REAL Server 2009

User's Guide

REAL Server 2009 Release 1 User's Guide

Documentation by David Brandt.

Printed in U.S.A.

Mailing Address	REAL Software, Inc. PO Box 162181 Austin, TX 78716
Web Site	http://www.realsoftware.com
ftp Site	ftp://ftp.realsoftware.com
Support	REALbasic Feedback at the REAL Software web site.
Bugs/ Feature Requests	Submit via REALbasic Feedback at the REAL Software web site.
Database Plug-ins	The REALbasic CD; the most recent versions are at www.realsoftware.com.
Sales	sales@realsoftware.com
Phone	512-328-REAL (7325)
Fax	512-328-7372

Version 2009 Release 1, May, 2009

Contents

CHAPTER 1

REAL Server Installation

This chapter describes the components of the REAL Server and explains how to install and launch the server. This chapter covers:

- REAL Server installation and registration
- The contents of the REAL Server folder
- Launching the server
- The "databases" and "backups" folders
- The REALbasic REALSQLServerDatabase class

Installation

On Macintosh and Windows, the REAL Server uses a standard installer. The installation process is largely automated. On Linux, the REAL Server is shipped as a .tgz file and is installed from the command line.

REAL Server includes a script that uninstalls any previous versions of REAL SQL Server that may be installed. If you have REAL SQL Server on your machine, run this script first.

Macintosh

Double-click the Macintosh .dmg file to mount a volume that contains the REAL Server installer.

To install REAL Server on Macintosh, do this:

1 **Double-click the Install REAL Server package.**

The installer's welcome screen appears.

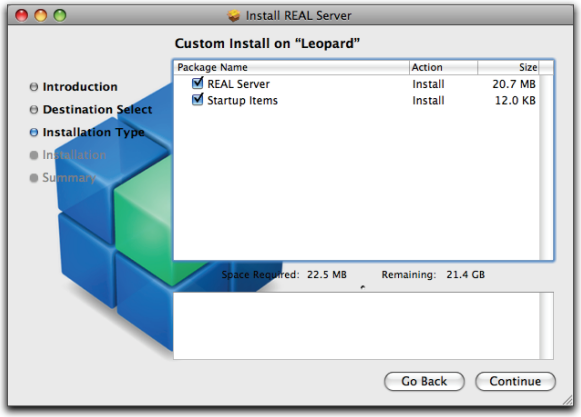
Figure 1. The REAL Server installer welcome screen.

The installer will create the folder "REAL Server" in your Applications folder.

2 **Click Continue.**

The Install dialog appears.

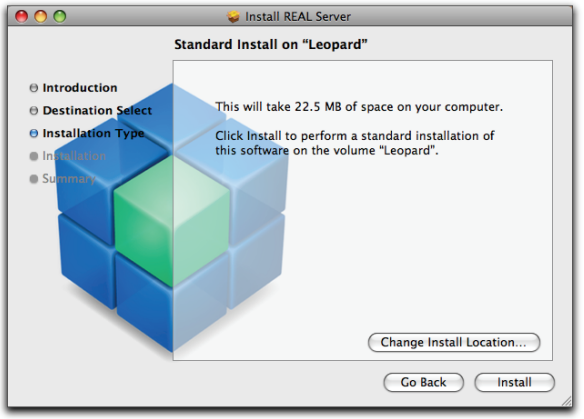
Figure 2. The Install dialog box.



3 **Click the Continue button.**

The Easy Install screen appears. There are no other installation options.

Figure 3. The Standard Install screen.



4 **Click Install to complete the installation process.**

Mac OS X authentication dialog box appears.

5 **Enter your username and password and click OK.**

When the installation process completes, your Applications folder will contain the REAL Server folder.

Proceed to the section "Launching The Server" on page 14.

Windows

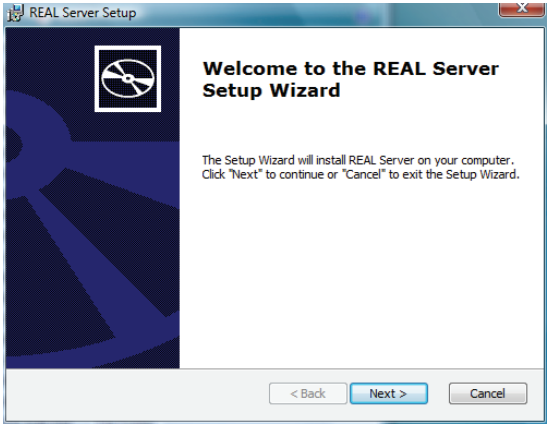
On Windows, a standard Installer package is also used.

To install on Windows, do this:

1 **Double-click the REALSQLServerSetup icon.**

The Setup Wizard screen appears.

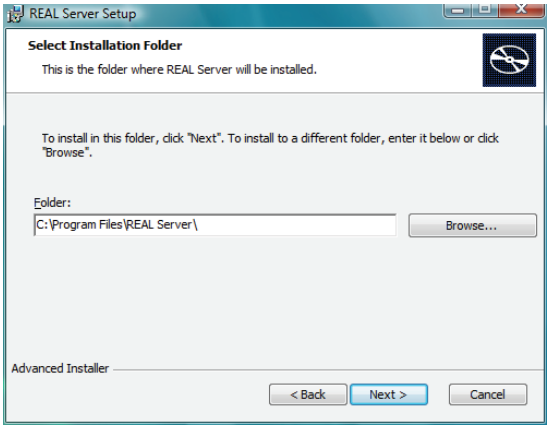
Figure 4. The Setup Wizard screen.



2 **Click Next.**

The Select Installation folder dialog appears.

Figure 5. The Select Destination Location dialog.

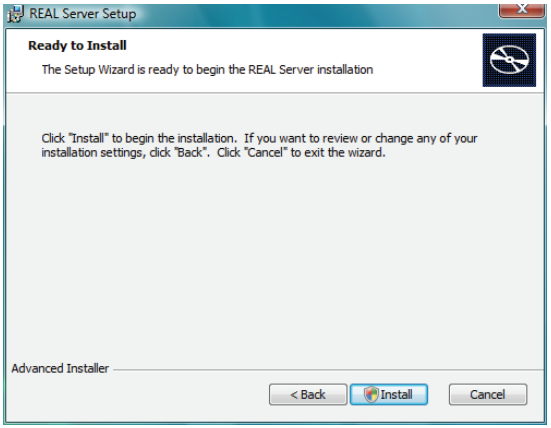


It suggests a folder in your Program Files directory.

3 **If you want to change the default name and/or location, click browse. Otherwise, click Next to accept the default folder.**

The Ready to Install screen appears. It asks you whether you want to proceed with the installation.

Figure 6. The Ready to Install screen.



4 **If you want to proceed with the installation, click Install.**

Vista will ask for you to authenticate the process. When you allow the process, installation begins. In a few moments, the Installation Completed screen appears. It gives you a chance to launch the REAL Server immediately.

Proceed to the section "Launching The Server" on page 14.

Linux

The REAL Server back end supports any x86-based Linux distribution. The REAL Server Admin application is a REALbasic application and has the same requirements as any REALbasic application:

- x86-based Linux distribution with GTK 2.0+ or higher,
- glibc-2.3 or higher,
- CUPS,
- libstdc++.so.5,

The following distributions are officially supported:

- Ubuntu 6.06 or higher,
- SUSE Linux Enterprise Desktop 10,
- Red Hat Linux Enterprise 5

The REAL Server is shipped as a .tgz file. Expand the archive with the command `tar xzvf realsqlserver.tgz`. CD to the realsqlserver directory and issue the command `sudo ./install.sh`.

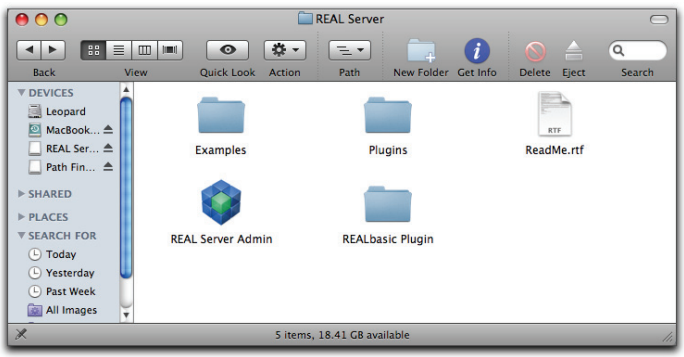
Proceed to the section "Launching The Server" on page 14.

To uninstall, issue the command `sudo ./uninstall.sh`.

The REAL Server Folder

The REAL Server folder contains the following items:

Figure 7. The REAL Server folder.



- **REAL Server Admin application:** Use the Admin application to launch, configure, and administer the server via a graphical user interface. For information about the Admin application, see the chapter "The REAL Server Admin Application" on page 21.
- **Documentation folder:** The REAL Server User's Guide and release notes.

- **Examples folder:** A folder of REALbasic database applications that use REAL Server. The examples require REALbasic 2005 Professional edition (or above) and the REALSQLServerPlugin.
- **Plugins folder:** Contains optional SQL Server plugins.
- **REALbasic Plugin folder**: Contains the REALSQLServer plugin. You need to install this plugin in your REALbasic folder in order to use the REAL Server commands in your REALbasic code.
- **ReadMe:** Contains information on what's new in REAL Server 2009.

The Databases, Backups, and Restore Folders

The server engine uses folders that contain the databases that it serves and any backups and restores that you create. They are not shipped with the product; rather, an empty "databases" folder is created the first time you connect to the server engine. A "backups" folder is created when you run your first backup.

You can review its default settings in the Settings panel in the Admin application. You can change the location of the se folders if you wish. It will then remember these locations in subsequent launches.

The Admin Application

The Admin application is the utility that enables you to administer the server via a graphical user interface. The Admin application can be moved to another location on your hard disk. In fact, it does not have to be running on the same computer or the same platform as the server that it administers. Also, you can administer two or more servers simultaneously.

Admin applications for all platforms are available. For information on the Admin application, see the chapter "The REAL Server Admin Application" on page 21.

You can also administer a server within a REALbasic application via the REAL Server command language. These commands are documented in the chapter "REAL Server Commands" on page 47.

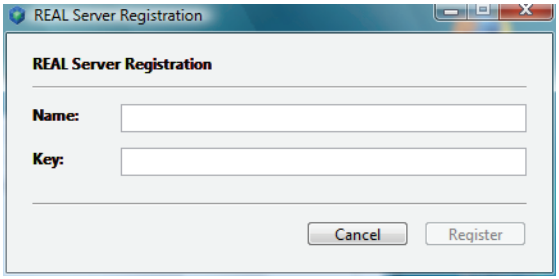
Launching The Server

The REAL Server suite is designed so that the server engine is launched by the REAL Server Admin application.

To launch the server, double-click the REAL Server Admin application.

If you obtain a license key, you can enter it into the dialog box shown in Figure 8. You get this dialog if you choose Register Server.

Figure 8. The Registration dialog box.



Enter the exact name the license was issued to and your registration key.

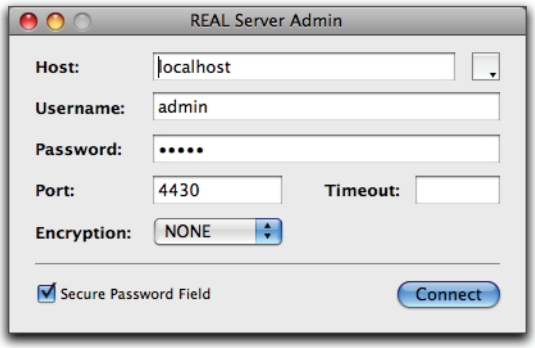
1 **Choose Server ▸ Start Local Server to launch the server as a background process.**

Your OS may ask you to authenticate the process.

2 **If the server does not launch, choose Edit ▸ Preferences (Admin ▸ Preferences on Macintosh). Click Default and then click Save.**

Then the server launches successfully, you can connect to it via the Admin dialog.

Figure 9. The Server Admin dialog box.

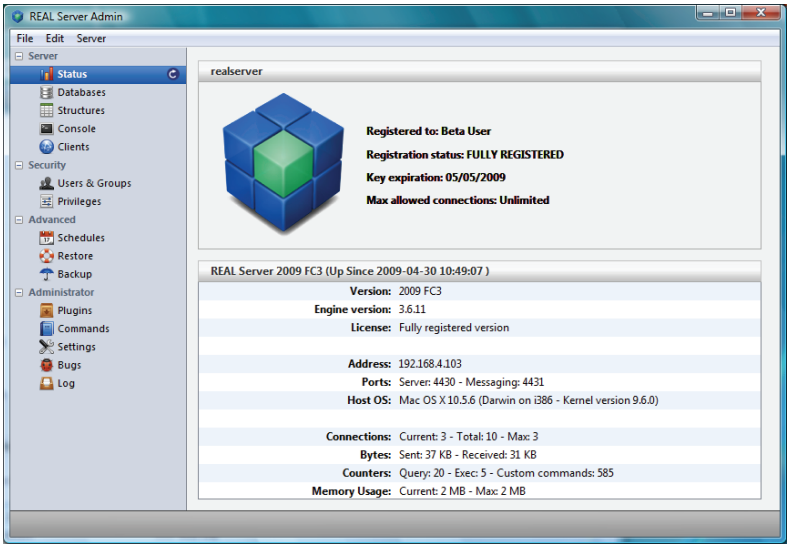


The default values for Host and Port will connect to the server engine on the local computer.

A newly installed server will have a default user named "admin" with the password set to "admin." The "admin" username is provided as the default in the connection dialog. If you are connecting to a newly installed server, then you can enter the default password and click the Connect button.

When the connection is successful, the Admin Status panel appears.

Figure 10. The Admin Status panel.



In subsequent launches, the Admin application will detect whether or not the server engine is running. If it is not running, you can choose Server ▶ Start Local Server to start the server.

Working with the Databases Folder

When the server launches, it looks for the databases folder at the location that is given in the Server Settings dialog (see Figure 13 on page 24). If the server does not find it, it creates a new empty folder.

The databases folder contains the databases to be served by the server. All the databases in the databases folder are served automatically.

There are several ways to get a database into the databases folder. One way is to create a database with the server, either in code or using the Admin application. To create a database via code, use the server's CREATE DATABASE command. For information on how to create a database via the Admin application, see the section "The Databases Panel" on page 25. The commands used to create databases and tables are covered in the chapter "REAL Server Commands" on page 47.

A second way to get a database into the databases folder is to use the Server ▶ Upload Database menu command in the Admin application or the UPLOAD DATABASE command in the language. The menu command is available when the Databases screen is displayed This command uploads an existing database to the databases folder. There is an example database in the Examples folder that illustrates uploading and downloading via the language.

Another way to get a database into the server's databases folder is to drag an existing database file into it. It does not matter whether the server is running when you install a database. The new database will be available immediately for use through the server. If you are running the Admin application, the new database will appear on the Databases screen very shortly after it is added.

REAL Server Admin Application

The Admin application provides a GUI for administering the server and backing up databases. It can be used to administer a server on either the local machine or a remote machine.

When you connect to the server, the Admin application's Status screen appears. For more details on how to use the REAL Server Admin application, see the chapter "The REAL Server Admin Application" on page 21.

A server must be running in order to administer it. If you fail to connect from the Admin application, check the status of the server in for that server to verify that the server engine is running. Use the File ▶ Connect command to reconnect to any server, local or remote.

Connecting from REALbasic

You can also administer a server from a REALbasic application. In order to connect to a REAL Server from REALbasic, you must place the REALSQLServerPlugin in REALbasic's Plugins folder. Plugins are available for all platforms that REALbasic supports. This plug-in adds the REALSQLServerDatabase class to the REALbasic framework.

The REALSQLServerDatabase Class

When you install the REALSQLServer plug-in in REALbasic's plugins folder, the REALSQLServerDatabase class is added to REALbasic's framework. With it, you can develop custom REAL Server applications. The REALSQLServerPlugin requires REALbasic 2005 Professional, Release 1 or later.

The REALSQLServerDatabase class works like any Database subclass, with some additional properties and methods. They are described in the following tables.

Properties

Name	Type	Description
Autocommit	Boolean	Determines whether the database connection commits changes automatically, or whether changes open an implicit transaction that you must explicitly close by calling Commit or Rollback. The default is False: Changes start implicit transactions that you must close explicitly. If you set AutoCommit to True, then that REALSQLServerDatabase connection will no longer open implicit transactions for you, and instead, will automatically commit each database change immediately. It is recommended that you leave AutoCommit set to False unless you understand the implications of setting it to True.
DatabaseName	String	The name of the database to connect to. You do not have to provide a DatabaseName to connect to the server.
Encryption	Integer	Determines the strength of encryption to use when connecting to a server. Encryption can only be set to the following values: 0, 128, 192, and 256. The higher the number, the stronger the encryption. A value of 0 means no encryption. The default value is 0. If you want to use encryption, it must be set before connecting to a server. If Encryption is set after a connection has been made, then further communication with the connected server will fail. You can, however, close the connection and reconnect at the new encryption setting. You do not need to override the default value in order to connect to a server.
Host	String	The IP address or hostname of the host computer. The value of 127.0.0.1 connects to the local machine. You must pass a value for Host in order to connect to a server.
Password	String	The password for the provided username. The default password for user “admin” on a newly installed server is “admin”. The “admin” user cannot be deleted and the username cannot be changed. The admin’s password can be changed. You must provide the correct password for the passed username in order to connect to a server.
Port	Integer	The port used to connect to the server engine. The default is 4430. If you are connecting to a remote computer, be sure that the firewalls are configured to allow communication on the selected port. You must pass a Port and Host in order to connect to a server.
Timeout	Integer	Indicates how long to wait for server responses before giving up and timing out. The default setting is 15 seconds.

Name	Type	Description
Username	String	The username to connect as. The default user for a newly installed server is “admin”. You must pass a Username in order to connect to a server.

Methods

Name	Parameters	Return Type	Description
Connect		Boolean	Connects to the server or tests the connection if you have already connected. The Host, Port, Username, and Password properties are required for a successful connection. If you have the DatabaseName property set when you call Connect, then the REALSQLServerDatabase will attempt to use the database named by DatabaseName. Connect returns True if the attempt to connect was successful and False otherwise. If the connection has been dropped, Connect attempts to reconnect. Connect returns True only if the attempt to reconnect was successful.
EndChunk		Boolean	Call EndChunk at any time while executing the DOWNLOAD DATABASE command to determine if the last chunk has been received. EndChunk returns True if the most recent ReceiveChunk was the last chunk and False if it was not. If EndChunk returns True, then the last ReceiveChunk issued won't have any data. See the example for the DOWNLOAD DATABASE command in the Command Reference.
IsConnected		Boolean	IsConnected returns True if you are connected to the server and False otherwise. It differs from Connect in that it does not attempt to establish a connection. It simply reports on the status of the connection. Use IsConnected to check on the status of the connection without trying to create one. Connect will try to re-establish a dropped connection.
ReceiveChunk		Integer	Call ReceiveChunk repeatedly after executing the DOWNLOAD DATABASE command to download chunks of the database. ReceiveChunk returns the number of bytes in the chunk. See the Upload/Download example in the Examples folder.
SendChunk	chunk as String		Call SendChunk repeatedly after executing the UPLOAD DATABASE command to upload chunks of the database file to the server. A good way to use SendChunk is to read a database file in 100K chunks and send each chunk to the server using SendChunk. See the Upload/Download example in the Examples folder.

Name	Parameters	Return Type	Description
SendEndChunk			Call SendEndChunk after sending the last chunk of a database file to the server using SendChunk. SendEndChunk informs the server that the last chunk has been sent. See the Upload/Download example in the Examples folder.

To connect to a server, use code such as the following:

```
Dim db as REALSQLServerDatabase
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.Encryption=128 //encrypted connection to the server
db.UserName="admin"
db.Password="admin"

If db.connect then
  //issue SQL commands here
Else
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
        +db.ErrorMessage
End if
```

Please see the Command Reference for information on the commands that administer the server and schedule and execute backups.

CHAPTER 2

The REAL Server Admin Application

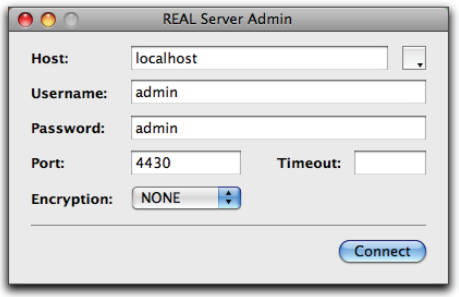
You can use the REAL Server application to administer the server engine and the backup utility. It can be used to administer both locally running servers and servers running on remote machines.

You can also administer a server via the special REAL Server commands in the language. In fact, if you have the REAL Server Admin application connected to the server when commands are executed via the language, the Admin interface will update to show your changes.

Connecting to the Server

When you first launch the REAL Server Admin application, you will be presented with a Connect dialog box. This dialog enables you to connect to either the server on the same computer or one that is visible on your network. It is shown in Figure 11.

Figure 11. The Connect dialog box.



The default values for Host and Port will connect to the server engine on the local computer.

A newly installed server will have a default user named "admin" with the password set to "admin." The "admin" username is provided as the default in the connection dialog. If you are connecting to a newly installed server, then you can enter the default password and click the Connect button. If you are connecting to a server that has already been configured, be sure to provide the correct admin password. If you are connecting to a remote server, be sure that the firewalls are configured so that you can communicate on the selected port.

