Rocket U2 Clients and APIs

Administrative Server Settings and Logging for U2 Clients

Version 5.3.0

October 2022
UCC-530-SUPP-UG-01

Notices

Edition

Publication date: October 2022
Book number: UCC-530-SUPP-UG-01
Product version: Version 5.3.0

Copyright

Trademarks

Examples

This information might contain examples of data and reports. The examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

License agreement

Note: This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when exporting this product.

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

Chapter 1: Introduction

The following APIs are provided for writing client application programs that connect to UniVerse and UniData databases.

They are called *common APIs* because programs written in them can access data in both databases.

The Client APIs are:

- UCI (UniVerse only)
- InterCall
- UniObjects for Java
- JDBC Driver for UniVerse and UniData
- Visual Schema Generator (VSG) (UniData only)
- U2 ODBC
- UniObjects for .NET
- U2 Toolkit for .NET

UCI (UniVerse only)

UCI is a C-language API. It lets developers write UNIX and Windows client programs that use SQL statements to access and manipulate data in UniVerse databases.

UCI is modeled on the ODBC standard as defined in the Microsoft ODBC 2.0 specification. It models only the API side of the ODBC standard, not the driver/transport side. Unlike the standard ODBC interface, UCI is more closely integrated with the extended relational database model used by UniVerse, with its nested tables, transaction processing support, and so forth.

InterCall

InterCall is an open API that lets client application programs developed on UNIX or Windows systems access data on UniVerse or UniData servers.

On UNIX systems, developers can write client programs using any tool that accesses static libraries, typically a C compiler. On Windows platforms, developers can write client programs using any tool that accesses DLLs, for example, Visual Basic, C, or Visual C/C++.

Note: InterCall replaces and supersedes ICI (Integrated Calling Interface).

UniObjects for Java

UniObjects for Java is an API that lets developers create Java-based applications that access UniVerse and UniVerse databases.

UniObjects for Java, based on the UniObjects model, is a 100% Pure Java Class Library whose objects can take full advantage of any Java-based IDE (Integrated Development Environment).

JDBC Driver for UniVerse and UniData

The JDBC driver for UniVerse and UniData is an interface to UniVerse and UniData databases from JDBC applications.

This book is for experienced programmers and application developers who are familiar with UniVerse and UniData, Java, JDBC, and who want to write JDBC applications that access these databases.

Note: The JDBC Driver for UniVerse and UniData does not require any configuration for UCI and will not need an entry within the UCI configuration file.

Visual Schema Generator (UniData only)

Visual Schema Generator (VSG), or Schema API, lets you prepare your database files for desktop access through UniData ODBC, UniData JDBC, or UniOLEDB.

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, JDBC, or UniOLEDB tools such as Visual Basic, PowerBuilder, and MS Access for ODBC, or WebSphere, EJB, or any J2EE-compliant tool for JDBC.

U2 ODBC

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.

UniObjects for .NET

UniObjects for .NET is an API that lets developers create .NET-based applications that access UniVerse and UniVerse databases.

UniObjects for .NET is fully integrated with the Microsoft environment.

Note: UniObjects for .NET is a deprecated product. It is replaced by U2 Toolkit for .NET.

U2 Toolkit for .NET

The U2 Toolkit for .NET Provider provides a comprehensive ADO.NET provider, LINQ to Entity provider, and the native UniObjects for .NET API for the U2 databases. Use Microsoft Visual Studio 2010 or later, to build applications and take advantage of the powerful Microsoft .NET Framework and CLR.

The U2 Toolkit for .NET Developers allows you to easily design U2 applications within Visual Studio. The U2 Toolkit for .NET works with both 32-bit and 64-bit Windows operating systems.

U2 Toolkit for .NET is made up of three primary components:

- U2 Toolkit for .NET Provider (ADO.NET Provider)
- U2 Entity Data Provider for .NET (LINQ to Entity)
- U2 Toolkit for .NET Developer (Visual Studio Add-ins for Server Explorer Integration)

Developers can use U2 Toolkit for .NET to take advantage of server-based capabilities, such as:

- Automatic Data Encryption (ADE)
- Secure Sockets Layer (SSL)
- Connection Pooling
- TOXML ('FillWithTOXML Method' in the on-line documentation)
- TOJSON ('ExecuteJson Method' in the on-line documentation)

U2 users can find more information in the *U2 Toolkit for .NET* manual.

U2 Client build information

Following is the build information for the 5.2.0 U2 Clients.

- UniObjects for .NET Framework 4.5 is built with Visual Studio 2013
- All other products are built with Visual Studio 2010 and Java 1.8

Chapter 2: Maintaining the UniRPC

The UniRPC lets UniVerse/UniData systems communicate with remote UNIX or Windows systems. The communicating systems must use TCP/IP networking software to make connections.

In this chapter the terms local and remote refer to client and server programs or systems. However, because client programs can connect to server programs running on the same computer, remote does not necessarily imply that the server is on another physical computer system.

Note: On UNIX UniData 7.3 or early versions, udt or udapi_slave processes load the `libodbc.xx` dynamic library based on the `/.udlibs73` link setting, similar to:

```
 ldd udt
/.udlibs73/libodbc.so (0xf7f4c000)
```

In UniVerse 11.3.1 and UniData 8.1 or higher, it might load a different `libodbc.xx` based on the $LD_LIBRARY_PATH variable setting, similar to:

```
[root@dentr64 bin]# ldd udt
libodbc.so => /usr/local/lib/libodbc.so (0x00007f6cef337000)
```

When UniData BCI and EDA users do not set the LD_LIBRARY_PATH environment variables correctly, it will not work with their ODBC connection.

In UniVerse 11.3.1 and UniData 8.1 or higher, we suggest setting the LIBPATH or LD_LIBRARY_PATH environment variable.

This chapter describes:

- The UniRPC daemon (on UNIX servers)
- The UniRPC service (on Windows servers)
- The contents of the `unirpcservices` file

The UniRPC on UniVerse servers requires little maintenance, other than starting and stopping the UniRPC daemon or service. On UniVerse servers, you can also do the following:

- Change the port number
- Add entries to a UNIX `hosts` file
- UNIX UniRPC daemon debugging log
- Windows UniRPC Service debugging log

System requirements

Before installing layered or third-party products that use the UniRPC, such as the UniDK, UniOLEDB, the JDBC Driver for UniVerse and UniData, or XAdmin, you must install and configure TCP/IP using the instructions supplied by the TCP/IP facility vendor.

On UNIX UniVerse systems, you should then identify the systems to be networked with the database by defining them in the /etc/hosts file. See Maintaining the hosts file (UniVerse Only), on page 12 for more information.

How the UniRPC works

The UniRPC daemon unirpcd (or the UniRPC service unirpc) waits for a request from a client system to connect to a server process.

When it receives a connection request, it checks the `unirpcservices` files to verify that the client system is allowed to request a particular service. If it can, the UniRPC starts the requested service, then returns to the listening state. Each client process connects to its own server process. Each server process uses the same amount of system resources as a local database user.

Maintaining the UniRPC

This section describes the following:

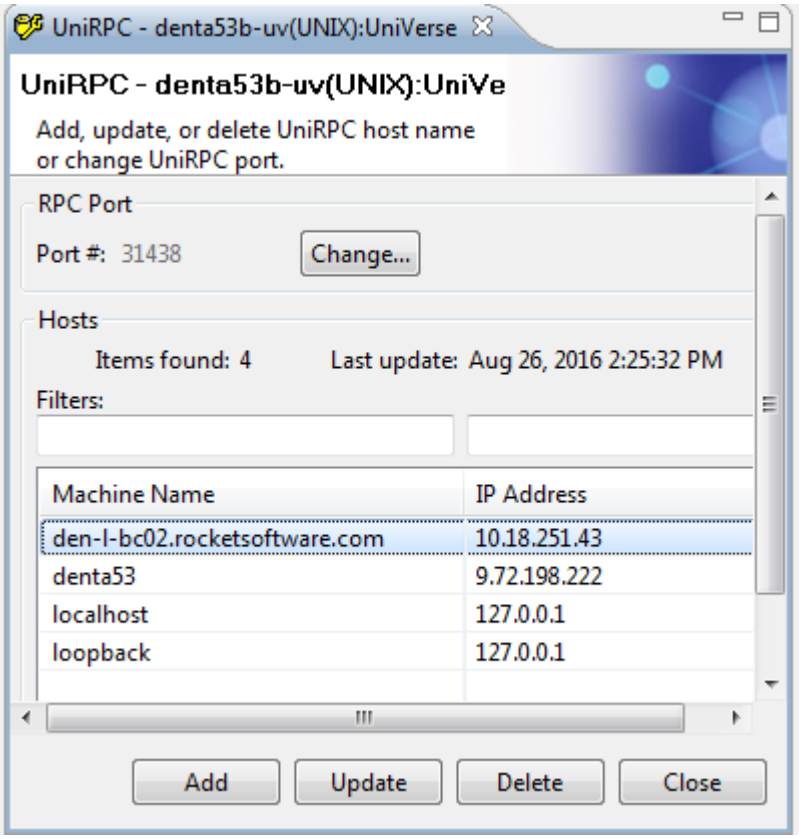
- How to change the UniRPC port number (UNIX UniVerse only)
- How to maintain a UNIX server's `hosts` file (UNIX UniVerse only)
- How to start and stop the UniRPC daemon (unirpcd)

UniRPC maintenance on UNIX UniVerse systems

Use XAdmin to:

- Define the UniRPC port number
- Maintain the *hosts* file on a UNIX server

Choose **Network Services** from the **XAdmin** menu. The Network Services window appears, as shown in the following example:



This window has the following components:

- Port # field. The current port number for the UniRPC daemon.
- Hosts list. Displays the machine name and IP address for each node in the `/etc/hosts` file.

