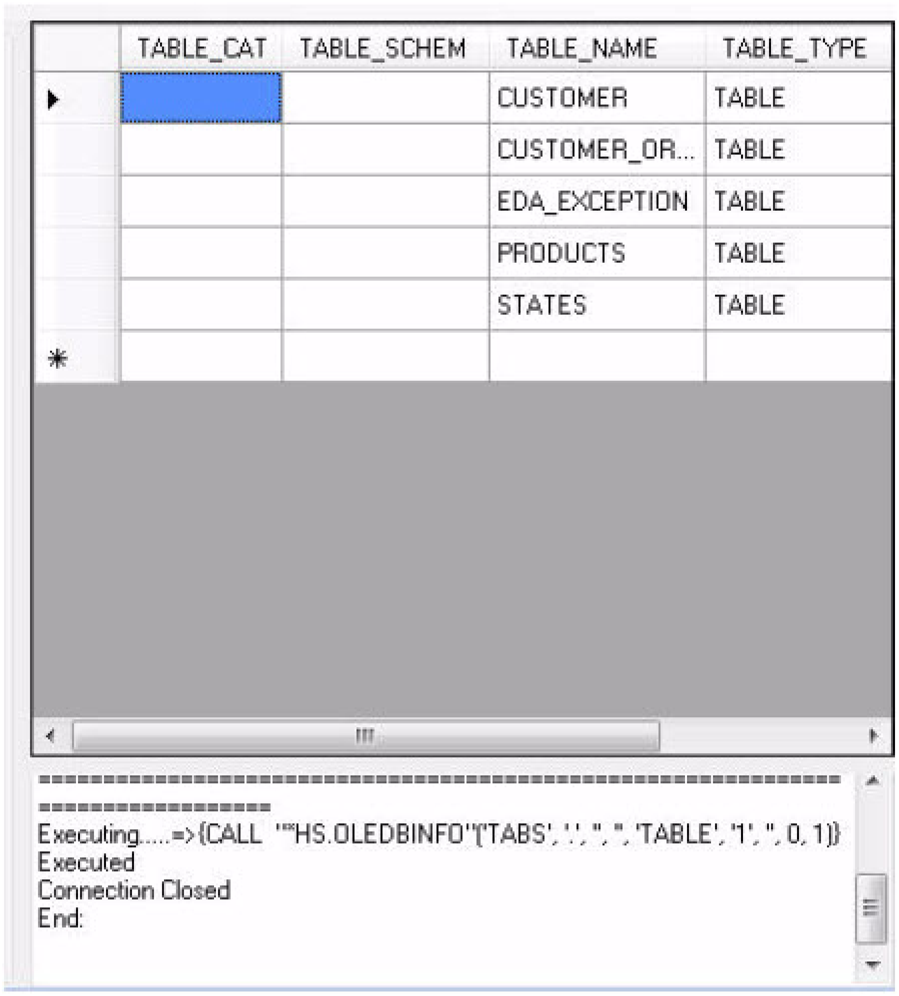
```
C:\U2\UniDK\U2ODBC_64bit\U2ODBC_Tester
```

To view the subroutine sample:

1. Click **Samples** from the menu options at the top of the tool, then select **CALL Subroutine (*HS.OLEDBINFO)** from the menu, as shown:



2. The tool runs the subroutine and the results appear in the output view, as shown:



| | TABLE_CAT | TABLE_SCHEM | TABLE_NAME | TABLE_TYPE |
|---|---|---|---|---|
| ▶ | | | CUSTOMER | TABLE |
| | | | CUSTOMER_OR... | TABLE |
| | | | EDA_EXCEPTION | TABLE |
| | | | PRODUCTS | TABLE |
| | | | STATES | TABLE |
| * | | | | |

```
=================================================================
==================
Executing.....=>{CALL '"*HS.OLEDBINFO"('TABS', '.', '', '', 'TABLE', '1', '', 0, 1)}
Executed
Connection Closed
End:
```

## Viewing the CALL TCL sample

The U2 ODBC Tester provides you with a sample CALL TCL function. The tester executes a CALL statement, similar to the one shown below:

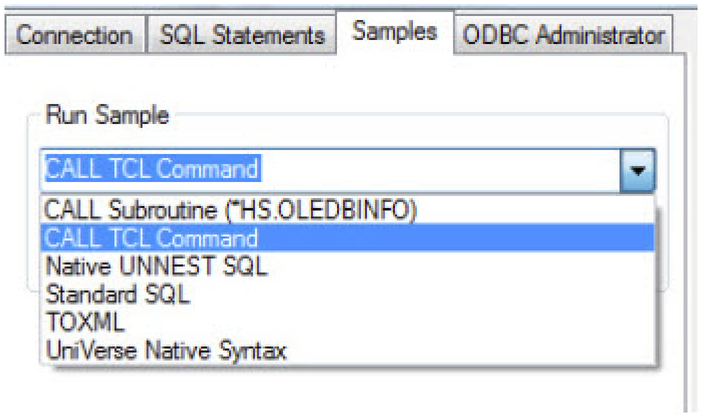
{CALL "LIST CUSTOMER WITH CUSTID=2"}

The CALL function executed by the test tool is located in the Form.cs file, found in the U2ODBC_Tester.sln file, which is located, by default, in the following directory:

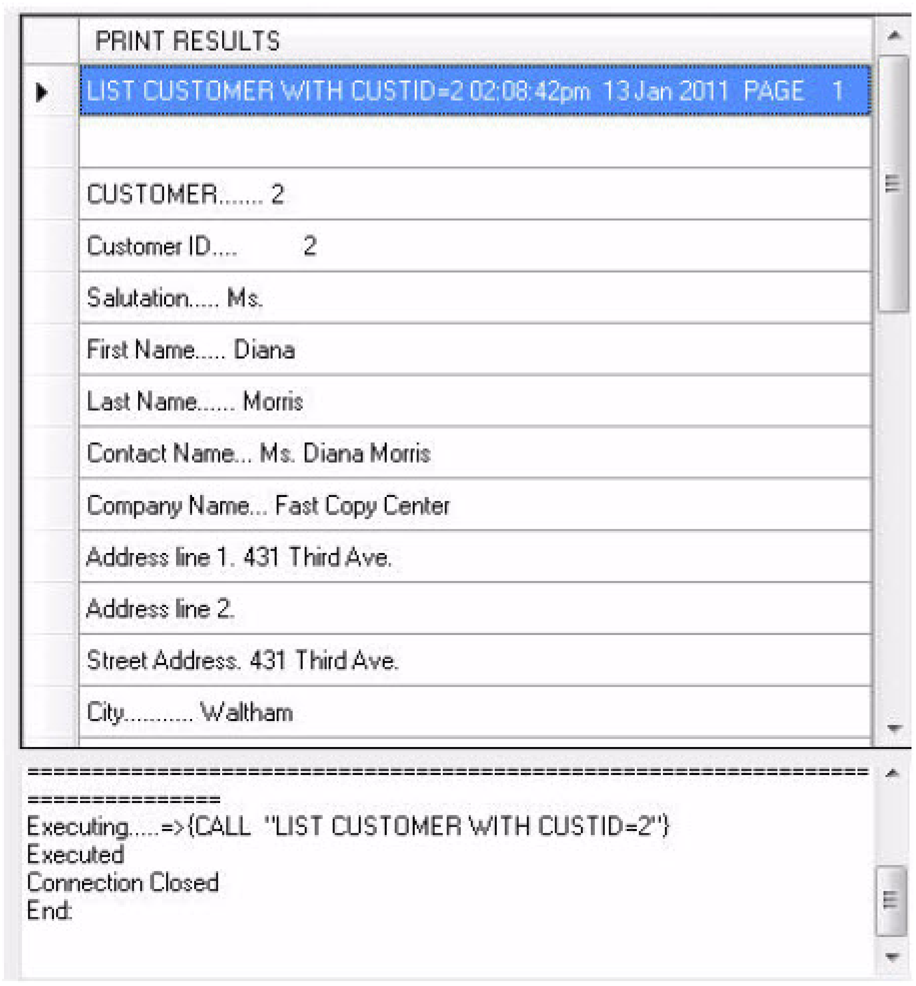
```
C:\U2\UniDK\U2ODBC_64bit\U2ODBC_Tester
```

To view the CALL function:

1. Select **Samples** from the menu options at the top of the tool, then select **CALL TCL Command** from the menu, as shown:



2. Click **Execute**. The results appear in the output view, as shown:



```
PRINT RESULTS
LIST CUSTOMER WITH CUSTID=2 02:08:42pm  13 Jan 2011  PAGE    1

CUSTOMER....... 2
Customer ID....          2
Salutation..... Ms.
First Name..... Diana
Last Name...... Morris
Contact Name... Ms. Diana Morris
Company Name... Fast Copy Center
Address line 1. 431 Third Ave.
Address line 2.
Street Address. 431 Third Ave.
City........... Waltham
```

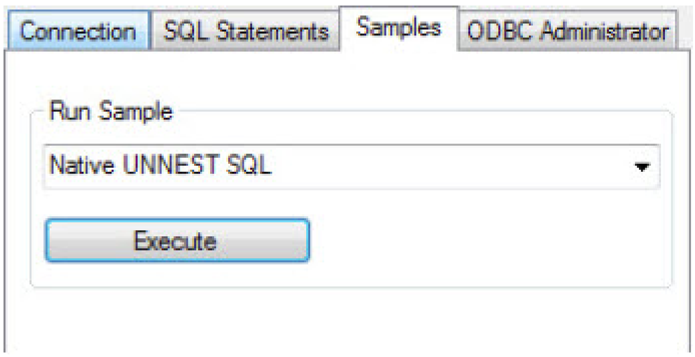
```
=================================================================
===============
Executing.....=>{CALL  "LIST CUSTOMER WITH CUSTID=2"}
Executed
Connection Closed
End:
```

## Viewing the Native SQL sample

The U2 ODBC Tester runs an UNNEST function by executing an UNNEST command.

To test an UNNEST function:

1. Select **Samples** from the menu options at the top of the tool, then select **Native UNNEST SQL** from the menu, as shown:

   The UNNEST function called by the test tool is located in the Form.cs source file found in the U2ODBC_Tester.sln file, which is located, by default, in the following directory: `C:\U2\UniDK\U2ODBC_64bit\U2ODBC_Tester`

2. Click **Execute**.

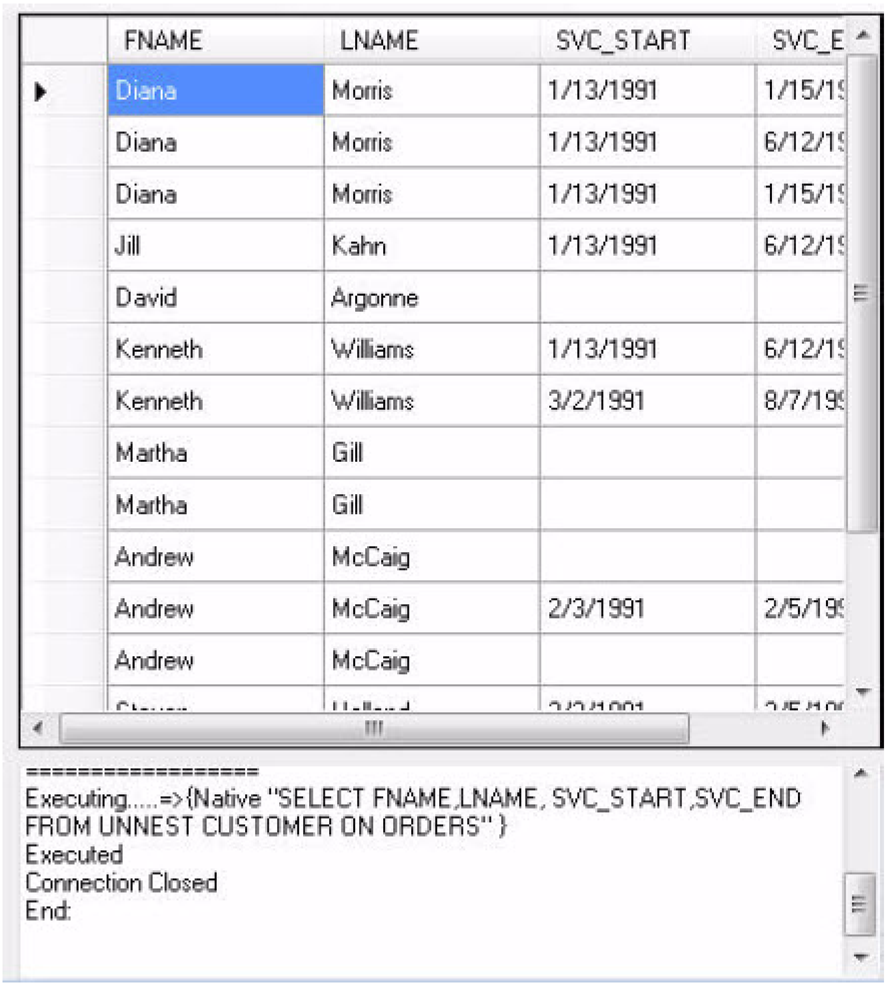
   The first time you attempt to test UNNEST functionality, an error message displays in the output view, as shown:

   ```
   ==========================================================
   To use the UNNEST keyword, open the ODBC ADMIN/U2ODBC
   Driver and deselect the "First Normal Form" checkbox.
   ==========================================================
   ```

   Change the 64-bit driver configuration and clear the First Normal Form checkbox before you can test the UNNEST function. To do this, open the ODBC Data Source Administrator, navigate to the DSN you are working with, and then deselect the First Normal Form box. For more information about using the ODBC Data Source Administrator, refer to Configuring UCI for the U2 ODBC driver, on page 16.

