cubeSQL

Commands Reference

version 5.7.0

INTRODUCTION

In addition to the standard SQL statements supported by sqlite, cubeSQL also understands a number of server-specific commands. This manual lists those commands and describes what each command does.

In general, commands that begin with SHOW are intended to query the server for information and the server returns the information as a 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 with Xojo:

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

(This assumes that "db" is a cubeSQLServer 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.sqlite")
```

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 manual 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).

Please note that all the examples in this manual uses the Xojo language for simplicity, but all the commands can be executed using the C SDK or the JSON protocol.

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.

Listing the Contents of a RecordSet with Xojo

The example code for commands that return a RecordSet call the method "DisplayRecordSet" to display the records in the RecordSet in a ListBox named ListBox1. 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
    ListBox1.DeleteAllRows
    ListBox1.HasHeading = true
    ListBox1.ColumnCount = rs.FieldCount

    // show header
    for i=1 to rs.FieldCount
        ListBox1.Heading(i-1) = rs.IdxField(i).Name
    next

    // show records
    While Not rs.eof
        ListBox1.AddRow ""
        for i=1 to rs.FieldCount
            ListBox1.Cell(List.LastIndex, i-1) = rs.IdxField(i).getString
        next
    rs.MoveNext
    Wend

End Sub
```

USERS & GROUPS

ADD USER TO GROUP

Syntax
ADD USER username TO GROUP groupname

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 "Marco" to the group "Developers".

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("ADD USER 'Marco' TO GROUP 'Developers';")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

CREATE GROUP

Syntax
CREATE GROUP groupname

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 CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("CREATE GROUP 'Engineers';")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

CREATE USER

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

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.
In the WITH HASH PASSWORD variant, the password parameter is not sent in the clear but it must be the SHA1 of the password to use.

Example
This example creates the user "marco" with no password set, then the user "giuly" with a clear password and finally the user "moki" with an hash password.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("CREATE USER 'marco';")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

db.SQLExecute("CREATE USER 'giuly' WITH PASSWORD 'mypass';")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

// SHA1 is a Xojo external plug-in which must be installed
Dim myPass as String = EncodeBase64(SHA1(SHA1("summit")))
db.SQLExecute("CREATE USER 'moki' WITH HASH PASSWORD " + myPass)
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

DROP GROUP

Syntax
DROP GROUP groupname

Privileges
PRIVILEGES

Description
DROP GROUP deletes the group with the passed name.

Example
This example deletes the group "Engineers".

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("DROP GROUP 'Engineers';")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

DROP USER

Syntax
DROP USER username

Privileges
PRIVILEGES

Description
DROP USER deletes the passed user.

Example
This example drops the user "marco".

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("DROP USER 'marco';")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

MOVE USER TO GROUP

Syntax
MOVE USER username TO GROUP groupname

Privileges
Privileges

Description
MOVE USER TO GROUP moves the passed user to the passed group.

Example
This example move the user "marco" to the group "engineers".

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("MOVE USER 'marco' TO GROUP 'engineers';")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

REMOVE USER

Syntax
REMOVE USER username FROM GROUP groupname

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 "marco" from the group "engineers".

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("REMOVE USER 'marco' FROM GROUP 'engineers';")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

RENAME GROUP

Syntax
RENAME GROUP oldGroupName 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 "engineers" to "developers".
The group "engineers " has already been created on the server.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("RENAME GROUP engineers TO developers;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

RENAME 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 "marco".
This user has already been created on the server.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("RENAME USER marco TO sqlabs;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

SET [HASH] PASSWORD FOR USER

Syntax
SET PASSWORD password FOR USER username
SET HASH PASSWORD password FOR USER username

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 clear password for the new user "marco" and also sets a hash password for the user "giuly".

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("SET PASSWORD 'test' FOR USER marco;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

Dim myPass as String = EncodeBase64(SHA1(SHA1("summit")))
db.SQLExecute("SET HASH PASSWORD " + myPass + " FOR USER giuly;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

SET MY [HASH] PASSWORD

Syntax
SET MY PASSWORD TO newPassword
SET MY HASH PASSWORD TO newPassword

Privileges
NONE

Description
SET MY HASH PASSWORD or SET MY PASSWORD sets the password for the current user. Password must be set in clear or in hashed version.

Example
This example sets a dummy password for myself.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("SET MY PASSWORD TO 'X79biff';")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

Dim myPass as String = EncodeBase64(SHA1(SHA1("X79biff")))
db.SQLExecute("SET MY HASH PASSWORD TO " + myPass)
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

SHOW GROUPS

Syntax
SHOW GROUPS

Privileges
PRIVILEGES

Description
SHOW GROUPS returns a list of all the groups on the server as a RecordSet.
Returned RecordSet contains just the groupname column.

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

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW GROUPS;")
DisplayRecordSet (rs)
```

SHOW GROUPS FOR USER

Syntax
SHOW GROUPS FOR USER username

Privileges
PRIVILEGES

Description
SHOW GROUPS FOR USER returns the groups the passed user is in as a RecordSet. Returned RecordSet contains just the groupname column.

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

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW GROUPS FOR USER Dave;")
DisplayRecordSet (rs)
```

SHOW MY GROUPS

Syntax
SHOW MY GROUPS

Privileges
NONE

Description
SHOW MY GROUPS returns the name of the groups I am in as a recordset.