Note: If you are using the Network Information Services (NIS, also known as Yellow Pages), you do not need to use the `/etc/hosts` file to define, change, and delete network nodes. See the UNIX networking documentation provided with your system for more information.

Defining the UniRPC port number (UNIX UniVerse Only)

Before you can use the UniRPC, you must specify the number of the port that the UniRPC is to use. You specify the port number on the client and the server systems. If you specify a port number other than the default, it must be the same on all systems that communicate via the UniRPC.

The current UniRPC daemon port number is displayed in the **Port #** field in the Network Services window. To change the number, do the following:

1. Click **Change**.
   The Change Port Number dialog box appears.
2. Enter a new number in the **Enter new Port number** field.
3. Click **OK**.
   The new port number is saved in the `UNIX/etc/services` file and the Network Services window is updated with the new setting.

To use the new port number, you must restart the UniRPC daemon (see Starting the UniRPC daemon on UNIX UniVerse systems, on page 14)

Maintaining the hosts file (UniVerse Only)

Use the Network Services option of XAdmin to add, modify, and remove nodes in the hosts file. These tasks are performed from the Network Services window.

Adding a node

To add a new node to the hosts file:

1. Click **Add** on the Network Services window.

   The Add Node dialog box appears.
2. Enter the node name in the **Machine Name** field.
3. Enter the node address in the **IP Address** field.
4. Click **OK**.

   The new node's machine name and IP address are checked against existing entries in the `hosts` file. If the new node matches an existing entry, a message box appears. You must acknowledge the message before you can enter alternative values. If the new node details are unique, the new node definition is added to the `hosts` file and the Network Services window is updated.

Modifying a node

To modify the name or IP address of an existing entry in the hosts file:

1. Choose the node to modify by doing one of the following:
   - Double-click the node in the **Hosts** list.
   - Choose the node and click **Modify**.

     The Modify Node dialog box appears.
2. Edit the entries in the **Machine Name** and **IP Address** fields.
3. Click **OK**.

   The node's machine name and IP address are checked against existing entries in the `hosts` file. If the node details match an existing entry, a message box appears. You must acknowledge the message before you can enter alternative values. If the node details are unique, the node definition is added to the `hosts` file and the Network Services window is updated.

Removing a node

To remove a node definition from the hosts file:

1. Select the node from the **Hosts** list.
2. Click **Remove**.

   A message box appears.
3. Click **Yes**. The node definition is removed from the `hosts` file and the Network Services window is updated.

Starting the UniRPC on Windows platforms

On UniVerse or UniData systems you cannot use Xadmin to start the UniRPC daemon because it uses the UniRPC daemon to connect to the UniVerse or UniData server. On Windows platforms, you can start the UniRPC daemon or service in one of three ways:

- From the Windows Control Panel

- From the UniVerse Control Panel
- At the MS-DOS prompt

From the Control Panel

1. Double-click the **Services** icon.
2. Scroll down the list of services until you find three entries for UniVerse: UniVerse Resource Service, UniRPC Service, and UniVerse Telnet Service.
3. Choose **UniRPC Service**, then choose **Start**.
4. Click **Startup**, then click **Automatic**.

   This ensures that UniVerse starts automatically when the server is rebooted.

From the UniVerse Control Panel

1. Choose **Start > Programs > Rocket U2 > UniVerse Control.**
2. Click the **Start All Services** button to start all UniVerse services.

At the MS-DOS prompt

Enter the following command:

```
D:\users>net start unirpc
```

The system reports the name of the service it is starting and whether the startup is successful.

Note: The UniVerse services are started automatically when the operating system is loaded unless you clear the automatic startup boxes during UniVerse installation.

Stopping the UniRPC on Windows platforms

You can shut down the UniRPC daemon or service in one of three ways:

- From the Windows Control Panel
- From the UniVerse Control Panel
- At the MS-DOS prompt

Note: If users are connected to the services when they are shut down, the users do not lose their connections; the connections remain active until the users terminate them. However, it is not possible for new users to connect to UniVerse. If you want to do a complete shutdown of UniVerse to restart the services, be sure that all connections are terminated first.

From the Control Panel

1. Double-click the **Services** icon.
2. Scroll down the list of services until you find three entries for UniVerse:
   - UniVerse Resource Service
   - UniRPC Service
   - UniVerse Telnet Service
3. Choose **UniRPC Service**, then choose **Stop**.
4. Click **OK**.

   The UniRPC daemon or service is shut down.

From the UniVerse Control Panel

1. Choose **Start > Programs > Rocket U2 > UniVerse Control**.
2. Click **Stop All Services** to stop all UniVerse services. Wait for all services to stop.
3. Click **OK** to exit the UniVerse Control Panel.

   All four services are shut down.

At the MS-DOS prompt

You can shut down the UniRPC daemon or service in one of three ways:

1. Enter the following command at the MS-DOS prompt:

   ```
   D:\users>net stop unirpc
   ```

   A message appears prompting you to confirm that you want to stop the UniRPC.
2. Enter `Y` to stop the UniRPC daemon or service.

Starting the UniRPC daemon on UNIX UniVerse systems

Use the UniVerse System Administration menus on the UniVerse server to start the UniRPC daemon. See *Administering UniVerse on Windows and UNIX Platforms* for more information.

1. Choose **Rpc administration** from the **Package** menu, then choose **Start the rpc daemon**.
2. At the prompt, do one of the following to handle any error messages.
   - Enter the name of the file to send all error and system messages to.
   - Enter a space to display messages on your screen.
   - Press Enter if you do not want to display or save message.

   Once you start the UniRPC daemon, it automatically restarts whenever you boot UniVerse.
3. At the next prompt, click **Yes** to start the UniRPC daemon or **No** to return to the **Rpc administration** menu.

> **Note:** The file that receives all error and system messages can grow unchecked unless you monitor it periodically.

Starting the UniRPC daemon on UNIX UniData systems

Enter the following command:

```
# $UDTBIN/startunirpcd
```

Stopping the UniRPC daemon on UNIX UniVerse systems

Use the UniVerse System Administration menus on the UniVerse server to stop the UniRPC daemon. See *Administering UniVerse on Windows and UNIX Platforms* for more information.

1. Choose **Rpc administration** from the **Package** menu, then choose **Halt the rpc daemon**.
2. At the next prompt, click **Yes** to stop the UniRPC daemon or **No** to return to the **Rpc administration** menu.

> **Note:** Stopping the UniRPC daemon does not interrupt active UniRPC processes.

Stopping the UniRPC daemon on UNIX UniData systems

Enter the following command:

```
# $UDTBIN/stopunirpcd
```

UniRPC maintenance on U2 servers

On UniVerse servers, UniRPC maintenance is minimal. You are not required to change the port number of the UniRPC, and there is no need to maintain a `hosts` file.

On UniiData servers, Use the `stopunirpcd` or `stopud` command to stop the UniRPC daemon or service. Use the `startunirpcd` or `startud` command to start the UniRPC.

About the unirpcservices file

Each process that uses the UniRPC automatically configures the *unirpcservices* file when it first starts. If no *unirpcservices* file exists, it is created in the *unishared* directory.

- On UNIX systems the default location of this file is `/usr/unishared/unirpc`.
- On Windows platforms the default location is `<drive>:\u2\unishared\unirpc`.

To determine the location of the `unirpcservices` file on your system, do the following:

- On UNIX systems, execute the command:

  ```
  $ cat /.unishared
  ```

- On Windows platforms, find the registry entry under the subkey `\HKEY_LOCAL_MACHINE \SOFTWARE\Rocket Software\unishared`.

When a client system requests a connection to a service on a server system, the UniRPC daemon (unirpcd) on the server uses the `unirpcservices` file to verify that the client system can start the requested service.

The UniRPC software uses field 3 of the `unirpcservices` file to verify that a machine making a request for a service is allowed to do so.

The following table lists the fields in the `unirpcservices` file:

Field	Contents
1	The name of the UniRPC service (for example, *uvserver*).
2	The full path of the service engine executed by the UniRPC daemon.
3	The names of nodes allowed to execute this service. This field is multivalued, with values separated by commas (no spaces). If the field contains * (asterisk), all hosts defined in `/etc/hosts` can execute this service.
4	The network transport mechanism for the service (TCP/IP).
5	Reserved for future use.
6	The value (in seconds) specifying how long an open connection can be idle before automatic closure from the remote connection. The default is 3600, or 60 minutes.

UniVerse systems

On UniVerse systems,the *unirpcservices* file might contain entries such as the following:

```
uvnet /usr/uv/bin/uvnetd host1,host2,host3 TCP/IP 3 3600
uvdrsrv /usr/uv/bin/uvdrsrvd * TCP/IP 0 3600
uvcs /usr/uv/bin/uvapi_server * TCP/IP 0 3600
uvfilefix /usr/uv/bin/uvfilefix_server * TCP/IP 0 3600
uvserver /usr/uv/bin/uvsrvd * TCP/IP 0 3600
```

The version of *uv.rc* shipped with UniVerse systems (`/usr/uv/sample/uv.rc`) contains commands that:

- Check for the existence of the `unirpcservices` file
- Verify that services are defined in it
- Start the UniRPC daemon if the file contains services

The UniRPC daemon is executed as part of the UniVerse reboot procedure.

UniData systems

On UniData systems the `unirpcservices` file might contain:

```
udcs /usr/ud81/bin/udapi_server * TCP/IP 0 3600
udserver /usr/ud81/bin/udsrvd * TCP/IP 0 3600
```

UNIX UniRPC daemon debugging log

Start the `unirpcd` daemon with debugging mode in a UNIX environment.