   **Note:** You must exit and restart the U2 ODBC Tester in order to apply any configuration changes made to the DSN.

3. Repeat Step 1 after making the changes to the DSN, and then click Execute. The results appear in the output view, as shown:



| | FNAME | LNAME | SVC_START | SVC_E |
|---|---|---|---|---|
| ▶ | Diana | Morris | 1/13/1991 | 1/15/1 |
| | Diana | Morris | 1/13/1991 | 6/12/1 |
| | Diana | Morris | 1/13/1991 | 1/15/1 |
| | Jill | Kahn | 1/13/1991 | 6/12/1 |
| | David | Argonne | | |
| | Kenneth | Williams | 1/13/1991 | 6/12/1 |
| | Kenneth | Williams | 3/2/1991 | 8/7/19 |
| | Martha | Gill | | |
| | Martha | Gill | | |
| | Andrew | McCaig | | |
| | Andrew | McCaig | 2/3/1991 | 2/5/19 |
| | Andrew | McCaig | | |
| | [illegible] | [illegible] | [illegible] | [illegible] |

```
==================
Executing.....=>{Native "SELECT FNAME,LNAME, SVC_START,SVC_END
FROM UNNEST CUSTOMER ON ORDERS" }
Executed
Connection Closed
End:
```

## Viewing the SQL SELECT sample

The U2 ODBC Tester provides you with a sample SQL SELECT function. The tester executes an SQL statement, similar to the one shown below:

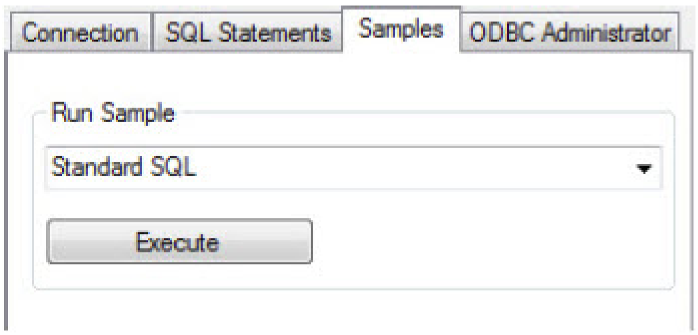
```
SELECT * FROM CUSTOMER
```

The SQL SELECT function executed by the test tool is located in the Form.cs file, found in the U2ODBC_Tester.sln file, which is located, by default, in the following directory:

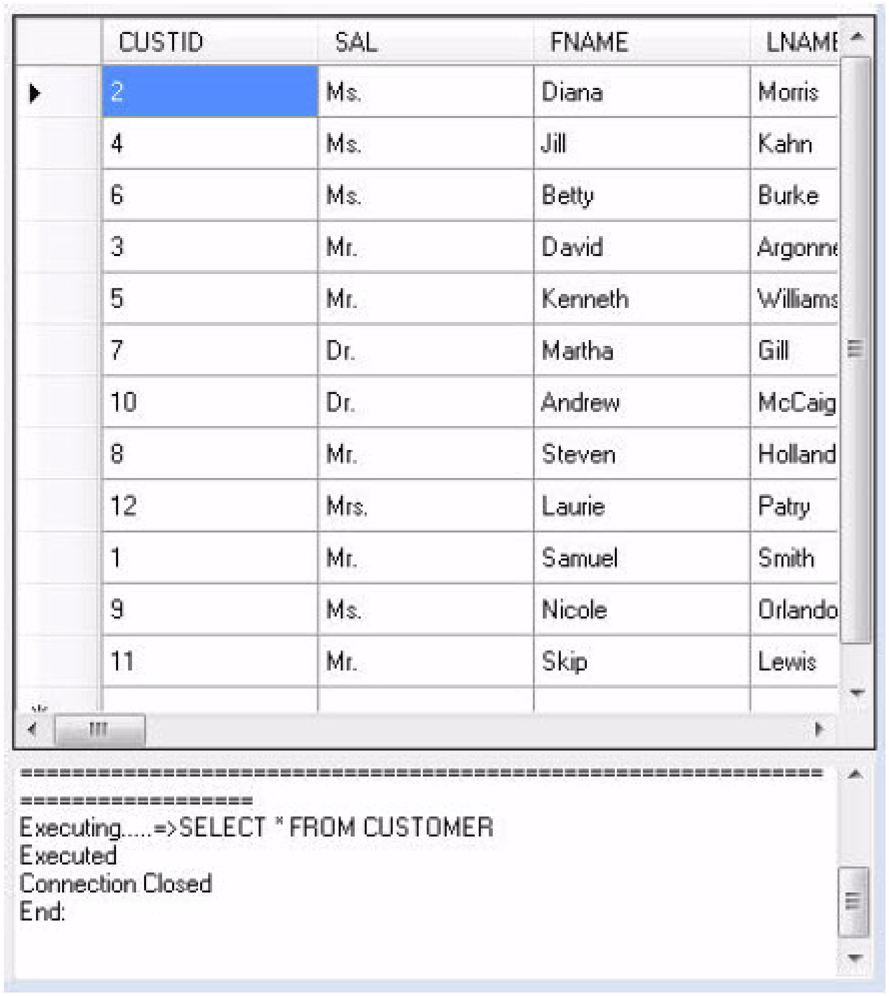
```
C:\U2\UniDK\U2ODBC_64bit\U2ODBC_Tester
```

To test an SQL SELECT function:

1. Select **Samples** from the menu options at the top of the tool, then select **Standard SQL** from the menu, as shown:



2. Click **Execute**. The tool runs the SELECT statement and the results display in the grid and output views, as shown:



| | CUSTID | SAL | FNAME | LNAMI |
|---|---|---|---|---|
| ▶ | 2 | Ms. | Diana | Morris |
| | 4 | Ms. | Jill | Kahn |
| | 6 | Ms. | Betty | Burke |
| | 3 | Mr. | David | Argonn |
| | 5 | Mr. | Kenneth | Williams |
| | 7 | Dr. | Martha | Gill |
| | 10 | Dr. | Andrew | McCaig |
| | 8 | Mr. | Steven | Holland |
| | 12 | Mrs. | Laurie | Patry |
| | 1 | Mr. | Samuel | Smith |
| | 9 | Ms. | Nicole | Orlando |
| | 11 | Mr. | Skip | Lewis |

```
==================================================================
===================
Executing.....=>SELECT * FROM CUSTOMER
Executed
Connection Closed
End:
```

## Viewing the TOXML sample

The U2 ODBC Tester provides you with a sample TOXML function. It does this by executing an XML statement, similar to the one below:

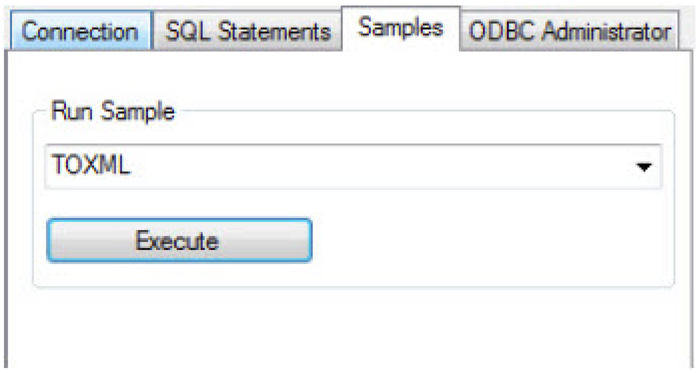
```
{CALL "LIST CUSTOMER TOXML"}
```

The TOXML function called by the test tool is located in the Form.cs source file found in the U2ODBC_Tester.sln file, which is located, by default, in the following directory:

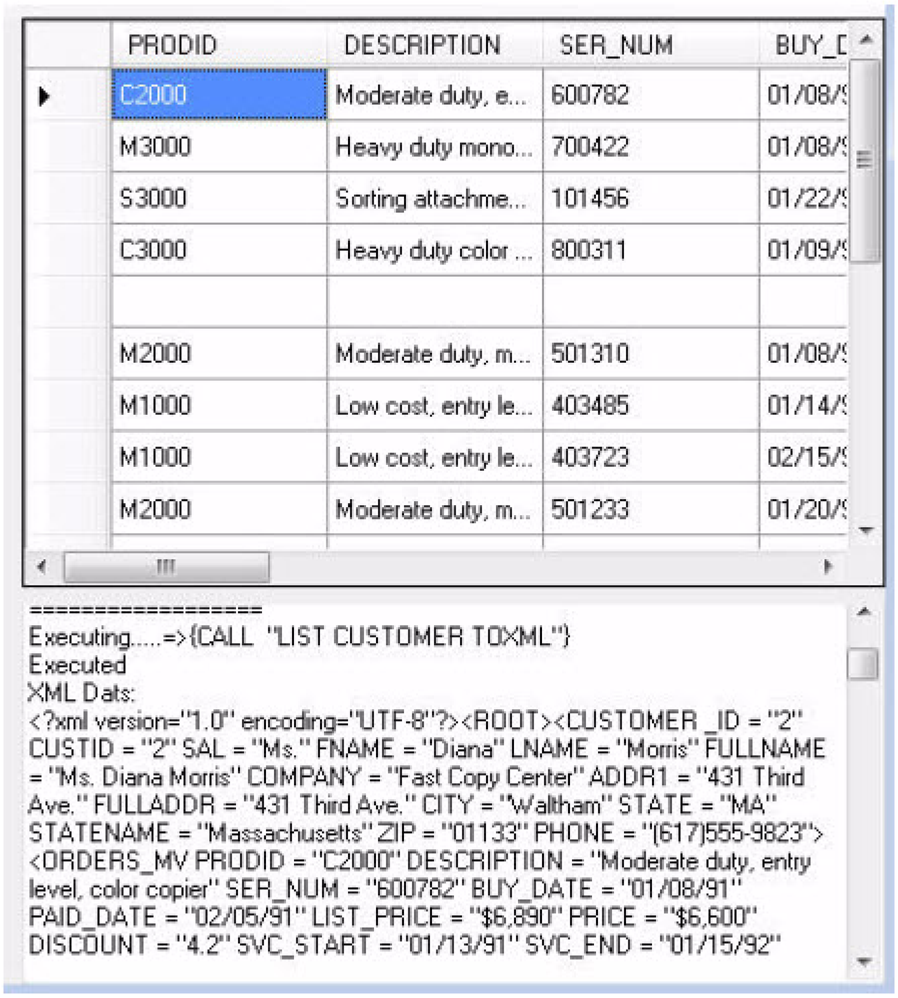
```
C:\U2\UniDK\U2ODBC_64bit\U2ODBC_Tester
```

To view a TOXML function:

1. Click **Samples** from the menu options at the top of the tool, then select **TOXML** from the menu, as shown:



2. Click **Execute**. The tool runs the TOXML function by executing an XML statement. The resulting XML document is displayed, in its entirety, in the output view. The XML document is converted into a dataset, which displays in the grid view, as shown:



| | PRODID | DESCRIPTION | SER_NUM | BUY_I |
|---|---|---|---|---|
| ▶ | C2000 | Moderate duty, e... | 600782 | 01/08/ |
| | M3000 | Heavy duty mono... | 700422 | 01/08/ |
| | S3000 | Sorting attachme... | 101456 | 01/22/ |
| | C3000 | Heavy duty color ... | 800311 | 01/09/ |
| | | | | |
| | M2000 | Moderate duty, m... | 501310 | 01/08/ |
| | M1000 | Low cost, entry le... | 403485 | 01/14/ |
| | M1000 | Low cost, entry le... | 403723 | 02/15/ |
| | M2000 | Moderate duty, m... | 501233 | 01/20/ |

```
==================
Executing.....=>{CALL  "LIST CUSTOMER TOXML"}
Executed
XML Data:
<?xml version="1.0" encoding="UTF-8"?><ROOT><CUSTOMER _ID = "2"
CUSTID = "2" SAL = "Ms." FNAME = "Diana" LNAME = "Morris" FULLNAME
= "Ms. Diana Morris" COMPANY = "Fast Copy Center" ADDR1 = "431 Third
Ave." FULLADDR = "431 Third Ave." CITY = "Waltham" STATE = "MA"
STATENAME = "Massachusetts" ZIP = "01133" PHONE = "(617)555-9823">
<ORDERS_MV PRODID = "C2000" DESCRIPTION = "Moderate duty, entry
level, color copier" SER_NUM = "600782" BUY_DATE = "01/08/91"
PAID_DATE = "02/05/91" LIST_PRICE = "$6,890" PRICE = "$6,600"
DISCOUNT = "4.2" SVC_START = "01/13/91" SVC_END = "01/15/92"
```

## Viewing the UniVerse Native Syntax sample

The U2 ODBC Tester tool provides you with a UniVerse Native syntax sample. It does this by executing a valid UniVerse Native statement, similar to the one shown below:

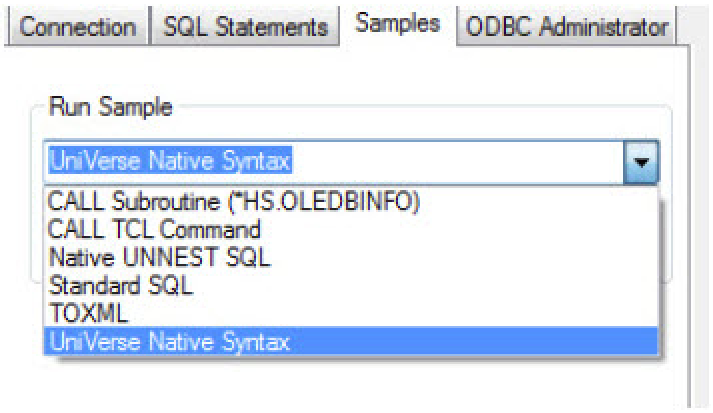
```
{Native "SELECT * FROM CUSTOMER" }
```

The UniVerse Native function called by the test tool is located in the Form.cs file, found in the U2ODBC_Tester.sln file, which is located, by default, in the following directory:

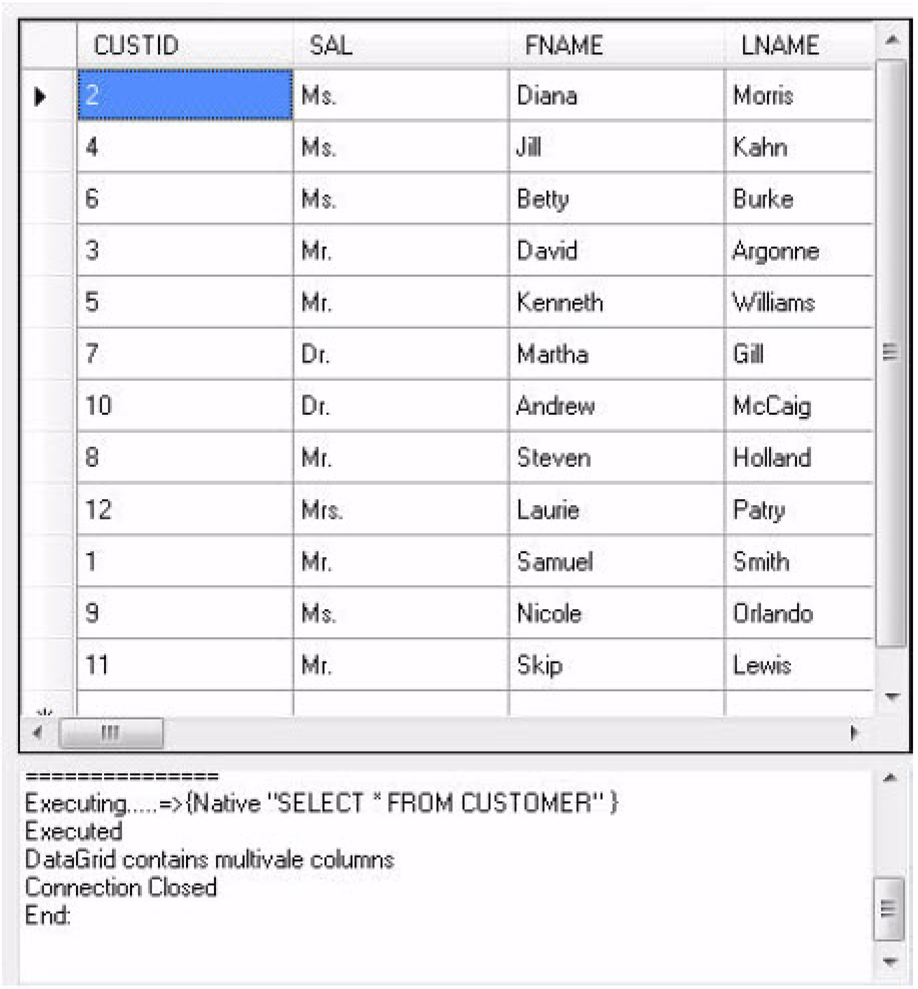
```
C:\U2\UniDK\U2ODBC_64bit\U2ODBC_Tester
```

To view a Native function:

1. Select **Samples** from the menu options at the top of the tool, then select **UniVerse Native Syntax** from the menu, as shown:



2. Click **Execute**. The results appear in the output view, as shown:



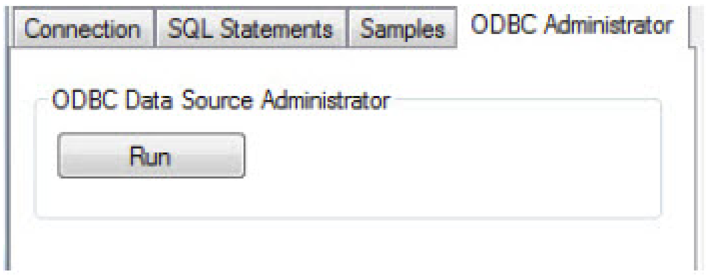
| CUSTID | SAL | FNAME | LNAME |
|---|---|---|---|
| 2 | Ms. | Diana | Morris |
| 4 | Ms. | Jill | Kahn |
| 6 | Ms. | Betty | Burke |
| 3 | Mr. | David | Argonne |
| 5 | Mr. | Kenneth | Williams |
| 7 | Dr. | Martha | Gill |
| 10 | Dr. | Andrew | McCaig |
| 8 | Mr. | Steven | Holland |
| 12 | Mrs. | Laurie | Patry |
| 1 | Mr. | Samuel | Smith |
| 9 | Ms. | Nicole | Orlando |
| 11 | Mr. | Skip | Lewis |

```
===============
Executing.....=>{Native "SELECT * FROM CUSTOMER" }
Executed
DataGrid contains multivale columns
Connection Closed
End:
```

## Accessing the ODBC Data Source Administrator

The ODBC Data Source Administrator can be accessed from within the U2 ODBC Tester. This allows you to easily edit your DSNs.