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW MY GROUPS;")
DisplayRecordSet (rs)
```

SHOW MY USERNAME

Syntax
SHOW MY USERNAME

Privileges
NONE

Description
SHOW MY USERNAME returns current client username in as a recordset.

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW MY USERNAME;")
DisplayRecordSet (rs)
```

SHOW USERS [IN GROUP]

Syntax
SHOW USERS
SHOW USERS IN GROUP groupname

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.
RecordSet contains the username, password and groupname columns.
If a user is in more than one group, use SHOW GROUPS FOR USER to get all the groups.
To get the users in a specific group use the SHOW USERS IN GROUP command.

Example
This example gets the list of all the users on the server and displays it in a ListBox then ask for all the users in the "engineers" group.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW USERS;")
DisplayRecordSet (rs)

Dim rs2 as RecordSet = db.SQLSelect("SHOW USERS IN GROUP engineers;")
DisplayRecordSet (rs2)
```

PRIVILEGES

GRANT

Syntax
GRANT privilege TO GROUP groupname
GRANT privilege TO GROUP groupname FOR DATABASE database
GRANT privilege TO GROUP groupname FOR TABLE table IN DATABASE database

Privileges
PRIVILEGES

Description
GRANT grants the group the passed server privilege. That privilege can be restricted to a specific database or to a specific table inside a tables based on the syntax of the used command.

The following privileges can be granted:

Privilege	Description
ADMIN	Super user privileges.
DATABASES	Can create and drop databases.
PRIVILEGES	Can edit user privileges.
PREFERENCES	Can see and edit server preferences.
CREATE	Can create tables, indexes, view, triggers (can also alter tables).
DROP	Can drop tables, indexes, view, triggers.
SELECT	Can execute query.
INSERT	Can insert records.
UPDATE	Can update records.
DELETE	Can delete records.
UPLOAD	Can upload databases to the server.
DOWNLOAD	Can download databases from the server.
PRAGMA	Can execute special sqlite pragma operations.
SERVER	Can execute server's tasks.
RESTORE	Can enable/disable/execute restore operations.
BACKUP	Can manage backups.
PLUGIN	Can enable/disable plugins.

Example
This example grants DATABASES privileges to the group "developers" and also grants CREATE privileges to the "engineers" group for the "Images" database and grants SELECT privileges to the "managers" group for the table "data" in the database "Images".

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
```

```
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("GRANT DATABASES TO GROUP developers;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

db.SQLExecute("GRANT CREATE TO GROUP engineers FOR DATABASE images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

db.SQLExecute("GRANT SELECT TO GROUP managers FOR TABLE data IN DATABASE
                images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

REVOKE

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

Syntax
REVOKE privilege FROM GROUP groupname
REVOKE privilege FROM GROUP groupname FOR DATABASE database
REVOKE privilege FROM GROUP groupname FOR TABLE table IN DATABASE database

Privileges
PRIVILEGES

Description
This syntax revokes privileges from the passed group.
See GRANT for a list of all possible privileges that can be revoked. A special ALL privilege has been added in order to be able to revoke

Example
This example removes all the privileges from the group "developers".

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("REVOKE ALL FROM GROUP developers;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

SHOW ALL PRIVILEGES

Syntax
SHOW ALL PRIVILEGES

Privileges
NONE

Description
SHOW ALL PRIVILEGES returns a recordset that report all privileges defined into the server. The fields in the RecordSet are databasename, groupname, privilege and tablename. (A "*" indicates all databases or all tables).

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW ALL PRIVILEGES;")
DisplayRecordSet (rs)
```

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.

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW MY PRIVILEGES;")
DisplayRecordSet (rs)
```

SHOW PRIVILEGES FOR GROUP

Syntax
SHOW PRIVILEGES FOR GROUP groupname

Privileges
PRIVILEGES

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

Example
The following example gets the privileges for the "developers" group.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW PRIVILEGES FOR GROUP developers;")
DisplayRecordSet (rs)
```

SHOW PRIVILEGES TABLE

Syntax
SHOW PRIVILEGES TABLE

Privileges
NONE

Description
SHOW PRIVILEGES TABLE returns the privileges table that can be defined on the server.

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW PRIVILEGES TABLE;")
DisplayRecordSet (rs)
```

SCHEDULES

ATTACH SCHEDULE

Syntax
ATTACH SCHEDULE name TO DATABASE database

Privileges
SERVER

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 CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("ATTACH SCHEDULE myBackup TO DATABASE Images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

CREATE SCHEDULE

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

Privileges
SERVER

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.

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 CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

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 occurred: " + db.ErrorMessage
```

DETACH SCHEDULE FROM DATABASE

Syntax
DETACH SCHEDULE name FROM DATABASE database

Privileges
SERVER

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 CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("DETACH SCHEDULE myBackup FROM DATABASE Images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

DROP SCHEDULE

Syntax
DROP SCHEDULE name

Privileges
SERVER

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 CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("DROP SCHEDULE myBackup")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

RENAME SCHEDULE

Syntax
RENAME SCHEDULE oldName TO newName

Privileges
SERVER

Description
RENAME SCHEDULE renames the passed old scheduleName with the new newScheduleName. The schedule retains all their settings.