The admin user has full privileges for administering the server. That is, no operations are prohibited. Although it is possible to perform some administrative tasks as a user other than an admin user, it is strongly recommended that you connect to the server either as the admin user or as someone in the admin group. For information on adding users and groups, see the sections "The Users & Groups Panel" on page 37. Please refer to the section "The Privileges Panel" on page 41 for information on granting privileges to groups.

Because the admin user has full privileges, it is strongly recommended that you change admin's password as soon as it is convenient. For information on changing a user's password, see the section "Modifying Users" on page 40.

Once the server connection has been established, the Admin Status panel will be displayed (Figure 12 on page 23).

Each Admin Status panel identifies the server that it is connected to by the IP address of the machine and the Port used to communicate with it.

The REAL Server Admin Window

The Admin window's panels are organized into the following topics: Server, Security, Advanced, and Administrator. The Admin application refreshes the current panel periodically to keep the information in that panel up-to-date. For example, if you add

a database by dragging it into the databases folder, it will appear on the Databases panel automatically.

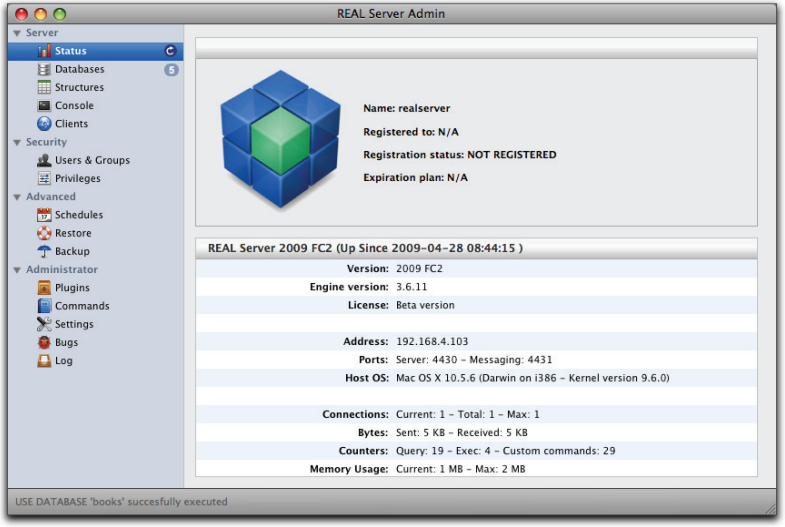
Also, the operations that can be done via the REAL Server's GUI can also be performed progammatically, via the special REAL Server commands that are described in the Command Reference. When you execute any such command, the REAL Server application's GUI is updated accordingly.

You can connect to more than one server at the same time. Simply choose File ▶ Connect from an open Admin application window or from the main menu bar (Macintosh only). A new Connect dialog will appear, enabling you to establish a connection to another server. You will then get a new Admin window for that server.

The Status Panel

The Status panel shows you the state of the server, either Running or Stopped. If it is running, it reports the number of connections and the length of time since it was started or restarted. It also gives you the IP address of the server machine, the port that the Admin application uses to communicate with the server engine, the version of the server, the IP address of the server machine, the port that Admin is using to connect to the server, and the OS on which the server is running.

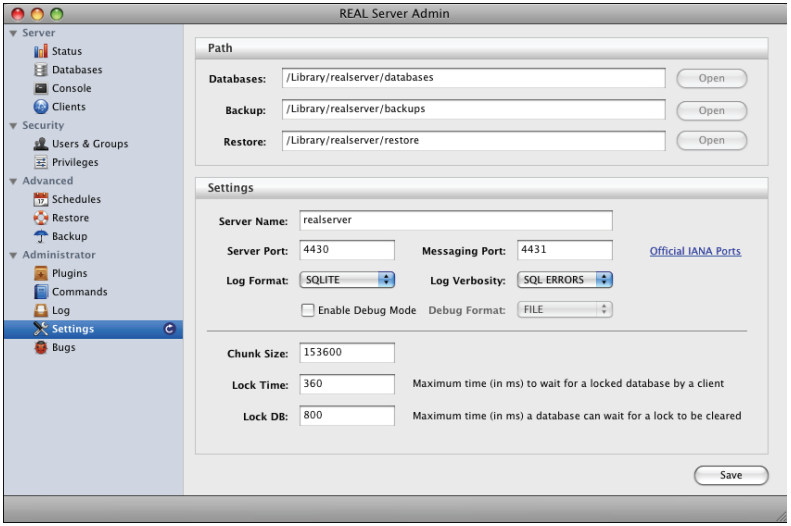
Figure 12. The Status panel.



Changing Server Settings

To make changes to the server's settings, click the Settings item in the Administrator group in the browser. The dialog box shown in Figure 13 appears.

Figure 13. The Server Settings panel.



Path Settings

The Path area enables you to set the paths to the Databases, Backups, and Restore folders. You can modify these paths entering a different path.

Server Settings

If desired, enter a new name for the server. You don't lose functionality if the server has no name.

The Port property is the port that the Admin application uses to communicate with the server engine. Under most circumstances, the default value of 4430 will work fine. If you need to change the port for some reason, enter a new port number. If you change the port, the new port setting won't take affect until the server is restarted. You may need to close the current REAL Server window and open a new one for the server after you change the server's port and restart it.

Also, you may need to update the computers' firewall settings to permit communication on the selected port. If you encounter problems connecting, check your firewall settings.

The Locks Timeout is the amount of time (in seconds) that the server will maintain a lock on a database without any activity on the part of the user who locked it. This refers to a multiuser issue. When a user is writing to a database, other users are locked out of that database until the user with write privileges releases the database to other users. The Locks Timeout setting provides a way to prevent a lock on a database from lasting too long. See the entries for the RESET ERROR and SET PREFERENCES commands in the Command reference.

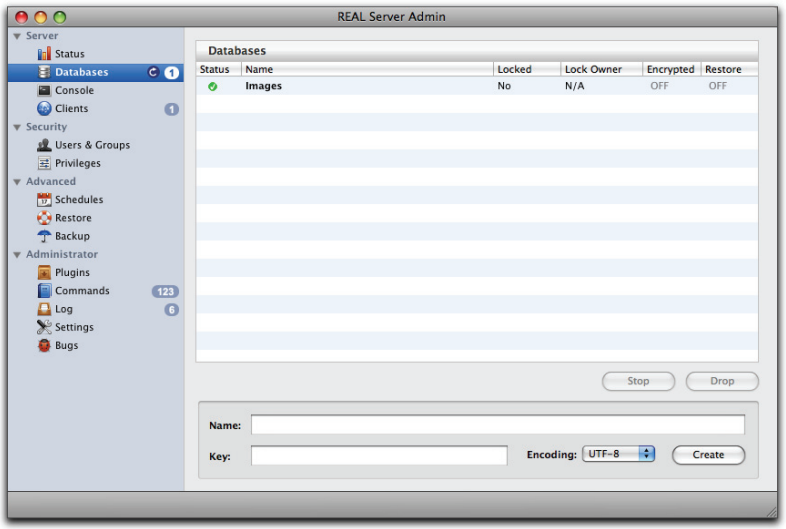
The Verbose Logging option controls the type of information that is written to the log. There are four settings: None, SQL Errors, and SQL Commands., and Debug.

Debug is selected by checking the Enable Debug checkbox. You can write out the log in either of two formats, SQLite and Text. The two formats are shown in the section "The Log Panel" on page 45. When you change formats, you must restart the server for the change to take effect.

The Databases Panel

You use the Databases panel to manage the databases that the server serves. You can create new databases, get database statistics, nd start, stop, and drop databases. Figure 14 shows the Databases panel for a server that is serving one database.

Figure 14. The Databases panel.



Creating a Database

Use the area below the list of databases to create a new database..

Figure 15. The Create Database area.

Enter the name of the new database. If you want to encrypt the database, enter the key for the database. Leave this field blank if you want to leave the database unencrypted. I f you need to change the If you decide to encrypt the database at a later point, you can do so by issuing the SET KEY FOR DATABASE command. If you want to change the encoding for the database, choose another encoding from the pop-up menu. Click Create to add the database.

The new database will then be listed in the Databases list. It will have no tables until you add them.

To delete an existing database, select the database and click the Drop button.

Uploading a Database

If you have an existing database that is not currently being served by REAL Server, you can install it in the databases folder by uploading it. You can upload from either the Admin application or with the UPLOAD DATABASE command in the language.

The Upload Database command in the Admin application assumes that there is not a database with the same name already installed in the databases folder. If it encounters a database with the same name, it will fail and issue an error. The UPLOAD DATABASE command in the language, however, includes an option to overwrite an existing database with the same name.

If you want to replace a database from the Admin application, first delete the database to be replaced and then upload the replacement.

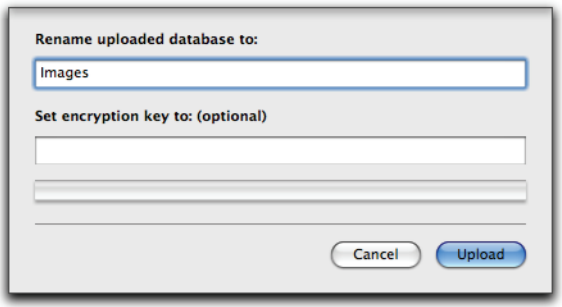
To upload a database from the Admin application, do this:

1. **Display the Databases panel in the Admin application.**
2. **Choose Server ▸ Upload Database.**

   A standard open-file dialog box appears.
3. **Select the desired database and click Open.**

   An Rename dialog appears, enabling you to rename the database and enter the encryption key for this database if it is encrypted.

Figure 16. The Encryption Key dialog box.



You must enter the key in order to use an encrypted database; if it is not encrypted, leave the entry area blank and click Upload.

When you click Upload, the REAL Server copies the database into the databases folder and immediately begins serving it.

You can also upload a database to the server by copying it into the “databases” folder from the desktop.

If the database that you uploaded is encrypted and the server does not know its encryption key, then it cannot serve it. For that reason, it appears in gray in the Admin

application. You need to tell it the key. You can do this in either the Admin application or via the language.

- In the Admin application, select the database in the list and choose Server ▶ Set Database Key. A dialog will appear in which you can enter the database's encryption key.
- In the language, call the SET KEY FOR DATABASE command, passing the database's key in the command.

Downloading a Database

Downloading a database reverses the process. It takes a database that is installed in the "databases" folder and copies it to another location on your hard disk. When you download a database, it does not remove it from the "databases" folder. It continues to be served.

You can also download a database using the language, with the DOWNLOAD DATABASE command.

To download a database from the Admin application, do this:

1. **Switch to the Databases panel in the Admin application.**
2. **Click on the database you wish to download in the Databases list.**
3. **Choose Server ▶ Download Database.**

   A standard save-file dialog box appears.
4. **Navigate to the folder in which you want to save the database and click Choose.**

   REAL Server displays a progress dialog.
5. **Click Start to begin the download.**

   The REAL Server copies the database to the location you chose.

Backups

The REAL Server includes an integrated backup utility. You can create backup schedules and do immediate backups either programmatically or via the Admin application. In the Admin application, you can do immediate backups and/or create one backup schedule per database.

In the Command Reference, the commands for scheduling backups are listed in the Schedules topic. Scheduled backups via the language offer somewhat more versatility than is currently supported in the Admin application. With the language, you create backup schedules independently of databases and attach a backup schedule to a database. You can attach more than one backup schedule to a database and you can assign a backup schedule to more than one database.

If you want to manage backups with the language instead of the Admin application, please see the example custom application "Backup" in the Examples folder in your REAL Server folder.

The language allows you to name your backup schedules, while the Admin application names each backup for a database "*DatabaseName* backup". If you create a backup schedule for a database via the language and do not use this naming convention, it will not appear in the Databases panel as a backup. The screen will erroneously imply that there is no backup attached to the database. What it means is that there is no backup schedule *of this name* attached to the database.

Since the Admin application only shows backup schedules that observe this naming convention, it cannot show multiple schedules attached to the same database.

To create or modify a backup schedule for a database, click the Schedules item in the Advanced group. Backups are scheduled by day and time. You can have the backup run on more than one day per week but it will run at the same time on each day. Backups are performed only if the "Schedule Enabled" checkbox is selected.

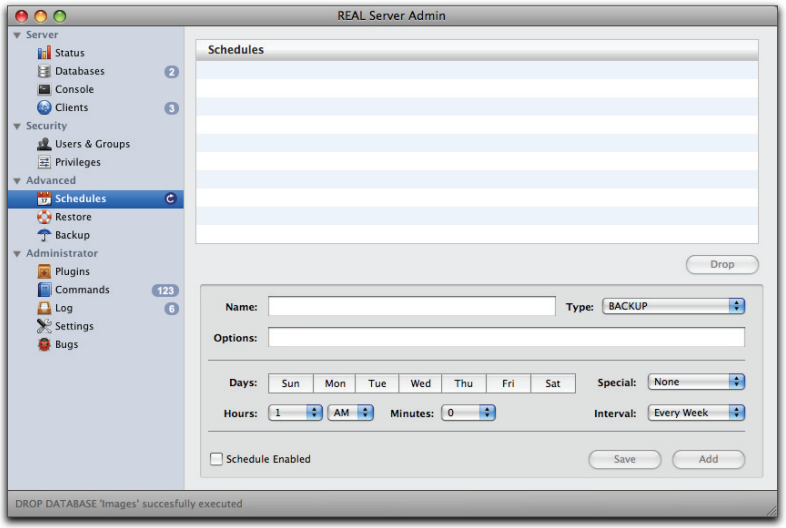
To create a backup schedule, do this:

1 **Select the database in the Databases list for which you want to create the backup schedule.**

2 **Select the Schedules panel in the Admin app.**

The Schedule panel appears.

Figure 17. The Schedules dialog box.



The Backups dialog box appears, with the selected database in the Title bar (Windows and Linux only), as shown in Figure 17.

3 **Click the "Schedule backups for" checkbox.**

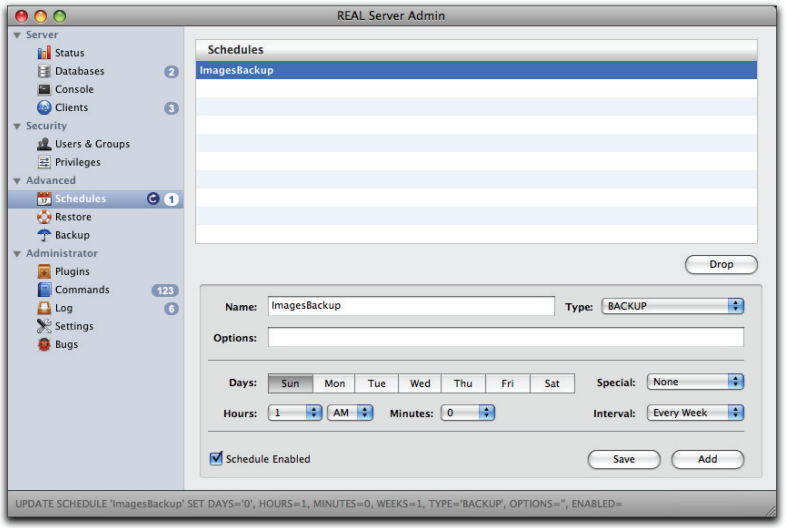
The controls for setting the days and time for the backup become active.

4 **Name the backup and click one or more of the Day buttons to choose the day or days on which the backup will run.**

The button for a selected day will remain depressed when you click on it. To deselect the day, click it again.

5 **Choose the time of day for the backup from the hour, minute, and AM/PM drop-down menus.**

Figure 18. A backup schedule.



6 **To create another schedule, repeat steps 4 and 5 and click Add.**

To modify a backup, schedule, click on the schedule, make any changes, and click Save.

You can do an immediate backup from the Admin application. Click the Back up item in the Advanced topic. Choose the database to be backed up and click Backup Now. With the language, you can do an immediate backup by issuing the BACKUP NOW command.

Your backups will be located in the "backups" folder, at the location shown in the Settings panel. Each backup will be contained in a folder that is identified by a date-time stamp.

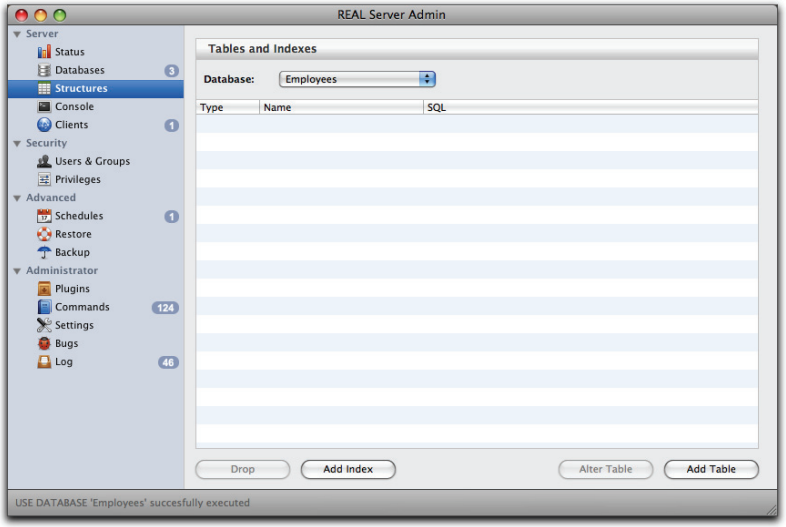
The Structures Panel

You use the Structures panel to create tables, fields and indexes for your databases.

Creating a Table

To create a new table for a database, switch to the Structures panel.

Figure 19. The Structures panel.



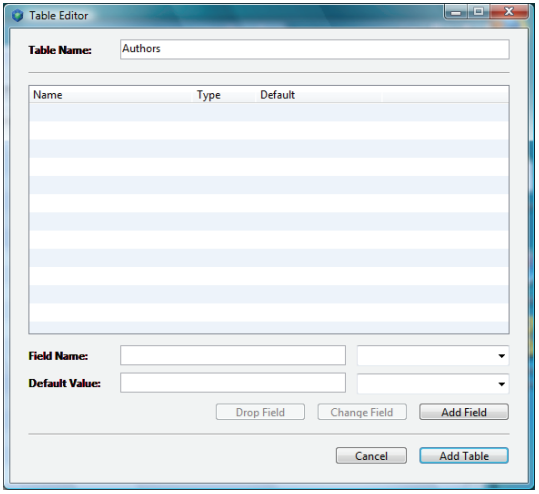
The pop-up menu at the top of the panel lists all the databases on the server. If a database contains tables or indexes, then they are listed the body of the panel.

To add a table to a database, do this:

1 **Choose the database from the pop-up menu and click Add Table.**

The Table Editor appears.

Figure 20. The Table Editor.



2 **Enter the name of the table in the Table Name field.**

You must now add at least one field to the table.

3 Enter a field name in the Field Name field and choose its data type from the pop-up menu to its right.

Your choices are as follows:

Data Type	Description
Binary	Stores code, images, and hexadecimal data of any size. Binary is the same as Blob.
Blob	Stores a binary object. The REAL Server database supports blobs of up to any size. A blob can be stored in a column of any declared data type. Use Blobs to store images.
Boolean	Stores the values of TRUE or FALSE. The number "0" and the string "False" are treated as False and "1" and "True" are treated as True. The behavior of any other values is undefined if retrieved using DatabaseField.BooleanValue. DatabaseField.StringValue, on the other hand, should be able to retrieve the original data if it can't be identified as a boolean by the REAL Server. If the REAL Server database can identify the value as a boolean, however, then "False" will always return "False" and "True" will always return "True", regardless of how those values are stored in the database. This is a side effect of the way in which database engines interact with the database API in the REALbasic framework. This change to DatabaseField.BooleanValue should only be an issue if you are trying to store non-boolean data in a boolean column.
Currency	This is a 64-bit fixed-point number format that holds 15 digits to the left of the decimal point and 4 digits to the right.
Date	Stores year, month, and day values of a date in the format *YYYY-MM-DD*. The year value is four digits; the month and day values are two digits.
Double	Stores double-precision floating-point numbers.
Float	Stores floating-point numeric values with a precision that you specify, i.e., FLOAT (5).
Integer	A numeric data type with no fractional part. The REAL Server supports 8-byte (64-bit) integers. Initially, Integers were limited to 32 bits. The DatabaseField.IntegerValue function can now return values that require 64-bit Integers to store. The field type returned by Database.FieldSchema is now 19 instead of 3 (32-bit integer). If you are expecting a 3, be sure to recode.
SmallInt	A numeric data type with no fractional part. The maximum number of digits is implementation-specific, but is usually less than or equal to INTEGER. The REAL Server supports 4-byte SmallInts. Database.FieldSchema returns 3 for SmallInts.

Data Type	Description
Text	Stores alphabetic data in which the number of characters can vary from record to record. The REAL Server uses UTF-8 text encoding. If the text is not already in UTF-8, it is converted. If you want to preserve a different encoding, use a Blob column instead. Same as VarChar.
Time	Stores hour, minute, and second values of a time in the format HH:MM:SS. The hours and minutes are two digits. The seconds values is also two digits, may include a optional fractional part, e.g., 09:55:25.248. The default length of the fractional part is zero.
TimeStamp	Stores both date and time information in the format YYYY-MM-DD HH:MM:SS. The lengths of the components of a TimeStamp are the same as for Time and Date, except that the default length of the fractional part of the time component is six digits rather than zero. If a TimeStamp values has no fractional component, then its length is 19 digits. If it has a fractional component, its length is 20 digits, plus the length of the fractional component.
VarChar	Stores alphabetic data in which the number of characters can vary from record to record. The REAL Server uses UTF-8 text encoding. If the text is not already in UTF-8, it is converted. If you want to preserve a different encoding, use a Blob column instead.