To access the ODBC Data Source Administrator from within the U2 ODBC Tester, select the **ODBC Administrator** tab and then click **Run**, as shown:



The ODBC Data Source Administrator opens. Note that you must exit the U2 ODBC Tester when you edit a DSN to apply the changes.

# Chapter 8: Tracing and logging

This chapter describes the logging facilities available in U2 ODBC.

## U2 ODBC log files

The U2 ODBC driver maintains a log file called u2odbc.log to monitor the progress of U2 ODBC and provide administrative feedback on the use of U2 ODBC. These log files are also useful for diagnosing problems.

For each connection, these log files contain information about startup and connecting, followed by information about each client request paired with the UniData or UniVerse database server's response, in sequence. The log can also include messages generated by certain UniData or UniVerse database server errors.

### Log file names

The log file names in U2 ODBC have the following format:

```
u2odbc_130312114312.log
```

## Creating the U2 ODBC log file

Usually log files are temporary and are automatically deleted following a client disconnect request and normal database server exit. However, if an error occurs, you may want to retain the log file to track down problems. To create the U2 ODBC log file using environment variables, complete the following steps:

1. Navigate to System Properties by clicking **Start > Control Panel > System** and selecting **Advanced system settings**.
2. Select the **Advanced** tab and click **Environment Variables**.
3. From the System variables group, click **New**.
4. In the New System Variable dialog box, enter **U2ODBC_LOG_PATH** as the name and **c:\temp** as the Variable value.

**Note:** After diagnostics have completed, click **Delete** to delete the U2ODBC_LOG_PATH environment variable.

### Locating the U2 ODBC log file

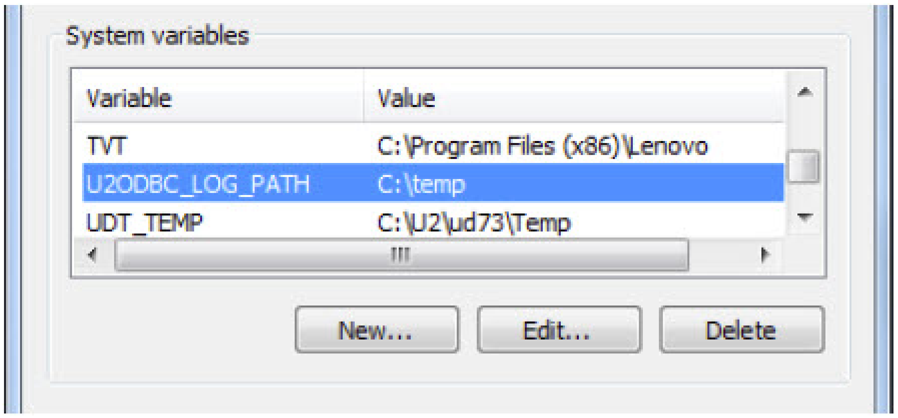
The u2odbc.log file is created in the current working directory. You can distinguish between the 32-bit and the 64-bit version of the U2 ODBC driver and the UniVerse 64-bit ODBC driver by the header information found in the log file.

U2 ODBC users who want to view the log file are encouraged to define a default path to the log file, U2ODBC_LOG_PATH, using the system's environment variables. For more information, see Defining a trace file in the environment variables, on page 80.

## Defining a trace file in the environment variables

By using the tracing feature, you can create logs of events between clients and the database through the server. Logs enable support personnel to help troubleshoot problems. You can create a trace file for database entries in the application's environment variables.

1. To access the environment variables, navigate to **Start > Control Panel > All Control Panel Items > System**.
2. Click **Advanced system settings** and select the **Advanced** tab.
3. Select **Environment Variables** to add or edit the environment variables for your system.
4. From the **System Variable** menu, select **New**.
5. In the name field, define the variable name as **U2ODBC_LOG_PATH**.
6. In the variable value field, define a location for the log file. In the following example, the log file path is defined as c:\temp:



System variables

| Variable | Value |
|---|---|
| TVT | C:\Program Files (x86)\Lenovo |
| U2ODBC_LOG_PATH | C:\temp |
| UDT_TEMP | C:\U2\ud73\Temp |

7. Click **OK**.

**Note:** When the U2ODBC_LOG_PATH variable is defined, a log file is generated each time the ODBC client is invoked. For this reason, it is highly recommended that users remove this variable when the log file is no longer required for debugging purposes. Failure to remove the U2ODBC_LOG_PATH variable when the Create Debug Log option is not selected in the U2 ODBC Data Source Setup may cause performance issues. When the Create Debug Log option is selected, the log files are preserved on the client, which may cause space issues to develop over time.

# Part V: Supplementary information for UniVerse

# Chapter 9: Working with Microsoft Excel Power Query

The U2 ODBC client supports Microsoft Excel Power Query from U2 Common client. It offers several options for loading queries into Microsoft Excel workbook.

1. From Microsoft Excel, select **Data** → **Get Data** → **From other Sources** → **Blank Query** to start Power Query Editor.
2. In the Power Query Tool, select **New Source** → **Other Sources** → **ODBC**.
3. On the From ODBC pane, select the ODBC data source name (DSN) you want from the drop-down box.

   The DSN name was created using the Microsoft ODBC Administrator tool.
4. From the Navigator pane, select the table name you want from the list.

   When you select the table name, the table output should be on the right side of Navigator pane.
5. Click **OK**.

   The Power Query Editor pane shows the output result.

# Chapter 10: ODBC usage notes

## Data types in SQL

U2 ODBC supports the following SQL data types. Use the UniVerse SQL data types in CREATE TABLE and ALTER TABLE statements, and the corresponding ODBC SQL data types in calls to the ODBC API.

| UniVerse SQL Type | ODBC SQL Type |
|---|---|
| CHARACTER [(n)] | SQL_CHAR |
| DATE | SQL_DATE |
| DECIMAL [(*p*[,*s*])] | SQL_DECIMAL |
| DOUBLE PRECISION | SQL_DOUBLE |
| FLOAT [(*p*)] | SQL_FLOAT |
| INTEGER | SQL_INTEGER |
| NUMERIC [(*p*[,*s*])] | SQL_NUMERIC |
| REAL | SQL_REAL |
| SMALLINT | SQL_SMALLINT |
| TIME | SQL_TIME |
| VARCHAR [(*n*)] | SQL_VARCHAR[1] |

[1]The 254-character limit for strict VARCHAR definition is relaxed to allow longer data values, although some ODBC application tools may assume this limit and truncate longer values.

For information about specifying UniVerse SQL data types in UniVerse files, see SQL data types, on page 40.

## Support for core ODBC SQL grammar

In response to **SQLGetInfo**, U2 ODBC says that it supports the core ODBC SQL grammar. More accurately, it supports *most* of the core grammar.

The parts it does not support are:

- Table name qualifiers in DDL.

  For example: `DROP TABLE MY_SCHEMA.MY_TABLE;`
- The CASCADE and RESTRICT qualifiers of the REVOKE statement (UniVerse only). For example:

  ```
  REVOKE UPDATE, DELETE ON MY_TABLE FROM PUBLIC CASCADE;
  ```
- LIKE USER. For example:

  ```
  SELECT * FROM MY_TABLE WHERE NAME LIKE USER;
  ```
- Qualified index names in CREATE INDEX, and unqualified names in DROP INDEX. For example:

  ```
  CREATE INDEX MY_SCHEMA.MY_INDEX ON MY_TABLE (MY_COLUMN);DROP INDEX
  MY_INDEX;
  ```

## Support for extended ODBC SQL grammar

U2 ODBC supports some of the extended ODBC SQL grammar.

- SELECT…FOR UPDATE statement
- Outer joins
- UNION clause
- ESCAPE clause in the LIKE predicate
- Date and time types
- Scalar functions

### SELECT…FOR UPDATE statement

In UniVerse, U2 ODBC supports the SELECT…FOR UPDATE version of the SELECT statement. The syntax is as follows:

```
SELECT [ ALL | DISTINCT ] column_specifications
FROM table_specification
[ WHERE clause ]
FOR UPDATE [ OF column [ , column ] … ]
```

SELECT…FOR UPDATE locks selected rows with exclusive record or file locks, letting users update or delete the selected rows without having to wait for other users' or processes' locks to be released. For example:

```
SELECT ORDERS.CUSTNAME, CUSTOMER.NAME
FROM ORDERS, CUSTOMER
WHERE ORDERS.CUSTNO = CUSTOMER.CUSTNO
FOR UPDATE OF CUSTOMER.NAME;
```

**Parent topic:** Support for extended ODBC SQL grammar

### Outer joins

U2 ODBC supports left outer joins in statements such as:

```
SELECT CODE AS STATE, COMPANY
FROM OJ STATES
LEFT OUTER JOIN
CUSTOMER
ON STATES.CODE = CUSTOMER.STATE;
```

However, U2 ODBC does not support nested left outer joins, such as:

```
SELECT CODE AS STATE, COMPANY, PRODID
FROM OJ STATES
LEFT OUTER JOIN
CUSTOMER
LEFT OUTER JOIN
CUSTOMER_ORDERS
ON CUSTOMER.CUSTID = CUSTOMER_ORDERS.CUSTID
ON STATES.CODE = CUSTOMER.STATE;
```

In UniVerse SQL, the table expression on the left side of a left outer join can itself be a left outer join, but the table expression on the right side cannot. In ODBC SQL, the reverse is true: only the table expression on the right side can be a left outer join. Therefore, in U2 ODBC SQL, neither the left nor the right table expression in a left outer join can be a left outer join.

**Parent topic:** Support for extended ODBC SQL grammar

## UNION clause

U2 ODBC supports the UNION clause as given in the ODBC grammar. For example:

```
SELECT SCHEMA_NAME FROM CATALOG.UV_SCHEMA
UNION
SELECT TABLE_NAME FROM CATALOG.UV_TABLES;
```

**Parent topic:** Support for extended ODBC SQL grammar

## ESCAPE clause in the LIKE predicate

U2 ODBC supports the ESCAPE clause in the LIKE predicate, which lets you use % and _ as literal characters in a search pattern. For example:

```
SELECT * FROM PROJECTS
WHERE STATUS LIKE '%80\% complete%' ESCAPE '\';
```

**Parent topic:** Support for extended ODBC SQL grammar

## Date and time types

U2 ODBC supports DATE and TIME in CREATE TABLE and ALTER TABLE statements. It also supports date and time literals. For example:

```
CREATE TABLE HALS (BIRTHDAY DATE);
INSERT INTO HALS VALUES ({d '1997-01-12'});
```

**Parent topic:** Support for extended ODBC SQL grammar

## Scalar functions

The following table lists U2 ODBC support of scalar functions.

| | Function | Supported in UniData | Supported in UniVerse |
|---|---|---|---|
| String | ASCII | Yes | No |
| | CHAR | No | No |

| | Function | Supported in UniData | Supported in UniVerse |
|---|---|---|---|
| | CONCAT | | Yes |
| | DIFFERENCE | | No |
| | DOWNCASE | Yes | No |
| | INSERT[1] | | Yes |
| | INITCAP | Yes | No |
| | INSTR | Yes | No |
| | LCASE | | Yes |
| | LEFT | | Yes |
| | LENGTH | Yes | Yes |
| | LOCATE | No | No |
| | LPAD | Yes | No |
| | LTRIM | Yes | Yes |
| | REPEAT | No | No |
| | REPLACE | No | No |
| | RESUSE | Yes | No |
| | RIGHT[1] | No | Yes |
| | RPAD | Yes | No |
| | RTRIM | Yes | Yes |
| | SOUNDEX | Yes | No |
| | SPACE | No | No |
| | SUBR | Yes | No |
| | SUBSTR | Yes | No |
| | SUBSTRING | Yes | Yes |
| | UCASE | No | Yes |
| | UPCASE | Yes | No |
| | UPPER | Yes | No |
| | VSIZE | | |
| | | | |
| Numeric | ABS | Yes | No |
| | ACOS | Yes | No |
| | ASIN | Yes | No |
| | ATAN | Yes | No |
| | ATAN2 | No | No |
| | AVG | Yes | No |
| | CEILING | No | No |
| | COS | Yes | No |
| | COT | No | No |
| | Count | Yes | No |
| | DECODE | Yes | No |
| | DEGREES | No | Yes |
| | EXP | No | No |
| | FLOOR | No | No |
| | INT | Yes | No |

| | Function | Supported in UniData | Supported in UniVerse |
|---|---|---|---|
| | LN | Yes | No |
| | LOG | No | No |
| | LOG10 | No | No |
| | Max | Yes | No |
| | Min | Yes | No |
| | MOD | Yes | No |
| | PI | No | Yes |
| | POWER | Yes | No |
| | RADIANS | No | Yes |
| | RAND | No | No |
| | ROUND | Yes | No |
| | SIGN | Yes | No |
| | SIN | Yes | No |
| | SQRT | Yes | No |
| | STDDEV | Yes | No |
| | SUM | Yes | No |
| | TAN | Yes | No |
| | TRUNC | Yes | No |
| | TRUNCATE | No | No |
| | VARIANCE | Yes | No |
| | | | |
| Time and Date | ADD_MONTHS | Yes | No |
| | CURDATE | No | No |
| | CURTIME | No | No |
| | DATEADD | Yes | No |
| | DATEDIFF | Yes | No |
| | DATEPART | Yes | No |
| | DAYNAME | No | No |
| | DAYOFMONTH | No | No |
| | DAYOFWEEK | No | No |
| | DAYOFYEAR | No | No |
| | GETDATE | Yes | No |
| | HOUR | No | No |
| | LAST_DAY | Yes | No |
| | MINUTE | No | No |
| | MONTH | No | No |
| | MONTHNAME | No | No |
| | MONTHS_BETWEEN | Yes | No |
| | NEXT_DAY | Yes | No |
| | NL1_KEY | Yes | No |
| | NL2_KEY | Yes | No |
| | NOW | No | No |
| | QUARTER | No | No |

| | Function | Supported in UniData | Supported in UniVerse |
|---|---|---|---|
| | SECOND | No | No |
| | TIMESTAMPADD | No | No |
| | TIMESTAMPDIFF | No | No |
| | WEEK | No | No |
| | YEAR | No | No |
| | | | |
| System | DATABASE | Yes | Yes |
| | IFNULL | No | No |
| | ISNULL | Yes | No |
| | NVL | Yes | No |
| | USER | | Yes |
| | USER_ID | Yes | No |
| | USER_NAME | Yes | No |
| | | | |
| Conversion | CONVERT[2] | | Yes |
| | ICONV | Yes | No |
| | OCONV | Yes | No |
| | TO_CHAR | Yes | No |
| | TO_DATE | Yes | No |
| | TO_NUMBER | Yes | No |

1 INSERT and RIGHT may evaluate their arguments more than once, consequently you cannot use a parameter marker as an argument to these functions, and you must watch out for side effects if you use an I-descriptor.

2 U2 ODBC supports the same conversions that UniVerse SQL CAST supports.

An example of a call to a scalar function is as follows:

```
SELECT * FROM UV_SCHEMA
WHERE SCHEMA_NAME = {fn DATABASE()};
```

ODBC implies that no scalar functions can take an arithmetic expression or a parameter marker as an argument. U2 ODBC tolerates violations of this rule, but in the future it may optionally catch them and give an error message.

**Parent topic:** Support for extended ODBC SQL grammar

## UniVerse extensions to ODBC SQL grammar

U2 ODBC supports the following driver-specific extensions to the SQL grammar.