Follow these steps:

1. Connect to the UNIX server as the root user.
2. Use the `cat /.unishared` command to find the `unishared` folder.
3. Change directory (`cd`) to the `unishared` folder.
4. Change directory (`cd`) to the `unirpc` folder.
5. Run the following commands:

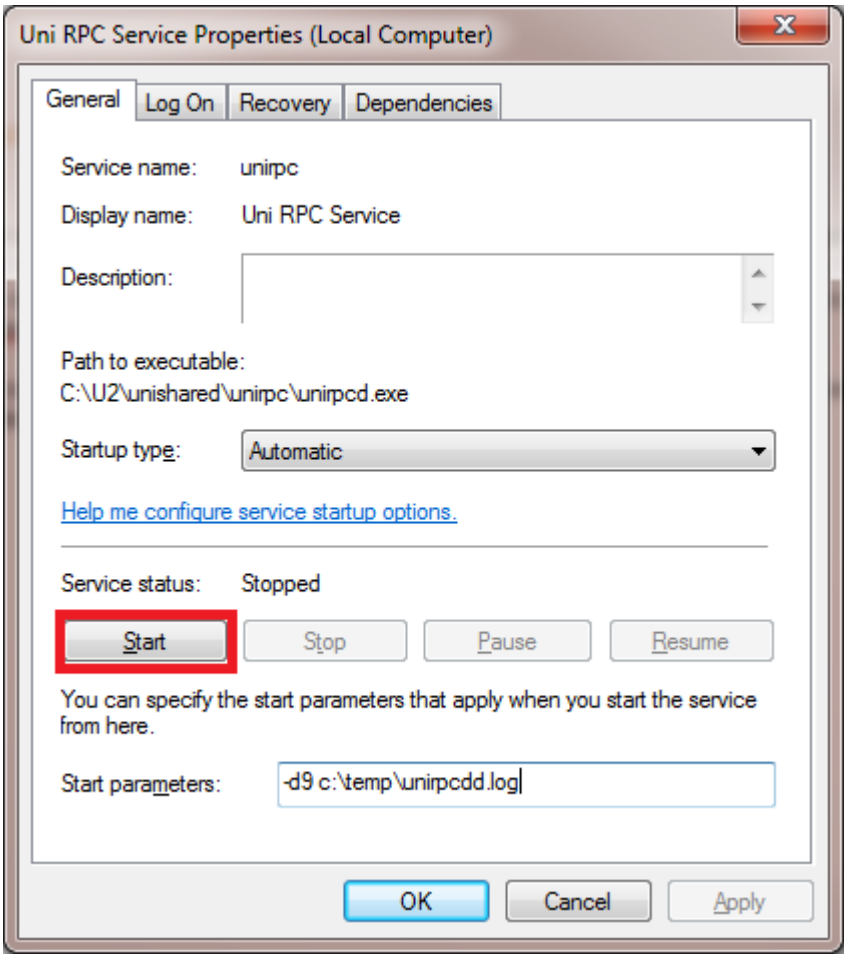
   ```
   ./unirpcd -d9 -timeout3 > /tmp/unirpcd.log 2>&1 &
   ```

   where *-d9* is debug level 9. Another option is *-d1* to reduce the unirpcd daemon log size, and where *-timeout3* sets the 3 seconds timeout to ignore a new client request if the packet is unknown.

Windows UniRPC service debugging log

In the Windows Services menu, restart the UniRPC Service using the Start parameters with the *-d9* option. It creates a `unirpcd.log` file in the system32 directory by default. The log file can be specified at the specific output folder. The Start parameters setting will not be saved to the service properties, and it will not create the unirpcd debug log automatically when you reboot the machine.

The following illustrates how to start the UniRPC service with debug *-d9* option and new log file created in the `c:\temp folder`:



Note: You must click on the **Start** button instead of the **OK** button to create the log file.

Chapter 3: Accessing UniData Accounts

UniData databases are organized into accounts. A consumer 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. You can also include the account path or database name in the UCI data source definition in the UCI configuration file.

For information about setting up the UCI Configuration file, see the *UCI Configuration Editor* documentation.

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

If you want to access an account that has a UDTHOME directory different from the default UDTHOME directory, you must include a definition for that account in the `ud_database` file on the server. On UNIX systems this file is located in the `/usr/ud81/include` path. On Windows platforms it is located in `\udthome\include`. You can find the path for udthome by looking in the registry under `HKEY_LOCAL_MACHINE\SOFTWARE\Rocket Software\UniData\8.1`. Use any text editor to modify the `ud_database` file.

To determine your default UniData home directory, use the UNIX `env` command. Output from this command includes the default setting for the UDTHOME environment variable.

The following Windows example shows an entry in the `ud_database` file for a database named `dbase2`:

```
DATABASE=dbase2
UDTHOME=d:\disk2\test81
UDTACCT=d:\disk2\test81\testacct
TRACE_LEVEL=0
```

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

Note: The account name defined in the UD.ACCOUNT file is not used by the UniData client driver.

Running concurrent UniData versions

When you install UniData 8.1 on a machine where 7.3 was previously installed, the unirpcservices file is overwritten with UniData 8.1 information. If you want to run UniData 7.3 concurrently with 8.1, you must edit certain files to enter the UniVerse 7.3 or 8.1 definitions.

Running UCI, UniData ODBC, or UniOLEDB concurrently

If you are running UCI, UniData ODBC, or UniOLEDB with concurrent versions of UniData, you must edit the unirpcservices file and the uci.config file to define locations of executables from the previous version of UniData.

The following example illustrates unirpcservices file entries when running UniData 7.3 concurrently with UniData 8.1:

```
udserver_81 d:\u2\ud81\bin\udsrvd.exe * TCP/IP 0 3600
udserver_73 c:\u2\ud73\bin\udsrvd.exe * TCP/IP 0 3600
```

Make sure the uci.config file contains an entry for the server for each version of UniData, as shown in the following example:

```
<UniData81>
DBMSTYPE = UniData
network = TCP/IP
service = udserver_81
host = server1

<UniData73>
DBMSTYPE = UniData
network = TCP/IP
service = udserver_73
host = server2
```

Running InterCall, UniObjects, or UniObjects for Java concurrently

If you are running InterCall, UniObjects, or UniObjects for Java with concurrent versions of UniData, you must edit the unirpcservices file to define the location of the udapi_server executable for the previous version you are running.

The following example illustrates unirpcservices file entries when running UniData 7.3 concurrently with UniData 8.1:

```
udcs_81 C:\u2\ud81\bin\udapi_server.exe * TCP/IP 0 3600
udcs_73 C:\u2\ud73\bin\udapi_server.exe * TCP/IP 0 3600
```

You can now set your service name to either service defined in the unirpcservices file, for example, udcs_72 for UniVerse 7.2 or udcs for UniVerse 7.3.

Tracing events for U2 ODBC, JDBC, or U2 Toolkit UCI connection server

You can use the tracing feature to create logs of events between clients and the database through the server. Logs enable support personnel to help troubleshoot problems. You can define trace levels for database entries in the `ud_database` file. The following table describes the valid trace levels and the associated information that is written to the trace log:

Trace level	Description
0	Includes all fatal error information.
1	Includes all UCI functions in addition to the information provided by trace level 0.
2	Includes parameter information and column descriptions in addition to the information provided by trace levels 0 and 1.
3	Includes data values in addition to the information provided by trace levels 0, 1, and 2.

The following UNIX example shows a tracing level setting for a database named *dbase2*:

```
DATABASE=dbase2
UDTHOME=/disk1/ud81
UDTACCT=/home/test/udtest
```

```
TRACE_LEVEL=3
```

Turning on the UniObjects server, UOJ, or U2 Toolkit native connection log based on specific service name

On a U2 server, you can turn the UniObjects server log on or off by creating a new `serverdebug` file in the UniData home folder. When the server log is turned on, it might generate a lot of log files for all of the UniObjects server processes. This makes it difficult to debug the specific application running on the live server. On the newer U2 server versions, it is possible for U2 users to create the log files for a specific service name.

You must provide the logging information with the service name, log level, and prefix log file name in the existing folder name.

- Example 1 - serverdebug for a UNIX UniData server.

  ```
  udcs 10 /tmp/udcs.log
  ```

  Note: The service name must be udcs for UniData. It does not matter if the application is using a different service name like defcs. The log level can be set to 0, 1, 3, 9, or 10.

  Log level 1: All UO server and slave process connection and communication.

  Log level 9: Add all client function debugging information.

  Log level 10: Additional calling basic debugging information.

  The log folder must exist and be writable. For this sample setting, the user must have write permissions in the `/tmp` folder. The log file name will be `udcs.log_<pid>` in the `/tmp` folder.

- Example 2 - serverdebug for a Windows UniData server.

  ```
  udcs 10 c:\temp\udcs.log
  ```

  Note: The Windows user must have write permissions in the existing `c:\temp` folder. The serverdebug file name should not have a .txt extension, such as `serverdebug.txt`.

- Example 3 - serverdebug log associated to a specific service name.

  From UniData version 7.3.x or later, you can turn on the UniObjects server log associated to a specific service name. The service name must also be defined in the `unirpcservices` file.