Example
This example rename the schedule "myBackup" that was previously created with CREATE SCHEDULE to a new "newBackup" name.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("RENAME SCHEDULE myBackup TO newBackup;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

RESET SCHEDULE

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

Privileges
SERVER

Description
RESET SCHEDULE updates a schedule with the passed name. See the CREATE SCHEDULE command for a list of parameters that can be set.

Example
The following example updates the schedule "myBackup".

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("RESET 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 occurred: " + db.ErrorMessage
```

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 CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW DATABASES FOR SCHEDULE myBackup;")
DisplayRecordSet (rs)
```

SHOW ID FOR SCHEDULE

Syntax
SHOW ID FOR SCHEDULE scheduleName

Privileges
SERVER

Description
SHOW ID FOR SCHEDULE returns a recordset with one field, schedule. This field is used to uniquely identify schedule.

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

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW ID FOR SCHEDULE myBackup;")
DisplayRecordSet (rs)
```

SHOW SCHEDULE

Syntax
SHOW SCHEDULE scheduleName

Privileges
SERVER

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).
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.
schedid	Unique schedule ID.

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 CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if
```

```
Dim rs as RecordSet = db.SQLSelect("SHOW SCHEDULE 'Images backup';")
DisplayRecordSet (rs)
```

SHOW SCHEDULES

Syntax
SHOW SCHEDULES

Privileges
SERVER

Description
SHOW SCHEDULES returns a list of all schedules known to the server as a RecordSet. The fields in the RecordSet are schedname and schedid.

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

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW SCHEDULES;")
DisplayRecordSet (rs)
```

SHOW SCHEDULES FOR DATABASE

Syntax
SHOW SCHEDULES FOR DATABASE database

Privileges
SERVER

Description
SHOW SCHEDULES returns a list of all the schedules attached to the passed database as a RecordSet with one schedname column.

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 CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW SCHEDULES FOR DATABASE Images;")
DisplayRecordSet (rs)
```

BACKUP

BACKUP

Syntax
BACKUP NOW databaseName
BACKUP ASYNC databaseName

Privileges
BACKUP

Description
BACKUP NOW backs up the passed database immediately. BACKUP NOW always attempts to retain old backups. Use the ASYNC command if you want to just start a backup process without waiting for the entire operation to complete.
To be unique backup name will be in the form: databaseName_YYYYMMDD_HHMMSS.

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

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("BACKUP NOW Images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

BACKUP SETTINGS

Syntax
BACKUP SETTINGS

Privileges
ADMIN

Description
BACKUP SETTINGS makes a backup of the settings file.
To be unique backup name will be in the form: cubesql.settings_YYYYMMDD_HHMMSS.

Example
The following example backs up the server's settings.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("BACKUP SETTINGS;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

DOWNLOAD BACKUP DATABASE WITH TIMESTAMP

Syntax
DOWNLOAD BACKUP DATABASE databaseName WITH TIMESTAMP timeStampValue

Privileges
DOWNLOAD

Description
Download a backup database specified by its name and its timestamp values. A list of backups for a given database can be retrieved using the SHOW BACKUPS FOR DATABASE command.

Example
Code is the same as the DOWNLOAD DATABASE example.

DROP BACKUP FOR DATABASE

Syntax
DROP BACKUP FOR DATABASE databaseName
DROP BACKUP FOR DATABASE databaseName WITH TIMESTAMP value

Privileges
BACKUP

Description
DROP BACKUP FOR DATABASE deletes the passed backup name.
DROP 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 the backup for the Images database.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("DROP BACKUP FOR DATABASE Images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

RESTORE BACKUP FOR DATABASE

Syntax
RESTORE BACKUP FOR DATABASE databaseName 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 CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("RESTORE BACKUP FOR DATABASE Images WITH TIMESTAMP
                    20110428_122215;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