4 **(Optional) Specify a default value for the field.**

5 **(Optional) Specify any field attributes from the Attributes pop-up menu, located to the right of the Default Value field.**

Your choices are:

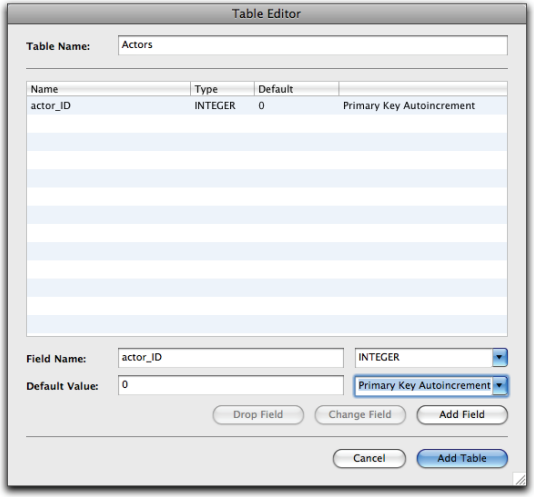
- Primary Key
- Primary Key auto increment
- Unique (single)
- Unique (multi)
- Not NULL

If you want to clear the selection, select the text and press Delete.

6 **When all the attributes of the field are specified, click the Add Field button to add the field to the field list.**

The new field is added to the list.

Figure 21. The Table Editor after adding a field.



7 Continue adding fields as needed by clearing the entries and replacing them with new values. When you are finished, click Add Table to save the table to the database.

Creating an Index

After adding your fields, you may want to index certain fields. Indexes improve the performance of the database. You should index fields on which you plan to do the majority of your searches and sorts. Information retrieval and relational operations among tables are faster when indexed fields are used. Primary key fields are indexed automatically.

Fields that are not good candidates for indexing are those that only take on a few values (i.e., gender or race), fields that are rarely searched on, and any fields in small tables where search and sort times are unlikely to be long.

You should not index too many fields because each index adds to the size of the database and, as records are added and deleted, it takes time for the system to update each index.

Indexes are added via the Structures panel. From the Structures panel, select a database that has at least one table. Select the table in the list and click the Add Index button.

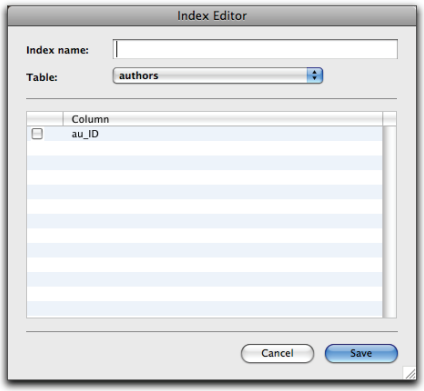
To create the indexes for a table, do this:

1 Choose a database from the pop-up menu and the name of a table in the Tables list.

2 Click the Add Index button below the list.

The Index Editor appears.

Figure 22. The Index Editor.



The Index Editor shows the fields in the selected table. To create an index, click all the fields that will comprise the index. If you choose more then one field, it will create a composite index rather than more than one distinct index.

3 **Enter the name of the index in the field at the top of the dialog.**

Usually you will use the names of the field or fields that comprise the index.

4 **Click the checkbox for the field or fields that you want to include in the index.**

If you click two or more fields, a composite index will be created, not separate indexes for each field.

For example, you may want to create a composite index on LastName and FirstName fields so that the database can quickly work with different people with the same last name.

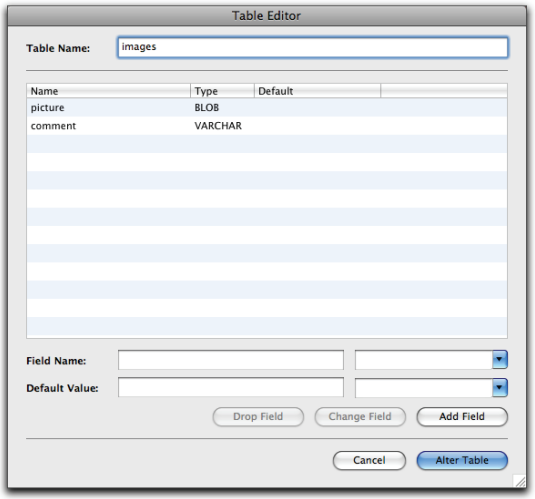
5 **Click Save to save the index.**

If desired, repeat this process to create additional indexes. To create indexes in another database, select another database in the Databases list and repeat this process.

Editing a Table Schema

You can edit the table schema for any existing table. To do so, display the Structures panel, select the database you want to modify. Click the Alter Table button. The Table Editor for that database appears.

Figure 23. The Table Editor for the Images table.



When you are in the process of creating fields, you can alter the schema by deleting new fields and adding other fields.

- To delete a field, highlight the field in the list and click the Delete button.
- To add a field, deselect any existing field and then repeat the process of adding fields described in the previous section.

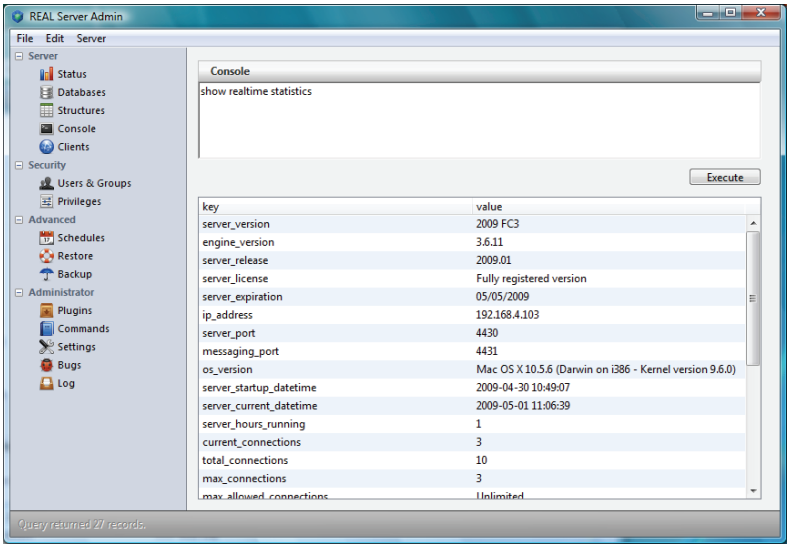
When you click Alter Table to display the Table Editor, you cannot drop or change fields, only add fields

When you are finished, click Alter Table to save your changes and put away the dialog box.

The Console Panel

The Console panel enables you to execute REAL Server commands interactively. Any command in the Command reference can be executed.

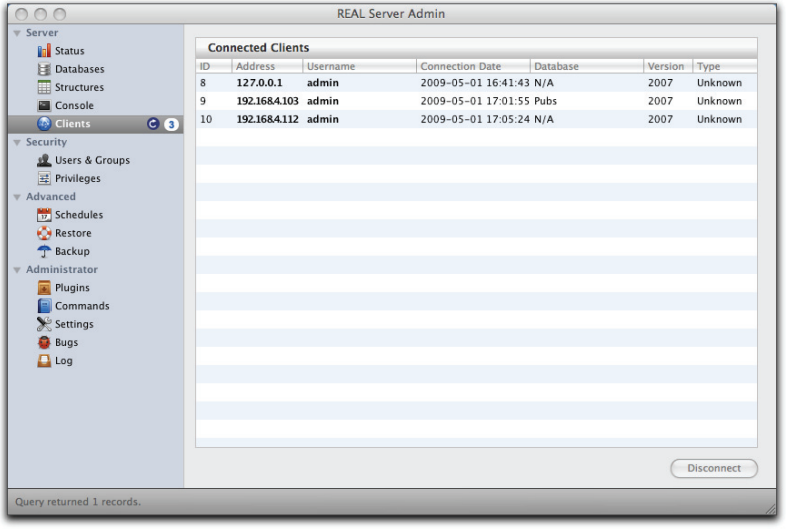
Figure 24. The Console panel with an executed command.



Clients Panel

The Clients panel displays a list of connected clients that are connected to the server that the instance of the Admin window is connected to. It lists their client ID, IP address, the username, date of connection, database, version, and type. You have the ability to disconnect a client from this panel.

Figure 25. The Clients panel showing three connections.



The Users & Groups Panel

Use the Users and Groups panel to create and edit the list of users who have access to the server and the databases. This includes both server administrators and end users.

Figure 26. The Users and Groups panel.

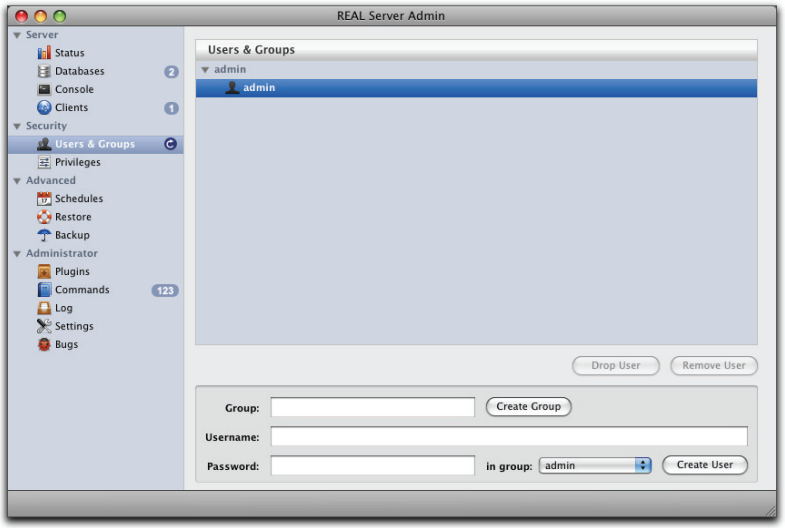


Figure 26 shows the default configuration. There is one group with one user. You can add other users to the admin group and create new groups that will be assigned fewer privileges.

You use the Users & Groups panel to first create and edit the list of groups. You can then assign users to groups.

To create the privileges system, do this:

- Create groups that will have different access privileges,
- Create a username and password for each user and place users into groups,
- In the Privileges panel, to grant privileges to each group.

The users in a group automatically acquire the privileges of their group. A user can be in more than one group. Such a user gets the privileges of those groups.

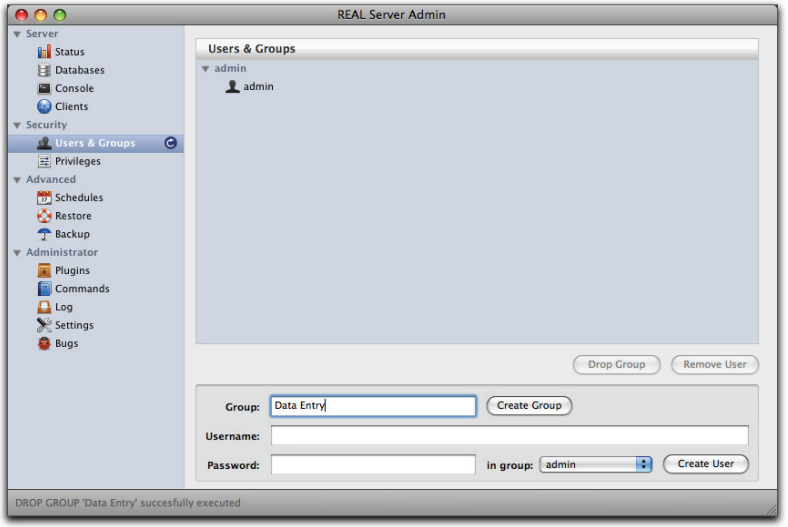
The admin user is the default user that ships with the server. It is a member of the admin group. The admin group has been granted all possible privileges and this cannot be changed. Privileges are granted to groups, not users. A user automatically gets all the privileges that have been granted to the groups that user is in. For example, you might have a Data Entry group that has permissions to write data only to the tables that concern them.

In addition to administrative functions, privileges control whether a user can select, add, modify, or delete individual records. Privileges can be granted at the level of individual tables, for the database as a whole, and for the server itself.

Creating Groups

To create a new group, enter a new group name click the Create Group button located under the Users & Groups list. The new group is added to the list of groups. In this case, the group "DataEntry" is about to be created.

Figure 27. The Users & Groups panel.



To delete an existing group, select the group and click the Drop Group button. You cannot delete the Admin group.

To rename a group, double-click on the group name and click Modify. The Rename Group dialog appears.

Figure 28. The Rename Group dialog box.

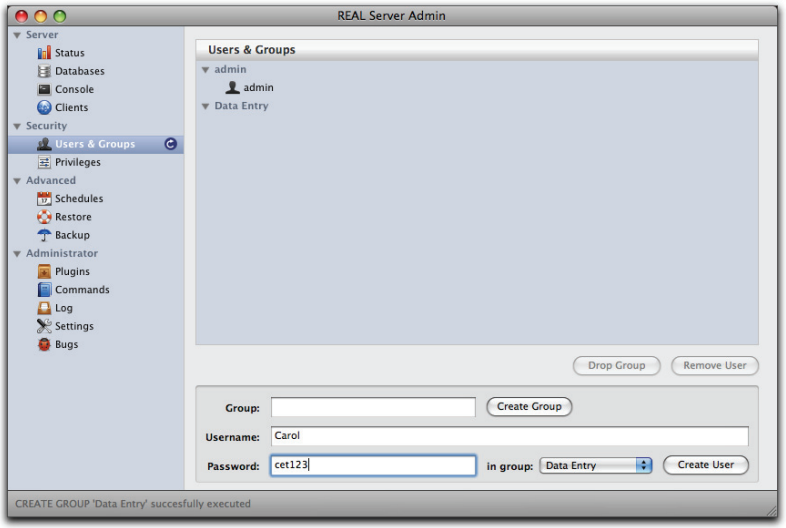
Enter the new name for the group and click Save.

You cannot rename or delete the admin group.

Creating Users

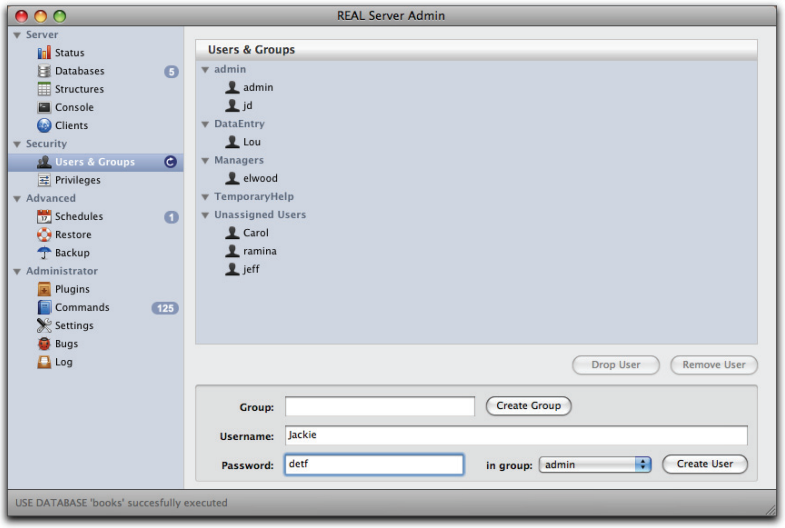
To create a new user, enter the user name and password into the username and password fields. Choose a group that the user will be in from the In Group pop-up menu and click Create User. (You can assign a user to more than one group later.)

Figure 29. Adding a user to the Data Entry group.



Using the CREATE USER command, it is possible to create a user without assigning him or her to a group. In this case, the user will appear in the "Unassigned Users" category. You can assign them to groups as explained in the following section.

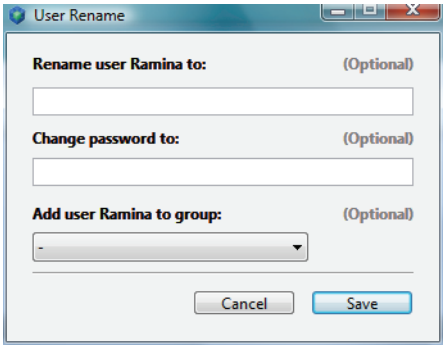
Figure 30. Unassisgned Users in the Users & Groups panel



Modifying Users

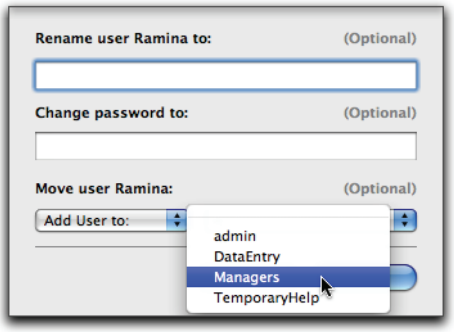
To rename a user, change the password, or add a user to another group, double-click the user in the Users list. The dialog shown in Figure 31 will appear.

Figure 31. The Modify User dialog box.



If you want to rename the user or change the password, enter the new information in the appropriate fields. You cannot change the "admin" user's name but you can change the admin's password. To add the user to another group, use the Move User area. The pop-up menu has items for moving the user and adding the user. Select the desired option and choose the group from the second pop-up menu. select the new group from the Add User pop-up menu.

Figure 32. Adding a user to a group.



If you want to assign the user to another group, repeat this process. When you are finished making changes, click Save.

Deleting Users

You can delete a user from the list of users by and clicking Drop User. The Drop User command drops the user from all groups and the server. If you want to remove a user from only one group, select the user in that group and click Remove User.

The Privileges Panel

The REAL Server uses a system of access privileges that you use to allow or deny actions on server resources. Privileges are assigned to server groups. Users in a group have all of the privileges of that group. A user can belong to more than one group. If a user is in more than one group, then the user has all the privileges of every group in which the user is a member.

By default, REAL Server has one group named admin. Every user in the admin group is considered an admin user, and as such, has every possible privilege. The admin user is a permanent member of the admin group and cannot be renamed or removed. Neither the admin user nor the admin group may be removed from a server. Also, privileges cannot be removed from the admin group. However, users other than admin can be removed from the admin group, stripping them of admin privileges.

In order to give a user a particular set of privileges, a group must exist with those privileges. By adding the user to the group, the user will automatically gain all the privileges of the group. Deleting a group removes the group's privileges from the members of that group.

There are three types of privileges:

- **Server privileges:** the privilege is granted for the entire server
- **Database privileges:** the privilege is granted only for a specified database
- **Table privileges:** the privilege is granted only for a specified table in a database

You set the scope of a privilege when you grant a privilege to a group. The bottom portion of the Privileges panel contains pop-up menus for Group, Database, and Table. If a privilege is to be granted for the entire server, you specify only the Group and set Database to "*". To grant a privilege at the database level, then specify the database from this pop-up and leave Table blank. To grant a privilege for only a table in a database, specify the database and the table from those two pop-up menus.

Privileges

Privileges include:

- **Admin:** has all the privileges of the Admin group
- **Privileges:** can edit group privileges
- **Preferences:** can see and edit server settings
- **Databases:** can create and delete databases
- **Upload:** can upload databases to the server
- **Download:** can download all databases from the server
- **Create:** can create tables in the database
- **Drop:** can drop tables in the database

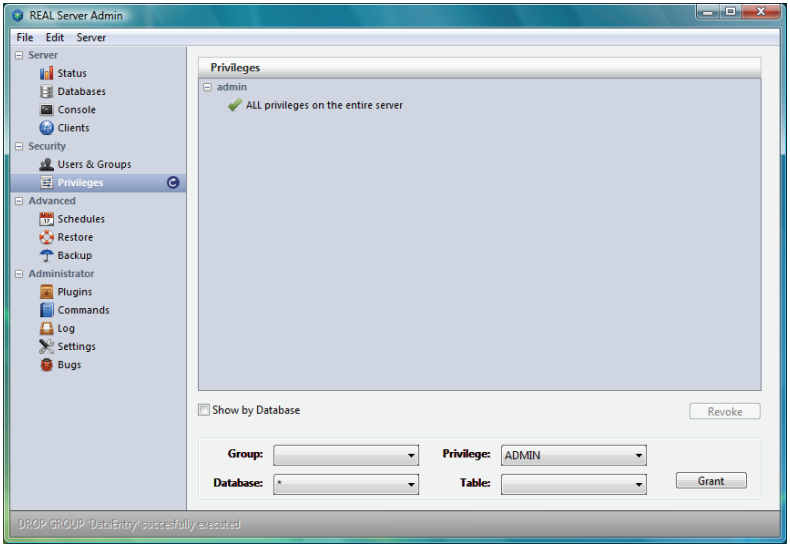
- **Select:** can query any table in the database
- **Insert:** can insert records into any table in the database
- **Update:** can update records in any table in the database
- **Delete:** can delete records in any table in the database
- **Upload:** can upload the database to the server, replacing it
- **Download:** can download the database
- **Pragma:** can execute PRAGMA commands to change database settings
- **Plugin:** can execute the plug-in commands, DISABLE PLUGIN and ENABLE PLUGIN
- **Restore:** can restore a backup
- **Backup:** can execute backups
- **Select:** can query the table
- **Insert:** can insert records into the table
- **Update:** can update records in the table
- **Delete:** can delete records in the table

In the Command Reference, the privileges that are required to execute each command are noted.