The following extensions are always available:

- Mixed-case keywords
- Multiline statements
- End-of-line comments beginning with two hyphens (--)
- The {NATIVE}statement
- The ORDER BY clause in SELECT…FOR UPDATE statements
- The COUNT (*expression*) clause

- The EXPLAIN and NOWAIT keywords
- The concatenation operator (||)
- The following scalar functions:
  - CHAR[ACTER]_LENGTH
  - CONCAT
  - LOWER
  - UPPER
- The following values in the VALUES clause of an INSERT statement:
  - Literals
  - The null value
  - USERODBC scalar functions
  - The UniVerse scalar functions CHAR_LENGTH, CONCAT, LOWER, and UPPER
  - Combinations of the previous with arithmetic operators, the concatenation operator, or ODBC scalar functions
- SOME instead of ANY or ALL before a subquery

Refer to the following additional information regarding UniVerse extensions to ODBC SQL grammar:

- NOWAIT keyword
- EXPLAIN keyword
- SQL usage notes and restrictions
- Procedures

## NOWAIT keyword

U2 ODBC supports the NOWAIT keyword at the end of the following statements:

- SELECT (including SELECT…FOR UPDATE)
- INSERT
- UPDATE
- DELETE

NOWAIT prevents users from having to wait if another user or process has a file or record lock that prevents the database operation from proceeding. For example:

```
INSERT INTO SALESMEN
SELECT * FROM EMPS
WHERE DEPTCODE = 'SALES'
NOWAIT;
```

**Parent topic:** UniVerse extensions to ODBC SQL grammar

## EXPLAIN keyword

U2 ODBC supports the EXPLAIN keyword for use in SELECT statements. EXPLAIN returns information about how the statement will be processed, letting users decide if they want to rewrite the query more efficiently. The SELECT statement is not executed.

**Parent topic:** UniVerse extensions to ODBC SQL grammar

## SQL usage notes and restrictions

The following issues also affect SQL usage in U2 ODBC:

- INSERT statements, on page 90
- Parameter markers, on page 90
- No DDL on files, on page 90
- Tolerance of ODBC-compliant names, on page 90
- Column aliases, on page 90

**Parent topic:** UniVerse extensions to ODBC SQL grammar

### INSERT statements

INSERT statementsUniVerse handles SQL INSERT statements that do not include an explicit column list as documented in *UniVerse SQL Reference*. For files, the values listed in the VALUES clause should match the fields defined in the @INSERT phrase. For tables, the values should match the columns defined in the SICA.

### Parameter markers

Parameter markersODBC SQL has certain restrictions on the use of parameter markers. U2 ODBC tolerates some violations of these restrictions, but in the future it may optionally catch them and give an error message.

### No DDL on files

SQL DDL statements that create, drop, alter, grant, or revoke privileges on tables are permitted on tables and views only. To use DDL statements on a file, the file must be converted to a table.

### Tolerance of ODBC-compliant names

Tolerance of ODBC-compliant namesWhen name mapping is on (the default), and the **Retry as UniVerse SQL** option is off, U2 ODBC returns an error if an SQL statement includes an ODBC qualifier name, table name, or column name that is not ODBC-compliant. However, U2 ODBC does not check the ODBC compliance of index names, correlation names, or column aliases.

### Column aliases

The U2 ODBC interpretation of column aliases is closer to that of SQL-92 than to that of UniVerse. This interpretation is better suited for the default behavior of ODBC. U2 ODBC permits references to column aliases only in ORDER BY clauses. As of Release 9, UniVerse permits references to column aliases in other places.

## Procedures

All UniVerse procedures are available through U2 ODBC as ODBC procedures. These include, but are not limited to, UniVerse BASIC programs and paragraphs. For details about UniVerse procedures, see *UCI Developer's Guide* or *UniVerse BASIC SQL Client Interface Guide*. For details about ODBC procedures, see *Microsoft ODBC 3.0 Programmer's Reference and SDK Guide*.

To invoke procedures using U2 ODBC, you need to write custom SQL statements, and, depending on your ODBC application, you may need to write custom ODBC calls.

**Parent topic:** UniVerse extensions to ODBC SQL grammar

### Syntax

U2 ODBC supports the following ODBC syntax for procedure calls:

- In UniVerse: {CALL *procedure* [(*parameter1*, … *parameter* n)] }
- In UniData: CALL *procedure* [(*parameter1*, … *parameter* n)]

The braces are part of the syntax and must be specified. You can specify parameters as bound parameters (?) or as literals.

You can specify parameters only when calling a UniVerse BASIC subroutine. When calling other UniVerse procedures, enclose the entire command in double quotation marks. For example:

```
{CALL "LIST CUSTOMER WITH CUSTID = 2"}
```

If procedure contains periods or is otherwise not ODBC-compliant, you must enclose it in double quotation marks. For example:

```
{CALL "MY.BASIC.SUBROUTINE" (?, ?, ?)}
```

You might like to call MY.BASIC.PROGRAM as MY_BASIC_PROGRAM, but this is not possible because U2 ODBC does not apply its name-mapping feature to procedure names.

### Parameters

You can bind a procedure parameter using any SQL type that U2 ODBC supports (see Data types in SQL, on page 83). However, the UniVerse engine itself treats all procedure parameters as character strings. Therefore, binding such a parameter using an SQL type other than SQL_CHAR or SQL_VARCHAR makes U2 ODBC check that the value of the parameter is compatible with the SQL type, and generate an error or warning if it is not.

You can bind any procedure parameter as input, input/output, or output. If you bind a procedure parameter as input/output or output and successfully execute the procedure, U2 ODBC always returns some value for the parameter.

In U2 ODBC, the values of output parameters are available immediately after the return of **SQLExecute** or **SQLExecDirect**. But this is not true of all ODBC drivers; according to *Microsoft ODBC 3.0 Programmer's Reference and SDK Guide*, a driver is not guaranteed to return the output parameters of a procedure until after all of the procedure's results have been fetched. For maximum interoperability, you should not attempt to read output parameter values until then.

If an error occurs during the process of returning a value for an output parameter, U2 ODBC may return a null value for the parameter.

## Result sets

A procedure may or may not return a result set.

The number and types of columns in the result set of a procedure are not available until the procedure is executed. **SQLPrepare** immediately followed by **SQLNumResultCols**, **SQLDescribeCol**, or **SQLColAttributes** gives an error.

The number and types of columns in the result set of a procedure may change from execution to execution.

## Behavior of SQLRowCount

**SQLRowCount** returns a value of –1 after a procedure that returns a result set.After a procedure which does not return a result set, **SQLRowCount** returns what the UCI **SQLRowCount** function returns. This might not be what you expect; see *UCI Developer's Guide* for more information.

## Limitations

U2 ODBC does not support the following ODBC procedure functionality:

- Return values from procedures. For example, the following statement generates an error:

{? = CALL MYPROC(?, ?)}The ? = is used for procedures with return values. UniVerse procedures can return data through any parameter, but they do not have return values as such, so U2 ODBC does not support this feature.

- Procedure names qualified with a qualifier or an owner. For example, the following statement generates an error:

{CALL MYSCHEMA.MYPROC(?, ?)}This syntax is legal in ODBC data sources in which procedures can be qualified with a qualifier or owner name, or both. It is illegal in U2 ODBC because UniVerse does not support qualified procedure names. See *UCI Developer's Guide*or *UniVerse BASIC SQL Client Interface Guide* for more information.If your procedure name contains periods, it must be quoted, as noted earlier.

- Catalog information.

ODBC defines two functions that return catalog information about procedures. **SQLProcedures** returns a list of procedures. **SQLProcedureColumns** returns a list of parameters and of columns in the result sets of procedures. In U2 ODBC these functions always return empty result sets.Consequently, to call a procedure, you must already know its name and the number and types of its parameters. Each time you execute a procedure, if the execution produces a result set, you can find out about the number and types of columns in the result set.

- Default values of procedure parameters.

## Interoperability with generic ODBC applications

The U2 ODBC implementation of procedures differs from that of other ODBC drivers in these respects:

- **SQLProcedures** and **SQLProcedureColumns** always return empty result sets.

- You cannot call **SQLPrepare**, and then call **SQLNumResultCols**, **SQLDescribeCol**, or **SQLColAttributes** without first calling **SQLExecute**.
- You cannot prepare a procedure, then execute it multiple times, with the guarantee that the number and types of columns in its result set will be the same after each execution.

  A generic ODBC application may not expect this behavior. Therefore, to use U2 ODBC procedures, you probably need to write custom SQL calls, and depending on your ODBC application, you may need to write custom ODBC calls.

## Executing UniVerse SQL

You can pass an ODBC application's SQL syntax directly to UniVerse. This bypasses U2 ODBC's SQL processor, which performs operations such as parsing, name mapping, and validating SQL syntax, and directly accesses UniVerse's SQL interpreter. You can do this using one of the following methods:

- Using the U2 ODBC-specific {NATIVE} syntax extension
- Configuring the ODBC data source to retry statements as UniVerse SQL
- Configuring the ODBC data source not to require strict ODBC syntax

### The {NATIVE} Syntax extension

To let you use native SQL syntax that conflicts with ODBC-standard SQL, U2 ODBC supports an SQL extension that allows the direct pass-through of native statements (SQL or other database-specific commands) to the database without interpretation. The syntax for this extension is:

{NATIVE *statement* }

The braces are part of the syntax and must be specified. *statement* is the single-or double-quoted string to pass to the underlying database without trans-lation. If you use single quotation marks to delimit *statement* and want to include a literal single quotation mark in *statement*, double the single quotation mark. Similarly, if you use double quotation marks to delimit *statement* and you want to include a literal double quotation mark in *statement*, double the double quotation mark.

U2 ODBC does not support name mapping or file access control in this extension.

### Retrying statements as UniVerse SQL

In an ODBC connection, retrying statements as UniVerse SQL can be on or off. It is off by default.

When retrying statements as UniVerse SQL is on, if U2 ODBC fails to interpret an SQL statement as a valid ODBC SQL statement, it retries the statement; that is, it tries to interpret the statement as though it were written in UniVerse SQL. If your ODBC application was developed to use the special features of UniVerse SQL, you may want to turn this option on by setting the HS_RETRY_AS_UV_SQL environment variable to YES or ON, or by clicking **Yes** at the **Retry as UniVerse SQL** in the U2 ODBC Data Source Setup dialog box.

When this option is off, U2 ODBC always tries to interpret SQL as ODBC SQL (unless a statement conforms to U2 ODBC's native SQL extension, in which case, U2 ODBC always passes the statement through). If you want your ODBC application to be portable across multiple databases, do not activate this option. In addition, you can set one or more of the options that enforce ODBC syntax. By doing so you ensure that the application's SQL statements are portable across databases and provide uniform error messages when SQL does not meet portability standards.

Retrying statements as UniVerse SQL takes effect only when name mapping is off. This option bypasses the translation of the client's SQL statement from ODBC SQL to UniVerse SQL, in which ODBC names are converted to UniVerse equivalents. If you want to use UniVerse extensions directly through this option, you also must use UniVerse native file, table, and column names in your SQL statements.

U2 ODBC does not support file access control in retried statements.

Retrying statements as UniVerse SQL is useful mainly to support custom ODBC applications that "know" they are accessing UniVerse databases. You should turn this option off when you use it with commercial ODBC-aware applications.

## SQL-related connection options

The client can use the U2 ODBC Data Source Setup dialog box to set many SQL-related U2 ODBC connection options. For more information, see Configuring UCI for the U2 ODBC driver, on page 16.

## Transaction support

U2 ODBC supports two modes of transaction behavior:

- Autocommit mode (the default when the server starts)
- Manual mode

### Autocommit mode

In autocommit mode, U2 ODBC treats every SQL statement received from a client as a separate transaction within the data session. It commits every statement that could have modified the database.

DDL statements such as CREATE TABLE are allowed only when the driver is in autocommit mode.

#### Multiple statement handles

Whenever a transaction is committed or rolled back, all open statement handles on that connection are closed. If your application is using autocommit mode and it executes an INSERT, UPDATE, or DELETE SQL statement on one ODBC statement handle, all other statement handles (such as for SELECT statements) are closed and cannot be used for additional SQLFetch operations. If you want to use SQLFetch to fetch rows one at a time from one SELECT statement, and you want to execute INSERT, UPDATE, or DELETE statements on another ODBC statement handle (this might be typical for a records-maintenance application), use manual mode.

For more information on cursor commit behavior and transaction modes in ODBC, see *Microsoft ODBC 3.0 Programmer's Reference and SDK Guide*.

### Manual mode

In manual mode the server runs the SQL statements as they come in from the client, but it defers committing changes to the database until it receives an explicit commit or rollback command from the client.

ODBC client applications request transaction commits or rollbacks internally by calling the ODBC SQLTransact function on the U2 ODBC driver. Each ODBC client application has its own user interface to control transactions.

## ODBC table types

An ODBC table's type is presented in the TABLE_TYPE column of the SQLTables result set. U2 ODBC maps UniVerse file structures to ODBC table types as follows:

| UniVerse File Type | ODBC Table Type |
|---|---|
| Table | TABLE |
| View | VIEW |
| File | TABLE |

# Chapter 11: UniVerse Error Messages

## Message descriptions

Each message description includes the Error ID, Severity, and Facilities codes. Because some errors originate in other products such as your database or TCP/IP product, it is not possible to list every error message. Instead, the word *STRING* refers to a portion of the message that varies depending on the server platform or database. In these cases, you may need to refer to your product's documentation for details. Errors are returned as a set of codes to an application program or as text to a user. In either case, an error consists of four components:

| Component | Description |
|---|---|
| Message | A brief description of the error, displayed on the screen. |
| Error ID | The numeric code that identifies the specific error or warning. |
| Severity | The error's level of severity. Informational or warning messages do not prevent further processing. Errors, severe errors, or fatal errors usually do. The severity levels are as follows:<br><br>SUCCESS – Successful completion.<br><br>INFO – Successful completion with extra information.<br><br>WARNING – Completion with information such as truncation or roundoff errors.<br><br>ERROR – Failure to complete operation. U2 ODBC is still functioning, but an exceptional situation is reported (such as a locked record or invalid parameters).<br><br>SEVERE – Failure to complete operation and U2 ODBC system failure. Try to save what you can and discontinue.<br><br>FATAL – Failure of the U2 ODBC system. This is most commonly caused by the lack of virtual memory and by communications failures. Continued use of the U2 ODBC connection is likely to fail. |
| Facility | The source of the error, usually the software module from which the error originated. This information is useful when calling technical support. |

## Messages

A CREATE VIEW statement's name list did not correspond to the query defining the view.

Error ID: 10, Severity: ERROR, Facility: DBCAPERR

ADDing a column to a table failed because there is already a column with the specified name.

Error ID: 30, Severity: ERROR, Facility: DBCAPERR

An INSERT statement contained the wrong number of values for the target table.

An INSERT statement contained the wrong number of values for the target table. Verify that the number of parameter markers and values supplied are consistent with the table's definition.

Error ID: 9, Severity: ERROR, Facility: DBCAPERR

An invalid HSTMT was encountered.

The HSTMT associated with this SQL statement was either invalid or equal to SQL_NULL_HSTMT when it should have been valid.

Error ID: 61, Severity: ERROR, Facility: DBCAPERR

An invalid parameter type has been specified.

Error ID: 38, Severity: ERROR, Facility: DBCAPERR

An operation in progress could not be canceled.

Error ID: 24, Severity: ERROR, Facility: DBCAPERR

A positioned UPDATE or DELETE failed to modify any rows.

An UPDATE or DELETE WHERE CURRENT OF cursorname did not modify any rows in the SELECT FOR UPDATE statement on cursorname.

Error ID: 7, Severity: ERROR, Facility: DBCAPERR

A positioned UPDATE or DELETE modified more than 1 row.

An UPDATE or DELETE WHERE CURRENT OF cursorname affected more rows than the last row fetched by cursorname.