SHOW BACKUPS FOR DATABASE

Syntax
SHOW BACKUPS FOR DATABASE databaseName

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 CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW BACKUPS FOR DATABASE Images;")
DisplayRecordSet (rs)
```

DATABASES

ATTACH DATABASE

Syntax
ATTACH DATABASE full_path AS dbName

Privileges
NONE

Description
ATTACH DATABASE adds another database file to the currently used database. Full_path must points to the database to attach and MUST BE outside the shared databases folder. dbName is the name that will internally be used to identify that newly attached database inside current connection.

Example
The following code attach a database foo.sqlite to the current bar.sqlite database.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = “admin"
db.DatabaseName = “bar.sqlite”

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("ATTACH DATABASE /user/marco/foo.sqlite AS foo;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

CLOSE DATABASE

Syntax
CLOSE DATABASE dbName

Privileges
NONE

Description
CLOSE DATABASE completely unload a database from cubeSQL. Database cannot be in use by someone else.

Example
The following code close and unload a database.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("CLOSE DATABASE myDatabase;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

CREATE DATABASE

Syntax
CREATE DATABASE dbName
CREATE DATABASE dbName IF NOT EXISTS
CREATE DATABASE dbName WITH ENCODING encodingValue
CREATE DATABASE dbName WITH ENCODING encodingValue IF NOT EXISTS
CREATE DATABASE dbName WITH KEY keyValue
CREATE DATABASE dbName WITH KEY keyValue IF NOT EXISTS
CREATE DATABASE dbName WITH KEY keyValue WITH ENCODING encodingValue
CREATE DATABASE dbName WITH KEY keyValue WITH ENCODING encodingValue 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.

You can supply additional parameters to the command like the WITH KEY keyValue one and in this case the server creates a new encrypted database with the passed name and encryption key.

If you specify the WITH ENCODING encodingValue parameter than you can supply an additional encoding value for the newly created database (default value is UTF-8) and other supported values are: "UTF-16", "UTF-16le" (little-endian UTF-16 encoding) and "UTF-16be" (big-endian UTF-16 encoding).

Example
This example connects to the local server and creates the database "Employees", then the database "Invoices" with a dummy password and a "Chinese" database specifying an UTF-16 encoding.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"
```

```
If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("CREATE DATABASE Employees IF NOT EXISTS;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

db.SQLExecute("CREATE DATABASE Invoices WITH KEY qwerty IF NOT EXISTS;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

db.SQLExecute("CREATE DATABASE Chinese WITH ENCODING UTF-16 IF NOT EXISTS;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

DECRYPT DATABASE

Syntax
DECRYPT DATABASE dbName

Privileges
DATABASES

Description
DECRYPT DATABASE decrypt a previously encrypted database.

Example
The following code decrypts a previously encrypted database.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("DECRYPT DATABASE mySecretDB;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

DETACH DATABASE

Syntax
DETACH DATABASE dbName

Privileges
NONE

Description
DETACH DATABSE detaches an additional database connection previously attached using the ATTACH statement. When not in shared cache mode, it is possible to have the same database file attached multiple times using different names, and detaching one connection to a file will leave the others intact. In shared cache mode, attempting to attach the same database file more than once results in an error.

Example
The following code detach database foo (previously attached to database bar.sqlite).

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = “admin"
db.DatabaseName = “bar.sqlite”

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("ATTACH DATABASE /user/marco/foo.sqlite AS foo;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

db.SQLExecute("DETACH DATABASE foo;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

DROP DATABASE

Syntax
DROP DATABASE dbName
DROP DATABASE dbName 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 CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("DROP DATABASE Employees;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

ENCRYPT DATABASE

Syntax
ENCRYPT DATABASE dbName WITH KEY keyValue

Privileges
DATABASES

Description
ENCRYPT DATABASE encrypts a non encrypted (clear) database using AES128 OFB algorithm.

Example
The following code encrypt a database.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("ENCRYPT DATABASE myDB WITH KEY 'abc';")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

RENAME DATABASE

Syntax
RENAME DATABASE dbName1 TO dbName2

Privileges
DATABASES

Description
RENAME DATABASE just renames dbName1 to a dbName2.

Example
The following code renames myDatabase1 to myDatabase2.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("RENAME DATABASE myDatabase1 TO myDatabase2;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

SET KEY FOR DATABASE

Syntax
SET KEY key FOR DATABASE dbName

Privileges
DATABASE

Description
SET KEY FOR DATABASE sets the encryption key for the passed database. If you have imported/dragged/uploaded an encrypted database into the databases folder, you can set its encryption key it by calling this command. Database MUST BE encrypted in order to be able to use this command.

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

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("SET KEY toad453tm FOR DATABASE images_encypted;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

SET [TEMP] PATH FOR DATABASE

Syntax
SET PATH fullPath FOR DATABASE dbName
SET TEMP PATH fullPath FOR DATABASE dbName

Privileges
NONE

Description
SET PATH FOR DATABASE sets a custom path for database identified by dbName. Each time you'll send a USE DATABASE dbName command, cubeSQL will open the database pointed by the fullPath specified by this command instead of trying to open the database inside the default shared databases folder. if you set the TEMP keyword then this settings will not be preserved between server restarts.

Example
The following example set a custom path for the foo database.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("SET PATH /user/marco/foo.sqlite FOR DATABASE foo;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

SHOW CURRENT DATABASE

Syntax
SHOW CURRENT DATABASE

Privileges
NONE

Description
SHOW CURRENT DATABASE returns a RecordSet with the name of the database set by the USE DATABASE command ("N/A" if no database has been set).

Example
The following code display current database in use

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW CURRENT DATABASE;")
DisplayRecordSet (rs)
```

SHOW DATABASE INFO

Syntax
SHOW DATABASE INFO dbName

Privileges
NONE

Description
SHOW DATABASE INFO returns information about the passed database in a key/value RecordSet. Current keys returned are databasename, nconnections, nwrite, nread and dbsize.