Using the Privileges Panel

You use the Privileges panel to assign privileges to groups. To give a user privileges, create a group with the desired privileges and then add the user to that group. You may also change the privileges for a group after users have been added to the group, of course.

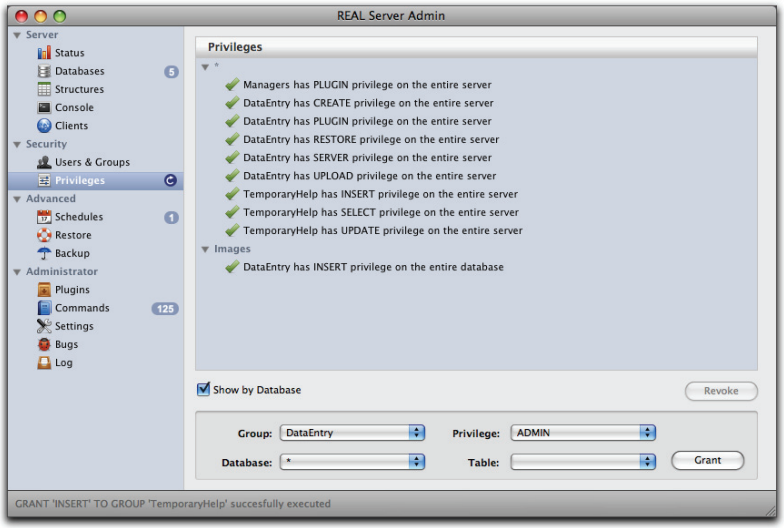
Figure 33. The Privileges panel with only the Admin group.



The Privileges panel is divided into two sections:

- The top section lists all the groups and the privileges assigned to them. The bottom section is used to assign privileges to groups. Privileges are nested in groups. The default view organizes the information by group. Click the Show by Database check-box to organize the information by database. In this display, the privileges are divided into a section for the entire server and separate sections for each database on the server.

Figure 34. The Database view for the Privileges panel.



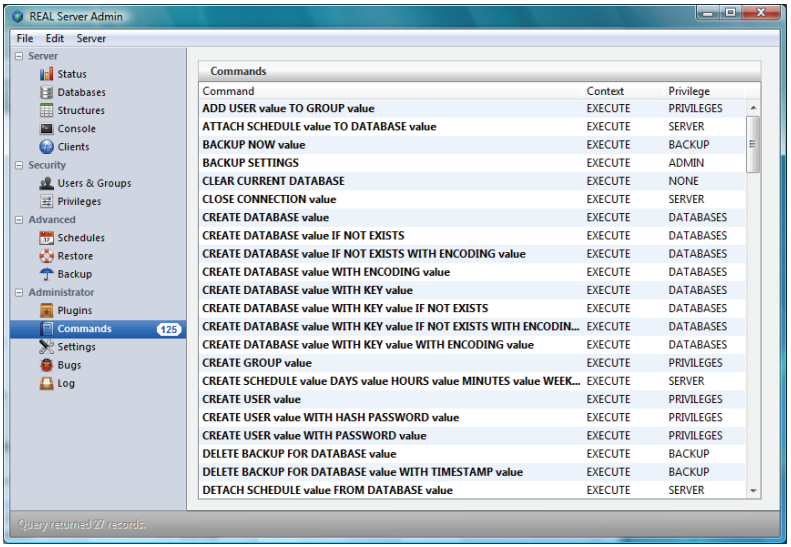
- The bottom section is used to assign privileges. You use the Database and Table pop--up menus to specify the scope of the privilege to be granted. To specify Server privileges, specify only the Group to be granted the privilege and the privilege to be granted. To specify Database privileges, specify the Group, the database, and the privilege. To specify Table privileges, specify the Group, the database, the Table, and the privilege.

  There are no limitations or constraints: you can check as many or as few privileges as you like.

The Commands panel

The Commands panel lists the syntaxes of all the special REAL Server commands, with the privileges that are required to execute the commands. The Context refers to whether you call the command with SQLExecute or SQLSelect.

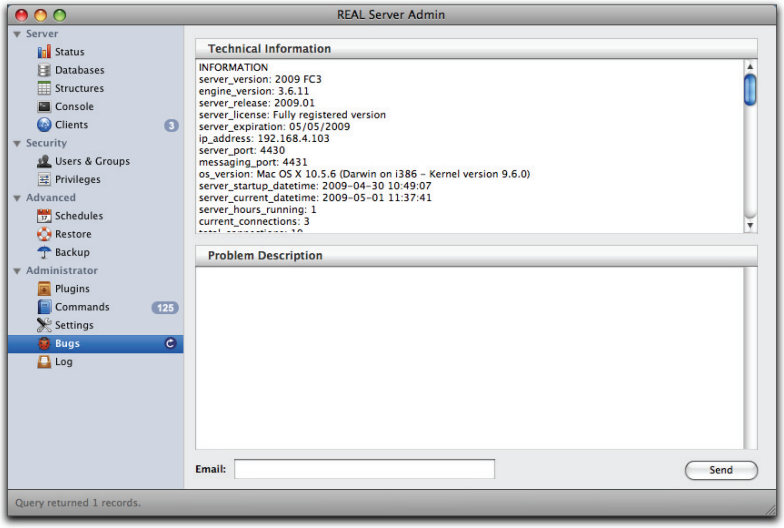
Figure 35. The Commands panel.



The Bugs Panel

Use the bugs panel to submit any bugs that you find in REAL Server. The top section contains information on the state of the server and its operating environment. The bottom section is for you to describe the bug in enough detail so the REAL Server engineers can replicate it, Enter your email in the email field and click Send to report your bug.

Figure 36. The Bugs panel.

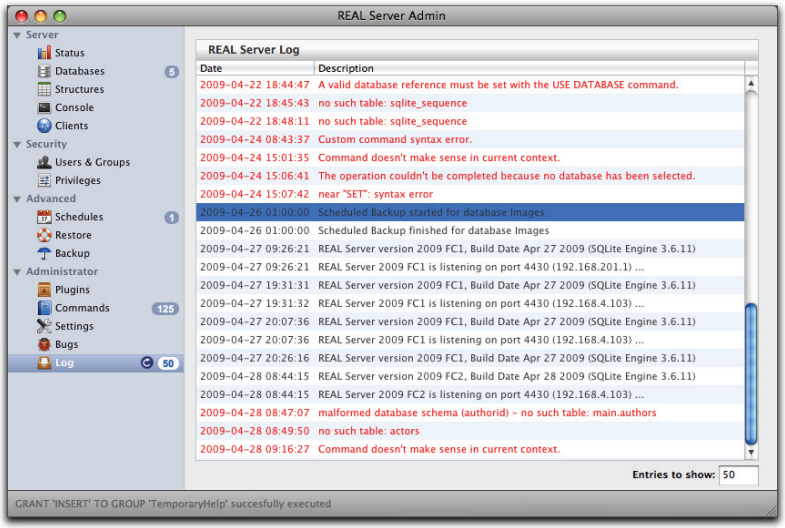


The Log Panel

You use the Log panel to review the server log. The Log panel displays the last *XX* entries in the log. Enter the number of entries that you want to see into "Entries to show." field in the Log panel. If you want to see entries other than the last *XX* rows, use the SHOW LOG command. It allows you to specify a date range instead of the last rows.

If you want to change the format, change value of the Log Format in the Settings panel (see "Changing Server Settings" on page 23). You can also set the verbosity (SQL errors, SQL commands, or none), enable or disable debug mode, and choose the debug format.

Figure 37. The Log panel.



The log includes columns for the datetime stamp, description, operation, address of the client machine from which the command originated, the username, and the database, if relevant. Only the timestamp and the description are shown within the Admin application. You can access all the fields by submitting the SHOW LOG command.

CHAPTER 3

REAL Server Commands

In addition to the SQL statements supported by the REAL SQL Database, the REAL Server also understands a number of server-specific commands. This chapter lists those commands and describes what each command does.

In general, commands that begin with SHOW are intended to query the server for information. The server returns the information as a REALbasic RecordSet. Therefore, each SHOW command needs to be issued with a SQLSelect statement. For example, here is how you would get a list of all of the databases on the server:

```
Dim rs as RecordSet
rs = db.SQLSelect("SHOW DATABASES")
```

(This assumes that "db" is a REALSQLServerDatabase object that already exists.)

Commands that do not begin with SHOW are intended to make a change on the server. Those commands must be issued with the SQLExecute statement. For example, here is how you would create a new database on the server:

```
db.SQLExecute("CREATE DATABASE newdatabase")
```

Most commands require special privileges to execute. If you are logged into a server with an account that has insufficient privileges to execute a particular command, then the server will return an error.

This chapter describes all of the special server commands the server understands, together with a description and example. The privileges needed to execute a command appear after each command's syntax. Specific privileges, such as DATABASES or PRIVILEGES are shown in uppercase. Other privileges are explained, such as "admin user" (must be an admin user), or "read privilege on database" (must have some read privileges on the given database).

In the Syntax line for each command, the keywords that make up the command are shown in uppercase and the values passed as parameters are shown in italics. If a value contains any spaces, it is enclosed in single quote marks. The entire SQL statement is passed to SQLExecute or SQLSelect as a string.

What's New in REAL Server 2009 r1

The new version of the server has a completely rewritten architecture that is able to handle a huge number concurrent connections. Thanks to the new architecture the server is now much more scalable and flexible. For each supported operating system, we now use a state-of-the-art event API (kqueue on Mac OS X, epool on Linux, and I/O Completion Ports on Windows). The new REAL Server is one of the few DBMS server available that is able of solving the C10K problem (more on http://www.kegel.com/c10k.html).

The most important new features of the new internal architecture are:

Event based

Asynchronous sockets

One thread per CPU/Core (setting adjustable by the user)

READ is COMMITTED

Multiversion Concurrency Control (MVCC)

Multiversion Concurrency Control (MVCC) is a standard technique for avoiding conflicts between read and write operations of the same object. MVCC guarantees that each transaction sees a consistent view of the database by reading non-current data for objects modified by concurrent transactions. MVCC is a fairly common technique in database implementation and all the modern DBMS adopt an implementation of the MVCC algorithm.

The MVCC algorithm was developed by Jim Starkey while he was working at DEC. He is the database architect who developed InterBase, the first relational database to support multi-versioning and he is currently working for MySQL AB.

Implementation of a complete MVCC algorithm inside a database adds a heavy overhead, so a lot of DBMSs use a relaxed version in order to achieve better performance. REAL Server 2009 uses an optimistic variant of MVCC that provides a consistent execution time.

Other features:

FTS3 support

RTree support

Plugins support

Restore and backup support

SQLite 3.6.11 with register based Virtual Machine

Listing the Contents of a RecordSet

The example code for commands that return a REALbasic RecordSet call the method "DisplayRecordSet" to display the records in the RecordSet in a ListBox. DisplayRecordSet is a method of the window that contains the example code and is as follows:

```
Sub DisplayRecordSet(rs as RecordSet)
 Dim i as Integer
 Dim V as String
 Listbox1.DeleteAllRows
 Listbox1.Columncount = rs.FieldCount
 Listbox1.AddRow rs.IdxField(1).Name
 Listbox1.CellBold(Listbox1.LastIndex, 0) = True
 For i = 2 to rs.FieldCount
  Listbox1.Cell(Listbox1.lastIndex, i - 1) = rs.IdxField(i).Name
  Listbox1.CellBold(Listbox1.LastIndex, i - 1) = True
 Next
 While Not rs.eof
  v = rs.IdxField(1).StringValue   //display first column
  Listbox1.AddRow v
  For i = 2 to rs.fieldcount
   Listbox1.Cell(Listbox1.LastIndex, i - 1) = rs.IdxField(i).StringValue
  Next
  rs.MoveNext
 Wend
```

Error messages that are returned by the server are listed in the Appendix, "Error Codes," on page 175.

ADD USER TO GROUP

Syntax **ADD USER *user* TO GROUP *group***

Privileges PRIVILEGES

Description ADD USER adds an existing user to an existing group. You will get an error if the user does not already exist.

Example This example adds user "Jeff" to the group "middleManagement."

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("ADD USER Jeff TO GROUP middleManagement")
 If db.error then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
         +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also CREATE GROUP, CREATE USER, RENAME GROUP, RENAME USER, SHOW USERS, SHOW USERS IN GROUP.

ATTACH SCHEDULE

Syntax **ATTACH SCHEDULE *name* TO DATABASE *database***

Privileges ADMIN user

Description ATTACH SCHEDULE attaches a previously created schedule to a database. Use this command after creating the schedule with CREATE SCHEDULE. You can attach a schedule to as many databases as you like. You can also attach more than one schedule to a database.

Example This example attaches the schedule "myBackup" to the database "Images."

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("ATTACH SCHEDULE myBackup TO DATABASE Images")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned."_
         +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also CREATE SCHEDULE, DETACH SCHEDULE, DROP SCHEDULE, SHOW DATABASES FOR SCHEDULE, SHOW SCHEDULE, SHOW SCHEDULES, SHOW SCHEDULES FOR DATABASE, UPDATE SCHEDULE.

BACKUP NOW

Syntax **BACKUP NOW *databaseName***

Privileges ADMIN user

Description BACKUP NOW backs up the passed database immediately. BACKUP NOW always attempts to retain old backups.

Example The following example backs up the "Images" database.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("BACKUP NOW Images")
 If db.error then
  MsgBox "An error was returned from the server " + db.ErrorMessage + _
            " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

BACKUP SETTINGS

Syntax **BACKUP SETTINGS**

Privileges ADMIN

Description BACKUP SETTINGS makes a backup of the Settings fie. The restore feature will be available in 2009 version 2.

Example

The following example backs up the server's settings.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("BACKUP SETTINGS")
 If db.error then
  MsgBox "An error was returned from the server " + db.ErrorMessage + _
            " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

CLEAR CURRENT DATABASE

Syntax

CEAR CURRENT DATABASE

Privileges

None

Description

CLEAR CURRENT DATABASE clears the database that was previously set by a call to the USE DATABASE command.

Example

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("CLEAR CURRENT DATABASE")
 if db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+ _
       " was returned with message "+db.ErrorMessage
  Return
 End if
//do stuff with the database here
Else
 MsgBox "Connection failed."
End if
```

See Also USE DATABASE.

CLOSE CONNECTION

Syntax **CLOSE CONNECTION *ConnectionID***

Privileges ADMIN

Description CLOSE CONNECTION closes the connection for the passed user connection ID. You can get the connection IDs from the ID field in the RecordSet returned by SHOW CONNECTIONS.

See the "Connections" project in the Examples folder for an example that gets all the current connections and closes the selected connection.

Example The following example closes the connection for the passed user ID.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("CLOSE CONNECTION 10")
 If db.error then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
         +db.ErrorMessage
  Return
 else
  MsgBox "Close was successful."
 End if
Else
 MsgBox "Close failed."
End if
```

See Also SHOW CONNECTIONS, SHOW INFO.

CREATE DATABASE WITH KEY

Syntax **CREATE DATABASE d*atabase* WITH KEY [IF NOT EXISTS] *key***

Privileges DATABASES

Description CREATE DATABASE WITH KEY creates a new encrypted database on the server with the passed name and encryption key. The server must have the key in order to serve the encrypted database.

If the database already exists and the optional IF NOT EXISTS parameter is not passed, CREATE DATABASE returns error code 7028. If the optional IF NOT EXISTS parameter is used and the database exists, the command does not return an error.

Example This example connects to the local server and creates the database "Employees" on the server.

```
 db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
  db.SQLExecute ("CREATE DATABASE employees WITH KEY IF NOT EXISTS"_
                 +" xfwr267")
  If db.Error then
    MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "+_
      db.ErrorMessage
    Return
  Else
    MsgBox "Database created or already exists!"
  End if
Else
  MsgBox "Connection failed!"
End if
```

See Also DROP DATABASE, LOCK DATABASE, SHOW DATABASES, SHOW DATABASES WITH DETAILS, SET KEY FOR DATABASE, START DATABASE, STOP DATABASE, USE DATABASE.

CREATE DATABASE

Syntax **CREATE DATABASE *database* [IF NOT EXISTS]**

Privileges DATABASES

Description CREATE DATABASE creates a new database with the passed name on the server. If the database already exists and the optional IF NOT EXISTS parameter is not passed, CREATE DATABASE returns error code 7028. If the optional IF NOT EXISTS parameter is used and the database exists, the command does not return an error.

Example

This example connects to the local server and creates the database "Employees" on the server.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
  db.SQLExecute ("CREATE DATABASE employees IF NOT EXISTS")
  If db.Error then
    MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "+_
    db.ErrorMessage
    Return
  Else
    MsgBox "Database created or already exists!"
  End if
Else
  MsgBox "Connection failed!"
End if
```

See Also

CREATE DATABASE WITH KEY, DROP DATABASE, LOCK DATABASE, SET KEY FOR DATABASE, SHOW DATABASES, SHOW DATABASES WITH DETAILS, START DATABASE, STOP DATABASE, USE DATABASE.

CREATE DATABASE WITH ENCODING

Syntax

CREATE DATABASE *database* [IF NOT EXISTS] WITH ENCODING *value*

Privileges

DATABASES

Description

CREATE DATABASE creates a new database with the passed name on the server. If the database already exists and the optional IF NOT EXISTS parameter is not passed, CREATE DATABASE returns error code 7028. If the optional IF NOT EXISTS parameter is used and the database exists, the command does not return an error. The encoding parameter enables you to specify the encoding that the database will use, such as UTF-8.

Example This example connects to the local server and creates the database "Employees" on the server.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
  db.SQLExecute ("CREATE DATABASE employees IF NOT EXISTS WITH_
        ENCODING UTF--16")
  If db.Error then
    MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "+_
    db.ErrorMessage
    Return
  Else
    MsgBox "Database created or already exists!"
  End if
Else
  MsgBox "Connection failed!"
End if
```

See Also CREATE DATABASE WITH KEY, DROP DATABASE, LOCK DATABASE, SET KEY FOR DATABASE, SHOW DATABASES, SHOW DATABASES WITH DETAILS, START DATABASE, STOP DATABASE, USE DATABASE.

CREATE GROUP

Syntax **CREATE GROUP group**

Privileges PRIVILEGES

Description CREATE GROUP creates a new group with the passed name. Call ADD USER TO GROUP to assign users to this group.

Example This example creates the group "Engineers."

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("CREATE GROUP Engineers")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
                  +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also ADD USER TO GROUP, CREATE USER, DROP GROUP, DROP USER, REMOVE USER, RENAME GROUP, SHOW GROUPS, SHOW GROUPS FOR USER, SHOW USERS, SHOW USERS IN GROUP.

CREATE SCHEDULE

Syntax **CREATE SCHEDULE *name* DAYS *days* HOURS *hours* MINUTES *minutes* WEEKS *weeks* TYPE *value* WITH OPTIONS *options* ENABLED *enabled***

Privileges ADMIN user

Description CREATE SCHEDULE creates a schedule with the given name. Create the schedule with CREATE SCHEDULE and apply the schedule to one or more databases with the ATTACH SCHEDULE command. A database can have more than one schedule attached to it.

Note: The Admin application will not display a backup schedule for a database unless it is named "*DatabaseName* backup". The Admin application uses this name when you create a backup schedule and recognizes a backup with this name that was created with this command and attached to the database.

Here are descriptions of the parameters that CREATE SCHEDULE requires:

Field Name	Description
name	The name of the schedule.
days	A list of the days in the week that the schedule will execute (0=Sunday and 6=Saturday). If you want to execute the schedule on more than one day per week, list the day numbers without any delimiter. For example, if you want the schedule to execute on Sunday and Wednesday, pass '03'.
hours	The hour that the schedule will execute (0 to 23 in local time).
minutes	The minute that the schedule will execute (0-59 in local time).
weeks	How often (in weeks) the schedule will execute. 1 means every week, 2 means every second week, 3 means every third week, and so forth. The range is from 0 to 53.
type	The type of schedule. The acceptable values are BACKUP, SQL, or SHELL. If type is Backup, the schedule sets up a database backup. If type is SQL, it schedules a SQL command to execute. If type is SHELL, then it schedules a shell script to run.
options	Schedule options. There are different types of options for each schedule type. If type is BACKUP, the valid options are either RETAIN_OLD or NOT_RETAIN. If type is SQL, then the Options parameter contains the SQL commands that you want to execute. For example, "VACUUM" performs a vacuum according to the schedule. If type is SHELL, then the options parameter is the full path to the application to execute.
enabled	Equals 1 if the schedule is enabled or 0 if it is not.

Example

The following example creates a backup schedule, "myBackup" for the Images database.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("CREATE SCHEDULE myBackup DAYS 1 HOURS 7 "_
       +" MINUTES 0 WEEKS 1 TYPE BACKUP WITH OPTIONS RETAIN_OLD"_
       +" ENABLED 1")
 db.SQLExecute("ATTACH SCHEDULE myBackup TO DATABASE Images")
 if db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
         +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

ATTACH SCHEDULE, DETACH SCHEDULE, DROP SCHEDULE, SHOW DATABASES FOR SCHEDULE, SHOW SCHEDULE, SHOW SCHEDULES, SHOW SCHEDULES FOR DATABASE, UPDATE SCHEDULE.