Error ID: 8, Severity: ERROR, Facility: DBCAPERR

Argument to Scalar Function not compatible with parameter type.

A literal value supplied to a scalar function is not compatible with the required parameter type.

Error ID: 42, Severity: ERROR, Facility: FPSRVERR

Argument value out of range.

An argument value was outside the allowable range for this request.

Error ID: 45, Severity: ERROR, Facility: FPSRVERR

Attempt to connect to a database which uses string and/or identifier delimiters which U2 ODBC does not support.

U2 ODBC supports only those databases in which the delimiter character for SQL string literals is single quotation marks, and the delimiter character for SQL identifiers is double quotation marks. The user attempted to connect to a database that did not have this property.

Error ID: 66, Severity: ERROR, Facility: DBCAPERR

Bad socket number.

This error is an internal communications error. Disconnect and reconnect. If that fails, reboot the PC.

Error ID: 101, Severity: SEVERE, Facility: LINKERR

CREATE INDEX failed because the specified index already exists.

CREATE INDEX failed because the specified index already exists.

Error ID: 28, Severity: ERROR, Facility: DBCAPERR

CREATE TABLE/VIEW failed because a table or view with that name already exists.

CREATE TABLE or CREATE VIEW failed because a table or view with that name already exists.

Error ID: 26, Severity: ERROR, Facility: DBCAPERR

Cannot interpret input data. Versions incompatible.

Attempt to read a file or connect to a server failed because of incompatible software versions.

Error ID: 15, Severity: FATAL, Facility: OIOERR

Cannot read a directory as a file.

Attempt to read data from a directory failed. Check your file pathname.

Error ID: 11, Severity: ERROR, Facility: OIOERR

Client timed out waiting for initial server response.

The U2 ODBC driver timed out waiting for the initial response from the UniVerse database server. If the client's login timeout interval is too short, you can increase it by setting the SQL_LOGIN_TIMEOUT connection option to a higher value.

Error ID: 5, Severity: ERROR, Facility: OCLIERR

Column number out of bounds.

A numeric reference to a column failed because the column number was out of bounds.

Error ID: 32, Severity: ERROR, Facility: DBCAPERR

Communications driver not loaded.

TCP/IP network driver software has not been loaded or is not running. Check to see that it has been installed correctly.

Error ID: 3, Severity: FATAL, Facility: OIOERR

Communications error. STRING.

A communications error has occurred. The connection has been closed. Retry the connection. If this message persists, contact U2 Customer Support.

Error ID: 123, Severity: SEVERE, Facility: LINKERR

Connection closed from server.

The UniVerse database server process has terminated abnormally. This could be caused by the host computer being rebooted, or it could indicate a problem in the server. Check that the server is running. Disconnect and reconnect. If that fails, reboot the PC.

Error ID: 109, Severity: SEVERE, Facility: LINKERR

Connection has been aborted.

This error can result from a variety of situations, such as the host is shut down, the host operator force logged out the server process, the host inactivity time-out expired and logged out the server process, TCP/IP has failed, or a fatal error occurred to the server process. Retry the connection. If this fails, reboot your PC and try again. If this fails, check to see if the host and TCP/IP are running.

Error ID: 102, Severity: SEVERE, Facility: LINKERR

Connection has closed.

The U2 ODBC driver is having trouble initiating communication to the server. This is often caused by invalid or no parameters in the configuration file uci.config.

This error can also result from a variety of situations, such as the host is shut down, the host operator force logged out the server process, the host inactivity time-out expired and logged out the server process, TCP/IP has failed, or a fatal error occurred to the server process. Retry the connection. If this fails, reboot your PC and try again. If this fails, check to see if the host and TCP/IP are running.

Error ID: 103, Severity: SEVERE, Facility: LINKERR

Connection is not open.

The U2 ODBC driver is having trouble initiating communication to the server. This is often caused by a host being down or by invalid link parameters in the configuration file.

Use UCI Config Editor to check the spelling of the host name.

Retry the connection. If this fails, reboot your PC and try again. If this fails, check to see if the host and TCP/IP are running.

Error ID: 104, Severity: SEVERE, Facility: LINKERR

Connection or statement option was set to a value other than that specified.

The driver does not support the requested value of the specified option and substituted a similar value.

Error ID: 6, Severity: WARNING, Facility: DBCAPERR

Connection timed out.

This reports an attempt to connect to a host that did not accept the connection within the time limit set in the local PC TCP software (usually several minutes). This can happen if the host is not running TCP/IP, or is not running at all. It can also occur if the system is very heavily loaded and other U2 ODBC users are also trying to connect at the same time. Check the host system and try again.

This error can also occur during the course of normal communications if the host or client system becomes very heavily loaded or hangs. In this case, check both the host and the client. If the problem persists, reboot both systems and try again.

Error ID: 107, Severity: SEVERE, Facility: LINKERR

Conversion method unknown.

The type of an argument supplied to a conversion function is not compatible with the type expected by that function. If the ODBC data source is configured with Enforce ODBC Function Argument Types set, arguments to conversion functions must be compatible with expected data types.

Error ID: 44, Severity: ERROR, Facility: FPSRVERR

Conversion of a value to a numeric type caused an underflow.

A text- or number-to-number conversion failed because the number was too small or too far negative to be represented in the specified format.

Error ID: 26, Severity: ERROR, Facility: DATUMERR

Could not login to database STRING as user STRING.

A database session could not be established because the database name, user name, or password was invalid.

Error ID: 3, Severity: FATAL, Facility: DBCAPERR

Could not start server. STRING.

The server process could not be started for the indicated reason. Correct the problem and try again.

Error ID: 124, Severity: SEVERE, Facility: LINKERR

Could not make name unique: more than 10 million names with the same stem.

Could not find a _nnnnnnn suffix for a name, which would make it unique within a space of names because 9,999,999 suffixes have already been given out for that name. This is an extremely rare error. Contact U2 Customer Support.

Error ID: 45, Severity: ERROR, Facility: DATUMERR

Cursor could not honor a FETCH_SAME_AS_LAST request.

The database interface could not fetch the same number of rows as the last request because there was no last request.

Error ID: 11, Severity: ERROR, Facility: FPSRVERR

Cursor not found with given name.

The indicated cursor name does not correspond to any allocated cursor.

Error ID: 5, Severity: ERROR, Facility: FPSRVERR

Cursor number not currently allocated.

The indicated cursor number is not allocated.

Error ID: 6, Severity: ERROR, Facility: FPSRVERR

DDL statements are illegal within a Transaction.

DDL statements are illegal within a transaction.

Error ID: 62, Severity: ERROR, Facility: DBCAPERR

Database capsule creation failed.

Error ID: 9, Severity: FATAL, Facility: FPSRVERR

Database error.

The database encountered an error while performing the operation. See your database manual for an explanation of the message and external error code.

Error ID: 1, Severity: ERROR, Facility: DBCAPERR

Database name/uid/password parameters must be textual.

User name and password strings must be alphanumeric. If your user name or password could be interpreted as a number, currency, date, or time, enclose it in double quotation marks.

Error ID: 13, Severity: SEVERE, Facility: LINKERR

Database reported a miscellaneous warning.

The database reported a warning to U2 ODBC but did not specify the nature of the problem.

Error ID: 37, Severity: WARNING, Facility: DBCAPERR

Database table integrity constraint violation.

An operation was attempted that would have violated integrity constraints on a database table.

Error ID: 17, Severity: ERROR, Facility: DBCAPERR

Database warning.

The database generated warnings while performing the operation. See your database manual for an explanation of the message and external error code.

Error ID: 2, Severity: WARNING, Facility: DBCAPERR

Data conversion error: STRING.

The data in this column is invalid for the defined SQL data type. The most common reason is nonnumeric data in numeric or date/time columns.

Error ID: 67, Severity: WARNING, Facility: DBCAPERR

Data source does not support default values of parameters of procedures.

Error ID: 62, Severity: ERROR, Facility: FPSRVERR

Data source does not support table name qualifiers.

Error ID: 37, Severity: ERROR, Facility: FPSRVERR

Data source not capable.

A construct was passed to the data source for which no support is provided.

Error ID: 36, Severity: ERROR, Facility: FPSRVERR

Data source not capable due to Dynamic Parameter.

This data source does not support a dynamic parameter value in this position.

Error ID: 43, Severity: ERROR, Facility: FPSRVERR

Data truncated.

All of the data for the specified column, icol, could not be retrieved in a single call to the function.

Error ID: 28, Severity: WARNING, Facility: OIOERR

Data truncated entering or leaving the database.

Either more data was provided than the defined column width, or the column contained more data than the output buffer could hold.

Error ID: 4, Severity: WARNING, Facility: DBCAPERR

Data type conflict.

The data type of a bound statement parameter or literal conflicted with the corresponding table column.

Error ID: 13, Severity: ERROR, Facility: DBCAPERR

Date/Time/Timestamp literal too long.

Error ID: 22, Severity: ERROR, Facility: FPSRVERR

Date/Time overflow.

A date/time parameter or literal in an SQL statement overflowed its allowed range.

Error ID: 14, Severity: ERROR, Facility: DBCAPERR

Date/Time overflow.

A date/time parameter or literal in an SQL statement overflowed its allowed range.

Error ID: 70, Severity: WARNING, Facility: DBCAPERR

Date or Time data truncated entering or leaving the database.

Error ID: 51, Severity: WARNING, Facility: DBCAPERR

Datetime field overflow.

The data for the column was not a valid date, time, or timestamp value.

Error ID: 30, Severity: ERROR, Facility: OIOERR

Disconnection failed because a COMMIT or ROLLBACK was in progress.

The disconnection failed because a COMMIT or ROLLBACK was in progress.

Error ID: 19, Severity: ERROR, Facility: DBCAPERR

Divide by zero error.

A divide-by-zero exception occurred during the execution of an SQL statement.

Error ID: 15, Severity: ERROR, Facility: DBCAPERR

Duplicate cursor name.

A cursor name assignment failed because the name already refers to another cursor.

Error ID: 22, Severity: ERROR, Facility: DBCAPERR

Empty string found in date/time column.

An empty string was found in a column defined by UniVerse SQL as SQL_DATE or SQL_TIME.

Error ID: 68, Severity: WARNING, Facility: DBCAPERR

End of data encountered.

The database indicated that the end of the data associated with this request has been reached.

Error ID: 10, Severity: INFORMATION, Facility: FPSRVERR

Error checking "STRING": STRING.

Some problem occurred when U2 ODBC tried to check the status of a file. Check the external error number against your operating system’s documentation to determine the cause.

Error ID: 6, Severity: ERROR, Facility: OCLIERR

Error fetching BASIC result set: [STRING] STRING.

U2 ODBC encountered an error fetching results from one of the cataloged BASIC helper programs it uses to obtain UniVerse data dictionary information.

Error ID: 57, Severity: ERROR, Facility: FPSRVERR

Error opening "STRING": STRING.

Some problem occurred when U2 ODBC tried to open a file, as reported by the stream. Check the external error number against your operating system’s documentation to determine the cause.

Error ID: 2, Severity: ERROR, Facility: OCLIERR

Error writing "STRING": STRING.

Some problem occurred when U2 ODBC tried writing to a file, as reported by the stream. This error usually occurs when the system is running out of disk space. Check the external error number against your operating system documentation to determine the cause.

Error ID: 3, Severity: ERROR, Facility: OCLIERR

Escape character not length 1.

The LIKE operator escape character must be a single character.

Error ID: 38, Severity: ERROR, Facility: FPSRVERR

Failure reading data from file.

A premature end-of-file or some other problem occurred while an application was reading streamed data from a file. Your file may have been truncated because of the lack of disk space, or somehow corrupted. Reconcile the external error number against your operating system’s error number list for more details.

Error ID: 1, Severity: FATAL, Facility: XIOERR

Failure writing data to file.

Some problem occurred while an application was writing streamed data to a file. Your file may have been truncated because of the lack of disk space, or your disk may be corrupted. Reconcile the external error number against your operating system’s error number list for more details.

Error ID: 2, Severity: FATAL, Facility: XIOERR

File write error, disk probably full.

A client was unable to write data to a disk file. Check free space on your disk.

Error ID: 19, Severity: ERROR, Facility: OIOERR

Floating-point value was truncated to integer.

A floating-point value was truncated during conversion to integer.

Error ID: 43, Severity: WARNING, Facility: DATUMERR

Identifier too big.

An identifier component (qualifier, owner name, table name, or column name) is longer than 128 characters.

Error ID: 27, Severity: ERROR, Facility: FPSRVERR

Incorrectly formatted date/time.

You typed a date/time value that could not be interpreted. Use the ANSIstandard YYYY-MM-DD HH:MM:SS HH:MM format. Note, however, that you cannot include any elements that are outside the precision of the date/time column (that is, you cannot include the time if the column is defined as date-only). Also note that the time zone is optional.

Error ID: 13, Severity: ERROR, Facility: DATUMERR

Incorrectly formatted integer.

You typed a value that could not be interpreted as an integer. Integers consist only of digits, with an optional + or – sign.

Error ID: 22, Severity: ERROR, Facility: DATUMERR

Incorrectly formatted interval.

You typed an interval value that could not be interpreted. Use the ANSIstandard YYYY-MM or DD HH:MM:SS format. Note, however, that you cannot include any elements that are outside the precision of the interval column.

Error ID: 20, Severity: ERROR, Facility: DATUMERR

Incorrectly formatted number.

You typed a value that could not be interpreted as a number. Numbers consist of an optional plus (+) or minus (–) sign, integer part, decimal point, fractional part, and exponent (including “e” for exponent, optional sign, and integer exponent value). For example, 1.2 and –3.4e56

Error ID: 23, Severity: ERROR, Facility: DATUMERR

Index names must be unqualified in CREATE INDEX and qualified in DROP INDEX.

In U2 ODBC SQL, index names must be unqualified in CREATE INDEX statements and qualified in DROP INDEX statements.

Error ID: 61, Severity: ERROR, Facility: FPSRVERR

Input/output parameter bound to non-procedure statement.

Input/output parameters provide only output values when used with procedure calls. Other statements treat I/O parameters as input-only.

Error ID: 45, Severity: WARNING, Facility: DBCAPERR

Internal error. Please report details to your UV/ODBC Support contact: Assorted Internal Errors.

You have encountered an unexpected internal error. Record what you were doing and the complete error information (such as the Error ID, Extern ID, Severity, and Facility). Then report this information to U2 Customer Support.

Invalid Cursor State.

A cursor unprepared by an autocommit caused by a previously executed statement.

Error ID: 8, Severity: ERROR, Facility: OCLIERR

Invalid Date literal.

A date literal is in an invalid format. If the ODBC data source is configured with Enforce ODBC Dates/ Times checked, all four digits of year and two digits of month and day must be specified.

Error ID: 23, Severity: ERROR, Facility: FPSRVERR

Invalid STRING argument to STRING.

An invalid argument was passed to a request.

Error ID: 3, Severity: ERROR, Facility: FPSRVERR

Invalid Time literal.

A time literal is in an invalid format. If the ODBC data source is configured with Enforce ODBC Dates/Times checked, both digits of hour, minute, and second must be specified.

Error ID: 39, Severity: ERROR, Facility: FPSRVERR

Invalid Timestamp literal.

A timestamp literal is in an invalid format. If the ODBC data source is configured with Enforce ODBC Dates/Times checked, all four digits of year and two digits of month, day, hour, minute, and second must be specified.

Error ID: 40, Severity: ERROR, Facility: FPSRVERR

Invalid connection or statement option value.

The client tried to set a connection or statement option to an invalid value.

Error ID: 43, Severity: ERROR, Facility: DBCAPERR

Invalid cursor position.

Error ID: 40, Severity: ERROR, Facility: DBCAPERR

Invalid datatype.

Error ID: 35, Severity: ERROR, Facility: FPSRVERR

Invalid entry in character mapping file STRING line NUMBER.