  With the `unirpcservices` file defined as:

  ```
  defcs C:\U2\ud73\bin\udapi_server.exe * TCP/IP 0 3600
  udcs C:\U2\ud73\bin\udapi_server.exe * TCP/IP 0 3600
  udcsx C:\U2\ud73\bin\udapi_server.exe * TCP/IP 0 3600
  udcsy C:\U2\ud73\bin\udapi_server.exe * TCP/IP 0 3600
  ```

  The resulting serverdebug logs look similar to:

  ```
  udcsx 10 c:\temp\udcsx.log
  udcsy 10 c:\temp\udcsy.log
  ```

Note: For general udcs or defcs connections, a log file will not be created. When the UniObjects client application is set to either the udcsx or udcsy service name, it will generate the log files in the `c:\temp` folder.

- Example 4 - without a log path setting

```
udcs 10
```

The log file will be stored as `udapiserver_<pid>.log` in the `tmp` folder.

Chapter 4: Accessing UniVerse Accounts

UniVerse databases are organized into accounts. A consumer connects to a UniVerse account and can access the files there. You optionally can define the account name in the `UV.ACCOUNT` file on the server. You can also include the account path or database name in the UCI data source definition in the UCI configuration file.

For information about setting up the UCI Configuration file, see *UCI Configuration Editor* .

You can also specify the account path or account name each time you try to connect to the account. In this case you need not include the account path or account name in the UCI configuration file. When you try to connect, you are prompted to specify either the full path to the account or the account name.

Tracing events for U2 ODBC, JDBC, or U2 Toolkit UCI connection server

Beginning at UniVerse version 10.1, TRACEuci provides more complete logging. The location of the log file is created in the `UVTEMP` directory.

To enable UCI logging:

1. Create a text file in the UV home directory, named `TRACEuci` (case-sensitive), using any text editor.
2. Leave the file empty.

To disable logging:

1. Rename or remove `TRACEuci` from the UV home directory.

The trace file is called `UVUCIlog_<pid>` and is created in the `$UVTEMP` directory. On UNIX, `$UVTEMP` defaults to `/tmp`. On Windows, it defaults to `c:\U2\uv\uvtemp` for UV 11.1 version or higher.

Note: The `TRACEuci` file must not have the .txt extension on Windows system.

Turning on the UniData UniObjects server, UOJ, or U2 Toolkit native connection log based on specific service name

On a U2 server, you can turn the UniObjects server log on or off by creating a new `serverdebug` file in the UniData home folder. When the server log is turned on, it might generate a lot of log files for all of the UniObjects server processes. This makes it difficult to debug the specific application running on the live server. On the newer U2 server versions, it is possible for U2 users to create the log files for a specific service name.

You must provide the logging information with the service name, log level, and optional prefix log file name in the existing folder name. There are several settings options for UniData UniObjects server logging.

- Example 1 - serverdebug for a UNIX UniData server.

```
uvcs 10 /tmp/udcs.log
```

> **Note:** The service name must be udcs for UniData. It does not matter if the application is using a different service name like defcs. The log level can be set to 0, 1, 3, 9, or 10. When the `udcs` service name is defined at the first line of serverdebug file, all UniObjects server process logs will be created based on the `udcs` entry setting. It will ignore all other settings defined in the `serverdebug` log file. It is the same rule applied for UNIX and Windows platform.

Log level 1: All UO server and slave process connection and communication.

Log level 9: Add all client function debugging information.

Log level 10: Additional calling basic debugging information.

The log folder must exist and be writable. For this sample setting, the user must have write permissions in the `/tmp` folder. The log file name will be `udcs.log_<pid>` in the `/tmp` folder.

- Example 2 - serverdebug for a Windows UniData server.

```
udcs 10 c:\temp\udcs.log
```

> **Note:** The Windows user must have write permissions in the existing `c:\temp` folder. The serverdebug file name should not have a .txt extension, such as `serverdebug.txt`.

- Example 3 - serverdebug log associated to a specific service name.

From UniData version 7.3.0 or later, you can turn on the UniObjects server log associated to a specific service name. The following setting is designed to create a log file on some specific processes, but not all processes. The `udcs` entry should not be defined in the serverdebug file. The service name must also be defined in the `unirpcservices` file.

With the `unirpcservices` file defined as:

```
defcs C:\U2\uv\bin\udapi_server.exe * TCP/IP 0 3600
udcs C:\U2\uv\bin\udapi_server.exe * TCP/IP 0 3600
udcsx C:\U2\uv\bin\udapi_server.exe * TCP/IP 0 3600
udcsy C:\U2\uv\bin\udapi_server.exe * TCP/IP 0 3600
```

The resulting serverdebug logs look similar to:

```
udcsx 10 c:\temp\udcsx.log
udcsy 10 c:\temp\udcsy.log
```

> **Note:** For general udcs or defcs connections, a log file will not be created. When the UniObjects client application is set to either the udcsx or udcsy service name, it will generate the log files in the `c:\temp` folder.

- Example 4 - without a log path setting

```
udcs 10
```

The log file will be stored as `udapiserver_<pid>.log` in the `tmp` folder.

- Example 5 – mixed `udcs` and other specific service name setting (not recommended)

```
udcsx 10 /tmp/udcsx.log
```

```
udcs 10 /tmp/udcs.log
udcsy 10 /tmp/udcsy.log
```

Note: The `udcsx` and `udcs` service setting will be respected, but the `udcsy` service setting will be ignored due to violate the first rule (see example 1). Another option is to change the `udcs` setting to the last line of `serverdebug` file.

```
udcsx 10 /tmp/udcsx.log
udcsy 10 /tmp/udcsy.log
udcs 10 /tmp/udcs.log
```

Note: All UniObjects server processes will create new log files. The `udcsx`" and `udcsy` service will create different log file name based on the setting. All other service log file names will be created based on the `udcs` setting.

Turning on the UniVerse UniObjects server, UOJ, or U2 Toolkit native connection log based on specific service name

On a U2 server, you can turn the UniObjects server log on or off by creating a new `serverdebug` file in the UniVerse home folder. When the server log is turned on, it might generate a lot of log files for all of the UniObjects server processes. This makes it difficult to debug the specific application running on the live server. On the newer U2 server versions, it is possible for U2 users to create the log files for a specific service name.

You must provide the logging information with the service name, log level, and optional prefix log file name in the existing folder name. There are several settings options for UniVerse UniObjects server logging.

- Example 1 - serverdebug for a UNIX UniVerse server.

  ```
  uvcs 10 /tmp/uvcs.log
  ```

  Note: The service name must be uvcs for UniVerse. It does not matter if the application is using a different service name like defcs. The log level can be set to 0, 1, 3, 9, or 10. When the `uvcs` service name is defined at the first line of serverdebug file, all UniObjects server process logs will be created based on the `uvcs` entry setting. It will ignore all other settings defined in the `serverdebug` log file. It is the same rule applied for UNIX and Windows platform.

  Log level 1: All UO server and slave process connection and communication.

  Log level 9: Add all client function debugging information.

  Log level 10: Additional calling basic debugging information.

  The log folder must exist and be writable. For this sample setting, the user must have write permissions in the `/tmp` folder. The log file name will be `uvcs.log_<pid>` in the `/tmp` folder.

- Example 2 - serverdebug for a Windows UniVerse server.

  ```
  udcs 10 c:\temp\uvcs.log
  ```

Note: The Windows user must have write permissions in the existing `c:\temp` folder. The serverdebug file name should not have a .txt extension, such as `serverdebug.txt`.

- Example 3 - serverdebug log associated to a specific service name.

  From UniVerse version 11.2.0 on UNIX or 11.2.3 on Windows, you can turn on the UniObjects server log associated to a specific service name. The following setting is designed to create a log file on some specific processes, but not all processes. The `uvcs` entry should not be defined in the serverdebug file. The service name must also be defined in the `unirpcservices` file.

  With the `unirpcservices` file defined as:

  ```
  defcs C:\U2\uv\bin\uvapi_server.exe * TCP/IP 0 3600
  uvcs C:\U2\uv\bin\uvapi_server.exe * TCP/IP 0 3600
  uvcsx C:\U2\uv\bin\uvapi_server.exe * TCP/IP 0 3600
  uvcsy C:\U2\uv\bin\uvapi_server.exe * TCP/IP 0 3600
  ```

  The resulting serverdebug logs look similar to:

  ```
  uvcsx 10 c:\temp\uvcsx.log
  uvcsy 10 c:\temp\uvcsy.log
  ```

  Note: For general uvcs or defcs connections, a log file will not be created. When the UniObjects client application is set to either the uvcsx or uvcsy service name, it will generate the log files in the `c:\temp` folder.

- Example 4 - without a log path setting

  ```
  uvcs 10
  ```

  The log file will be stored as `uvapiserver_<pid>.log` in the `tmp` folder.

- Example 5 – mixed `uvcs` and other specific service name setting (not recommended)

  ```
  uvcsx 10 /tmp/uvcsx.log
  uvcs 10 /tmp/uvcs.log
  uvcsy 10 /tmp/uvcsy.log
  ```

  Note: Once the ‘uvcs’ entry is found in serverdebug, anything listed after it will be ignored. However, as long as the specific entries are defined first, the individual log files will be used. The `uvcsx` and `uvcs` service setting will be respected, but the `uvcsy` service setting will be ignored due to violate the first rule (see example 1). Another option is to change the `uvcs` setting to the last line of `serverdebug` file.

  ```
  uvcsx 10 /tmp/uvcsx.log
  uvcsy 10 /tmp/uvcsy.log
  uvcs 10 /tmp/uvcs.log
  ```

  Note: All UniObjects server processes will create new log files. The `uvcsx`” and `uvcsy` service will create different log file name based on the setting. All other service log file names will be created based on the `uvcs` setting.

Chapter 5: Device licensing

This chapter describes how device licensing works.

For more information about device licensing, see *Administering UniVerse on Windows and UNIX Platforms*.