CREATE USER

Syntax

CREATE USER *username*

Privileges

PRIVILEGES

Description

CREATE USER creates a new user with the passed username. Give the user a password with the SET PASSWORD command. Add the user to a group using ADD USER TO GROUP.

You can instead call CREATE USER WITH PASSWORD or CREATE USER WITH HASH PASSWORD to create the user and set the password in one step.

Example This example creates the user "Jeff."

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("CREATE USER Jeff")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
            +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also ADD USER TO GROUP, CREATE GROUP, CREATE USER WITH PASSWORD, CREATE USER WITH HASH PASSWORD, DROP USER, REMOVE USER, RENAME USER, SET PASSWORD, SHOW USERS, SHOW USERS IN GROUP.

CREATE USER WITH PASSWORD

Syntax **CREATE USER *username* WITH PASSWORD *password***

Privileges Privileges

Description CREATE USER WITH PASSWORD creates a user with the passed *username* and *password*. It is equivalent to calling CREATE USER and then calling SET PASSWORD FOR USER.

Example This example creates the user "Jeff" with password "summit".

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("CREATE USER Jeff WITH PASSWORD summit")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
            +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also ADD USER TO GROUP, CREATE GROUP, CREATE USER, CREATE USER WITH HASH PASSWORD, DROP USER, REMOVE USER, RENAME USER, SET PASSWORD, SHOW USERS, SHOW USERS IN GROUP.

CREATE USER WITH HASH PASSWORD

Syntax **CREATE USER *username* WITH HASH PASSWORD *password***

Privileges Privileges

Description CREATE USER WITH HASH PASSWORD creates a user with the passed *username* and *password*. The *password* parameter is not sent in the clear. It must be the hash of the password to use. It requires an SHA1 function. Since this is not built into REALbasic, you must use a plug-in that provides this capability. The example assumes that you have installed such a plug-in.

Example This example creates the user "Jeff" with password "summit".

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 //SHA1 is an external plug-in which must be installed
 Dim myPass as String = EncodeBase64(SHA1(SHA1("summit")))
 db.SQLExecute("CREATE USER Jeff WITH HASH PASSWORD "+ myPass)
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
            +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also ADD USER TO GROUP, CREATE GROUP, CREATE USER, CREATE USER WITH PASSWORD, DROP USER, REMOVE USER, RENAME USER, SET PASSWORD, SHOW USERS, SHOW USERS IN GROUP.

DELETE BACKUP FOR DATABASE

Syntax **DELETE BACKUP FOR DATABASE *value***

Privileges BACKUP

Description DELETE BACKUP FOR DATABASE deletes the passed backup.

Example This example deletes the backup for the Images database.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
  db.SQLExecute("DELETE BACKUP FOR DATABASE Images")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
            +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also DELETE BACKUP FOR DATABASE WITH TIMESTAMP.

DELETE BACKUP FOR DATABASE WITH TIMESTAMP

Syntax **DELETE BACKUP FOR DATABASE *value* WITH TIMESTAMP *value***

Privileges BACKUP

Description DELETE BACKUP FOR DATABASE WITH TIMESTAMP deletes the backup identified by both the databasename and a timestamp. The timestamp must be in the same format as with SHOW BACKUPS FOR DATABASE, i.e, SQL DateTime, YYYYMMDD_HHMMSS.

Example This example deletes a backup indicated by its timestamp.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
  db.SQLExecute("DELETE BACKUP FOR DATABASE Images WITH "_
"             TIMESTAMP 20090428_122215")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
            +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also DELETE BACKUP FOR DATABASE.

DETACH SCHEDULE FROM DATABASE

Syntax **DETACH SCHEDULE *name* FROM DATABASE *database***

Privileges Admin user

Description DETACH SCHEDULE removes a schedule from a database. Use ATTACH SCHEDULE to attach the schedule.

Example This example detaches the schedule "myBackup" from the database "Images." It was previously attached with ATTACH SCHEDULE.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("DETACH SCHEDULE myBackup FROM DATABASE Images")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned."_
       +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also ATTACH SCHEDULE, CREATE SCHEDULE, DROP SCHEDULE, SHOW DATABASES FOR SCHEDULE, SHOW SCHEDULE, SHOW SCHEDULES, SHOW SCHEDULES FOR DATABASE, UPDATE SCHEDULE.

DISABLE PLUGIN

Syntax **DISABLE PLUGIN *value***

Privileges PLUGIN

Description DISABLE PLUGIN disables the passed plugin.

Example

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("DISABLE PLUGIN fts1")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned."_
       +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also ENABLE PLUGIN, SHOW PLUGINS.

DISABLE RESTORE ON DATABASE

Syntax **DISABLE RESTORE ON DATABASE *value***

Privileges SERVER

Description DISABLE RESTORE ON DATABASE disables the restore command for the passed database.

Example

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("DISABLE RESTORE ON DATABASE Images")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned."_
       +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also ENABLE RESTORE ON DATABASE.

DOWNLOAD DATABASE

Syntax **DOWNLOAD DATABASE database**

Privileges DOWNLOAD

Description DOWNLOAD DATABASE exports a copy of a database in the "databases" folder to another location on your computer.

A call to DOWNLOAD DATABASE copies the passed database to another folder on your hard disk. After a successful call, a copy of the database is placed in that folder.

The Examples folder includes a REALbasic application that illustrates uploading and downloading databases.

Example

This example downloads the selected database to the directory that was chosen from the SelectFolder call.

```
//mDatabase is a REALSQLServerDatabase property of the window
// sanity check
if mDatabase = nil or not mDatabase.IsConnected then
 MsgBox "You are not connected to a server. Connect first before"_
                +" attempting an upload or download."
 Return
End if

  // ask the user for a folder to download to
Dim f as FolderItem = SelectFolder
if f = Nil then
 Return
End if

  // get the name of the database to download
Dim dbName as String = DatabasesListBox.Text
f = f.Child(dbName)

// initiate the download with the server (notice how we quote the
//database name with single quotes in case the name contains spaces)
mDatabase.SQLExecute "DOWNLOAD DATABASE '" + dbName + "'"
if mDatabase.Error then
// couldn't initiate download; report the error and bail
 MsgBox "Couldn't initiate download with server: " +_
                mDatabase.ErrorMessage
 Return
End if
```

```
 // try to create the new file as a BinaryStream
Dim bs as BinaryStream = f.CreateBinaryFile("")
if bs = Nil then
 Return
End if

  // call ReceiveChunk in a loop until all chunks have been received
While True
    // read the next chunk from the server
 dim chunk as String = mDatabase.ReceiveChunk

 // there was an error receving a chunk, report the error and bail
 if mDatabase.Error then
  MsgBox "Error receiving chunk from server: " + mDatabase.ErrorMessage
  Return
 end if

// see if we have reached the end of the chunks and exit the loop if we have
 If mDatabase.EndChunk then
  Exit
 End if
  // write the chunk out to the file and loop again
 bs.Write chunk
Wend

  // report success
  MsgBox "Download completed."
```

See Also SET CHUNK SIZE, UPLOAD DATABASE

DROP DATABASE

Syntax **DROP DATABASE database**

Privileges DATABASES

Description DROP DATABASE deletes the database with the passed name from the server.

Example The following code deletes a database that was created by the CREATE DATABASE example.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

if db.Connect then
 db.SQLExecute("DROP DATABASE Employees")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "+_
              db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also CREATE DATABASE, SHOW DATABASES, SHOW DATABASES WITH DETAILS, SHOW LASTROWID, SHOW STATISTICS, SHOW TABLES, SHOW TABLES FOR DATABASE, START DATABASE, STOP DATABASE, USE DATABASE.

DROP DATABASE IF EXISTS

Syntax **DROP DATABASE *value* IF EXISTS**

Privileges DATABASES

Description DROP DATABASE deletes the database with the passed name from the server.

Example The following code deletes a database that was created by the CREATE DATABASE example.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

if db.Connect then
 db.SQLExecute("DROP DATABASE Employees IF EXISTS")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "+_
              db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also CREATE DATABASE, SHOW DATABASES, SHOW DATABASES WITH DETAILS, SHOW LASTROWID, SHOW STATISTICS, SHOW TABLES, SHOW TABLES FOR DATABASE, START DATABASE, STOP DATABASE, USE DATABASE.

DROP GROUP

Syntax **DROP GROUP *group***

Privileges PRIVILEGES

Description DROP GROUP deletes the group with the passed name.

Example This example deletes the group "Engineers."

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("DROP GROUP Engineers")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
         +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also ADD USER TO GROUP, CREATE GROUP, RENAME GROUP, SHOW GROUPS, SHOW GROUPS FOR USER, SHOW USERS IN GROUP.

DROP SCHEDULE

Syntax **DROP SCHEDULE *name***

Privileges ADMIN user

Description DROP SCHEDULE deletes the schedule with the passed name.

Example This example deletes the schedule "myBackup" that was previously created with CREATE SCHEDULE.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("DROP SCHEDULE myBackup")
 If db.error then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
            +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also ATTACH SCHEDULE, CREATE SCHEDULE, DETACH SCHEDULE, SHOW DATABASES FOR SCHEDULE, SHOW SCHEDULE, SHOW SCHEDULES, SHOW SCHEDULES FOR DATABASE, UPDATE SCHEDULE.

DROP USER

Syntax **DROP USER *username***

Privileges PRIVILEGES

Description DROP USER deletes the passed user.

Example This example drops the user "Pat."

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("DROP USER  Pat")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
          +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also ADD USER TO GROUP, CREATE USER, REMOVE USER, RENAME USER, SET PASSWORD, SHOW GROUPS FOR USER, SHOW USERS, SHOW USERS IN GROUP.

ENABLE PLUGIN

Syntax **ENABLE PLUGIN *value***

Privileges PLUGIN

Description To use a plug-in, just drag into the folder to install it. You can call DISABLE PLUGIN to disable it. Use ENABLE PLUGIN only if you have previously called DISABLE PLUGIN.

Example

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("ENABLE PLUGIN fts1")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned."_
       +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

DISABLE PLUGIN, SHOW PLUGINS.

ENABLE RESTORE ON DATABASE

Syntax

ENABLE RESTORE ON DATABASE ***value***

Privileges

Server

Description

ENABLE RESTORE ON DATABASE enables the restore functionality if it has been disabled by DISABLE RESTORE ON DATABASE.

Example

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("ENABLE RESTORE ON DATABASE Images")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned."_
       +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

DISABLE RESTORE ON DATABASE

GRANT

There are three syntaxes to the GRANT command.

Syntax

GRANT priv*ilege* TO GROUP *group*

Privileges

PRIVILEGES

Description

GRANT grants the group the passed server privilege. See the section "The Privileges Panel" on page 41 for more information about privileges.

The following Server-level privileges can be granted:

Privilege	Description
DATABASES	Can create and drop databases.
DOWNLOAD	Can download databases from the server.
PREFERENCES	Can see and edit server preferences.
PRIVILEGES	Can edit user privileges.
UPLOAD	Can upload databases to the server.

Example

This example grants databases privileges to the group "middleManagement."

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

if db.Connect then
 db.SQLExecute("GRANT databases TO GROUP middleManagement")
 if db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
              +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

Syntax

GRANT *privilege* TO GROUP *group* FOR DATABASE *database*

Privileges

PRIVILEGES

Description

This syntax grants the group the passed Database privilege for the given database. See the section "The Privileges Panel" on page 41 for more information about privileges. This syntax is for granting Database-level privileges.

The following Database preferences can be granted.

Privilege	Description
CREATE	Can create tables and indexes in the database.
DROP	Can drop tables and indexes in the database.
SELECT	Can query any table in the database.
INSERT	Can insert records into any table in the database.
UPDATE	Can update records in any table in the database.
DELETE	Can delete records in any table in the database.
UPLOAD	Can upload the database to replace the existing one.
DOWNLOAD	Can download the database.

Example

The following example grants CREATE privileges to the middleManagement group for the "Images" database.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("GRANT create TO GROUP middleManagement FOR" _
      +" DATABASE Images")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned."_
       +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

Syntax

GRANT *privilege* TO GROUP *group* FOR TABLE *table* IN DATABASE *database*

Privileges

PRIVILEGES

Description

This syntax grants the group the passed Table privilege for the passed table in the passed database. See the section "The Privileges Panel" on page 41 for more information about privileges. This syntax is for granting Table-level privileges.

The following Table privileges can be granted.

Privilege	Description
SELECT	Can query the table.
INSERT	Can insert records in the table.
UPDATE	Can update records in the table.
DELETE	Can delete records in the table.

These four privileges can also be granted at the database level. If granted, they automatically apply to all tables in the database.

Example The following example grants SELECT privileges to the middleManagement group for the table "images" in the database "Images."

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("GRANT select TO GROUP middleManagement "_
          +"FOR TABLE images IN DATABASE Images")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
           +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also REVOKE, SHOW PRIVILEGES, SHOW PRIVILEGES FOR GROUP.

LOCK DATABASE

Syntax **LOCK DATABASE *database***

Privileges DATABASES

Description LOCK DATABASE locks the database. Use it when you need exclusive access to the database. You will be the only one able to write to a locked database. Call UNLOCK DATABASE when you no longer need exclusive write access.

The Locks Timeout preference that can be set in the Server Settings dialog or via the SET PREFERENCE command is used to release a locked database if there has been no activity for the specified amount of time and another user is trying to gain write access.

Example This example locks the Images database.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("USE DATABASE Images")
 db.SQLExecute("LOCK DATABASE Images")
  //you have exclusive write access here...
 db.SQLExecute("UNLOCK DATABASE Images")

 if db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+ _
       " was returned with message "+db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also LOCK READ DATABASE, LOCK WRITE DATABASE, UNLOCK DATABASE, UNLOCK DATABASE WITH FORCE, USE DATABASE..

LOCK READ DATABASE

Syntax **LOCK READ DATABASE *value***

Privileges DATABASE

Description LOCK READ DATABASE locks the database against reading. Use it when you need exclusive access to the database. You will be the only one able to write to a locked database. Call UNLOCK READ DATABASE when you no longer need exclusive write access.

The Locks Timeout preference that can be set in the Server Settings panel or via the SET PREFERENCE command is used to release a locked database if there has been no activity for the specified amount of time and another user is trying to gain write access.

Example This example locks the Images database.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
  db.SQLExecute("LOCK READ DATABASE Images")
  //you have exclusive write access here...
 db.SQLExecute("UNLOCK READ DATABASE Images")

 if db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+ _
       " was returned with message "+db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also UNLOCK READ DATABASE.

LOCK RECORD

Syntax **LOCK RECORD *rowid* ON TABLE *tableName***

Privileges Null

Description LOCK RECORD locks the record with the passed rowid for the table with the passed name. When the record is locked, you have exclusive write access. The rowid field is created and maintained by the REAL Server database automatically. Call UNLOCK RECORD when you no longer need exclusive write access to the record.

Example This example locks the record with rowid 8 and then unlocks it.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("USE DATABASE Images")
 db.SQLExecute("LOCK RECORD 8 ON TABLE images")
 //you have exclusive write access here...
 db.SQLExecute("UNLOCK RECORD 8 ON TABLE images")

 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+ _
       " was returned with message "+db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also UNLOCK RECORD, USE DATABASE.

LOCK WRITE DATABASE

Syntax **LOCK WRITE DATABASE *value***

Privileges Databases

Description LOCK WRITE DATABASE locks the passed database against write operations. Call UNLOCK WRITE DATABASE to release the lock.

The Locks Timeout preference that can be set in the Server Settings panel or via the SET PREFERENCE command is used to release a locked database if there has been no activity for the specified amount of time and another user is trying to gain write access.

Examples

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
  db.SQLExecute("LOCK WRITE DATABASE Images")
  //you have exclusive write access here...
 db.SQLExecute("UNLOCK WRITE DATABASE Images")

 if db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+ _
       " was returned with message "+db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

UNLOCK WRITE DATABASE.

MOVE USER TO GROUP

Syntax

MOVE USER *value* TO GROUP *value*

Privileges

Privileges

Description

MOVE USER TO GROUP moves the passed user to the passed group. The Users and Groups panel reflects the change the next time it is displayed.

Example

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
  db.SQLExecute("MOVE USER jeff TO GROUP Managers")

 if db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+ _
       " was returned with message "+db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

ADD USER TO GROUP.

PING

Syntax

PING

Privileges

None.

Description

Used to determine if the server is alive. It either completes successfully, or returns an error.

Example The following pings a remote server.

```
db=New REALSQLServerDatabase
db.Host="192.168.1.103"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("PING")
 If db.error then
  MsgBox "An error was returned from the server " + db.ErrorMessage + _
      " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also SHOW CONNECTIONS, SHOW INFO, SHOW SERVER STATUS.

QUIT SERVER

Syntax **QUIT SERVER**

Privileges Admin

Description QUIT SERVER quits the server you are connected to.

Examples

```
db=New REALSQLServerDatabase
db.Host="192.168.1.103"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("QUIT SERVER")
 If db.error then
  MsgBox "An error was returned from the server " + db.ErrorMessage + _
      " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

START SERVER.

REMOVE USER

Syntax

REMOVE USER u*ser* FROM GROUP *group*

Privileges

PRIVILEGES

Description

REMOVE USER removes the passed user from the passed group. Both the user and the group must already exist.

Example This example removes the user "Pat" from the group "middleManagement."

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("REMOVE USER Pat FROM GROUP middleManagement")
 If db.error then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
         +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also ADD USER TO GROUP, CREATE USER, DROP USER, SHOW USERS, SHOW USERS IN GROUP.

RENAME GROUP

Changes the name of the passed group.

Syntax **RENAME GROUP old*GroupName* TO *newGroupName***

Privileges PRIVILEGES

Description RENAME GROUP renames the passed old GroupName with the newGroupName. The group retains the privileges it enjoyed under the old group name.

Example This example renames the group "dataEntry." This group has already been created on the server.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("RENAME GROUP dataEntry TO middleManagement")
 If db.error then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
          +db.ErrorMessage
  Return
 else
  MsgBox "Rename was successful."
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also ADD USER TO GROUP, CREATE GROUP, CREATE USER, DROP GROUP, DROP USER, RENAME USER, SHOW GROUPS FOR USER, SHOW USERS, SHOW USERS IN GROUP.

RENAME USER

Changes the name of the passed user.

Syntax **RENAME USER *oldUserName* TO *newUserName***

Privileges PRIVILEGES

Description RENAME USER renames the passed oldUserName with the passed newUserName.

Example This example renames the user "Jeff." This user has already been created on the server.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("RENAME USER Jeff  TO Pat")
 If db.error then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
          +db.ErrorMessage
  Return
 else
  MsgBox "Rename was successful."
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also ADD USER TO GROUP, CREATE GROUP, CREATE USER, DROP GROUP, DROP USER, RENAME GROUP, SHOW GROUPS FOR USER, SHOW USERS, SHOW USERS IN GROUP.

RESET AUTOINCREMENT

Syntax **RESET AUTOINCREMENT TO *value* IN TABLE *value***

Privileges (null)

Description RESET AUTOINCREMENT resets the counter that autoincrements the primary key field.