The indicated line in the character mapping file contains an error. Each entry line in a character map file has the following format: local = remote. local and remote are 8-bit character code values in the range 1–255, and the codes for ASCII a–z, A–Z, 0–9, and !%*.<=>_ cannot be mapped. Check the indicated line in the indicated file for extraneous characters, invalid values, or conflicts with other lines. The system has ignored this line in setting up the character map.

Error ID: 11, Severity: INFORMATION, Facility: XIOERR

Invalid monetary unit STRING.

The monetary unit specified is not a recognized monetary unit. Consult documentation on the money data type for allowable monetary units.

Error ID: 39, Severity: ERROR, Facility: DATUMERR

Invalid number.

A string is not in a recognizable numeric format and cannot be converted to a number.

Error ID: 17, Severity: ERROR, Facility: FPSRVERR

Invalid or unrecognized character.

Error ID: 24, Severity: ERROR, Facility: FPSRVERR

Invalid procedure created.

The procedure was created but errors were encountered. Errors could include invalid parameters, compilation errors, and so on.

Error ID: 48, Severity: WARNING, Facility: FPSRVERR

Invalid string or buffer length.

Error ID: 60, Severity: ERROR, Facility: DBCAPERR

Licensing Error: MESSAGE.

This error occurred while trying to establish a license for U2 ODBC. For more information about UniVerse licensing, see Administering UniVerse or contact U2 Customer Support.

Error ID: 75, Severity: ERROR, Facility: DBCAPERR

Licensing error: STRING.

This error occurred while trying to establish a license for U2 ODBC. For more information regarding UniVerse licensing, see Administering UniVerse or contact U2 Customer Support.

Error: 75, Severity: ERROR, Facility: DBCAPERR

Line NUMBER, column NUMBER: STRING.

Error ID 33 should not appear; message text was used when constructing other errors.

Error ID: 33, Severity: INFORMATION, Facility: FPSRVERR

Line NUMBER, column NUMBER (around "STRING"): STRING.

Error ID 34 should not appear; message text was used when constructing other errors.

Error ID: 34, Severity: INFORMATION, Facility: FPSRVERR

No cursor of the specified name.

A reference to a cursor by name failed because there is no cursor with the specified name.

Error ID: 20, Severity: ERROR, Facility: DBCAPERR

No data to read from file or pipe.

A nonblocking read on a pipe or stream found no data available.

Error ID: 12, Severity: ERROR, Facility: OIOERR

No more database cursors available.

All the available database cursors are in use on this connection and none is available. Close one or more other cursors and retry the operation.

Error ID: 4, Severity: ERROR, Facility: FPSRVERR

No more result sets available.

All the result sets for this request have already been returned. There are no more result sets available.

Error ID: 8, Severity: INFORMATION, Facility: FPSRVERR

No more Windows timers left.

All the available Windows timers are in use on this serial connection. Close other Windows applications or exit and reenter Windows.

Error ID: 113, Severity: SEVERE, Facility: LINKERR

Number too big.

A numeric literal is longer than 254 characters.

Error ID: 28, Severity: ERROR, Facility: FPSRVERR

Numeric conversion or arithmetic expression error fetching column.

A numeric conversion or arithmetic expression error occurred while an application was fetching a value from a column. A null value was returned.

Error ID: 64, Severity: ERROR, Facility: DBCAPERR

Numeric data truncated entering or leaving the database.

Numeric data was truncated while entering or leaving the database.

Error ID: 50, Severity: WARNING, Facility: DBCAPERR

Numeric value out of range.

The supplied buffer is too small to hold the integer part of the requested numeric value. Provide a larger character buffer or a buffer of a numeric type with a larger range.

Error ID: 29, Severity: ERROR, Facility: OIOERR

Object input data format error: versions incompatible.

Your software tried to read object data that was created by a more recent version of the product. The data in question was probably a saved query file.

Error ID: 22, Severity: ERROR, Facility: OIOERR

ODBC-compliance syntax error.

A non-ODBC construct has been encountered at the indicated line and column number of the SQL statement. Verify that the SQL statement follows proper ODBC SQL syntax.

Error ID: 51, Severity: ERROR, Facility: FPSRVERR

ODBC escape clause not properly terminated.

Error ID: 21, Severity: ERROR, Facility: FPSRVERR

Operation canceled.

Error ID: 44, Severity: ERROR, Facility: DBCAPERR

Operation has been canceled.

The request was interrupted and cancelled. Results from the cancelled request may not be reliable.

Error ID: 49, Severity: ERROR, Facility: FPSRVERR

Out of memory.

The U2 ODBC driver has run out memory. Reduce the size of the result set you request, close other database connections with large query result sets, or close down other Windows applications that are consuming memory, then retry the operation. If this message persists, you may need to shut down

Windows and restart it or reboot your PC. If this message occurs frequently, you need more memory on your PC or more virtual memory on your data source.

Error ID: 1, Severity: FATAL, Facility: MEMERR

Out of result memory in the client.

The result set of a query will not fit in available client memory. Reduce the size of the result set you request, close other database connections with large query result sets, or close down other Windows applications that are consuming memory, then retry the operation. If this message persists, you may need to shut down Windows and restart it or reboot your PC. If this message occurs frequently, you need more memory on your PC.

Error ID: 2, Severity: SEVERE, Facility: LINKERR

Out of sequence request.

The request could not be performed because prerequisite requests have not yet been made.

Error ID: 47, Severity: ERROR, Facility: FPSRVERR

Overflow converting a value to a numeric type.

A text- or number-to-number conversion failed because the number was too large to be represented in the specified format.

Error ID: 24, Severity: ERROR, Facility: DATUMERR

Path truncated.

You typed a value that was longer than a valid path plus filename. The value was truncated to the maximum length defined for the column. This message is informational only, and requires no further action.

Error ID: 30, Severity: WARNING, Facility: DATUMERR

Premature EOF reading file.

End-of-file was reached while an application was prematurely reading data from a file.

Error ID: 13, Severity: ERROR, Facility: OIOERR

Read from invalid file.

An attempt was made to read from an invalid file descriptor.

Error ID: 9, Severity: ERROR, Facility: OIOERR

Read permission denied.

An attempt to read a file failed because you have no access rights to the file.

Error ID: 10, Severity: ERROR, Facility: OIOERR

Received unexpected data: STRING.

An event on the data source system caused unexpected data to be sent over the network. Usually this indicates that the server process has terminated, and the unexpected data is a message indicating what happened. Try again.

This error can also happen if the shell configuration scripts on your UNIX host echo data to the terminal. In this case, you must modify the scripts to move or remove the terminal-accessing commands.

If this problem persists, contact U2 Customer Support.

Error ID: 7, Severity: SEVERE, Facility: LINKERR

Receive timeout, closing connection.

The U2 ODBC driver timed out after waiting for data from the host server. The host system is either very slow or has failed, or the host server program has failed. Contact your system administrator.

Error ID: 16, Severity: FATAL, Facility: OIOERR

Restricted data type attribute violation.

The data value cannot be converted to the C data type specified by the argument fCType.

Error ID: 27, Severity: ERROR, Facility: OIOERR

Serialization failure.

The transaction to which the prepared statement belonged was terminated to prevent deadlock.

Error ID: 39, Severity: ERROR, Facility: DBCAPERR

Server received an increment of input character data longer than the defined chunk length.

The U2 ODBC driver received an increment of input character data longer than the defined chunk length. The data was right-truncated.

Error ID: 11, Severity: WARNING, Facility: DBCAPERR

SQL numeric value out of range.

An SQL statement contained a numeric parameter or literal that was out of range for its context.

Error ID: 12, Severity: ERROR, Facility: DBCAPERR

SQL numeric value out of range.

An SQL statement contained a numeric parameter or literal that was out of range for its context.

Error ID: 69, Severity: WARNING, Facility: DBCAPERR

SQL REVOKE failed because of insufficient privilege.

The user had insufficient privilege to revoke a privilege for another user.

Error ID: 5, Severity: ERROR, Facility: DBCAPERR

SQL statement contained ambiguous column reference.

An SQL statement used a column name that had more than one possible referent. The statement probably contained a join in which two tables each had a column with that name.

Error ID: 73, Severity: ERROR, Facility: DBCAPERR

SQL statement referenced a nonexistent column in a table or view.

Error ID: 31, Severity: ERROR, Facility: DBCAPERR

SQL statement referenced a nonexistent index.

Error ID: 29, Severity: ERROR, Facility: DBCAPERR

SQL statement referenced a nonexistent table or view.

Error ID: 27, Severity: ERROR, Facility: DBCAPERR

SQL statement used undefined table name.

An SQL statement used tablename.columnname or tablename.* but did not correctly use tablename elsewhere in the statement—for example, in the FROM clause.

Error ID: 74, Severity: ERROR, Facility: DBCAPERR

SQL Syntax error.

The database reported an SQL syntax error.

Error ID: 21, Severity: ERROR, Facility: DBCAPERR

Stack overflow (lexer).

The SQL statement contains too many levels of nested escape sequences.

Error ID: 41, Severity: ERROR, Facility: FPSRVERR

String data/length mismatch.

Less character data was provided than was promised for filling in an input parameter.

Error ID: 16, Severity: ERROR, Facility: DBCAPERR

STRING is illegal in ODBC SELECT...FOR UPDATE statements.

In the ODBC SQL grammar, only a limited form of the SELECT statement can be used with FOR UPDATE.

Error ID: 64, Severity: ERROR, Facility: FPSRVERR

STRING is not a valid UniVerse account or schema.

The UniVerse account is not valid. If an account pathname was specified, verify that the directory exists and is a UniVerse account. If an account name was specified, verify that the UV.ACCOUNT entry is correct. If an SQL schema name was specified, verify that the UV_SCHEMA entry is correct.

Error ID: 76, Severity: ERROR, Facility: DBCAPERR

Syntax error.

A syntax error has been discovered at the indicated line and column number of the SQL statement. Verify that the SQL statement follows proper ODBC SQL syntax.

Error ID: 29, Severity: ERROR, Facility: FPSRVERR

Table name or correlation name conflicts with U2 ODBC's correlation name cookie <STRING>, so cannot be used.

U2 ODBC sometimes generates correlation names of the form <correlation name cookie><n>, where n is a nonnegative integer. You cannot use a table or correlation name that also has this form.

Error ID: 60, Severity: ERROR, Facility: FPSRVERR

TCP driver software not loaded.

The TCP/IP driver is not running, or the client TCP/IP driver software has not been loaded. Check to see if the networking software is installed properly.

Error ID: 111, Severity: SEVERE, Facility: LINKERR

Text truncated.

You typed a value that was longer than the space provided by the database for values in this column. The value was truncated to the maximum length defined for the column. This message is informational only, and requires no further action.

Error ID: 10, Severity: WARNING, Facility: DATUMERR

Text truncated to 63,950 characters.

Because the Text class is currently implemented using far pointers (instead of huge pointers), it is not possible to create a Text class containing more than 63,950 characters. If you need to store more than 63,950 characters of textual information, you might want to consider storing the text as binary data (using the Binary class).

Error ID: 44, Severity: WARNING, Facility: DATUMERR

The database reported an error when attempting to access this data item.

Error ID: 54, Severity: ERROR, Facility: DBCAPERR

The minutes and seconds parts of this date/time have been set to 0.

You typed a date and hours value for a column that is defined to include both a date and a complete time. The unspecified time portions have been set to 0. This message is informational only, and requires no further action.

Error ID: 18, Severity: WARNING, Facility: DATUMERR

The name STRING violates ODBC rules for names.

The ODBC 3.0 standard requires names of database objects to begin with a letter, and to contain only letters, digits, and the underscore character.

Error ID: 53, Severity: ERROR, Facility: FPSRVERR

The number did not fit within the specified precision; it was rounded off.

You typed a value that has more significant digits than the precision specified for its column. The number you typed was rounded off to fit within the column's defined precision. This message is informational only, and requires no further action.

Error ID: 27, Severity: WARNING, Facility: DATUMERR

The number is too large; it was rounded to infinity.

You typed a value that was larger than the precision specified for its column. Because the column supports infinite values, the number you typed was converted to infinity. This message is informational only, and requires no further action.

Error ID: 25, Severity: WARNING, Facility: DATUMERR

The operation modified or deleted all rows in the table.

An UPDATE or DELETE statement affected every row in the table, possibly because it had no WHERE clause.

Error ID: 34, Severity: WARNING, Facility: DBCAPERR

The requested operation was inapplicable to the specified cursor.

The requested operation was invalid on the specified cursor. Examples: UPDATE was positioned before FETCH; column attributes were on a non- SELECT statement.

Error ID: 18, Severity: ERROR, Facility: DBCAPERR

The requested value is not available.

Error ID: 46, Severity: ERROR, Facility: FPSRVERR

The Schema (UVSCHEMA) defining the catalog could not be accessed.

Error ID: 57, Severity: ERROR, Facility: DBCAPERR

The Schema defined by the client does not exist or could not be accessed.

Error ID: 55, Severity: ERROR, Facility: DBCAPERR

The search pattern argument STRING contains an escape character which is not itself escaped and which is not followed by %, _, or the escape character.

U2 ODBC's interpretation of ODBC is that in search pattern arguments to catalog functions, every escape character that is not itself escaped must be followed by %, _, or the escape character.

Error ID: 52, Severity: ERROR, Facility: FPSRVERR

The seconds part of this date/time has been set to 0.

You typed a date and hours/minutes value for a column that is defined to include both a date and a complete time. The unspecified time portions have been set to 0. This message is informational only, and requires no further action.

Error ID: 19, Severity: WARNING, Facility: DATUMERR

The server does not support the requested function.

Error ID: 25, Severity: ERROR, Facility: DBCAPERR

The Table defined by the client does not exist or could not be accessed.

Error ID: 56, Severity: ERROR, Facility: DBCAPERR

The table expansion/explosion was halted at the row limit of NUMBER rows.

Expansion/explosion of the table would generate more than the specified number of rows. The expanded/exploded table was limited to this number of rows and does not represent all the underlying data.

Error ID: 38, Severity: INFORMATION, Facility: DATUMERR

The Table of Columns (UVCOLUMNS) in the Catalog could not be accessed.

Error ID: 59, Severity: ERROR, Facility: DBCAPERR

The Table of Tables (UVTABLES) in the Catalog could not be accessed.

Error ID: 58, Severity: ERROR, Facility: DBCAPERR

The time part of this date/time has been set to midnight.

You typed a date value for a column that is defined to include both a date and a time. The time portion has been set to 0, which represents midnight. This message is informational only, and requires no further action.

Error ID: 17, Severity: WARNING, Facility: DATUMERR

The transaction isolation level cannot be changed during an open transaction.

The transaction isolation level cannot be changed during an open transaction. The current transaction must be committed or rolled back before the transaction isolation level can be changed.

Error ID: 72, Severity: ERROR, Facility: DBCAPERR

The underlying database does not provide support for output parameters.

Error ID: 52, Severity: ERROR, Facility: DBCAPERR

The U2 ODBC driver or the underlying database do not provide support for the requested SQL statement.

The U2 ODBC driver or the underlying database does not provide support for the requested SQL statement. Error ID: 49, Severity: ERROR, Facility: DBCAPERRThe U2 ODBC driver or the underlying database do not provide the requested feature.

Error ID: 33, Severity: ERROR, Facility: DBCAPERR

The year in this date must be AD.

You typed a date/time value whose year is BC (negative), but the date/time column can support only AD (positive) years.

Error ID: 15, Severity: ERROR, Facility: DATUMERR

This SQL statement was ignored.

This SQL statement is not supported by the underlying database. This SQL statement was not executed by the U2 ODBC driver. Success with information is returned instead of an error because some applications abort if an attempted execution of this type of SQL statement returns an error.

Error ID: 63, Severity: WARNING, Facility: DBCAPERR

Too many open sockets.

The connection could not be made because all available TCP/IP sockets are in use. This can happen if many TCP/IP service sessions (telnet, ftp) or many U2 ODBC connections are open. Close one or more telnet, ftp, or U2 ODBC connections. You may be able to configure your TCP/IP software package to provide more concurrent sockets. See your PC TCP/IP package documentation.