Licensing modes

UniVerse and UniData provide two licensing modes:

- Session licensing
  Session licensing is like the licensing system used before Release 9.5 of UniVerse and Release 5.1 of UniData. Every connection from telnet or an API, even from the same PC, consumes one database license.
- Device licensing
  Device licensing, sometimes called client-side licensing, tries to combine all remote connections from a single device to a database server at both the database license level and the package level.

Session licensing

Session licensing is like the licensing system used before Release 9.5 of UniVerse and Release 5.1 of UniData. Every connection from telnet or an API, even from the same PC, consumes one database license.

On UniVerse systems, session licensing has been enhanced to include a new licensing tool, *uvlictool*, that reports on the current licensing state and cleans up current licensing.

Parent topic: Licensing modes

Device licensing

Device licensing, sometimes called client-side licensing, tries to combine all remote connections from a single device to a database server at both the database license level and the package level.

Device licensing works with the following connection types (among others):

- UCI
- InterCall
- UniObjects for Java
- JDBC
- U2 ODBC
- UniObjects for .NET
- U2 Toolkit for .NET

Parent topic: Licensing modes

Why do I need device licensing?

Users accessing a database server through one or more client application programs may want to put their licensing scheme on a one-license-per-device basis. Such applications often open multiple connections to a database server. For example, an application might use one connection to browse, another connection to check data, yet another connection to update the database, and so forth.

Before UniVerse Release 9.5 and UniData Release 5.1, each connection to the server consumed its own separate license, even though only one user was using all those connections from one PC. Device licensing lets such users consume one database license and the number of connections for which they are licensed, up to ten, to the server from a single PC.

Device licensing requirements

Device licensing has the following requirements:

- Clients must run on a Windows platform.
- Clients must run on a LAN or TCP/IP with an Ethernet card.

Connection types

There are three ways to connect to a database server:

- Direct connection. This is not a client/server connection.
- Two-tier client/server connection.
- Multiple-tier client/server connection.

Each PC can have up to ten connections to the server, but not all connections from a PC can be combined.

Direct connections

Direct connections are not really client/server connections because there is no real client. Examples of direct connections are:

- Directly invoking the database on a system
- TTY serial line

Two-tier connections

Two-tier connections are typical client/server connections where a client application connects to a database server either on the same machine or on a different machine. Telnet connections to the database are an example of a two-tier connection.

Client applications running on PCs different from the database server appear to the server with unique identifiers.

Multiple-tier connections

Multiple-tier connections are client applications that connect from a PC to a database server either through one or more different PCs, or through an application server component.

Examples of multiple-tier connections are:

- An HTTP server running scripts that use UniObjects or UniObjects for Java.
- An application that connects first to an application server either on a different PC or on the server system. The application server connects to the database server.

Using device subkeys

Each PC that connects immediately to the database server can have up to ten connections.

Using multiple-tier connections, each PC that connects to an intermediate application component consumes a separate license. But each of these PCs, at one or more removes from the server, can have up to ten connections.

In order for a PC to have multiple connections to the database server and still consume only one license, users must ensure that each PC connecting to the server through another system specify a unique device subkey before requesting a connection to the server. This subkey is a string of up to 24 characters. All client applications on a given device that connect to one database server must use the same unique subkey.

Chapter 6: U2 server security control subroutine and U2 environment settings

This chapter describes the U2 server security subroutine and U2 server environment settings.

UniData and UniVerse native client connection control

UniData server supported the UOLOGIN security subroutine from version 5.1. However, there was no way to limit access to UniVerse by middleware clients. Beginning at UniVerse version 11.1.1, the UniObjects server now supports the use of the optional UOLOGIN subroutine to impose security on client connections being made to the server.

The UOLOGIN subroutine is used to control access to the UniData database or UniVerse accounts from middleware clients. UniObjects, UniObjects for .NET, UniObjects for Java, and InterCall check for its existence. The subroutine supports all the capabilities available in UniData or UniVerse BASIC so it can be tailored to meet the security needs of your specific environment.

UOLOGIN is executed when a client connection is initiated. It must be cataloged globally for UniVerse. It can be locally or globally for UniData. If UniVerse UOLOGIN is not cataloged globally, and the UniObjects server log is enabled, the error message `*UOLOGIN is not in the CATALOG space` is written to the server debug log file. When UOLOGIN is called by a client connection, a return value of 0 results when the subroutine fails, and error 80011 is sent to the client.

The UOLOGIN subroutine contains two arguments:

```
SUBROUTINE UOLOGIN(RTNVAL,APPNAME)
```

- RTNVAL - If RTNVAL is a nonzero value, the connection is allowed. If RTNVAL is 0, the connection is not allowed and an error message is returned.
- APPNAME - The name of the client application trying to establish the connection.

> **Note:** The application name is set on the client side. This is done by changing the value of the username before establishing a connection to Your_Application_Name:username. The additional data will be used in the UOLOGIN subroutine as the APPNAME parameter.

UniData and UniVerse ODBC client connection control

The UniData or UniVerse ODBC server provides the ability to impose security on ODBC connections made to the server through the use of the ODBCLOGIN subroutine.

UniData supports the ODBCLOGIN security subroutine from version 7.1. The ODBCLOGIN subroutine was added to UniVerse at the 11.2.4.4710 release or later.

UniData or UniVerse ODBC connections check for the existence of the ODBCLOGIN subroutine, and, if found, will execute the subroutine. If the subroutine is not found, the connection is made without any restriction.

ODBCLOGIN is a globally cataloged BASIC subroutine containing two arguments. The subroutine can be coded in any manner desired to provide the security requirements for the site. The security implemented will be based on the login ID of the user making the connection.

For example, you might want to prevent the connection unless the connection was being made by the root user. The example described here accomplishes that. When the ODBCLOGIN subroutine below is cataloged, any connection will fail unless the root user is making the connection.

If you have globally cataloged the ODBCLOGIN subroutine on the server, when an ODBC connection is made, the *ODBCLOGIN cataloged subroutine will be run first. If the return value is 0, the server process will terminate and return a login error. Otherwise, the connection will be allowed.

The ODBCLOGIN subroutine is optional for any connection. However, if the globally cataloged *ODBCLOGIN subroutine exists on the server, it will be executed first on all connections.

The ODBCLOGIN subroutine has two parameters, as described below.

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

- RTNVAL - If RTNVAL is a nonzero value, the connection is allowed. If it is zero, the connection is disallowed and an error message is returned.
- 2. USERNAME - The following sample subroutine demonstrates restricting ODBC connections to only the root user.

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

U2 server environment setting

All U2 server processes requested from U2 clients will not be able to set up any environment based on the user's profile or any LOGIN script setting defined in the VOC file. All environment variable settings are inherited from the parent unirpc daemon. The library path setting might be the most important environment variable setting for some third-party products to work correctly. They must be set correctly, before the unirpcd daemon is started. The server dynamic library environment setting syntax is dependent on system platform. For an AIX system, it set to the LIBPATH environment variable. All other UNIX systems are use the LD_LIBRARY_PATH environment variable.

For UniData users, the SETENV function can be used to set up the environment variable in the UOLOGIN or ODBCLOGIN subroutine to work with their environment. Some system environment variable settings. such as library path, UDTHOME, and UDTBIN are not allowed to be set in the subroutine level. The UDTHOME environment variable setting is described in Chapter 3.

A UniData server supports the umask setting from UniData 8.1.0 and later using the SETENV() function to change the UNIX umask value. This can be defined in the UOLOGIN subroutine. The optional UOLOGIN subroutine is implemented on UniData 7.1 or later.

Following is a UOLOGIN sample subroutine for UniData. The following sample changes the UNIX umask value to 002 in the current UniObjects session.

```
SUBROUTINE UOLOGIN(ALLOWED, PARAM.IN)
     ALLOWED=1
     RET=SETENV("umask","002")
END
```

UniVerse supports the similar ENV function to set the environment variables. The variables can be defined in the UOLOGIN or ODBCLOGIN security subroutine.

A UniVerse server supports the `UMASK TCL` command to change the UNIX umask value. This can be defined in the UOLOGIN subroutine. The UOLOGIN subroutine is optional and is implemented on UniVerse 10.1 and later.

Following is a UOLOGIN sample subroutine for UniVerse. The following sample changes the UNIX umask value to 011 in the current UniObjects session.

```
SUBROUTINE UOLOGIN(ALLOWED, PARAM.IN)
     ALLOWED=1
     EXECUTE 'UMASK 011'
END
```

Date format change for U2 native client

In the international format, the day of the month appears first. In the United States format, the month appears first. U2 users can use the `DATE.FORMAT` command to set or change the default date format.

Starting with UniVerse 11.3.1 and UniData 8.1.1, the `UOLoginCommand` file has been implemented in the `$UVHOME` or `$UDTHOME` directory upon the startup of a UniObjects session. The `UOLoginCommand` file now accepts one or multiple commands defined in the VOC file.

Chapter 7: U2 client connection debugging log and settings

This chapter describes the U2 client debugging log and settings.

U2 ODBC client logging

The logging setting is defined in the Windows U2ODBC_LOG_PATH environment variable. The `u2odbc_<date_time>.log` file will be created in the `%U2ODBC_LOG_PATH%` folder.

U2 UniObjects for Java client logging

The logging setting is implemented in the UniJava class. The setDebug method implements full logging functionality for the UOJ driver (e.g. UniJava.setDebug(true)). It creates a new `uoj_trace.log` file in the Java program folder.

U2 JDBC client logging

The logging setting is defined in the tracelevel variable in the JDBC URL. The trace level is from 0 to 5.