Examples

```
db=New REALSQLServerDatabase
db.Host="192.168.1.103"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("RESET AUTOINCREMENT TO 10 IN TABLE Images")
 If db.error then
  MsgBox "An error was returned from the server " + db.ErrorMessage + _
      " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

RESET ERROR

Syntax **RESET ERROR**

Privileges None

Description Call RESET ERROR to reset a persistent error that is not cleared automatically by a subsequent SQLExecute statement and must be cleared explicitly by the client. A persistent error occurs when the Locks Timeout value is exceeded. You should call RESET ERROR when the Lock Timeout error occurs (Error 7666).

Note that a persistent error is cleared without a call to RESET ERROR if the client calls the USE DATABASE command.

Example In this example, the client computer is doing a series of INSERTs while maintaining a lock on the database. In the midst of his work, he goes out for a hamburger without releasing the lock. The code to clear the persistent error is:

```
db.SQLExecute("INSERT INTO...")
If db.ErrorCode = kLockTimeout then  //test for locks timeout exceeded
 MsgBox "Transaction aborted by the sever"
 // Clear persistent error
 db.SQLExecute("RESET ERROR")
End if
```

RESTORE BACKUP FOR DATABASE

Syntax **RESTORE BACKUP FOR DATABASE *value* WITH TIMESTAMP *value***

Privileges BACKUP

Description RESTORE BACKUP FOR DATABASE restores the backup indicated by the passed timestamp.

Example This example restore the backup for the Images database identified by the passed timestamp.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
  db.SQLExecute("RESTORE BACKUP FOR DATABASE Images WITH "_
"             TIMESTAMP 20090428_122215")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
            +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also RESTORE DATABASE, RESTORE DATABASE TO ID.

RESTORE DATABASE

Syntax **RESTORE DATABASE *value***

Privileges RESTORE

Description RESTORE DATABASE restores the passed database.

Example This example restores the database Images.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
  db.SQLExecute("RESTORE DATABASE Images"
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
            +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also RESTORE DATABASE ON TABLE, RESTORE DATABASE TO ID.

RESTORE DATABASE TO ID

Syntax **RESTORE DATABASE *value* TO ID *id***

Privileges RESTORE

Description RESTORE DATABASE TO ID restores the passed database using the backup specified by its ID. The ID can be obtained from the SHOW RESTORE LOG FOR DATABASE command.

Example This example restores the database Images.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
  db.SQLExecute("RESTORE DATABASE Images TO ID 2"
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
            +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also RESTORE DATABASE ON TABLE

RESTORE DATABASE ON TABLE TO ID

Syntax **RESTORE DATABASE *value* ON TABLE *value* TO ID *value***

Privileges RESTORE

Description RESTORE DATABASE ON TABLE restores the passed table using the passed ID. You can get the IDs of your backups with SHOW RESTORE LOG FOR DATABASE.

Example

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
  db.SQLExecute("RESTORE DATABASE Images ON TABLE images TO ID 2"
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
            +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

RESTORE DATABASE, RESTORE DATABASE TO ID.

REVOKE

The REVOKE command revokes privileges from groups for the server, databases, tables, or all possible privileges. It has four syntaxes.

Syntax

REVOKE *value* FROM GROUP *group*

Privileges

PRIVILEGES

Description

This syntax revokes all privileges from the passed group.

Example

This example removes all the privileges from the group "middleManagement."

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("REVOKE ALL FROM GROUP middleManagement")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
            +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

Syntax

REVOKE *privilege* FROM GROUP *group*

Privileges

PRIVILEGES

Description

This syntax revokes the passed privilege from the passed group. See the section "The Privileges Panel" on page 41 for more information about privileges. This syntax is for revoking Server-level privileges. The following Server-level privileges can be revoked.

Privilege	Description
DATABASES	Can create and drop databases.
DOWNLOAD	Can download databases to the server.
PREFERENCES	Can see and edit server preferences.
PRIVILEGES	Can edit user privileges.
UPLOAD	Can upload databases to the server.

Example

This example revokes the DATABASES privilege from the group "middleManagement."

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("REVOKE databases FROM GROUP middleManagement")
 If db.error then
 MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
            +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

Syntax

REVOKE *privilege* FROM GROUP *group* FOR DATABASE *database*

Privileges

PRIVILEGES

Description

This syntax revokes the passed privilege from the passed group for the passed database. See the section "The Privileges Panel" on page 41 for more information about privileges. This syntax is for revoking Database-level privileges.

The following Database-level privileges can be revoked.

Privilege	Description
CREATE	Can create tables and indexes in the database.
DROP	Can drop tables and indexes in the database.
SELECT	Can query any table in the database.
INSERT	Can insert records into any table in the database.
UPDATE	Can update records in any table in the database.
DELETE	Can delete records in any table in the database.
UPLOAD	Can upload a database of the same name, replacing the current database.
DOWNLOAD	Can download the current database.

Example

This example revokes the CREATE privilege from the group "middleManagement" for the database "Images."

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("REVOKE create FROM GROUP middleManagement "_
          +"FOR DATABASE Images")
 if db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
          +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

Syntax

REVOKE *privilege* FROM GROUP *group* FOR TABLE *table* IN DATABASE *database*

Privileges

PRIVILEGES

Description

This syntax revokes the passed privilege from the passed group for the passed table in the passed database. See the section "The Privileges Panel" on page 41 for more information about privileges. This syntax is for revoking Table-level privileges. The following Table-level privileges can be revoked.

Privilege	Description
SELECT	Can query the table.
INSERT	Can insert records in the table.
UPDATE	Can update records in the table.
DELETE	Can delete records in the table.

Example

This example revokes the SELECT privilege from the group "middleManagement" for the table "images" in the database "Images."

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("REVOKE select FROM GROUP middleManagement "_
        +"FOR TABLE images IN DATABASE Images")
 if db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
        +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

GRANT, SHOW PRIVILEGES, SHOW PRIVILEGES FOR GROUP.

SET AUTOCOMMIT

Syntax

SET AUTOCOMMIT TO ON|OFF

Privileges

None.

Description

The Autocommit property of the REALSQLServer class determines whether the database connection commits changes automatically, or whether changes open an implicit transaction that you must explicitly close by calling Commit or Rollback. The default is False (aka, OFF). Changes open implicit transactions that you must close explicitly. If you turn on Autocommit, then that REALSQLServerDatabase connection will no longer open implicit transactions for you, and instead, will automatically commit each database change immediately. It is recommended that you leave Autocommit alone unless you understand the implications of turning it on.

Example This example turns on the Autocommit property.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("SET AUTOCOMMIT TO ON'")
 If db.error then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
            +db.ErrorMessage
  Return
 Else
 MsgBox "Autocommit set to ON."
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also SHOW AUTOCOMMIT.

SET CHUNK SIZE

Syntax **SET CHUNK SIZE *value***

Privileges None

Description SETCHUNK SIZE sets the chunk_size preference, used in uploading and downloading. This is the same parameter as appears on the Setting panel in the Admin application. It is used to determine the maximum size that can have a RecordSet. The bigger the value, the faster you'll be able to receive the query, but you will use more memory.

It is important for very big queries. Its value depends on how many clients you expect to have connected at the same time. For example, if you set chunk size to 1 MB and you have ten clients connected and all doing a very big query, you can expect 10 MB memory usage just for the queries.

For example, a query that returns 800,000 rows and 140 MB of data gives the following results by chunk size.

Chunk Size	Time
150KB	35.8 seconds
500 KB	15 seconds
1.5 MB	8.66 seconds

Example

Here is an example that sets the chunk size.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("SET CHUNK SIZE TO 200")
 If db.error then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
            +db.ErrorMessage
  Return
 Else
 MsgBox "Autocommit set to ON."
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

SHOW PREFERENCES.

SET HASH PASSWORD FOR USER

Syntax

SET HASH PASSWORD *password* FOR USER *username*

Privileges

Privileges

Description

SET HASH PASSWORD FOR USER assigns the passed password to the passed user. The user must already exist. Users can be created via CREATE USER or via the Admin application.

The password parameter is not sent in the clear. It must be the hash of the password to use. It requires an SHA1 function. Since this is not built into REALbasic, you must

use a plug-in that provides this capability. The example assumes that you have installed such a plug-in.

You can instead create the user and assign the password with a call to CREATE USER WITH HASH PASSWORD or CREATE USER WITH PASSWORD.

Example

This example creates a hashed password for user "Jeff." Please note that the required SHA1 function is not built into REALbasic. You must install an external SHA1 plug-in.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 Dim myPass as String = EncodeBase64(SHA1(SHA1("summit")))
 db.SQLExecute("SET HASH PASSWORD "+ myPass+" FOR USER Jeff")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
          +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

CREATE USER, CREATE USER WITH PASSWORD, CREATE USER WITH HASH PASSWORD, DROP USER, REMOVE USER, SET PASSWORD FOR USER, SHOW USERS, SHOW USERS IN GROUP.

SET KEY FOR DATABASE

Syntax

SET KEY key FOR DATABASE *name*

Privileges

DATABASE

Description

SET KEY FOR DATABASE sets the encryption key for the passed database. If you have created an unencrypted database or imported/dragged an unencrypted database into the databases folder, you can encrypt it by calling this command.

Example The following example sets a key for the "images" database.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
  db.SQLExecute("SET KEY toad453tm FOR DATABASE images")

  If db.error then
    MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
          +db.ErrorMessage
    Return
  Else
    MsgBox "Key successfully assigned!"
  End if
End if
```

See Also CREATE DATABASE WITH KEY.

SET LANGUAGE

Syntax **SET LANGUAGE TO *value***

Privileges None

Description SET LANGUAGE sets the default language for the Server. You must use ISO 3166-1 alpha-2 country codes. See http://en.wikipedia.org/wiki/ISO_3166-1_alpha-2

In version 2009r1, it is ignored. It can be used for localization in 2009r2.

Example

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("SET LAUGUAGE TO English'")
 If db.error then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
            +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

SET MY HASH PASSWORD

Syntax **SET MY HASH PASSWORD TO *value***

Privileges None

Description SET MY HASH PASSWORD sets the password for the passed user. The user must already exist. Users can be created via CREATE USER or via the Admin application. You can create a hashed password with SET HASH PASSWORD FOR USER.

You can instead create the user and assign the password with a call to CREATE USER WITH HASH PASSWORD or CREATE USER WITH PASSWORD.

Example This example sets a password for the new user "Jeff."

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("SET PASSWORD Summit FOR USER Jeff")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
          +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also CREATE USER, CREATE USER WITH PASSWORD, CREATE USER WITH HASH PASSWORD, DROP USER, REMOVE USER, SET MY PASSWORD, SET HASH PASSWORD FOR USER, SHOW USERS, SHOW USERS IN GROUP.

SET MY PASSWORD

Syntax **SET MY PASSWORD TO *value***

Privileges None

Description SET MY PASSWORD sets the password for the current user.

Example This example sets a password for the current user.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("SET MY PASSWORD TO X79biff")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
          +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also CREATE USER, CREATE USER WITH PASSWORD, CREATE USER WITH HASH PASSWORD, DROP USER, REMOVE USER, SET HASH PASSWORD FOR USER, SET MY HASH PASSWORD, SET PASSWORD FOR USER, SHOW USERS, SHOW USERS IN GROUP.

SET PASSWORD FOR USER

Syntax **SET PASSWORD *password* FOR USER *user***

Privileges PRIVILEGES

Description SET PASSWORD sets the password for the passed user. The user must already exist. Users can be created via CREATE USER or via the Admin application. You can create a hashed password with SET HASH PASSWORD FOR USER.

You can instead create the user and assign the password with a call to CREATE USER WITH HASH PASSWORD or CREATE USER WITH PASSWORD.

Example This example sets a password for the new user "Jeff."

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("SET PASSWORD Summit FOR USER Jeff")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
          +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also CREATE USER, CREATE USER WITH PASSWORD, CREATE USER WITH HASH PASSWORD, DROP USER, REMOVE USER, SET HASH PASSWORD FOR USER, SET MY PASSWORD, SHOW USERS, SHOW USERS IN GROUP.

SET PING TIMEOUT

Syntax **SET PING TIMEOUT TO *value***

Privileges None

Description You must call USE DATABASE prior to calling SET PING TIMEOUT.

Example

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
  db.SQL Execute("USE DATABASE images")
 db.SQLExecute("SET PING TMEOUT TO 100")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
           +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

PING.

SET PREFERENCE

Syntax

SET PREFERENCE *key* TO *value*

Privileges

PREFERENCES

Description

SET PREFERENCE sets the preference for the passed key to the passed value. The following preferences can be set:

Preference key	Description
SERVER_NAME	The name you wish to give to the server.
SERVER_PORT	The port you want to use to communicate with the server. This can be any legal port number.
VERBOSE_LOG	Enables or disables logging. If enabled, it specifies the events to be logged. Set to any of the following values: 0: None. 10: SQL errors. 20: SQL statements. 66: Debug. The verbose logging options should be used only for debugging purposes. Verbose logging should not be used in a production server because it will degrade performance.

Preference key	Description
MAX_LOCKTIME	The lock timeout in seconds. It is enabled only if it has a value greater than zero.

Notes

The MAX_LOCKTIME preference prevents the server from maintaining a database in a locked state indefinitely. When a user gains write-access to a database, other users are locked out until the lock is released (i.e., for the duration of the transaction). This preference comes into play only when another client tries to access the database while it is locked. Consider this situation. The user who has locked the database is executing a sequence of INSERTs.

```
db.SQLExecute("INSERT INTO...")
db.SQLExecute("INSERT INTO...")
db.SQLExecute("INSERT INTO...")
  ...
// Take a 15 minute coffee break
  ...
db.SQLExecute("INSERT INTO...")
db.SQLExecute("INSERT INTO...")
db.SQLExecute("INSERT INTO...")
db.Commit
```

When MAX_LOCKTIME is invoked, a ROLLBACK command is sent to the database with respect to the open transaction.

When the user with exclusive write access tries to execute a SQL statement against the locked database, a LOCKTIMEOUT error will be triggered. The code that is executing with write access should test for this error.

```
db.SQLExecute("INSERT INTO...")
if db.ErrorCode = kLockTimeout then

//Clear persistent error
db.SQLExecute("RESET ERROR")
 MsgBox "Transaction aborted by the Server"
End if
```

The persistent error is also cleared automatically if the client issues a USE DATABASE command to the server.

Example The following example changes the server port preference.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"
db.SQLExecute("SET PREFERENCE SERVER_PORT 8430")
If db.connect then

 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
          +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also SHOW PREFERENCE, SHOW PREFERENCES.

SET REGISTRATION

Syntax **SET REGISTRATION TO *name* WITH KEY *serial_number***

Privileges None

Description SET REGISTRATION enables you to enter a license number for a user programmatically.

Example

This example would register the passed user—if a valid serial number had been passed. If an invalid serial number is passed, error 7054 is returned.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("SET REGISTRATION TO 'Elton Thomas' "_
              +"WITH KEY XXXXXXX-XXXXXX-XXXX'")
 If db.error then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
   +db.ErrorMessage
  Return
 else
  MsgBox "User successfully registered!"
 End if
Else
 MsgBox "Connection failed."
End if
```

SET TIMEOUT

Syntax

SET TIMEOUT TO *value*

Privileges

None

Description

SET TIMEOUT sets the Lock Time parameter that specifies how long to wait (in milliseconds) for a locked database. It can also be set in the Admin's Settings panel.

Example

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
  db.SQLExecute("SET TMEOUT TO 100")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
          +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

SHOW PREFERENCES.

SHOW ALL PRIVILEGES

Syntax

SHOW ALLL PRIVILEGES

Privileges

None

Description

SHOW ALL PRIVILEGES returns a recordset that report all privileges. The fields in the RecordSet are as follows:

Field Name	Description
databasename	The database to which the privilege pertains. A "*" indicates all databases
groupname	The value of the preference.
privilege	The privilege.
tablename	The name of the table to which the privilege pertains. A "*" indicates all tables for the specified databasename.

Example

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
  Dim rs as RecordSet
 rs=db.SQLSelect("SHOW ALL PRIVILEGES")
 If rs = Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
                  " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also SHOW PRIVILEGES, SHOW PRIVILEGES FOR GROUP.

SHOW AUTOCOMMIT

Syntax **SHOW AUTOCOMMIT**

Privileges None.

Description SHOW AUTOCOMMIT returns the value of the Autocommit property as a RecordSet. The only field in the RecordSet is named "autocommit."

Example This example reports the current setting of the Autocommit property as either "ON" or "OFF".

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 ListBox1.DeleteAllRows
 Dim rs as RecordSet
 rs=db.SQLSelect("SHOW AUTOCOMMIT")
 If rs = Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
                " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also SET AUTOCOMMIT.

SHOW BACKUPS FOR DATABASE

Syntax **SHOW BACKUPS FOR DATABASE *value***

Privileges Backup

Description SHOW BACKUPS FOR DATABASE lists all the backups for the specified database. The fields in the RecordSet are as follows:

Field Name	Description
databasename	The database that has the backup.
timestamp	The timestamp of the backup. Timestamp is a SQL datetime in the format YYYYMMDD_HHMMSS

Example

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 ListBox1.DeleteAllRows
 Dim rs as RecordSet
 rs=db.SQLSelect("SHOW BACKUPS FOR DATABASE Images")
 If rs = Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
                " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also DELETE BACKUP FOR DATABASE.

SHOW CHANGES

Syntax **SHOW CHANGES**

Privileges None

Description SHOW CHANGES returns a RecordSet with one field, "changes." SHOW CHANGES returns the number of database rows that were changed, inserted, or deleted by the most recently completed SQL statement on the database connection. Only changes that are directly specified by the insert, update, or delete operations. Auxiliary changes caused by triggers are not counted. Use SHOW TOTAL CHANGES to find the total number of changes including changes caused by triggers.

A "row change" is a change to a single row of a single table caused by an INSERT, DELETE, or UPDATE statement. Rows that are changed as side effects of REPLACE constraint resolution, rollback, ABORT processing, DROP TABLE, or by any other mechanisms do not count as direct row changes.

A "trigger context" is a scope of execution that begins and ends with the script of a trigger. Most SQL statements are evaluated outside of any trigger. This is the "top level" trigger context. If a trigger fires from the top level, a new trigger context is entered for the duration of that one trigger. Subtriggers create subcontexts for their duration.

Example

This example returns the number of row changes to the Images database.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
  db.SQLExecute("USE DATABASE IMAGES")
 Dim rs as RecordSet
 rs=db.SQLSelect("SHOW CHANGES")
 If rs = Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
                  " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also

SHOW TOTAL CHANGES, USE DATABASE.

SHOW COMMANDS

Syntax

SHOW COMMANDS

Privileges

None

Description

SHOW COMMANDS returns a list of the special server SQL commands in a RecordSet. The RecordSet contains the following fields, command, context, and privilege.

Field Name	Description
command	The command's syntax. The command is displayed in uppercase and the parameters are in lowercase.
context	The context in which the command is executed. It is either EXECUTE or SELECT. EXECUTE commands are passed to the server with the SQLExecute and SELECT commands are passed to the server with SQLSelect.
privilege	The required privileges for executing the command.

Example

The following example returns the list of commands and displays it in a RecordSet.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 Dim rs as New RecordSet
 ListBox1.DeleteAllRows

 rs=db.SQLSelect("SHOW COMMANDS")
 If rs <> Nil then
  DisplayRecordSet rs
 Else
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
         " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

SHOW CONNECTIONS, SHOW INFO, SHOW SERVER STATUS.

SHOW CONNECTIONS

Syntax

SHOW CONNECTIONS

Privileges

SERVER

Description

SHOW CONNECTIONS returns the following information about every connected user as a RecordSet. You can use the values of the ID field to close a connection with CLOSE CONNECTION.

The fields in the RecordSet are as follows:

Field Name	Description
ID	The user's ID.
Address	The user's IP address.
Username	The user's username.
ConnectionDate	The date the user connected. It is reported in SQL datetime format.

Example

The following example returns information about all the connected users.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 ListBox1.DeleteAllRows
 Dim rs as RecordSet
 rs=db.SQLSelect("SHOW CONNECTIONS")
 If rs = Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
           " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also

CLOSE CONNECTION, SHOW DATABASES FOR SCHEDULE, SHOW INFO, SHOW SERVER STATUS.

SHOW DATABASE INFO

Syntax

SHOW DATABASE INFO *databaseName*

Privileges

None

Description

SHOW DATABASE INFO returns information on the passed database in a RecordSet with the following fields:

Field	Description
Key	The name of the field.
Value	The value of the field.

Currently, the only value of Key is "Size" and it returns the size of the database in bytes.

Example

The following example gets the size of the Images database.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 ListBox1.DeleteAllRows
 Dim rs as RecordSet
 rs=db.SQLSelect("SHOW DATABASE INFO Images")
 If rs = Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
       " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
  DisplayRecordSet rs
 Else
 MsgBox "Connection failed."
End if
```

See Also

SHOW DATABASES, SHOW DATABASES WITH DETAILS, SHOW INDEXES FOR DATABASE.

SHOW DATABASES

Syntax

SHOW DATABASES

Privileges

None.

Description

SHOW DATABASES returns a RecordSet containing a list of the databases on the server. You will only receive those databases you have privileges to read.

The fields in the RecordSet are as follows.

Field	Description
databasename	The database name.

Example

The following example populates a ListBox with the names of the databases on the server.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 ListBox1.DeleteAllRows

 Dim rs as RecordSet = db.SQLSelect("SHOW DATABASES")
 If rs = Nil then
   MsgBox "Unable to get list from server: " + db.ErrorMessage + _
            " (" + Format(db.ErrorCode, "#") + ")
 Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also

SHOW DATABASES FOR SCHEDULE, SHOW DATABASES WITH DETAILS, SHOW LASTROWID, SHOW STATISTICS, SHOW TABLES, SHOW TABLES FOR DATABASE.

SHOW DATABASES FOR SCHEDULE

Syntax

SHOW DATABASES FOR SCHEDULE *scheduleName*

Privileges

SERVER

Description

SHOW DATABASES FOR SCHEDULE returns a RecordSet that contains the databases that have been attached to the passed schedule. You use ATTACH SCHEDULE to assign an existing schedule to a database.

The fields in the RecordSet are as follows:

Field	Description
dbName	The name of the database.

Example

This example gets the names of all of the databases that have the schedule "myBackup" attached to them.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
  ListBox1.DeleteAllRows
  Dim rs as RecordSet = db.SQLSelect("SHOW DATABASES FOR SCHEDULE"_
        +" myBackup")

  If rs = Nil then
    MsgBox "Unable to get list from server: " + db.ErrorMessage + _
          " (" + Format(db.ErrorCode, "#") + ")"
    Return
  End if
  DisplayRecordSet rs
Else
  MsgBox "Connection failed."
End if
```

See Also

ATTACH SCHEDULE, CREATE SCHEDULE, DETACH SCHEDULE, DROP SCHEDULE, SHOW SCHEDULE, SHOW SCHEDULES, SHOW SCHEDULES FOR DATABASE, UPDATE SCHEDULE.

SHOW DATABASES WITH DETAILS

Syntax

SHOW DATABASES WITH DETAILS

Privileges

None.

Description

SHOW DATABASES WITH DETAILS returns a RecordSet of the server's databases with the following information.

The fields in the RecordSet are as follows:

Field	Description
Databasename	The name of the database.
Stopped	Equals True if the database is stopped and False if it is not.
Locked	Equals 1 if the database is locked and 0 if it is not.
Lockowner	The name of the user who has locked the database.
Encrypted	Equals True if the database is encrypted and False if it is not.
Available	Equals True if the server recognizes it as a valid REAL Server database and has the key for it if it is encrypted. It is False if the server cannot recognize it as a database or it is encrypted and the server does not have the key for it.

Example

The following example populates a ListBox with the info on the databases on the server.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
  ListBox1.DeleteAllRows

  Dim rs as RecordSet = db.SQLSelect("SHOW DATABASES WITH DETAILS")
  If rs = Nil then
    MsgBox "Unable to get list from server: " + db.ErrorMessage + _
              " (" + Format(db.ErrorCode, "#") + ")"
    Return
  End if
  DisplayRecordSet rs
Else
  MsgBox "Connection failed."
End if
```

See Also

SHOW DATABASES, SHOW DATABASES FOR SCHEDULE, SHOW LASTROWID, SHOW STATISTICS, SHOW TABLES, SHOW TABLES FOR DATABASE.