Error ID: 122, Severity: SEVERE, Facility: LINKERR

Too many sockets open.

The local PC limit on the number of concurrently open sockets has been reached. This can occur if there are several concurrent sessions active over the TCP-IP/Ethernet connection, including telnet, ftp, and NFS. Exit one or more of the concurrent application sessions and try again. If this fails, reboot your PC and try again.

Error ID: 106, Severity: SEVERE, Facility: LINKERR

Transmit timeout, closing connection.

No acknowledgment has been received from the host/server for the last message sent. The connection has been closed. Retry the connection.

Error ID: 120, Severity: SEVERE, Facility: LINKERR

UCI Error. Func: SQL Connect(); State: IM002; UniVerse code: 0; Msg: [U2][SQL Client]The data source is not in configuration file.

Check the uci.config file to verify that the data source entry exists.

Error ID: 46, Severity: ERROR, Facility: DBCAPERR

UCI Error. Func: STRING; State: STRING; UniVerse code: NUMBER; Msg: STRING.

A UCI error was encountered. See UniVerse Call Interface Guide to learn more about this error.

Error ID: 46, Severity: ERROR, Facility: DBCAPERR

UCI Warning Func: STRING; State: STRING; UniVerse code: NUMBER; Msg: STRING.

A UCI warning was encountered. See UniVerse Call Interface Guide to learn more about this warning.

Error ID: 47, Severity: WARNING, Facility: DBCAPERR

Unable to determine the current schema for this connection.

Either no schema has been created for this account or the current schema could not be determined for this account from the account information provided in the Account field in the Database server/ Authorization Information section of the configuration file entry.

Error ID: 53, Severity: ERROR, Facility: DBCAPERR

Uni RPC daemon is not running.

Startup this daemon from the UniVerse System Administration menu before attempting to use UV/ ODBC. The UniRPC daemon must be running in order for U2 ODBC to establish a connection with UniVerse. The UniRPC daemon can be started from the UniVerse System Administration menu in the UV account. The UniRPC service can be started from the Windows NT Control Panel, the UniVerse Control Panel, or from an MS-DOS window.

Error ID: 71, Severity: ERROR, Facility: DBCAPERR

UniVerse SQL does not support UNION with FOR UPDATE.

Error ID: 65, Severity: ERROR, Facility: FPSRVERR Error ID: 10, Severity: FATAL, Facility: XIOERR

Unknown data type.

An unknown ODBC data type has been passed.

Error ID: 2, Severity: ERROR, Facility: FPSRVERR

Unterminated quoted string.

A quoted string is missing a closing single or double quotation mark.

Error ID: 18, Severity: ERROR, Facility: FPSRVERR

User does not have permission to execute this SQL statement.

Error ID: 23, Severity: ERROR, Facility: DBCAPERR

UV/ODBC BASIC program STRING failed to execute: [STRING] STRING.

U2 ODBC's attempt to execute a cataloged BASIC helper program failed for the reason mentioned. This could indicate a file access problem or an invalid U2 ODBC installation.

Error ID: 55, Severity: ERROR, Facility: FPSRVERR

UV/ODBC BASIC program STRING reported error: STRING.

A U2 ODBC cataloged BASIC program encountered this error while collecting information about UniVerse tables or files. Error ID: 56, Severity: WARNING, Facility: FPSRVERR

**Note:** This error message actually describes a warning from a BASIC program, not an error.

UV/ODBC BASIC program STRING reported error: STRING.

A cataloged BASIC program encountered this error while collecting information about UniVerse tables or files.

Error ID: 59, Severity: ERROR, Facility: FPSRVERR

UV/ODBC cannot describe the result set of a UniVerse procedure which has not been executed.

You requested information about the result set of a UniVerse procedure which has been prepared, but not yet executed. This information cannot be determined until you actually execute the procedure.

Error ID: 63, Severity: ERROR, Facility: FPSRVERR

UV/ODBC dictionary cannot prepare SQL statement [STRING].

U2 ODBC extracts information about columns of database tables from this database by preparing an SQL statement for execution. The database rejected the SQL statement, probably because access privileges were inadequate.

Error ID: 58, Severity: WARNING, Facility: FPSRVERR

UV/ODBC's file access control prevents writing to this file.

The user attempted to write to a file whose accessibility was specified in the HS_FILE_ACCESS file as READ or NONE.

Error ID: 54, Severity: ERROR, Facility: FPSRVERR

Write to invalid file.

An attempt was made to write to an invalid file descriptor.

Error ID: 17, Severity: ERROR, Facility: OIOERR

Write to invalid pipe.

An attempt was made to write to a pipe not open for writing.

Error ID: 18, Severity: ERROR, Facility: OIOERR

Wrong number of input parameters to SQL statement.

The client application provided more, or fewer, parameter values than there were place holders in the SQL statement.

Error ID: 41, Severity: ERROR, Facility: DBCAPERR

# Chapter 12: Sample UniVerse accounts

## Locating the sample UniVerse accounts

Each UniVerse database server has two sample accounts: HS.SALES and HS.SERVICE. When you install UniVerse, these sample accounts are created in the UV account directory, so you can use them to learn how to use U2 ODBC before working with your own data. The files for these accounts are located in two directories called HS.SALES and HS.SERVICE.

If you defined HS.SALES and HS.SERVICE as ODBC data sources using UCI Config Editor, they should be accessible to client products and sample applications.

## Database definitions

The HS.SALES and HS.SERVICE accounts each contain several files. The following sections list the files contained in each account, the columns found in each file, and the relationships among the files.

### HS.SALES account

The HS.SALES account contains information about copiers and fax machines sold by a fictitious manufacturer. It contains information about sales prospects, customers who bought machines, details of their purchases, and the products purchased. This information is typically used by the Sales department.

HS.SALES comprises three files: CUSTOMER, PRODUCTS, and STATES.

| CUSTOMER | PRODUCTS | STATES |
|---|---|---|
| CUSTID | PRODID | CODE |
| SAL | LIST | NAME |
| FNAME | DESCRIPTION | |
| LNAME | | |
| COMPANY | | |
| ADDR1 | | |
| ADDR2 | | |
| CITY | | |
| STATE | | |
| ZIP | | |
| PHONE | | |
| PRODID | | |
| SER_NUM | | |
| PRICE | | |
| BUY_DATE | | |
| PAID_DATE | | |
| SVC_PRICE | | |
| SVC_START | | |

| CUSTOMER | PRODUCTS | STATES |
|---|---|---|
| SVC_END | | |
| SVC_PAID_DATE | | |

## HS.SERVICE account

The HS.SERVICE account contains information about customer calls, commonly reported problems, and products under maintenance. This information is typically used by the Service department.

HS.SERVICE comprises three files: CALLS, PROBLEMS, and PRODS.

| CALLS | PROBLEMS | PRODS |
|---|---|---|
| CALL_ID | PROB_NUM | PRODUCT |
| CUST_NUM | PRODUCT | VERSION |
| F_NAME | VERSION | DESCRIPTION |
| L_NAME | PROB_DESCRIPTION | FIRST_SHIP |
| PHONE | STATUS | |
| PROB_NUM | FIX | |
| PRODUCT | NOTES | |
| VERSION | OCCURRENCES | |
| SERIAL | | |
| CALLED_ON | | |
| CALL_DATE | | |
| CALL_TIME | | |
| CALL_MINUTES | | |
| PROBLEM | | |
| RESOLUTION | | |

# Locating the sample UniVerse accounts

Each UniVerse database server has two sample accounts: HS.SALES and HS.SERVICE. When you install UniVerse, these sample accounts are created in the UV account directory, so you can use them to learn how to use U2 ODBC before working with your own data. The files for these accounts are located in two directories called HS.SALES and HS.SERVICE.

If you defined HS.SALES and HS.SERVICE as ODBC data sources using UCI Config Editor, they should be accessible to client products and sample applications.

# Database definitions

The HS.SALES and HS.SERVICE accounts each contain several files. The following sections list the files contained in each account, the columns found in each file, and the relationships among the files.

## HS.SALES account

The HS.SALES account contains information about copiers and fax machines sold by a fictitious manufacturer. It contains information about sales prospects, customers who bought machines, details of their purchases, and the products purchased. This information is typically used by the Sales department.

HS.SALES comprises three files: CUSTOMER, PRODUCTS, and STATES.

| CUSTOMER | PRODUCTS | STATES |
|---|---|---|
| CUSTID | PRODID | CODE |
| SAL | LIST | NAME |
| FNAME | DESCRIPTION | |
| LNAME | | |
| COMPANY | | |
| ADDR1 | | |
| ADDR2 | | |
| CITY | | |
| STATE | | |
| ZIP | | |
| PHONE | | |
| PRODID | | |
| SER_NUM | | |
| PRICE | | |
| BUY_DATE | | |
| PAID_DATE | | |
| SVC_PRICE | | |
| SVC_START | | |
| SVC_END | | |
| SVC_PAID_DATE | | |

## HS.SERVICE account

The HS.SERVICE account contains information about customer calls, commonly reported problems, and products under maintenance. This information is typically used by the Service department.

HS.SERVICE comprises three files: CALLS, PROBLEMS, and PRODS.

| CALLS | PROBLEMS | PRODS |
|---|---|---|
| CALL_ID | PROB_NUM | PRODUCT |
| CUST_NUM | PRODUCT | VERSION |
| F_NAME | VERSION | DESCRIPTION |
| L_NAME | PROB_DESCRIPTION | FIRST_SHIP |
| PHONE | STATUS | |
| PROB_NUM | FIX | |
| PRODUCT | NOTES | |

| CALLS | PROBLEMS | PRODS |
|---|---|---|
| VERSION | OCCURRENCES | |
| SERIAL | | |
| CALLED_ON | | |
| CALL_DATE | | |
| CALL_TIME | | |
| CALL_MINUTES | | |
| PROBLEM | | |
| RESOLUTION | | |

## Sample data

The following files show the sample data contained in the HS.SALES and HS.SERVICE accounts.

**Note:** If you update any of these files, the actual data you see can differ from these samples.

| CUSTID | SAL | FNAME | LNAME | COMPANY | ADDR1 | ADDR2 | CITY | STATE | ZIP | PHONE |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Mr. | Samuel | Smith | Better Beer, Inc. | 10 Commercial St | | Concord | NH | 02131 | (603) 555-3212 |
| 2 | Ms. | Diana | Morris | Fast Copy Center | 431 Third Ave. | | Waltham | MA | 01133 | (617) 555-9823 |
| 3 | Mr. | David | Argonne | Fast Copy Center | 75 Great Road | | Bedford | MA | 01182 | (617) 555-3468 |
| 4 | Ms. | Jill | Kahn | Fast Copy Center | 12 School St | | Boston | MA | 01103 | (617) 555-7396 |
| 5 | Mr. | Kenneth | Williams | Ocean State Fish Company | 837 Ocean Ave | | Provi-dence | RI | 03171 | (401) 555-6512 |
| 6 | Ms. | Betty | Burke | Lightning Computer Corp. | 400 Technology Path | MS 10-27 | White River | VT | 01644 | (802) 555-9854 |
| 7 | Dr. | Martha | Gill | Central Hospital | 555 Main Street | | Derry | NH | 04429 | (603) 555-5437 |
| 8 | Mr. | Steven | Holland | Copies, Inc. | 4325 Hill Road | | Lowell | MA | 01386 | (508) 555-2365 |
| 9 | Ms. | Nicole | Orlando | A1 Used Auto | 820 Middlesex Turnpike | | Burl-ington | MA | 01173 | |
| 10 | Dr. | Andrew | McCaig | HGT Dental Center | 999 Hill Road | | Brattle-boro | VT | 03356 | (802) 555-6534 |
| 11 | Mr. | Skip | Lewis | Skip's Whale Watch | 10 Dock Street | | Plym-outh | MA | 01382 | (508) 555-2368 |

| CUSTID | SAL | FNAME | LNAME | COMPANY | ADDR1 | ADDR2 | CITY | STATE | ZIP | PHONE |
|---|---|---|---|---|---|---|---|---|---|---|
| 12 | Mrs. | Laurie | Patry | Rustic Printers | 10 Rustic Trail | | Littleton | MA | 01142 | (508) 555-9426 |

| CUST_ID | PRODID | SER_NUM | PRICE | BUY_DATE | PAID_DATE |
|---|---|---|---|---|---|
| 1 | M2000 | 501278 | $4200 | 1991-01-07 | 1991-01-28 |
| 2 | C2000 | 600782 | $6600 | 1991-01-08 | 1991-02-05 |
| 2 | M3000 | 700422 | $12000 | 1991-01-08 | 1991-02-05 |
| 2 | S3000 | 101456 | $900 | 1991-01-22 | 1991-02-12 |
| 3 | M2000 | 501310 | $4250 | 1991-01-08 | 1991-01-10 |
| 4 | C3000 | 800311 | $16500 | 1991-01-09 | 1991-02-07 |
| 5 | M1000 | 403485 | $1900 | 1991-01-14 | 1991-02-14 |
| 5 | M1000 | 403723 | $1900 | 1991-02-15 | 1991-02-28 |
| 7 | M2000 | 501233 | $4490 | 1991-01-20 | |
| 7 | S2000 | 101212 | $990 | 1991-01-20 | |
| 8 | M3000 | 700514 | $1200 | 1991-01-21 | 1991-02-21 |
| 8 | S3000 | 201399 | $900 | 1991-01-21 | 1991-02-21 |
| 10 | M1000 | 203510 | $1990 | 1991-01-28 | 1991-02-14 |
| 10 | M1000 | 203600 | $1900 | 1991-01-29 | 1991-02-28 |
| 10 | C2000 | 600791 | $6500 | 1991-01-30 | |

| CUSTID | PRODID | SER_NUM | SVC_PRICE | SVC_START | SVC_END | SVC_PAID_DATE |
|---|---|---|---|---|---|---|
| 1 | M2000 | 501278 | $600 | 1991-01-13 | 1992-01-15 | 1991-01-28 |
| 2 | C2000 | 600782 | $900 | 1991-01-13 | 1992-01-15 | 1991-02-05 |
| 3 | M3000 | 700422 | $500 | 1991-01-13 | 1992-06-12 | 1991-02-05 |
| 4 | S3000 | 101456 | $150 | 1991-01-13 | 1991-01-15 | 1991-02-12 |
| 5 | M1000 | 403485 | $600 | 1991-01-13 | 1991-06-12 | 1991-02-14 |
| 6 | C3000 | 800311 | $600 | 1991-01-13 | 1991-06-12 | 1992-02-07 |
| 9 | M3000 | 700514 | $1000 | 1991-02-03 | 1992-02-05 | 1991-02-21 |
| 10 | S3000 | 201399 | $150 | 1991-02-03 | 1992-02-05 | 1991-02-21 |
| 11 | M1000 | 203600 | $400 | 1991-02-03 | 1992-02-05 | 1991-02-28 |
| 12 | C2000 | 600791 | $950 | 1991-02-15 | 1992-02-19 | |
| 13 | M1000 | 403723 | $600 | 1991-03-02 | 1991-08-07 | 1991-02-28 |

| PRODID | LIST | DESCRIPTION |
|---|---|---|
| M1000 | $1,990 | Low cost, entry level, light duty, monoc |
| M2000 | $4,490 | Moderate duty monochrome copier |
| C2000 | $6,890 | Moderate duty, entry level, color copier |
| M3000 | $12,990 | Heavy duty monochrome copier |
| C3000 | $17,990 | Heavy duty color copier |
| S2000 | $990 | Sorting attachment for M2000/C2000 |
| S3000 | $1,990 | Sorting attachment for M3000/C3000 |

| CODE | NAME | | CODE | NAME |
|---|---|---|---|---|
| AK | Alaska | | MT | Montana |