Example
The following example displays information about the Images database.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW DATABASE INFO Images;")
DisplayRecordSet (rs)
```

SHOW DATABASES

Syntax
SHOW DATABASES
SHOW DATABASES WITH DETAILS

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. If you don't specify the WITH DETAILS parameter than you'll just receive a RecordSet with a single databasename column. If you specify the WITH DETAILS parameter than additional columns are returned like stopped as BOOLEAN, locked as INTEGER, lockowner as TEXT, encrypted as BOOLEAN, available as BOOLEAN, restore_status as BOOLEAN and lockownerid as INTEGER.

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

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

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

START DATABASE

Syntax
START DATABASE dbName

Privileges
ADMIN

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 CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("START DATABASE images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

STOP DATABASE

Syntax
STOP DATABASE dbName

Privileges
ADMIN

Description
STOP DATABASE stops a running database. It can be restarted with START DATABASE. A stopped database cannot be server and cannot be used with the USE DATABASE command.

Example
This example stops a database on the server.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("STOP DATABASE images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

UNSET CURRENT DATABASE

Syntax
UNSET CURRENT DATABASE

Privileges
NONE

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

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("UNSET CURRENT DATABASE;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

USE DATABASE

Syntax
USE DATABASE dbName

Privileges
Any privilege on that database.

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 or INSERTs and so forth).

Example
This example selects a database on the server to use.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("USE DATABASE images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

LOCK & UNLOCK

LOCK DATABASE

Syntax
LOCK DATABASE dbName

Privileges
DATABASES

Description
LOCK DATABASE locks the database. Use it when you need exclusive write 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.

Example
This example locks the Images database.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("USE DATABASE images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

db.SQLExecute("LOCK DATABASE images;")
// you have exclusive write access here...
db.SQLExecute("UNLOCK DATABASE images;")
```

LOCK RECORD

Syntax
LOCK RECORD rowid ON TABLE tableName

Privileges
INSERT, UPDATE or DELETE

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 cubeSQL automatically. Call UNLOCK RECORD when you no longer need exclusive write access to the record. When a record is locked, other clients can read it but no other clients can modify it.

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

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("USE DATABASE images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

db.SQLExecute("LOCK RECORD 8 ON TABLE images;")
// you have exclusive write access to that record here...
db.SQLExecute("UNLOCK RECORD 8 ON TABLE images;")
```

LOCK TABLE

Syntax
LOCK TABLE tableName

Privileges
INSERT, UPDATE or DELETE

Description
LOCK TABLE locks the entire table with the passed name. When a table is locked, you have exclusive write access to it. Call UNLOCK TABLE when you no longer need exclusive write access to that table. When a table is locked, other clients can read it but no other clients can modify it.

Example
This example locks table data on database images and then unlocks it.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("USE DATABASE images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

db.SQLExecute("LOCK TABLE data;")
// you have exclusive write access to that table here...
db.SQLExecute("UNLOCK TABLE data;")
```

SHOW LOCKED RECORDS

Syntax
SHOW LOCKED RECORDS

Privileges
NONE

Description
SHOW LOCKED RECORDS returns a RecordSet containing a list of the locked rows and tables on the server. Returned RecordSet contains id, databasename, tablename, recordid and clientid columns (recordid equals to zero means that the entire table is locked).

Example
The following example populates a ListBox with the list of all the locks on the server.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW LOCKED RECORDS;")
DisplayRecordSet (rs)
```

UNLOCK DATABASE

Syntax
UNLOCK DATABASE dbName
FORCE UNLOCK DATABASE dbName

Privileges
DATABASES

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 have not locked use FORCE UNLOCK DATABASE.
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 CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("USE DATABASE images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

db.SQLExecute("LOCK DATABASE images;")
// you have exclusive write access here...
db.SQLExecute("UNLOCK DATABASE images;")
```

UNLOCK RECORD

Syntax
UNLOCK RECORD rowid ON TABLE tableName
UNLOCK RECORD WITH ID value

Privileges
INSERT, UPDATE, DELETE

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. You can also use the WITH ID parameter if you know the unique ID of the locked record (it's the unique id column returned by the SHOW LOCKED RECORDS command).

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

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("USE DATABASE images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

db.SQLExecute("LOCK RECORD 8 ON TABLE images;")
// you have exclusive write access to that record here...
db.SQLExecute("UNLOCK RECORD 8 ON TABLE images;")
```

UNLOCK TABLE

Syntax
UNLOCK TABLE tableName

Privileges
INSERT, UPDATE or DELETE

Description
UNLOCK TABLE unlocks the table with the passed name. Call UNLOCK TABLE when you no longer need exclusive write access to that table. When a table is locked, other clients can read it but no other clients can modify it.

Example
The following example unlocks a table after it was previously locked with LOCK TABLE command.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("USE DATABASE images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

db.SQLExecute("LOCK TABLE data;")
// you have exclusive write access to that table here...
db.SQLExecute("UNLOCK TABLE data;")
```

DATABASE UPLOAD/DOWNLOAD

DOWNLOAD DATABASE

Syntax
DOWNLOAD DATABASE dbName

Privileges
DOWNLOAD

Description
DOWNLOAD DATABASE exports a network copy of a database in the from the shared "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 Xojo application that illustrates uploading and downloading databases.

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

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

// ask the user for a folder to download to
Dim f as FolderItem = SelectFolder
if (f = nil) then return

// initiate the download with the server
db.SQLExecute("DOWNLOAD DATABASE images;")
If db.error then
    MsgBox "An error occurred: " + db.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 = db.ReceiveChunk

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

  // see if we have reached the end of the chunks and exit the loop if we have
  if db.IsEndChunk then
    exit
  end if

  // write the chunk out to the file and loop again
  bs.Write chunk
wend

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

UPLOAD DATABASE

Syntax
UPLOAD DATABASE dbName

Privileges
UPLOAD

Description
UPLOAD DATABASE installs the passed database into the shared "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.
The Examples folder includes a Xojo application that illustrates uploading and downloading databases.