SHOW GROUPS

Syntax

SHOW GROUPS

Privileges

PRIVILEGES

Description

SHOW GROUPS returns a list of all the groups on the server as a RecordSet.

The fields in the RecordSet are as follows:

Field Name	Description
groupname	The name of a group.

Example

The following example gets the list of all the groups that have been created on the server.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 ListBox1.DeleteAllRows
 Dim rs as New RecordSet
 rs=db.SQLSelect("SHOW GROUPS")
 If rs=Nil then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
            +db.ErrorMessage
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also

RENAME GROUP, SHOW GROUPS FOR USER, SHOW USERS, SHOW USERS IN GROUP.

SHOW GROUPS FOR USER

Syntax

SHOW GROUPS FOR USER *user*

Privileges

PRIVILEGES

Description

SHOW GROUPS FOR USER returns the groups the passed user is in as a RecordSet.

The fields in the RecordSet are as follows:

Field Name	Description
groupname	The name of a group.

Example The following example gets all the groups that the user "Dave" is in.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 ListBox1.DeleteAllRows
 Dim rs as New RecordSet
 rs=db.SQLSelect("SHOW GROUPS FOR USER Dave")
 If rs=Nil then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
          +db.ErrorMessage
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also RENAME GROUP, SHOW GROUPS,SHOW MY GROUP, SHOW USERS, SHOW USERS IN GROUP.

SHOW ID FOR SCHEDULE

Syntax **SHOW ID FOR SCHEDULE *value***

Privileges Server

Description SHOW ID FOR SCHEDULE returns a recordset with one field, schedule. This field is used to identify schedules in other commands, such as RESTORE DATABASE ON.

Example This example returns the ID for the user-created schedule named "Images."

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 ListBox1.DeleteAllRows
 Dim rs as New RecordSet
 rs=db.SQLSelect("SHOW ID FOR SCHEDULE Images")
 If rs=Nil then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
            +db.ErrorMessage
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also RESTORE DATABASE ON.

SHOW INDEXES FOR DATABASE

Syntax **SHOW INDEXES FOR DATABASE *databaseName***

Privileges (null)

Description SHOW INDEXES FOR DATABASE returns the names of the indexes in a RecordSet. The fields in the RecordSet are as follows:

Field Name	Description
indexname	The name of an index.

Example

The following example gets the list of indexes in the Images database.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 ListBox1.DeleteAllRows

 Dim rs as RecordSet
 rs=db.SQLSelect("SHOW INDEXES FOR DATABASE Images")
 If rs = Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
       " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also

SHOW DATABASE INFO, SHOW DATABASES, SHOW DATABASES WITH DETAILS.

SHOW INFO

Syntax

SHOW INFO

Privileges

None

Description

SHOW INFO returns server information as a RecordSet.

The fields in the RecordSet are as follows:

Field Name	Description
Key	The key name for a piece of information.
Value	The value corresponding to the Key.

The fields correspond closely to the fields displayed on the Status page of the Admin application. It returns the following fields:

Key	Description
Server_version	The version of REAL Server
SQLite_version	The version of SQLite on which REAL Server is based.
Address	IP address of the server machine.
Port	Port used to communicate with the server.
OS (a.k.a., "System")	Operating system used by the server machine.
upsince	Time that the server was started.
connections	Number of connections to the server.
server_path	Path to the server engine.
databases_path	Path to the databases folder.
support_path	Path to the support folder.
registered	1 if registered; 0 if not.
max_connections	The maximum number of allowed connections.
expiration	Date that the current license expires.
server_release	The version of REAL Server. The release version is given as a two digit number following the decimal point. For example, version 2009 Release 1 is given as 2009.01.
max_concurrent_connections	The maximum number of concurrent connections permitted by the REAL Server license.
sql_query_count	The number of SQL Query commands that have been issued.
sql_exec_count	The number of SQL Exec commands that have been issued.
custom_commands_count	The number of custom REQL SQL Server commands that have been issued.

Example This example returns the server info for a remote server.

```
db=New REALSQLServerDatabase
db.Host="192.168.1.107"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 ListBox1.DeleteAllRows

 Dim rs as RecordSet
 rs=db.SQLSelect("SHOW INFO")
 If rs = Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
           " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also SHOW CONNECTIONS, SHOW SERVER STATUS.

SHOW LAST ROWS FROM LOG

Syntax **SHOW LAST *value* ROWS FROM LOG**

Privileges ADMIN

Description SHOW LAST ROWS FROM LOG returns the most recent *value* entries to the log as a RecordSet. The parameter *value* is an integer.

The fields in the RecordSet are as follows:

Field Name	Description
Datetime	The date and time of the logged event, in SQL datetime format.
Description	A description of the event.
Operation	Either SYSTEM, SELECT or EXECUTE, depending on whether the event was system-initiated or user-initiated via a SELECT or an EXECUTE.
Address	The IP address of the machine that issued the SQL statement.

Field Name	Description
Username	The name of the user who issued the SQL statement.
Database	The name of the database that was queried, if relevant.

Example

The following retrieves the last 20 rows from the log.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
ListBox1.DeleteAllRows

Dim rs as RecordSet
rs=db.SQLSelect("SHOW LAST 20 ROWS FROM LOG")
 If rs = Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
           " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
  MsgBox "Connection failed."
End if
```

See Also

SHOW LOG.

SHOW LASTROWID

Syntax

SHOW LASTROWID

Privileges

N/A.

Description

SHOW LASTROWID returns the last rowid added to the current database as a RecordSet. You must specify the current database with a call to the USE DATABASE command.

The fields in the RecordSet are as follows:

Field Name	Description
LastRowID	The last rowID.

Example The following example gets the last row ID for the database "Images."

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 ListBox1.DeleteAllRows

 Dim rs as RecordSet
 db.SQLExecute("USE DATABASE Images")
 rs=db.SQLSelect("SHOW LASTROWID")
 If rs = nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
          " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also SHOW DATABASES, SHOW DATABASES WITH DETAILS, SHOW STATISTICS, SHOW TABLES, SHOW TABLES FOR DATABASE, USE DATABASE.

SHOW LOG

Syntax **SHOW LOG FROM *yyyy-mm-dd* TO *yyyy-mm-dd***

Privileges ADMIN

Description SHOW LOG returns the log information for the period specified by the passed dates in a RecordSet. Dates must be in GMT. SHOW LOG is supported only by the SQLITE format. This format can be set in the Settings dialog (see "Changing Server Settings" on page 23).

The fields in the RecordSet are as follows:

Field Name	Description
Datetime	The date and time of the logged event, in SQL datetime format.

Field Name	Description
Description	A description of the event.
Operation	Either SYSTEM, SELECT, or EXECUTE, depending on whether the event was system-initiated or user-initiated via a SELECT or EXECUTE statement.
Address	The IP address of the machine that issued the SQL statement.
Username	The name of the user who issued the SQL statement.
Database	The name of the database that was queried, if relevant.

Example The following example gets the log for a four-day period.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
ListBox1.DeleteAllRows

Dim rs as RecordSet
rs=db.SQLSelect("SHOW LOG FROM 2009-02-20 TO 2009-02-23")
 If rs = Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
           " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
  MsgBox "Connection failed."
End if
```

See Also SHOW LAST ROWS FROM LOG.

SHOW MY GROUP

Syntax **SHOW MY GROUP**

Privileges None

Description SHOW MY GROUP returns a recordset with one field.

The fields in the RecordSet are as follows:

Field Name	Description
groupname	The name of the group.

Example

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
ListBox1.DeleteAllRows

Dim rs as RecordSet
rs=db.SQLSelect("SHOW MY GROUP")
 If rs = Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
           " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
  MsgBox "Connection failed."
End if
```

See Also

SHOW GROUPS FOR USER, SHOW GROUPS

SHOW MY INFO

Syntax

SHOW MY INFO

Privileges

None

Description

SHOW MY INFO reports the values of variables pertaining to the currently logged in user as a RecordSet. The fields in the RecordSet are as follows:

Field Name	Description
key	The info label.
value	The value of the specified key.

The values of *key* are as follows:

Key
clientID
connectionID
transactionID
userID
groupID
timeout
pingTimeout
clientVersion
clientType
ip_address

Example

The following example retrieves the values for the keys shown above.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
ListBox1.DeleteAllRows

Dim rs as RecordSet
rs=db.SQLSelect("SHOW MY INFO")
 If rs = Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
           " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
  MsgBox "Connection failed."
End if
```

See Also

SHOW INFO.

SHOW MY PRIVILEGES

Syntax

SHOW MY PRIVILEGES

Privileges

None

Description

SHOW MY PRIVILEGES returns a recordset that lists the currently logged in user's privileges and the databases and tables to which each privilege pertains. The fields in the RecordSet are as follows:

Field Name	Description
privilege	The name of the privilege.
databasename	The database to which it pertains.
tablename	The table in the database to which it pertains.

Example

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
ListBox1.DeleteAllRows

Dim rs as RecordSet
rs=db.SQLSelect("SHOW MY PRIVILEGES")
 If rs = Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
           " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
  MsgBox "Connection failed."
End if
```

See Also

SHOW PRIVILEGES TABLE, SHOW PRIVILEGES FOR GROUP.

SHOW PLUGINS

Syntax

SHOW PLUGINS

Privileges

None

Description

SHOW PLUGINS returns a RecordSet that provides information about the installed plugins. The fields in the RecordSet are as follows:

Field Name	Description
name	The name of the plugin.
version	The version of the plugin.
copyright	Copyright information.
description	The description of the plugin
exename	The executable name.

Example

The following example returns the characteristics of the installed plugins.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then

 Dim rs as RecordSet
 rs=db.SQLSelect("SHOW PLUGINS")
 If rs = Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
                " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also

DISABLE PLUGIN, ENABLE PLUGIN.

SHOW PREFERENCE

Syntax **SHOW PREFERENCE *key***

Privileges PREFERENCES

Description SHOW PREFERENCE returns the value of the preference with the passed key as a RecordSet.

The fields in the RecordSet are as follows:

Field Name	Description
Key	The preference key or name.
Value	The value of the preference.

The preferences listed for the SHOW PREFERENCES command can be retrieved individually.

Example The following example returns the value of the preference "Server_Port."

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 ListBox1.DeleteAllRows

 Dim rs as RecordSet
 rs=db.SQLSelect("SHOW PREFERENCE SERVER_PORT")
 If rs = Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
                " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also SET PREFERENCE, SHOW PREFERENCES.

SHOW PREFERENCES

Syntax

SHOW PREFERENCES

Privileges

PREFERENCES

Description

SHOW PREFERENCES returns all the server preferences and their values as a RecordSet.

The fields in the RecordSet are as follows:

Field Name	Description
Key	The preference key or name
Value	The value of the preference.

The following preferences are reported:

Key	Description
BACKUP_PATH	The path to the backups folder. By default, it is in the REAL Server folder.
CHUNK_SIZE	The chunk size used for uploading and downloading databases.
DATABASES_PATH	The path to the databases folder. By default, it is in the REAL Server folder.
EXPIRATION_DATE	Expiration date of the license
JOURNAL_LOG	Not supported in Release 1.
KEY_NAME	Either demo or registered.
KEY_STATUS	Either demo or registered.
LOG_FORMAT	The format in which the log is written. Either SQLITE or TEXT.
MAX_LOCKTIME	The max_locktime value in seconds.
OS_VERSION	Version of the operating system on the server machine.
PREF_VERS	Version stamp of the preferences info that is returned, so you can remain compatible with older preferences.
SERVER_NAME	The name of the server.
SERVER_PATH	The path to the server engine.
SERVER_PORT	The port used to communicate with the server.
SERVER_VERSION	The version of the server, reported as a string.
USE AUTOTRANSACTIONS	1 if Autotransactions is on; 0 if is off.

Key	Description
VERBOSE_LOG	The setting of the VERBOSE_LOG preference. 0: None. 10: SQL errors. 20: SQL statements. 66: Debug.

Example The following example lists the preferences.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 ListBox1.DeleteAllRows

 Dim rs as RecordSet
 rs=db.SQLSelect("SHOW PREFERENCES")
 If rs = nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
              " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also SET PREFERENCE, SHOW PREFERENCE.

SHOW PRIVILEGES FOR GROUP

Syntax **SHOW PRIVILEGES FOR GROUP *group***

Privileges PRIVILEGES

Description SHOW PRIVILEGES FOR GROUP returns the privileges for the passed group as a RecordSet.

The fields in the RecordSet are as follows:

Field Name	Description
Privilege	The name of the privilege.
Databasename	The name of the database affected by the privilege.
Tablename	The name of the table affected by the privilege.

Example

The following example gets the privileges for the "dataEntry" group.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 ListBox1.DeleteAllRows
 Dim rs as RecordSet
 rs=db.SQLSelect("SHOW PRIVILEGES FOR GROUP dataEntry")
 if rs=Nil then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
              +db.ErrorMessage
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also

GRANT, REVOKE, SHOW PRIVILEGES.

SHOW PRIVILEGES TABLE

Syntax

SHOW PRIVILEGES TABLE

Privileges

None

Description

SHOW PRIVILEGES TABLE returns the privileges for the currently logged in user as a RecordSet. If the user is in the admin group, SHOW PRIVILEGES TABLE returns no privilege records.

The fields in the RecordSet are as follows:

Field Name	Description
name	The name of the privilege.
value	The value of the privilege.

Example

This example shows the privileges for the currently logged in user, "Jeff."

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="Jeff"
db.Password="103dYt7"

If db.connect then
 Dim rs as New RecordSet
 rs=db.SQLSelect("SHOW PRIVILEGES TABLE")
 if rs <> Nil then
  DisplayRecordSet rs
 Else
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
            " (" + Format(db.ErrorCode, "#") + ")"
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

SHOW PRIVILEGES FOR GROUP. SHOW MY PRIVILEGES.

SHOW REALTIME STATISTICS

Syntax

SHOW REALTIME STATISTICS

Privileges

None

Description

SHOW REALTIME STATISTICS returns a RecordSet with statistics on a variety of server parameters. The fields in the RecordSet are as follows:

Field Name	Description
Key	The preference key or name.
Value	The value of the preference.

The values of *key* are as follows:

Key
server_version
engine_version
server_release
server_license
server_expiration
ip_address
server_port
messaging_port
os_version
server_startup_datetime
server_current_datetime
server_hours_running
current_connections
total_connections
max_connections
max_allowed_connections
bytes_sent
bytes_received
server_path
databases_path
backups_path
restore_path
query_count
exec_count
commands_count
memory_usage
max_memory_usage

Example

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 ListBox1.DeleteAllRows
 Dim rs as RecordSet
 rs=db.SQLSelect("SHOW REALTIME STATISTICS")
 if rs=Nil then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
                +db.ErrorMessage
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also UPDATE STATISTICS.

SHOW RECORDLOCKS

Syntax **SHOW RECORDLOCKS**

Privileges None

Description SHOW RECORDLOCKS returns a list of locked records as a RecordSet.

The fields in the RecordSet are as follows:

Field Name	Description
DatabaseName	The name of the database with the locked record.
ConnectionID	The connectionID of the user who has the lock.
TableName	The name of the table that contains the locked record.
RowNumber	The rownumber of the locked record.

Example

This example returns a RecordSet containing all the locked records.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 Dim rs as New RecordSet
 rs=db.SQLSelect("SHOW RECORDLOCKS")
 if rs <> Nil then
  DisplayRecordSet rs
 Else
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
            " (" + Format(db.ErrorCode, "#") + ")"
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

LOCK RECORD, UNLOCK RECORD

SHOW RESTORE LOG FOR DATABASE

Syntax

SHOW RESTORE LOG FOR DATABASE *value*

Privileges

Restore

Description

SHOW RESTORE LOG FOR DATABASE returns a Recordset with the following fields. The fields in the RecordSet are as follows:

Field Name	Description
ID	The ID of the restore.
datetime	The SQL datetime of the restore.
tablename	The name of the table that was restored.
sql	The sql statement.

Example

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 Dim rs as New RecordSet
 rs=db.SQLSelect("SHOW RESTORE LOG FOR DATABASE Images")
 if rs <> Nil then
  DisplayRecordSet rs
 Else
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
            " (" + Format(db.ErrorCode, "#") + ")"
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

SHOW RESTORE LOG FOR DATABASE ON, SHOW RESTORE LOG FOR DATABASE TO, SHOW RESTORE LOG FOR DATABASE.

SHOW RESTORE LOG FOR DATABASE FROM

Syntax

SHOW RESTORE LOG FOR DATABASE *value* FROM *value* [TO *value*]

Privileges

Restore

Description

SHOW RESTORE LOG FOR DATABASE FROM returns a recordset that documents the restores that took place in the specified interval.

The fields in the recordset are as follows:

Field Name	Description
ID	The ID of the restore.
datetime	The SQL datetime of the restore.
tablename	The name of the table that was restored.
sql	The sql statement.

Example

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 Dim rs as New RecordSet
 rs=db.SQLSelect("SHOW RESTORE LOG FOR DATABASE Images FROM"_
                 " 2009-04-17 TO 2009-04-18")
 if rs <> Nil then
  DisplayRecordSet rs
 Else
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
            " (" + Format(db.ErrorCode, "#") + ")"
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

SHOW RESTORE LOG FOR DATABASE ON, SHOW RESTORE LOG FOR DATABASE TO, SHOW RESTORE LOG FOR DATABASE

SHOW RESTORE LOG FOR DATABASE ON

Syntaxes

SHOW RESTORE LOG FOR DATABASE *value* ON *value*

SHOW RESTORE LOG FOR DATABASE *value* ON *value* FROM value

SHOW RESTORE LOG FOR DATABASE *value* ON *value* FROM value TO value

SHOW RESTORE LOG FOR DATABASE *value* ON *value* TO *value*

Privileges

Restore

Description

Example

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 Dim rs as New RecordSet
 rs=db.SQLSelect("SHOW RESTORE LOG FOR DATABASE Images ON"_
                 " 2009-04-17 TO 2009-04-18")
 if rs <> Nil then
  DisplayRecordSet rs
 Else
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
            " (" + Format(db.ErrorCode, "#") + ")"
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

SHOW RESTORE LOG FOR DATABASE, SHOW RESTORE LOG FOR DATABASE TO

SHOW RESTORE LOG FOR DATABASE TO

Syntax

SHOW RESTORE LOG FOR DATABASE *value* TO *value*

Privileges

Restore

Description

SHOW RESTORE LOG FOR DATABASE TO returns a Recordset that contains a record for each restore until the passed date. The fields in the Recordset are as follows.

Field Name	Description
ID	The ID of the restore.
datetime	The SQL datetime of the restore.
tablename	The name of the table that was restored.
sql	The sql statement.

Example

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 Dim rs as New RecordSet
 rs=db.SQLSelect("SHOW RESTORE LOG FOR DATABASE Images TO"_
                 " 2009-04-18")
 if rs <> Nil then
  DisplayRecordSet rs
 Else
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
            " (" + Format(db.ErrorCode, "#") + ")"
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also

SHOW RESTORE LOG FOR DATABASE, SHOW RESTORE LOG FOR DATABASE FROM

SHOW RESTORE STATUS FOR DATABASE

Syntax

SHOW RESTORE STATUS FOR DATABASE *value*

Privileges

Server

Description

SHOW RESTORE STATUS FOR DATABASE returns a status code as a RecordSet. The fields in the RecordSet are as follows:

Field Name	Description
restore	The restore status code. 0 is deactivated; 1 is activated.

Example

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 ListBox1.DeleteAllRows
 Dim rs as New RecordSet
 rs=db.SQLSelect("SHOW RESTORE STATUS FOR DATABASE Images'")
 if rs=Nil then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
         +db.ErrorMessage
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also SHOW RESTORE LOG FOR DATABASE ON, SHOW RESTORE LOG FOR DATABASE TO, SHOW RESTORE LOG FOR DATABASE

SHOW SCHEDULE

Syntax **SHOW SCHEDULE scheduleName**

Privileges ADMIN user

Description SHOW SCHEDULE returns details about the passed schedule as a RecordSet.

The fields in the RecordSet are as follows:

Field Name	Description
Schedname	Name of the schedule.
Scheddays	A list of the days in the week that the schedule will execute (0=Sunday and 6=Saturday).
Schedhours	The hour that the schedule will execute (0 to 23 in local time).
Schedminutes	The minute that the schedule will execute (0-59 in local time).

Field Name	Description
Schedweeks	How often (in weeks) the schedule will execute. "1" means every week, "2" means every second week, "3" means every third week, and so forth.
Schedtype	The type of schedule. The acceptable values are BACKUP, SQL, or SHELL.
Schedoptions	Schedule options, specific to schedule type.
Schedenabled	Equals 1 if the schedule is enabled or 0 if it is not.