- 0 = No trace
- 1 = General error message
- 2 = data variable dump
- 3 = message passed to/from the server
- 4 = data shipped to/from the server
- 5 = methods entry and exist

Define the trace file for the log file name in the URL. For example:

```
String url =
"jdbc:rs.u2://localhost/demo?tracelevel=5;tracefile=ud_jdbc.trace;dbmstype=UNIDATA";
String url =
"jdbc:rs.u2://localhost/HS.SALES?tracelevel=5;tracefile=uv_jdbc.trace;dbmstype=
UNIVERSE";
```

UniObjects for .NET client logging

The UniObjects for .NET logging setting is UniObjects for .NET tracing defined in the app.config or web.config file.

In the .NET Framework, there are four trace levels:

- 1 (error)
- 2 (warning)
- 3 (info)
- 4 (verbose)

If the Microsoft Visual Studio project does not contain a configuration file (`app.config`), then from the **Project** menu, select **Add > New Item**. The `myListener.log` file will be created in the `c:\temp` folder. The application configuration file is created and replaced by the following content:

```
<?xml version="1.0"?>
< configuration>
  <system.diagnostics>
    <sources>
      <!-- This section defines the logging configuration for My.Application.Log -->
      <source name="DefaultSource" switchName="DefaultSwitch">
        <listeners>
          <add name="myListener"/>
        </listeners>
      </source>
    </sources>
    <switches>
      <!-- add name="DefaultSwitch" value="Information" /-->
      <!-- Set value property of Arithmetic switch to one of the following: 1(error),
          2(warning), 3(info), 4(verbose) -->
      <add name="UniTraceSwitch" value="4" />
    </switches>
    <sharedListeners>
      <add name="myListener"     type="System.Diagnostics.TextWriterTraceListener"
         initializeData="c:\temp\myListener.log" />
    </sharedListeners>
    <trace autoflush="true" indentsize="4">
      <listeners>
        <add name="myListener" />
      </listeners>
    </trace>
  </system.diagnostics>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0,Profile=Client"/>
  </startup>
< /configuration>
```

U2 Toolkit for .NET client logging

U2 Toolkit for .NET (U2NETDK) provides a standard tracing and logging facility to trace the execution of U2NETDK code and to log the data in a user-specified destination. You can set the configuration for tracing and logging using the application's environment variables. By default, tracing and logging are turned off in U2NETDK.

To access the environment variables:

1. 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:
   - 1 (error)
   - 2 (warning)
   - 3 (info)
   - 4 (verbose)

A U2NETDK application can select one of these four levels and specify a storage destination for the output of tracing and logging. The UCINETTRACE environment variable is used to define the log file folder. The UCINETTRACESWITCH environment variable is used to define the specific trace levels.

Following is an example:

```
UCINETTRACE=c:\temp
    UCINETTRACESWITCH=4
```

In this example, tracing is turned on and it is set to the 4(verbose) level. The log file name is `c:\temp\ucinet_trace_<pid>.txt`.

Tracing and logging in U2 Toolkit for .NET add-ins for Visual Studio

The U2 Toolkit for .NET add-ins for Visual Studio provide a standard tracing and logging facility to trace the execution of U2NETDK code, and log the data for both the installation process and in the Visual Studio environment. You can set the configuration for tracing and logging using the system's environment variables. By default, tracing and logging are turned off in U2NETDK.

Visual Studio trace files

To turn on the trace files option:

1. Navigate to **Start > Control Panel > All Control Panel Items > System**.
2. Click **Advanced system settings** and then select **Advanced > Environment Variables**.
3. In the **System variables** field, click **New**.
4. Enter `U2ADDINS_LOG` as the variable name and enter the location where the log file will be stored as the variable value. For example, `U2ADDINS_LOG=c:\temp`.

Chapter 8: U2 SSL client connection debugging log and settings

This chapter describes the U2 SSL client debugging log and settings.

UniData and UniVerse native SSL client connection

Server

When the U2 UniObjects server debug log is turned on, two log files are generated in the specified log path based on the `serverdebug` file setting. For example, the UniVerse UniObjects server log is `uvcs.log_<pid>`. It will generate another `uvcs.log_ssl_<pid>` on the UniVerse server machine for the SSL connection.

Client

In the Windows environment, the InterCall client application can generate new SSL logging information based on the `clntrace.conf` file that file must reside in the `c:\temp` folder.

Here is the sample setting in the `clntrace.conf` file to create the `itc.log` file in the `c:\temp` folder.

```
ITC 9 c:\temp\itc.log
```

The second column is log level from 0 to 9. The third column is defined for the log file.

UniData and UniVerse U2 ODBC SSL client connection

Server

When the UniVerse ODBC server debug log is turn on, there are two generated log files in the `UVTEMP` folder. For example, the UniVerse ODBC server log is `UVUCIlog_<pid>`. It will generate another `UVUCIlog_ssl_<pid>` file on the UniVerse server machine for the SSL connection. There is only one ODBC server log for UniData.

Client

In the Windows environment, the U2 ODBC client application can generate new SSL logging information based on the `clntrace.conf` file that file must reside in the `c:\temp` folder.

Following is a sample setting in the `clntrace.conf` file to create the `uci.log` file in the `c:\temp` folder.

```
UCI 9 c:\temp\uci.log
```

The second column is log level from 0 to 9. The third column is defined for the log file.

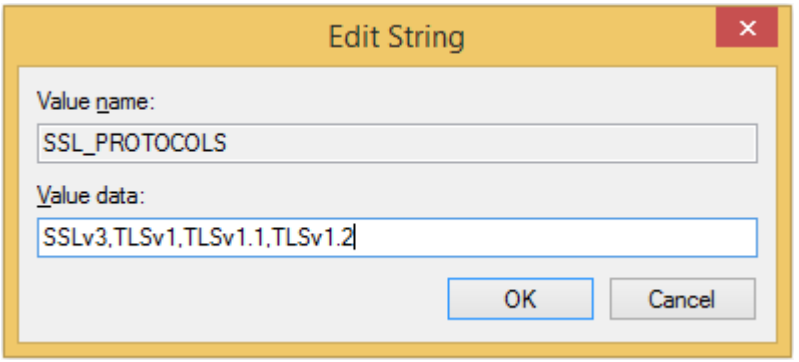
Turning on the U2 ODBC SSL client debugging log

In the U2 ODBC client 5.1.0 version, U2 users can set the U2SSL_DEBUG environment variable to 1. It generates a `u2ssl_debug.log` file in the `c:\temp` folder that can help debug U2 ODBC SSL issues.

U2 ODBC SSL Client compatibility

The U2 ODBC SSL Client can work with an older U2 Server that only supports the SSLv3 or TLSv1 protocol, with modification.

The U2 ODBC version 5.1.0 client only supports TLSv1.1 and TLSv1.2 protocol by default. With older UniVerse servers, such as 11.2.5, the user must add another SSLv3 or TLSv1 protocol option to `HKLM\SOFTWARE\Wow6432Node\Rocket Software\UniClient\SSL_PTOTOCOLS` for 32-bit U2 ODBC and `HKLM\SOFTWARE\Rocket Software\UniClient\SSL_PTOTOCOLS` setting for 64-bit U2 ODBC.



Chapter 9: Error Messaging

The following sections detail the error messaging within the U2 Client applications.

Database error codes

Value	Symbol	Meaning
1	IE_NLS_DEFAULT	
14002	IE_ENOENT	No such file or directory
14005	IE_EIO	I/O error
14009	IE_EBADF	Bad file number
14012	IE_ENOMEM	No memory available
14013	IE_EACCES	Permission denied
14022	IE_EINVAL	Invalid argument
14023	IE_ENFILE	File table overflow
14024	IE_EMFILE	Too many open files
14028	IE_ENOSPC	No space left on device
14551	IE_BW_NETUNREACH	Network is unreachable. When you see this error, you must quit and reopen the session.
22002	IE_BTS	Buffer too small
22004	IE_LRR	Last record already read
22005	IE_NFI	File identifier given does not correspond to an open file
22009	IE_STR	FILEINFO result is a string
30001	IE_RNF	Record not found
30002	IE_LCK	File or record locked by another user
30086	IE_UFI	FILEINFO request has not been implemented
30094	IE_BIL	Bad ID length
30095	IE_FIFS	File ID is incorrect for session
30096	IE_USC	Unsupported server command
30097	IE_SELFAIL	Select failed
30098	IE_LOCKINVALID	Lock number provided is invalid
30099	IE_SEQOPENED	The file was opened for sequential access and you have attempted hashed access
30100	IE_HASHOPENED	The file was opened for hashed access and you have attempted sequential access
30101	IE_SEEKFAILED	Seek command failed
30102	IE_DATUMERROR	Internal datum error
30103	IE_INVALIDATKEY	Invalid key used for GET/SET @variables
30104	IE_INVALIDFILEINFOKEY	FILEINFO key out of range
30105	IE_UNABLETOLOADSUB	Unable to load subroutine on host
30106	IE_BADNUMARGS	Bad number of arguments for subroutine (either too many or too few)
30107	IE_SUBERROR	Subroutine failed to complete successfully