Example
The following example uploads and install the selected local database in the shared "databases" folder.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

// select database to upload
Dim f as FolderItem = GetOpenFolderItem("Any")
if (f = nil) then return

// try to initiate an upload (notice how we quote the database name
// with single quotes in case the name contains spaces)
db.SQLExecute("UPLOAD DATABASE '" + f.Name + "';")
If db.error then
    MsgBox "An error occurred: " + db.ErrorMessage
    return
end if
```

```
 // open the file as a BinaryStream
 dim bs as BinaryStream = f.OpenAsBinaryFile
 if bs = nil then
  return
 end if

 // upload the file in chunks
 while not bs.EOF
  // read the next chunk from the file
  dim chunk as String = bs.Read(100*1024)

  // send the chunk to the server
  db.SendChunk chunk

  // if there was an error; report it and bail
  if db.Error then
   MsgBox "Error receiving chunk from server: " + db.ErrorMessage
   return
  end if

wend

// send the stop end chunk command (the server needs this
// to know that the file has been completely sent)
db.SendEndChunk

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

RESTORE

DISABLE RESTORE ON DATABASE

Syntax
DISABLE RESTORE ON DATABASE dbName

Privileges
SERVER

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

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("DISABLE RESTORE ON DATABASE images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

ENABLE RESTORE ON DATABASE

Syntax
ENABLE RESTORE ON DATABASE dbName

Privileges
SERVER

Description
ENABLE RESTORE ON DATABASE enables the restore functionality to the specified database.

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("ENABLE RESTORE ON DATABASE images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

RESTORE DATABASE

Syntax
RESTORE DATABASE dbName
RESTORE DATABASE dbName TO ID idvalue

Privileges
RESTORE

Description
RESTORE DATABASE full restores the passed database. If you need to restore the database to a specific point you can use the TO ID parameter (idvalue is the id field returned by the SHOW RESTORE LOG commands). RESTORE DATABASE can work only if RESTORE has been enabled on that database.

Example
The following example fully restore the database images1 and restore database image2 until a dummy ID.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("RESTORE DATABASE images1;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

db.SQLExecute("RESTORE DATABASE images2 TO ID 84;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

SHOW RESTORE LOG FOR DATABASE

Syntax
SHOW RESTORE LOG FOR DATABASE dbName
SHOW RESTORE LOG FOR DATABASE dbName FROM date
SHOW RESTORE LOG FOR DATABASE dbName TO date
SHOW RESTORE LOG FOR DATABASE dbName FROM date1 TO date2
SHOW RESTORE LOG FOR DATABASE dbName USERNAME user
SHOW RESTORE LOG FOR DATABASE dbName USERNAME user FROM date
SHOW RESTORE LOG FOR DATABASE dbName USERNAME user TO date
SHOW RESTORE LOG FOR DATABASE dbName USERNAME user FROM date1 TO date2

Privileges
RESTORE

Description
SHOW RESTORE LOG FOR DATABASE returns a Recordset with the following fields:

Field Name	Description
id	The ID of the restore.
datetime	The SQL datetime of the restore.
username	The username of the user who sent the sql statement.
sql	The sql statement.

You can restrict the number of records returned specifying a date (sql format) and or a username. Once you obtains the ID of the restore point you can use the RESTORE DATABASE command in order to restore it.

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW RESTORE LOG FOR DATABASE images;")
DisplayRecordSet (rs)
```

SHOW RESTORE STATUS FOR DATABASE

Syntax
SHOW RESTORE STATUS FOR DATABASE dbName

Privileges
SERVER

Description
If you need to know if RESTORE is enabled or disabled on a specified database, just executes the SHOW RESTORE STATUS FOR DATABASE command and a RecordSet with just one row and one column will be returned. That row will contains 1 if restore is ON otherwise 0.

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW RESTORE STATUS FOR DATABASE images;")
DisplayRecordSet (rs)
```

SECURITY

DISABLE LOGIN

Syntax
DISABLE LOGIN username

Privileges
PRIVILEGES

Description
Use the DISABLE LOGIN command if you need to temporary remove access to the server to a previously defined user. Disable a login doesn't drop the user from the server, it justs disable his possibility to log-in.

Example
The following example DISABLE LOGIN marco on the server.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("DISABLE LOGIN marco;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

ENABLE LOGIN

Syntax
ENABLE LOGIN username

Privileges
PRIVILEGES

Description
Use the ENABLE LOGIN command if you need to re-enable login access to a previously disabled user.

Example
The following example ENABLE a previously disabled user on the server.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("ENABLE LOGIN marco;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

SHOW DISABLED USERS

Syntax
SHOW DISABLED USERS

Privileges
PRIVILEGES

Description
Use the SHOW DISABLED USERS if you need to know a list of all the disabled users defined into the server. The command returns a RecordSet with the username of the users that cannot login.