Example

This example lists the parameters of the "Images Backup" schedule in a ListBox. Note that the name of the schedule is in single quote marks because it contains a space.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 ListBox1.DeleteAllRows
 Dim rs as New RecordSet
 rs=db.SQLSelect("SHOW SCHEDULE 'Images backup'")
 if rs=Nil then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
         +db.ErrorMessage
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also

SHOW DATABASES FOR SCHEDULE, SHOW SCHEDULES, SHOW SCHEDULES FOR DATABASE.

SHOW SCHEDULES

Syntax

SHOW SCHEDULES

Privileges

ADMIN user

Description

SHOW SCHEDULES returns a list of all schedules known to the server as a RecordSet.

The fields in the RecordSet are as follows:

Field Name	Description
Schedname	The name of the schedule.

Example

The following example lists the names of the schedules in a ListBox.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

if db.Connect then
 ListBox1.DeleteAllRows
 Dim rs as New RecordSet
 rs=db.SQLSelect("SHOW SCHEDULES")
 if rs=Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
           " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also

SHOW SCHEDULE, SHOW SCHEDULES FOR DATABASE.

SHOW SCHEDULES FOR DATABASE

Syntax

SHOW SCHEDULES FOR DATABASE *database*

Privileges

ADMIN user

Description

SHOW SCHEDULES returns a list of all the schedules attached to the passed database as a RecordSet.

The fields in the RecordSet are as follows:

Field Name	Description
Schedname	The name of the schedule.

Example

This example returns the schedules that are attached to the database "Images" via the ATTACH SCHEDULE command or via the Admin application.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

if db.Connect then
 Dim rs as New RecordSet
 rs=db.SQLSelect("SHOW SCHEDULES FOR DATABASE Images")
 If db.error then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
      +db.ErrorMessage
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also

SHOW DATABASES FOR SCHEDULE, SHOW SCHEDULE, SHOW SCHEDULES.

SHOW SERVER STATUS

Syntax

SHOW SERVER STATUS

Privileges

ADMIN user

Description

SHOW SERVER STATUS returns an indicator of whether the server is running or stopped as a RecordSet.

The fields in the RecordSet are as follows:

Field Name	Description
Status	The status of the server. Either "Running" or "Stopped", depending on the server's status.

Example This example returns the server status in a RecordSet with one record and one column.

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 ListBox1.DeleteAllRows

 Dim rs as RecordSet
 rs=db.SQLSelect("SHOW SERVER STATUS")
 If rs = nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
                " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also PING, SHOW CONNECTIONS, SHOW INFO.

SHOW TABLES FOR DATABASE

Syntax **SHOW TABLES FOR DATABASE *database***

Privileges PRIVILEGES

Description SHOW TABLES FOR DATABASE returns a list of tables for the passed database without having to call USE DATABASE. It returns a RecordSet with one field, tablename.

Example The following example gets the list of tables for the passed database and displays the tables in a ListBox.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 ListBox1.DeleteAllRows

 Dim rs as RecordSet
 rs=db.SQLSelect("SHOW TABLES FOR DATABASE Images")
 If rs = Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
      " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed"
End if
```

See Also SHOW DATABASES, SHOW DATABASES WITH DETAILS, SHOW LASTROWID, SHOW STATISTICS, SHOW TABLES.

SHOW TABLES

Syntax **SHOW TABLES**

Privileges PRIVILEGES

Description SHOW TABLES returns a list of tables for a database. The database must be specified with a prior call to USE DATABASE. It returns a RecordSet with one field, tablename. It provides the same functionality as SHOW TABLES FOR DATABASE, but requires the call to USE DATABASE.

Example

The following example gets the list of tables in the database specified by the USE DATABASE statement.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 ListBox1.DeleteAllRows
 db.SQLExecute("USE DATABASE Images")
 Dim rs as RecordSet
 rs=db.SQLSelect("SHOW TABLES")
 If rs = Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
      " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed"
End if
```

See Also

SHOW DATABASES, SHOW DATABASES WITH DETAILS, SHOW LASTROWID, SHOW STATISTICS, SHOW TABLES FOR DATABASE.

SHOW TOTAL CHANGES

Syntax

SHOW TOTAL CHANGES

Privileges

None

Description

SHOW TOTAL CHANGES returns a RecordSet with one field, "changes." SHOW TOTAL CHANGES returns the number of database rows that were changed, inserted, or deleted by the most recently completed SQL statement on the database connection, plus changes caused by triggers.. Use SHOW CHANGES to find the total number of changes excluding changes caused by triggers.

A "row change" is a change to a single row of a single table caused by an INSERT, DELETE, or UPDATE statement. Rows that are changed as side effects of REPLACE constraint resolution, rollback, ABORT processing, DROP TABLE, or by any other mechanisms do not count as direct row changes.

A “trigger context” is a scope of execution that begins and ends with the script of a trigger. Most SQL statements are evaluated outside of any trigger. This is the “top level” trigger context. If a trigger fires from the top level, a new trigger context is entered for the duration of that one trigger. Subtriggers create subcontexts for their duration.

Example

```
db=New REALSQLServerDatabase
db.Host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.connect then
 db.SQLExecute("USE DATABASE IMAGES")
 Dim rs as RecordSet
 rs=db.SQLSelect("SHOW TOTAL CHANGES")
 If rs = Nil then
  MsgBox "Unable to get list from server: " + db.ErrorMessage + _
                " (" + Format(db.ErrorCode, "#") + ")"
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also

SHOW CHANGES. USE DATABASE.

SHOW USERS

Syntax

SHOW USERS

Privileges

PRIVILEGES

Description

SHOW USERS returns all the users on the server as a RecordSet. This is the list of users that have been created on the server, not the list of currently logged-in users.

The fields in the RecordSet are as follows:

Field Name	Description
Username	The name of a user.

Field Name	Description
Password	The user's password.
Groupname	The group the user is in. If a user is in more than one group, use SHOW GROUPS FOR USER to get all the groups.

Example

This example gets the list of all the users on the server and displays it in a ListBox.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 ListBox1.DeleteAllRows

 Dim rs as New RecordSet
 rs=db.SQLSelect("SHOW USERS")
 If rs=Nil then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned."_
              +db.ErrorMessage
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also

RENAME USER, SHOW GROUPS, SHOW GROUPS FOR USER, SHOW USERS IN GROUP.

SHOW USERS IN GROUP

Syntax

SHOW USERS IN GROUP *group*

Privileges

PRIVILEGES

Description

SHOW USERS IN GROUP returns the users in the passed group as a RecordSet.

The fields in the RecordSet are as follows:

Field Name	Description
Username	The name of a user.

Field Name	Description
Password	The user's password.

Example

The following example gets the list of users in the group "Managers."

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 ListBox1.DeleteAllRows
 Dim rs as New RecordSet
 rs=db.SQLSelect("SHOW USERS IN GROUP Managers")
 if rs=Nil then
  MsgBox "An error code of "+Str(db.ErrorCode)+" was returned. "_
            +db.ErrorMessage
  Return
 End if
 DisplayRecordSet rs
Else
 MsgBox "Connection failed."
End if
```

See Also

RENAME USER, SHOW GROUPS, SHOW GROUPS FOR USER, SHOW USERS.

START DATABASE

Syntax

START DATABASE *database*

Privileges

ADMIN user

Description

START DATABASE starts a stopped database. A database on a running server can be stopped by the STOP DATABASE command.

Example This example starts a database on the server.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
db.SQLExecute("START DATABASE Images")
If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+ _
     " was returned with message "+db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also SHOW DATABASES, SHOW DATABASES WITH DETAILS, STOP DATABASE.

STOP DATABASE

Syntax **STOP DATABASE *database***

Privileges ADMIN user

Description STOP DATABASE stops a running database. It can be restarted with START DATABASE.

Example This example stops a database on the server.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("STOP DATABASE Images")
 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+ _
            " was returned with message "+db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also SHOW DATABASES, SHOW DATABASES WITH DETAILS, START DATABASE.

UNLOCK DATABASE

Syntax **UNLOCK DATABASE *database***

Privileges N/A.

Description UNLOCK DATABASE unlocks a previously locked database. This assumes that you have previously called USE DATABASE to get access to the database. UNLOCK DATABASE will fail unless you have previously called USE DATABASE and then locked it with LOCK DATABASE.

If you need to unlock a database that you are not using, use UNLOCK DATABASE WITH FORCE.

Call UNLOCK DATABASE after a call to LOCK DATABASE when you no longer need exclusive write access.

Example This example unlocks a previously locked database.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("USE DATABASE Images")
 db.SQLExecute"LOCK DATABASE Images")
  //you have exclusive write access here...
 db.SQLExecute ("UNLOCK DATABASE Images")

 if db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+ _
       " was returned with message "+db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also LOCK DATABASE, UNLOCK DATABASE WITH FORCE, USE DATABASE.

UNLOCK DATABASE WITH FORCE

Syntax **UNLOCK DATABASE *database* WITH FORCE**

Privileges ADMIN user

Description The WITH FORCE option for UNLOCK DATABASE enables a user with ADMIN privileges to unlock a database that he is not currently using. Without the WITH FORCE option, UNLOCK DATABASE must be preceded by the USE DATABASE command.

Example This example unlocks a previously locked database that is in use by another user.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("UNLOCK DATABASE Images WITH FORCE")
 if db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+ _
       " was returned with message "+db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also LOCK DATABASE, UNLOCK DATABASE, USE DATABASE.

UNLOCK READ DATABASE

Syntax **UNLOCK READ DATABASE *value***

Privileges Admin user and whoever issued the LOCK RECORD command.

Description UNLOCK RECORD unlocks the record with the passed rowid for the table with the passed name. Call UNLOCK RECORD after a call to LOCK RECORD as soon as you no longer need exclusive write access to a locked record.

Example The following example unlocks a record after it was previously locked with LOCK RECORD.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
  db.SQLExecute("LOCK READ DATABASE images")
//you have blocked write access to other users here...
 db.SQLExecute("UNLOCK READ DATABASE images")

 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+ _
       " was returned with message "+db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also LOCK RECORD, LOCK READ DATABASE, USE DATABASE.

UNLOCK RECORD

Syntax **UNLOCK RECORD *rowid* ON TABLE *tableName***

Privileges Admin user and whoever issued the LOCK RECORD command.

Description UNLOCK RECORD unlocks the record with the passed rowid for the table with the passed name. Call UNLOCK RECORD after a call to LOCK RECORD as soon as you no longer need exclusive write access to a locked record.

Example The following example unlocks a record after it was previously locked with LOCK RECORD.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("USE DATABASE Images")
 db.SQLExecute("LOCK RECORD 8 ON TABLE images")
//you have blocked write access to other users here...
 db.SQLExecute("UNLOCK RECORD 8 ON TABLE images")

 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+ _
       " was returned with message "+db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also LOCK RECORD, USE DATABASE.

UNLOCK WRITE DATABASE

Syntax **UNLOCK WRITE DATABASE** ***value***

Privileges Admin user and whoever issued the LOCK WRITE DATABASE command.

Description UNLOCK RECORD unlocks the record with the passed rowid for the table with the passed name. Call UNLOCK WRITE DATABASE after a call to LOCK WRITE DATABASE as soon as you no longer need exclusive write access to a locked record.

Example The following example unlocks a record after it was previously locked with LOCK RECORD.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
  db.SQLExecute("LOCK WRITE DATABASE images")
//you have blocked write access to other users here...
 db.SQLExecute("UNLOCK WRITE DATABASE images")

 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+ _
       " was returned with message "+db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also LOCK WRITE DATABASE, USE DATABASE.

UPDATE SCHEDULE

Syntax **UPDATE SCHEDULE *schedName* SET DAYS='*days*', HOURS=*hours*, MINUTES=*minutes*, WEEKS=*weeks*, TYPE='*type*', OPTIONS='*options*', ENABLED=*enabled***

Privileges ADMIN user

Description UPDATE SCHEDULE updates a schedule with the passed name. Currently, only backup operations can be scheduled. Here are descriptions of the parameters that UPDATE SCHEDULE requires:

Field Name	Description
schedName	Name of the schedule.

Field Name	Description
days	A list of the days in the week that the schedule will execute (0=Sunday and 6=Saturday). If you want to execute the schedule on more than one day in the week, then the day numbers are not separated by any delimiter. For example "03" specifies Sunday and Wednesday.
hours	The hour that the schedule will execute (0 to 23 in local time).
minutes	The minute that the schedule will execute (0-59 in local time).
weeks	How often (in weeks) the schedule will execute. "1" means every week, "2" means every second week, "3" means every third week, and so forth. The range is from 0 to 53.
type	The type of schedule. The acceptable values are BACKUP, SQL, or SHELL. If type is Backup, the schedule sets up a database backup. If type is SQL, it schedules a SQL command to execute. If type is SHELL, then it schedules a shell script to run.
options	Schedule options. There are different types of options for each schedule type. If type is BACKUP, the valid options are either RETAIN OLD or NOT RETAIN. If type is SQL, then the Options parameter contains the SQL commands that you want to execute. For example, "VACUUM" performs a vacuum according to the schedule. If type is SHELL, then the options parameter is the full path to the application to execute.
enabled	Equals 1 if the schedule is enabled or 0 if it is not.

Example The following example updates the schedule "myBackup."

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("USE DATABASE Images")
 db.SQLExecute("UPDATE SCHEDULE myBackup SET DAYS='0', HOURS=0, "_
    +"MINUTES=0, WEEKS=1, TYPE='BACKUP', OPTIONS='RETAIN OLD',"_
    +" ENABLED=1")

 If db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+" was returned. "_
         +db.ErrorMessage
  Return
 End if
Else
 MsgBox "Connection failed."
End if
```

See Also CREATE SCHEDULE, ATTACH SCHEDULE, DETACH SCHEDULE, SHOW DATABASES FOR SCHEDULE, SHOW SCHEDULE, SHOW SCHEDULES, SHOW SCHEDULES FOR DATABASE.

UPDATE STATISTICS

Syntax **UPDATE STATISTICS**

Privileges Admin

Description UPDATE STATISTICS causes an immediate update to server statistics. They can be viewed by calling SHOW REALTIME STATISTICS.

Example

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("UPDATE STATISTICS")
 dim rs as RecordSet
 rs=db.SQLSelect("SHOW REALTIME STATISTICS")
 if rs=nil then
   MsgBox "An error code of "+str(db.ErrorCode)+ _
      " was returned with message "+db.ErrorMessage
   Return
 End if
DisplayRecordSet rs
 if db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+ _
      " was returned with message "+db.ErrorMessage
  Return
 End if
//do stuff with the database here
Else
 MsgBox "Connection failed."
End if
```

See Also

SHOW REALTIME STATISTICS.

UPLOAD DATABASE

Syntax

UPLOAD DATABASE *database* [WITH REPLACE]

Privileges

UPLOAD

Description

UPLOAD DATABASE installs the passed database into the "databases" folder. This function is equivalent to the Upload Database menu command in the Admin application.

A call to UPLOAD DATABASE presents an open-file dialog box in which the user can choose the database to upload. After a successful call, a copy of the database is placed in the server's "databases" folder.

If the optional WITH REPLACE keyword is specified, then any existing database with the same name will be replaced. If WITH REPLACE is omitted, then uploading a database with the same name as an existing database will generate an error.

Currently, the Admin application's Upload Database menu command always uploads databases without replacement. If you want to replace a database with Admin instead of UPLOAD DATABASE, first drop the database, and then upload the new database.

The Examples folder includes a REALbasic application that illustrates uploading and downloading databases.

Example

The following example installs the selected database in the "databases" folder.

```
//mDatabase is a REALSQLServerDatabase property of the window
If not mDatabase.IsConnected then
 MsgBox "You are not connected to a server. Connect first before"_
        +"attempting an upload or download."
 Return
End if

Dim dbFile as FolderItem = GetOpenFolderItem("Any")
If dbFile = nil then
 Return
End if

  // Sanity check by trying to open the file as a REAL SQL Database
  // (if the file isn't a REAL SQL Database, then the server can't serve it)
Dim db as New REALSQLDatabase
db.DatabaseFile = dbFile
If not db.Connect then
 MsgBox "The file you selected does not appear to be a REAL SQL"_
        +"Database file."
 Return
End if

  // close the REAL SQL Database (we need to do this explicitly so that
  // the open database doesn't interfer with the upload process)
db.Close

  // try to initiate an upload (notice how we quote the database name
  // with single quotes in case the name contains spaces)
mDatabase.SQLExecute "UPLOAD DATABASE '" + dbFile.Name + "'"
if mDatabase.Error then
 MsgBox "SQLExecute error: " + mDatabase.ErrorMessage + "("+"_
          +"Str(mDatabase.ErrorCode) + ")"
 Return
End if
```

```
 // open the file as a BinaryStream
Dim bs as BinaryStream = dbFile.OpenAsBinaryFile

 // upload the file in chunks
While not bs.EOF
  // read the next chunk from the file
 Dim chunk as String = bs.Read(kChunkSize)

  // send the chunk to the server
 mDatabase.SendChunk chunk

  // if there was an error; report it and bail
 If mDatabase.Error then
  MsgBox "SendChunk error: " + mDatabase.ErrorMessage + "("+"_
          +Str(mDatabase.ErrorCode) + ")"
  Return
 End if

  // set the progress bar by the length of the chunk
 StepProgressBar chunk.Len
Wend

 // Send the stop end chunk command (the server needs this
 // to know that the file has been completely sent)
mDatabase.SendEndChunk

 // inform the user the upload succeeded
MsgBox "Upload completed."
```

See Also DOWNLOAD DATABASE.

USE DATABASE

Syntax **USE DATABASE *database***

Privileges DATABASE with any privilege.

Description USE DATABASE selects a database to use for subsequent operations. Call USE DATABASE prior to calling commands that query or modify the database (e.g., SQL SELECTs and so forth).

Example

This example selects a database on the server to use.

```
db=New REALSQLServerDatabase
db.host="127.0.0.1"
db.port=4430
db.UserName="admin"
db.Password="admin"

If db.Connect then
 db.SQLExecute("USE DATABASE Images")
 if db.error then
  MsgBox "An error code of "+str(db.ErrorCode)+ _
       " was returned with message "+db.ErrorMessage
  Return
 End if
//do stuff with the database here
Else
 MsgBox "Connection failed."
End if
```

See Also

CLEAR CURRENT DATABASE, SHOW DATABASES, SHOW DATABASES WITH DETAILS, START DATABASE, STOP DATABASE.

Appendix A

Error Codes

The following are REAL SQL Server-specific errors. You should never see these errors marked as internal errors. They indicate that something has gone seriously wrong in the internal server engine or in the REALSQLServerPlugin.

Table 1: Error Codes for REAL SQL Server

Error Code	Description
7001	Timeout error.
7002	Socket error.
7003	Memory error (such as out-of-memory).
7004	Database engine is incorrect (internal error).
7005	Wrong number of fields in results data (internal error).
7006	Results data is incorrect (internal error).
7007	Database not found on server.
7008	Authorization failure.
7009	A SQLite error has occurred.
7010	Something has gone wrong with an execute statement (internal error).
7011	An error has occurred running the SQLite virtual machine (internal error).
7012	Client and server using different protocol versions.
7013	Compression error.

Error Code	Description
7014	Encryption error.
7015	Unknown error.
7016	Server command error (such as a syntax error).
7017	Incorrect username.
7018	Incorrect password.
7019	The maximum database limit has been reached.
7020	The maximum connections limit has been reached.
7021	Insufficient privileges to complete command (read-only client).
7022	Insufficient privileges to complete command.
7023	Error with SELECT statement (internal error).
7024	Record already locked.
7025	A SELECT was issued in a SQLExecute, or a non-SELECT was issued in a SQLSelect.
7026	Syntax error.
7027	Database is stopped.
7028	Database already exists.
7029	Database in use.
7030	Database not yet set (need to call USE DATABASE first).
7031	SQLite version error (internal error).
7032	SQL statement is empty.
7033	Unknown user.
7034	Unknown group.
7035	Couldn't create user.
7036	Couldn't drop user.
7037	Duplicate user (user already exists).
7038	Command not yet implemented.
7039	Couldn't create database.
7040	Commands were given in the wrong sequence.
7041	Duplicate group (group already exists).
7042	Couldn’t open database.
7043	Couldn’t read server information.
7044	Couldn't drop group.
7045	Unknown privilege.
7046	Incomplete SQL statement.
7047	Incorrect arguments to command.

Error Code	Description
7048	Client disconnected.
7049	Wrong header.
7050	Input/Output error.
7051	Rename command error.
7052	Demo time expired.
7053	Key needed.
7054	Key wrong.
7666	Lock timeout.

The following are the SQLite errors that can be returned by the server.

Table 2: SQLite error codes returned by the server.

Error Code	Description
0	No error.
1	SQL error or missing database.
3	Access permission denied.
4	Callback routine requested an abort.
5	The database file is locked.
6	A table in the database is locked.
7	Out of memory.
8	Attempt to write to a read-only database.
10	Some kind of disk I/O error occurred.
11	The database disk image is malformed.
13	Insertion failed because database is full (usually means the disk is full).
14	Unable to open the database file.
15	Database lock protocol error.
16	Database is empty.
17	The database schema changed.
19	Abort due to constraint violation.
20	Data type mismatch.
21	Library used incorrectly.
22	Uses OS features not supported on host.
23	Authorization denied.
24	Auxiliary database format error.
26	File opened that is not a database file.

Alphabetical Command Index

Commands by Theme

Backups

Databases

Encryption

Logs

Plugins

Preferences

Privileges

Records

Schedules

Server

Users & Groups

User's Guide Index