Value	Symbol	Meaning
30108	IE_ITYPEFTC	I-type failed to complete correctly
30109	IE_ITYPEFAILEDTOLOAD	I-type failed to load
30110	IE_ITYPENOTCOMPILED	The I-type has not been compiled
30111	IE_BADTYPE	It is not an I-type or the I-type is corrupt
30112	IE_INVALIDFILENAME	Filename is null
30113	IE_WEOFFAILED	ic_weofseq failed
30114	IE_EXECUTEISACTIVE	An execute is currently active
30115	IE_EXECUTENOTACTIVE	An execute is currently inactive
30116	IE_BADEXECUTESTATUS	Internal execute error, execute has not returned an expected status
30117	IE_INVALIDBLOCKSIZE	Block size is invalid for call
30118	IE_BAD_CONTROL_CODE	Bad trans control code
30119	IE_BAD_EXEC_CODE	Execute did not send return codes back to client correctly
30120	IE_BAD_TTY_DUP	Failure to duplicate ttys
30124	IE_TX_ACTIVE	Transaction is active
30125	IE_CANT_ACCESS_PF	Cannot access part files
30126	IE_FAIL_TO_CANCEL	Failed to cancel an execute
30127	IE_INVALID_INFO_KEY	Bad key for `ic_session_info`
30128	IE_CREATE_FAILED	The creation of a sequential file failed
30129	IE_DUPHANDLE_FAILED	Failed to duplicate a pipe handle
31000	IE_NVR	No VOC record
31001	IE_NPN	No pathname in VOC record
33201	IE_PAR1	Bad parameter 1
33202	IE_PAR2	Bad parameter 2
33203	IE_PAR3	Bad parameter 3
33204	IE_PAR4	Bad parameter 4
33205	IE_PAR5	Bad parameter 5
33206	IE_PAR6	Bad parameter 6
33207	IE_PAR7	Bad parameter 7
33208	IE_PAR8	Bad parameter 8
33209	IE_PAR9	Bad parameter 9
33211	IE_BSLN	Bad select list number
33212	IE_BPID	Bad partfile ID
33213	IE_BAK	Bad secondary index file
39000	IE_BAD_COMMAND	Command not recognized by server
39101	IE_NODATA	The server is not responding
39119	IE_AT_INPUT	A program executed using `ic_execute` is waiting for terminal input
39120	IE_SESSION_NOT_OPEN	The session is not opened when an action is attempted
39121	IE_UVEXPIRED	The license has expired
39122	IE_CSVERSION	Client or server is out of date; client/server functions have been updated

Value	Symbol	Meaning
39123	IE_COMMSVERSION	Client or server is out of date; comms support has been updated
39124	E_BADSIG	You are trying to communicate with the wrong client or server
39125	IE_BADDIR	The directory you are connecting to does not exist or is not a database account
39126	IE_SERVERERR	An error has occurred on the server while trying to transmit an error code to the client
39127	IE_BAD_UVHOME	Unable to get the correct path to the installed database
39128	IE_INVALIDPATH	Bad path found in UV.ACCOUNT file
39129	IE_INVALIDACCOUNT	Account name given is not an account
39130	IE_BAD_UVACCOUNT_FILE	UV.ACCOUNT file could not be found and/or opened
39131	IE_FTA_NEW_ACCOUNT	Failed to attach to the account specified
39133	IE_FTS_TERMINAL	Failed to set up the terminal for server
39134	IE_ULR	User limit has been reached
39135	IE_NO_NLS	NLS is not enabled on the server
39200	IE_SR_CREATE_PIPE_FAIL	Server failed to create the slave pipes
39201	IE_SR_SOCK_CON_FAIL	Server failed to connect to socket
39202	IE_SR_GA_FAIL	Slave failed to give server the Go Ahead message
39203	IE_SR_MEMALLOC_FAIL	Failed to allocate memory for the message from the slave
39204	IE_SR_SLAVE_EXEC_FAIL	The slave failed to start correctly
39205	IE_SR_PASS_TO_SLAVE_FAIL	Failed to pass the message to the slave correctly
39206	IE_SR_EXEC_ALLOC_FAIL	Server failed to allocate the memory for the execute buffer correctly
39207	IE_SR_SLAVE_READ_FAIL	Failed to read from the slave correctly
39208	IE_SR_REPLY_WRITE_FAIL	Failed to write the reply to the slave (ic_inputreply)
39209	IE_SR_SIZE_READ_FAIL	Failed to read the size of the message from the slave
39210	IE_SR_SELECT_FAIL	Server failed to select on input channel. When you see this error, you must quit and reopen the session.
39211	IE_SR_SELECT_TIMEOUT	The select has timed out
80011	IE_BAD_LOGINNAME	Login failed (user name or password invalid)
80019	IE_BAD_PASSWORD	Password has expired
80036	IE_REM_AUTH_FAILED	Unable to set remote authorization
80144	IE_ACCOUNT_EXPIRED	The account has expired
80145	IE_PERM	Account has been locked (AIX only)
80146	IE_EAGAIN	User licenses are all in use
80147	IE_RUN_REMOTE_FAILED	Unable to run as the given user
80148	IE_UPDATE_USER_FAILED	Unable to update user details

Value	Symbol	Meaning
81001	UVRPC_BAD_CONNECTION	Connection is bad. When you see this error, you must quit and reopen the session.
81002	UVRPC_NO_CONNECTION	Connection is down. When you see this error, you must quit and reopen the session.
81003	UVRPC_NOT_INITED	The UniRPC has not be initialized
81004	UVRPC_INVALID_ARG_TYPE	Argument for message is not a valid type. When you see this error, you must quit and reopen the session.
81005	UVRPC_WRONG_VERSION	UniRPC version mismatch
81006	UVRPC_WRONG_VERSION	Packet message out of step. When you see this error, you must quit and reopen the session.
81007	UVRPC_NO_MORE_ CONNECTIONS	No more connections available
81008	UVRPC_BAD_PARAMETER	Bad parameter passed to the UniRPC. When you see this error, you must quit and reopen the session.
81009	UVRPC_FAILED	UniRPC failed. When you see this error, you must quit and reopen the session.
81010	UVRPC_ARG_COUNT	Bad number of arguments for message
81011	UVRPC_UNKNOWN_HOST	Bad host name, or host not responding
81012	UVRPC_FORK_FAILED	UniRPC failed to fork service correctly
81013	UVRPC_CANT_OPEN_SERV_ FILE	Cannot find or open the `unirpcsfervices` file
81014	UVRPC_CANT_FIND_SERVICE	Unable to find the service in the `unirpcservices` file
81015	UVRPC_TIMEOUT	Connection has timed out. When you see this error, you must quit and reopen the session (start the UniRPC daemon or service on the server).
81016	UVRPC_REFUSED	Connection refused, unirpcd not running. When you see this error, you must quit and reopen the session.
81017	UVRPC_SOCKET_INIT_FAILED	Failed to initialize network interface
81018	UVRPC_SERVICE_PAUSED	The UniRPC service has been paused
81019	UVRPC_BAD_TRANSPORT	An invalid transport type was used
81020	UVRPC_BAD_PIPE	Invalid pipe handle
81021	UVRPC_PIPE_WRITE_ERROR	Error writing to pipe
81022	UVRPC_PIPE_READ_ERROR	Error reading from pipe

UniRPC error codes

Remote procedure call (UniRPC) error codes appear if there is a problem in the communications between the client and the server, or if the server encounters one of several error conditions.

Error Code	Meaning
81001	Connection closed, reason unspecified.
81002	On an SQLConnect call, this indicates that the service name specified by the data source was not present on the server, the *unirpcservices* file was not found, or the service name was not found in the *unirpcservices* file.
81003	The UniRPC interface has not been initialized.
81004	Error occurred while trying to store an argument in the transmission packet.
81005	The client and server are running incompatible versions of the UniRPC protocol.
81006	A sequence number failure was detected on the connection.
81007	No more connections can be processed by the RPC interface.
81008	A bad UniRPC parameter was detected.
81009	An internal UniRPC error was detected.
81010	A mismatch in the number of arguments passed between the client and server was detected.
81011	Unknown host. The host name or IP address specified in the data source is not valid for the network.
81012	The UniRPC daemon (*unirpcd*) could not start the *uvserver* executable.
81013	Cannot open unirpcservices file.
81014	Cannot find service.
81015	The connection timed out.
81016	The connection was refused.
81018	The connection was refused.
81019	Invalid transport type.
81020	Invalid pipe handle.
81021	Error writing to pipe.
81022	Error reading from pipe.
81023	A connection specific error has occurred.
81024	This connection does not support multiplexing.
81025	This connection does not support encryption.
81026	This connection does not support compression.
81027	This type of encryption is not supported.
81028	This type of compression is not supported.
930098	The database server could not fork a helper process.

UniVerse SQL error codes

The following table shows the UniVerse SQL error codes and error message text associated with certain SQLSTATE codes. Some texts are shown in abbreviated form.

Code	Message
S0001	Table or view already exists
950458	UniVerse/SQL: Table “*tablename*” already exists in VOC.
950459	UniVerse/SQL: Table “*tablename*” is being created twice.
950528	UniVerse/SQL: View “*viewname*” already exists in VOC.