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW DISABLED USERS;")
DisplayRecordSet (rs)
```

DISABLE CONNECTION

Syntax
DISABLE CONNECTION CLEAR
DISABLE CONNECTION ENCRYPT
DISABLE CONNECTION JSON
DISABLE CONNECTION TOKEN

Privileges
ADMIN

Description
Use the DISABLE CONNECTION command if you want to disable one or more connection type from the server.

Example
The following example DISABLE CONNECTION JSON on the server.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("DISABLE CONNECTION JSON;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

ENABLE CONNECTION

Syntax
ENABLE CONNECTION CLEAR
ENABLE CONNECTION ENCRYPT
ENABLE CONNECTION JSON
ENABLE CONNECTION TOKEN

Privileges
ADMIN

Description
Use the ENABLE CONNECTION command if you want to re-enable a previously disabled connection type from the server.

Example
The following example ENABLE CONNECTION JSON on the server.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("ENABLE CONNECTION JSON;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

LOG

SHOW LAST ROWS FROM LOG

Syntax
SHOW LAST nrows ROWS FROM LOG
SHOW LAST nrows ROWS FROM LOG ORDER DESC

Privileges
SERVER

Description
SHOW LAST nrows ROWS FROM LOG returns the last nrows log entries as a RecordSet. The parameter value is an integer and you can specify the descending order with the ORDER DESC parameter.

The fields in the RecordSet are datetime as timestamp, description as text, operation as text, username as text, database as text, address as text, version as integer, connectionID as integer and seconds as real. Please note that for some entries some fields can be NULL.

Example
The following retrieves the last 20 rows from the log.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW LAST 20 ROWS FROM LOG;")
DisplayRecordSet (rs)
```

SHOW LOG FROM

Syntax
SHOW LOG FROM date1 TO date2
SHOW LOG FROM date1 TO date2 ORDER DESC

Privileges
SERVER

Description
SHOW LAST nrows ROWS FROM LOG returns log entries in a specified date range as a RecordSet. The parameters are in the form yyyy-mm-dd and you can specify the descending order with the ORDER DESC parameter. This command can be executed only if the LOG format is set to SQLITE3.

The fields in the RecordSet are datetime as timestamp, description as text, operation as text, username as text, database as text, address as text, version as integer, connectionID as integer and seconds as real. Please note that for some entries some fields can be NULL.

Example
The following retrieves log entries in a specified date range.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW LOG FROM 2011-02-20 TO 2011-02-23;")
DisplayRecordSet (rs)
```

TABLES & INDEXES

SHOW INDEXES

Syntax
SHOW INDEXES FOR DATABASE databaseName
SHOW INDEXES FOR TABLE tableName

Privileges
NONE

Description
SHOW INDEXES FOR DATABASE returns the names of all the indexes defined in the specified database. If you need to know all the indexes associated to a given table use the SHOW INDEXES FOR TABLE command.

Example
The following example gets the list of indexes in the data table (inside the images database).

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("USE DATABASE images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

Dim rs as RecordSet = db.SQLSelect("SHOW INDEXES FOR TABLE data;")
DisplayRecordSet (rs)
```

SHOW TABLES

Syntax
SHOW TABLES
SHOW TABLES FOR DATABASE databaseName

Privileges
NONE

Description
SHOW TABLES returns the names of all the tables defined in the current database. If you need to know all the tables defined in a given database then use the SHOW TABLES FOR DATABASE command.

Example
The following example gets the list of tables defined into my current images database.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("USE DATABASE images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

Dim rs as RecordSet = db.SQLSelect("SHOW TABLES;")
DisplayRecordSet (rs)
```

SHOW TABLE INFO

Syntax
SHOW TABLE INFO tableName

Privileges
NONE

Description
SHOW TABLE INFO returns information about the specified tableName. Result is equal to the PRAGMA table_info sqlite command. It returns a RecordSet with information about column name, data type, whether or not the column can be NULL, and the default value for the column.

Example
The following example gets information about the data table defined into my current images database.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("USE DATABASE images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

Dim rs as RecordSet = db.SQLSelect("SHOW TABLE INFO data;")
DisplayRecordSet (rs)
```

PREFERENCES

DROP PREFERENCE

Syntax
DROP PREFERENCE key

Privileges
PREFERENCES

Description
DROP PREFERENCE remove a previously defined preference key.

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("DROP PREFERENCE myPreferenceKey;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

SET AUTOCOMMIT

Syntax
SET AUTOCOMMIT TO ON|OFF

Privileges
NONE

Description
Server by default is in AUTOTRANSACTION mode, that means that every INSERT, UPDATE, DELETE sql operation is automatically executed inside a TRANSACTION that must be COMMITTED or ROLLEDBACK at the end. If you want to change the default behavior you can set AUTOCOMMIT to ON, that means that an implicit COMMIT is executed after every WRITE operation. This is a PER CONNECTION property and not a global server or a global database property. Please note that AUTOTRANSACTION cannot be turned ON when MVCC is enabled.

Example
This example turns ON the autocommit property.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("SET AUTOCOMMIT TO ON;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

SET CLIENT TYPE

Syntax
SET CLIENT TYPE TO value

Privileges
NONE

Description
Use the SET CLIENT TYPE TO value command to set a custom user defined value on server side associated to this connection. That value is then returned by the SHOW CONNECTIONS command. It's up to you to decide what to do with this value.