| CODE | NAME | | CODE | NAME |
|---|---|---|---|---|
| AL | Alabama | | NC | North Carolina |
| AR | Arkansas | | ND | North Dakota |
| AZ | Arizona | | NE | Nebraska |
| CA | California | | NH | New Hampshire |
| CO | Colorado | | NJ | New Jersey |
| CT | Connecticut | | NM | New Mexico |
| DE | Delaware | | NV | Nevada |
| FL | Florida | | NY | New York |
| GA | Georgia | | OH | Ohio |
| HI | Hawaii | | OK | Oklahoma |
| IA | Iowa | | OR | Oregon |
| ID | Idaho | | PA | Pennsylvania |
| IL | Illinois | | RI | Rhode Island |
| IN | Indiana | | SC | South Carolina |
| KS | Kansas | | SD | South Dakota |
| KY | Kentucky | | TN | Tennessee |
| LA | Louisiana | | TX | Texas |
| MA | Massachusetts | | UT | Utah |
| MD | Maryland | | VA | Virginia |
| ME | Maine | | VT | Vermont |
| MI | Michigan | | WA | Washington |
| MN | Minnesota | | WI | Wisconsin |
| MO | Missouri | | WV | West Virginia |
| MS | Mississippi | | WY | Wyoming |

| CUST_NUM | F_NAME | L_NAME | PHONE | PROB_NUM | PRODUCT | VERSION | SERIAL | CALLED_ON | CALL_MINUTES | PROBLEM | RESOLUTION |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | Diana | Morris | (617) 555-9823 | 4 | C2000 | 1D | 600782 | 1991-02-14 13:45:01 | 2 | | Didn’t read the manual |
| 10 | Andy | McCaig | (802) 555-6534 | | M1000 | 2 | 203600 | 1991-02-05 9:05:04 | 4 | Copier doesn’t work in the morning | Copier in 10 minute warm-up cycle; didn’t read manual and understand the flashing ’wait’ light |
| 10 | Andy | McCaig | (802) 555-6534 | | M1000 | 2 | 203600 | 1991-02-05 12:56:37 | 4 | Lines down the middle of the page | Drum needs cleaning; service technician dispatched. |
| 1 | Sam | Smith | (603) 555-3212 | 5 | M2000 | 1 | 501278 | 1991-03-02 9:32:45 | 3 | | |
| 2 | Diana | Morris | (617) 555-9823 | 7 | M3000 | 2 | 700422 | 1991-02-18 14:58:10 | 5 | | |

| CUST _NUM | F _NAME | L _NAME | PHONE | PROB _NUM | PRODUCT | VERSION | SERIAL | CALLED _ON | CALL _MINUTES | PROBLEM | RESOLUTION |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | Ken | Williams | (401) 555-6512 | 3 | M1000 | 1 | 403485 | 1991-02-20 15:30:02 | 2 | | |
| 5 | Ken | Williams | (401) 555-6512 | 8 | M1000 | 1A | 403723 | 1991-02-24 11:12:52 | | | |
| 8 | Steve | Holland | (508) 555-2365 | 7 | C3000 | 2 | 700514 | 1991-02-24 13:12:33 | | Stapler jams | Wasn't using proper kind of staples. |

| PROB_NUM | PRODUCT | VERSION | PROB _DESCRIPTION | STATUS | FIX | NOTES | OCCURRENCES |
|---|---|---|---|---|---|---|---|
| 1 | S200 | 1 | Sorter jams when collating stapled copies | fixed | Cycle sorter power before every collated, stapled print job | Fixed in rev 1A | 15 |
| 2 | M1000 | 1 | Toner light won't go out, even after adding more toner | fixed | Cycle power after adding toner | Fixed in rev 2 | 4 |
| 3 | M1000 | 1 | Frequent paper jams | fixed | Fan paper thoroughly, don't use moist paper | Rev 2 paper tray is much better | 27 |
| 4 | C2000 | 1 | Smeared colors after changing toner cartridge | doc | Make 20 copies of color test sheet after changing toner | The manual now recom-mends this as standard procedure | 25 |
| 5 | M2000 | 1 | Paper jam light indicates location 10, no paper jammed there | fixed | Paper is jammed between locations 9 and 10. Turn roller 9 several times to right and remove from location 10 | Rev 1A shows both loca-tions 9 and 10 for this kind of jam | 3 |
| 6 | C3000 | 1 | Smeared colors after changing toner cartridge | doc | Make 20 copies of color test sheet after changing toner | The manual now recom-mends this as standard procedure | 17 |

| PROB_NUM | PRODUCT | VERSION | PROB _DESCRIPTION | STATUS | FIX | NOTES | OCCURRENCES |
|---|---|---|---|---|---|---|---|
| 7 | M3000 | 2 | Frequent original jams when 2-sided copying large originals | pending | Feeder needs service. Should install ver. 2A feeder long-life upgrade. | | 0 |
| 8 | M1000 | 2 | Wrong paper size displayed when using 8.5x14 paper tray | doc | Use only rev2 paper trays | Explained in rev 3 manual, page 10. | 4 |

| Product | Version | Description | First_Ship |
|---|---|---|---|
| M1000 | 2 | Low cost, entry level, light duty, mono-chrome copier. Rated 1000 pages/ month. Annual service recommended. | 1991-01-31 |
| M2000 | 1A | Moderate duty monochrome copier. Rated 10,000 pages/ month. Bimonthly service recommended. | 1991-02-14 |
| C2000 | 1D | Moderate duty, entry level, color copier. Rated 10,000 pages/ month. Bimonthly service recommended. | 1991-02-14 |
| M3000 | 2 | Heavy duty monochrome copier. Rated 100,000 pages/ month. Monthly service recommended. | 1990-04-05 |
| C3000 | 3A | Heavy duty color copier. Rated 100,000 pages/month. Monthly service recommended. | 1990-05-10 |
| S2000 | 2 | Sorting attachment for M2000/C2000 | 1991-01-30 |
| S3000 | 2 | Sorting attachment for M3000/C3000 | 1991-01-30 |

# Chapter 13: Troubleshooting

## Isolating a problem

Most installation and operation difficulties with U2 ODBC fall into one of the following categories:

- Windows operating system configuration
- PC hardware or software configuration
- Windows security, permissions, quotas, or disk space
- Nonfunctioning communications paths

If a client encounters a problem accessing the UniData or UniVerse database server, first try to access the server from a similarly configured client system that is working. Many problems are communications-related, so isolating them to the client or server is important.

In the DOS Command Prompt window, enter the Ping command to verify the network connection. In the U2 ODBC Data Source Setup (called by the Microsoft ODBC Data Source Administrator tool), use the Test button to verify database access.

If Ping fails, the problem is communications-related, and you should examine whether other applications external to U2 ODBC (or even external to Windows) are visible to the server. Terminal emulators and FTP utilities should be checked to see if they fail similarly. For more information, see Manual communications verification, on page 126.

If Ping succeeds but Test fails, the problem is most likely on the server or in the configuration entry. Examine the configuration carefully. Correct any problems you find, then retry the test.

## Before reporting a problem

Before you report a U2 ODBC problem, you need relevant information about your system.

## Getting information about your system

This section describes how to find the information you may need for problem diagnosis or assistance. The following tables itemize the UNIX and Windows server information you may need. The Action column provides detailed procedures for getting the information.

| **Item** | **Action** |
|---|---|
| UniData or UniVerse version number | Enter `.L RELLEVEL` at the UniVerse prompt.<br><br>Enter `VERSION` at the UniData prompt. |
| Is the UniRPC daemon running? | Enter `ps -ef \| grep unirpcd` or `ps -aux \| grep unirpcd`, depending on your system. |
| TCP/IP host name of the UNIX server | Enter `hostname` at the UNIX prompt. |
| Contents of the U2 account directory, and permissions on them | Change directory to the U2account directory and enter `ls -l > logfile` to write the permissions settings to a file. |

| Item | Action |
|---|---|
| Contents of the U2 account directory, and permissions on them | Change directory to the U2 account directory and enter `ls -l > logfile` to write the permissions settings to a file. |
| Contents of the unishared directory, and permissions on them | Change directory to the *unishared* directory and enter `ls -l > logfile` to write the permissions settings to a file. |
| Contents of the unirpc directory, and permissions on them | Change directory to the *unirpc* directory and enter `ls -l > logfile` to write the permissions settings to a file. |
| Disk usage of the U2 account directory | Enter `du U2.account.dir`.<br>*U2.account.dir* is the directory path where UniData or UniVerse is installed. |
| Available disk space on the file system where UniData or UniVerse is installed | Enter `df filesystem` on that file system.<br>*filesystem* is the name of the UNIX file system that you are checking. |

| Item | Action |
|---|---|
| Windows version number | Enter `WINMSD` in the Run dialog box. Click the **Version** tab. |
| Is UniData or UniVerse running? | Try to use *telnet* to reach the Windows host name. From the Windows Control Panel, choose **Services**. Verify that UniData or UniVerse is in the list of installed services and the status is Started. |
| Is the UniRPC service running? | Choose **Services** from the Windows Control Panel. Verify that UniData or UniVerse UniRPC is in the list of installed services and the status is Started. |
| TCP/IP host name of the Windows NT server | Enter `hostname` at the MS-DOS prompt. |
| Contents of the U2 account directory, and permissions on them | Use the Windows Explorer. |
| Contents of the unishared directory, and permissions on them | Use the Windows Explorer. |
| Contents of the unirpc directory, and permissions on them | Use the Windows Explorer. |
| Available disk space on the file system containing UniData or UniVerse | Use the Windows Explorer. |
| Disk usage of the U2 account directory | Use the Windows Explorer. |

The following table itemizes the Windows client information you may need. The Action column provides detailed procedures for obtaining each item of information.

| Item | Action |
|---|---|
| U2 ODBC driver version number | Right-click U2ODBC.DLL to check the version number. |

| Item | Action |
|---|---|
| Configuration file entry on which the problem occurred | Check the UCI.CONFIG file. |
| PC processor type and clock speed | Choose **System** from the Control Panel, then click the **General** tab. |
| PC main and extended memory size | Choose **System** from the Control Panel, then click the **Performance** tab. |
| Windows version | Choose **System** from the Control Panel, then click the **General** tab. |
| Type of Ethernet card installed | Look at the vendor documentation. |
| Pathname of the U2 ODBC driver installation directory | Use the Windows Explorer (usually `<drive>:\U2\U2ODBC`). |
| PATH environment variable value | Choose **System** from the Control Panel, then click the **Environment** tab. |

For diagnostic purposes it may also be necessary to get copies of the following files at the time of the failure.

For UNIX only:

- *.profile*, *.login*, and *.cshrc* (or other environment files) files for the user
- /etc/services file
- /etc/inetd.conf file
- /etc/rc* files
- *ps –aux* or *ps –ef* output file

For Windows Platforms only:

- \WINNT\SYSTEM32\DRIVERS\etc\HOSTS file
- \WINNT\SYSTEM32\DRIVERS\etc\LMHOSTS file
- \WINNT\SYSTEM32\DRIVERS\etc\SERVICES file
- \WINNT\SYSTEM32\DRIVERS\etc\NETWORKS file
- C:\WINNT\VSL.INI file
- C:\WINNT\WIN.INI file
- The client TCP/IP software HOSTS file (if used)
- The contents of the U2 ODBC driver installation directory (usually C:\U2\U2ODBC), with file sizes and dates

## Manual communications verification

The UCI Config Editor and Microsoft ODBC Data Source Administrator tool are usually all that is needed to configure and validate U2 ODBC driver access to a UniData or UniVerse database server. However, if you are having difficulties, you can use manual procedures to further diagnose communications difficulties, as described in the following sections.

## Verifying a TCP/IP connection

TCP/IP products typically provide several utilities to verify connectivity between the U2 ODBC driver and the UniData or UniVerse database server. Consult the documentation for your TCP/IP software for detailed instructions for performing the following checks.

To verify a TCP/IP connection, use *ping* or *telnet*.

### Ping

Provide the host name. The *ping* program looks up the host name in the local hosts file or in a name server and sends one or a series of low-level messages to the host, which echoes them back.

If using *ping* with the host name does not work, you can use *ping* with the IP address of the host. If the IP address works but the host name does not, either the local hosts file or the name server in use does not know the host by the name you are using. Check the spelling and capitalization (host names are case-sensitive).

If using *ping* with the IP address does not work to a system that is known to be available, you may have the wrong IP address, or there may be gateways in the Ethernet path between your client and the UniData or UniVerse database server that are not passing your requests. In this case, you need to get help from your system administrator.

If *ping* works, you have verified that the TCP/IP software has been installed and configured correctly, and the host name you used is valid (you still do not know if it is the host you want). Do not proceed until *ping* works.

### Telnet

Provide the host name, then log on with a user name and password to an interactive host terminal session. For information about how to use the *telnet* utility, see your TCP/IP software manual.

When you can log on successfully to the correct host through *telnet*, you have verified that your local hosts file or name server has the correct IP address for that host name, and that the user name and password you used are valid on that host.

## Verifying other types of network connection

The procedure for verifying other types of communications link (including asynchronous communications) depends on the type of hardware and software you have.

## Connection problems

If your client application does not show the detailed error message, you may need to create either a client log file or a `u2odbc.log` file in the working directory to verify the error.

- To create a `u2odbc.log` file, use the Microsoft ODBC Data Source Administrator tool. Click the Option button in the U2 ODBC Data Source Setup dialog box and select the check box for enabling a log file.

For more information on log files, see U2 ODBC log files, on page 79.

## Windows server-related problems

- Is the correct version of the Windows operating system installed?
- Is the UniRPC service running?
- Is there enough memory and paging space?
- Is there correct access to the UV account directory for all U2 ODBC user IDs?
- Are the permissions set correctly on UniData or UniVerse accounts for intended users (this may require read/write access to the VOC file)?
- Is the UniData or UniVerse Resource service installed and started?
- Does other TCP-based software use too many TCP connections?

## UNIX server-related problems

- Is the correct version of the UNIX operating system installed?
- Is the UniRPC daemon (*unirpcd*) running?
- Is there enough memory and paging space?
- Is there correct access to the UV account directory for all U2 ODBC user IDs?
- Are the correct permissions set on UniData or UniVerse accounts for intended users? (This can require write access to the VOC file.)
- Does other TCP-based software use too many TCP connections?
- Make sure that *.cshrc* does not change terminal parameters (*.login* and *.profile* do not need to be run), and does not echo anything to standard output.
- Are there entries in */etc/hosts* for all U2 ODBC client machines?

## Windows and UNIX server-related problems

- Does the UniVerse System Administration menu option **Activate Access to Files in an Account** run successfully on all UniVerse accounts to be accessed (as evidenced by the HS.INSTALL.LOG file in the UniVerse account)?
- Are all functions used by virtual fields or fields with conversion or correlative codes cataloged and listed in the account's VOC?
- Are secondary indexes defined on likely query fields where possible?
- Are the file user permissions in the HS_FILE_ACCESS file in this account set correctly, as is the user-specific record based on the login used in the client's configuration file?
- Does the record ID field appear in the dictionary of an updatable file?

## Client-related problems

- Does the system have the required configuration of memory, processor type, and operating system?
- Is the supported Ethernet card installed correctly?
- Is the client TCP/IP software installed correctly, and is it verified to communicate with this host?

- If your client TCP/IP software is configured to use a HOSTS file, is the HOSTS file accessible by the path (PATH environment variable on Windows)?
- Do the host names in the configuration file uci.config match those in the HOSTS file (case must match) or the name server?
- Do the HOSTS file entries for the host names used contain the correct IP addresses?
- Does the HOSTS file specify the IP address of the first Ethernet controller when multiple controllers are present?
- Was the U2 ODBC driver installation completed successfully?
- Make sure that other Windows applications do not take up too much memory.
- Are the host names in the configuration file uci.config spelled and capitalized correctly?
- Are you using the correct user names and passwords?
- Are the database parameters in the configuration file uci.config absolute paths?
- Is the server parameter in the configuration file uci.config an absolute path, and does it correspond to the UV account directory?