Code	Message
950529	UniVerse/SQL: View “*viewname*” is being created twice.
S0002	**Table or view not found**
950311	UniVerse/SQL: “*viewname*” is a VIEW, not a BASE TABLE.
950313	UniVerse/SQL: “*tablename*” is a BASE TABLE, not a VIEW.
950390	UniVerse/SQL: Table “*tablename*” does not exist.
950455	UniVerse/SQL: View “*viewname*” does not exist.
950545	UniVerse/SQL: “*name*” is not a base table.
950596	UniVerse/SQL: “*associationname*” is an association; not valid for REFERENCES.
950597	UniVerse/SQL: “*associationname*” is an association, not a VIEW.
950598	UniVerse/SQL: “*associationname*” is an association, not a base table or view.
950599	UniVerse/SQL: “*name*” is not a base table; not valid for REFERENCES.
S0021	**Column already exists**
950416	UniVerse/SQL: Explicit column name “*columnname*” is not unique.
950570	UniVerse/SQL: Duplicate column name “*columnname*”.
S0022	**Column not found**
950418	UniVerse/SQL: Table constraint has an undefined column “*columnname*”.
950425	UniVerse/SQL: Column “*columnname*” not in table.
950428	UniVerse/SQL: Association key column not found.
950522	UniVerse/SQL: Invalid column “*columnname*” specified in constraint.
950523	UniVerse/SQL: Unknown column “*columnname*” specified in table constraint.
21S01	**Number of columns INSERTed doesn’t match number expected**
950059	UniVerse/SQL: Number of columns inserted doesn’t match number required.
21S02	**Number of columns SELECTed doesn’t match number defined in CREATE VIEW**
950415	UniVerse/SQL: More explicit column names than columns selected.
950417	UniVerse/SQL: More columns selected than explicit column names.
22005	**Error in assignment – Data type mismatch (ODBC)**
950043	UniVerse/SQL: *type1* and *type2* types are incompatible in this operation.
950121	UniVerse/SQL: Column “*columnname*” data type does not match insert value.
950122	UniVerse/SQL: Column “*columnname*” data type does not match update value.
950169	UniVerse/SQL: Inconsistent data types in multivalued literal.
950617	UniVerse/SQL: Incorrect data type for literal DEFAULT.

Code	Message
23000	**Integrity constraint violation**
923012	Integrity Constraint Violation, Index not active
923013	Integrity Constraint Violation, Index not UNIQUE
950136	UniVerse/SQL: *constraintname* Constraint Violation *name* on column “*columnname*”.
950568	UniVerse/SQL: Can’t update existing rows with NULL default for NOT NULL column.
950645	UniVerse/SQL: Unable to alter table “*tablename”*, Integrity constraint violation.
40000	**Transaction rolled back**
040065	FATAL: The locks necessary for database operations at the current isolation level (*level*) are not held by this process.
909046	Transaction aborted. Roll back attempted.
950604	Fatal error: ISOLATION level cannot be changed during a transaction.
40001	**An SQL statement with NOWAIT encountered a conflicting lock**
930157	UniVerse/SQL: Locking system failure in CursorOpen
950251	UniVerse/SQL: NOWAIT, Can’t lock record, conflict with another user.
950259	UniVerse/SQL: NOWAIT, Can’t lock file, conflict with another user.
950260	UniVerse/SQL: NOWAIT, Can’t lock record, conflict with user "*user*".
950261	UniVerse/SQL: NOWAIT, Can’t lock file, conflict with user "*user*".
42000	**User lacks SQL or operating system permissions**
001397	User does not have write privileges to current directory.
001422	Insufficient SQL permissions to read *name*.
001423	Insufficient SQL permissions to write *name*.
001424	Insufficient SQL permissions to delete *name*.
020142	Unable to open “*filename*” file.
036010	Permission Denied.
950072	UniVerse/SQL: Permission needed to delete records in table “*tablename*”.
950076	UniVerse/SQL: Permission needed to insert records in table “*tablename*”.
950078	UniVerse/SQL: Permission needed to update records in table “*tablename*”.
950131	UniVerse/SQL: Permission needed to update column “*columnname*” in table “*tablename*”.
950303	UniVerse/SQL: No read/write permission for *username*, cannot create schema.
950304	UniVerse/SQL: No rwx permission for *name*, cannot create schema.
950305	UniVerse/SQL: *username* does not have rwx permission for *name*, cannot create schema.
950306	UniVerse/SQL: *username* does not have rw permission for *name*, cannot create schema.
950338	UniVerse/SQL: *username* is not an SQL user.
950343	UniVerse/SQL: *username* does not have permission to drop schema.
950350	UniVerse/SQL: *username* does not have permission to create schemas.

Code	Message
950352	UniVerse/SQL: You must be DBA to create a schema for another user.
950361	UniVerse/SQL: *username* does not have DBA privilege.
950362	UniVerse/SQL: Command aborted, you may not revoke your own privileges.
950365	UniVerse/SQL: No read/write permission for *username*, cannot create table.
950391	UniVerse/SQL: You do not have sufficient privileges to REVOKE on this file.
950392	UniVerse/SQL: You do not have sufficient privileges to REVOKE SELECT on this file.
950393	UniVerse/SQL: You do not have sufficient privileges to REVOKE INSERT on this file.
950394	UniVerse/SQL: You do not have sufficient privileges to REVOKE DELETE on this file.
950395	UniVerse/SQL: You do not have sufficient privileges to REVOKE UPDATE on this file.
950398	UniVerse/SQL: Command aborted. *username* is not an SQL user.
950405	UniVerse/SQL: You do not have sufficient privileges to GRANT on this file.
950406	UniVerse/SQL: You do not have sufficient privileges to GRANT SELECT on this file.
950407	UniVerse/SQL: You do not have sufficient privileges to GRANT INSERT on this file.
950408	UniVerse/SQL: You do not have sufficient privileges to GRANT DELETE on this file.
950409	UniVerse/SQL: You do not have sufficient privileges to GRANT UPDATE on this file.
950534	UniVerse/SQL: Unable to alter table “*tablename*”.
950538	UniVerse/SQL: You do not have sufficient privileges to REVOKE ALTER on this file.
950539	UniVerse/SQL: You do not have sufficient privileges to REVOKE REFERENCES on this file.
950540	UniVerse/SQL: You do not have sufficient privileges to GRANT ALTER on this file.
950541	UniVerse/SQL: You do not have sufficient privileges to GRANT REFERENCES on this file.
950546	UniVerse/SQL: Permission needed to alter table *tablename*.
950548	UniVerse/SQL: Write permission needed to create or delete index.
950563	UniVerse/SQL: You don’t have enough privileges to DROP “*tablename*”.
950588	UniVerse/SQL: Cannot write to *tablename*.
950590	UniVerse/SQL: Unable to open *tablename*.
950607	UniVerse/SQL: Unable to create REFERENCES on table *tablename*.
950609	UniVerse/SQL: Permission needed to create REFERENCES to table *tablename*.

U2 ODBC error codes

The following table shows the U2 ODBC error codes and error message text.

Code	Message
ODB930000	Successful completion
ODB930001	Disconnect failure
ODB930002	Connection already established
ODB930003	Connection is not established
ODB930004	Invalid parameter length
ODB930005	Unsupported data type
ODB930006	The data source is not in the configuration
ODB930007	Invalid cursor state
ODB930008	Character string truncation
ODB930009	Numeric value out of range
ODB930010	Not all parameter markers have been resolved
ODB930011	Function call is illegal at this point
ODB930012	Data has been truncated
ODB930013	An invalid column number was specified
ODB930014	Unsupported function
ODB930015	Invalid transaction code
ODB930016	An invalid data type has been requested
ODB930017	Disconnect with an active transaction is illegal
ODB930018	An illegal network type was specified.
ODB930019	There is no configuration file, or an error was found in the file
ODB930020	An illegal configuration option was found
ODB930021	Connect failure
ODB930022	An illegal connect parameter was detected
ODB930023	A SequeLink middleware error was detected
ODB930024	An invalid cursor name was specified
ODB930025	A duplicate cursor name was specified
ODB930026	No cursor name was specified
ODB930027	An error occurred at the data source
ODB930028	An illegal SQL data type was supplied
ODB930029	A 0 or empty pointer was specified
ODB930030	An unsupported attribute was specified
ODB930031	An illegal parameter number was specified
ODB930032	An unsupported SQL data type was encountered
ODB930033	An illegal option value was specified
ODB930034	Fractional truncation
ODB930035	An unknown DBMS type has been specified
ODB930036	An illegal option value was specified
ODB930037	Non-numeric data was found where numeric required
ODB930038	Transaction commit failure
ODB930040	Failed opening SequeLink cursor
ODB930041	Illegal date/time value
ODB930051	Row exceeds maximum allowable width.
ODB930053	Information type out of range.

Code	Message
ODB930054	Only a single environment variable may be allocated.
ODB930055	Multi-valued data present. Single result returned.
ODB930056	Memory allocation failure.
ODB930057	Improper MAPERROR option.
ODB930058	Improper SQLTYPE option.
ODB930059	Error in Remote Procedure Call interface.
ODB930060	Connections to non-uniVerse data sources is not allowed.
ODB930061	Nested transactional operations to non-uniVerse databases are not permitted.
ODB930062	Communications link failed during operation.
ODB930063	Remote user id is required.
ODB930064	Remote password is required.
ODB930065	UniVerse user limit on server has been reached, try again later.
ODB930066	The uniVerse on the server has expired.
ODB930067	SQLGetData on column bound as multi-valued is illegal.
ODB930068	SQLBindMvCol/SQLBindMvParam illegal on 1NF connection.
ODB930069	Function type out of range.
ODB930070	Multi-valued parameter binding for CALL not allowed.
ODB930071	UCI connections to non UniVerse databases is not allowed.
ODB930072	Driver does not support this function.
ODB930073	Invalid string or buffer length.
ODB930074	OUTPUT parameter markers are only valid with procedure calls.
ODB930075	Parameter marker text size exceeds allocated space.
ODB930076	Operation invalid at this time.
ODB930077	NLS is not enabled.
ODB930078	NLS item not found.
ODB930079	NLS locale category entry is invalid.
ODB930080	NLS locale support is not enabled.
ODB930081	NLS locale name is not loaded into shared memory.
ODB930082	NLS locale category number is invalid.
ODB930083	NLS multivalued locale array is incomplete.
ODB930084	NLS internal error.
ODB930086	Unable to get UCI configuration file from Registry.
ODB930087	Unable to access specified UCI configuration file.
ODB930088	Illegal value for fetch direction.
ODB930089	Illegal Account name specified.