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("SET CLIENT TYPE TO 'My Great Application';")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

SET LANGUAGE

Syntax
SET LANGUAGE TO value

Privileges
NONE

Description
SET LANGUAGE sets the default language for this connection on the server. You must use ISO 3166-1 alpha-2 country codes. See http://en.wikipedia.org/wiki/ISO_3166-1_alpha-2. Please note that this command is reserved for future use and it's currently ignored by the server.

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("SET LANGUAGE TO IT;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

SET PING TIMEOUT

Syntax
SET PING TIMEOUT TO value

Privileges
NONE

Description
When a client does nothing for more than PING TIMEOUT seconds than the server automatically disconnects it in order to save resources. Default value is 300 (5 minutes) but you can use the SET PING TIMEOUT command to change that value.
Please note that SET PING TIMEOUT is a per connection property.

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("SET PING TIMEOUT TO 500;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

SET PREFERENCE

Syntax
SET PREFERENCE key TO value

Privileges
PREFERENCES

Description
SET PREFERENCE sets the preference for the passed key to the passed value. Server supports a lot of keys and customizations, for a list of supported keys just show the output of the SHOW PREFERENCES command.

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("SET PREFERENCE SERVER_NAME TO MyGreatApplication;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

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. Without a valid serial number server is in restricted mode, that means that only custom commands are enabled and the server is limited to accept just 2 concurrent connections.

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 CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("SET REGISTRATION TO 'Marco Bambini' "
                    +"WITH KEY XXXXXXX-XXXXXX-XXXX")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

SET TIMEOUT

Syntax
SET TIMEOUT TO value

Privileges
NONE

Description
SET TIMEOUT sets the maximum time (in seconds) a client is allowed to wait for a busy resource to became available. It could be for example a locked database or a locked table. Increase that value if you experience a lot of "resource unavailable" errors or if your server has a lot of writer. Default value is 12 seconds.

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("SET TIMEOUT TO 30;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

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" and it can be 1 or 0.

Example
This example reports the current setting of the autocommit property.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW AUTOCOMMIT;")
DisplayRecordSet (rs)
```

SHOW PREFERENCE

Syntax
SHOW PREFERENCE key

Privileges
PREFERENCES

Description
SHOW PREFERENCE returns the value of the preference with the passed key as a RecordSet with a key/value columns. 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 CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW PREFERENCE SERVER_PORT;")
DisplayRecordSet (rs)
```

SHOW PREFERENCES

Syntax
SHOW PREFERENCES

Privileges
PREFERENCES

Description
SHOW PREFERENCES returns all the server preferences and their values as a RecordSet with a key/value columns.

Example
The following example lists all the preferences.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW PREFERENCES;")
DisplayRecordSet (rs)
```

FILES I/O

FILE DOWNLOAD

Syntax
FILE DOWNLOAD fileName

Privileges
WRITE

Description
FILE DOWNLOAD downloads a previously uploaded file names fileName.
A special folder is created on server file to collect all uploaded files. Please note that each file is associated to the currently used database (a folder for each associated database is automatically created on server side).
The Examples folder includes a Xojo application that illustrates uploading and downloading files.

Example
This example downloads a foo.png file.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

// ask the user for a folder to download to
Dim f as FolderItem = SelectFolder
if (f = nil) then return

// initiate file download
db.SQLExecute("FILE DOWNLOAD foo.png;")
If db.error then
    MsgBox "An error occurred: " + db.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 = db.ReceiveChunk

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

  // see if we have reached the end of the chunks and exit the loop if we have
  if db.IsEndChunk then
    exit
  end if

  // write the chunk out to the file and loop again
  bs.Write chunk
wend

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

FILE UPLOAD [WITH REPLACE]

Syntax
FILE UPLOAD fileName
FILE UPLOAD fileName WITH REPLACE

Privileges
WRITE

Description
FILE UPLOAD uploads a file to the server. A special folder is created on server file to collect all uploaded files. Please note that each file is associated to the currently used database (a folder for each associated database is automatically created on server side). An error occurred if a file with the same name AND associated to the same database already exists on server side. To prevent this error and to automatically replace the file us the WITH REPLACE variants.

The Examples folder includes a Xojo application that illustrates uploading and downloading files.

Example
The following example uploads a file selected by the user.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

// select database to upload
Dim f as FolderItem = GetOpenFolderItem("Any")
if (f = nil) then return

// try to initiate an upload (notice how we quote the database name
// with single quotes in case the name contains spaces)
db.SQLExecute("FILE UPLOAD'" + f.Name + "';")
If db.error then
    MsgBox "An error occurred: " + db.ErrorMessage
```

```
      return
end if


 // open the file as a BinaryStream
 dim bs as BinaryStream = f.OpenAsBinaryFile
 if bs = nil then
  return
 end if

 // upload the file in chunks
 while not bs.EOF
  // read the next chunk from the file
  dim chunk as String = bs.Read(100*1024)

  // send the chunk to the server
  db.SendChunk chunk

  // if there was an error; report it and bail
  if db.Error then
   MsgBox "Error receiving chunk from server: " + db.ErrorMessage
   return
  end if

wend

// send the stop end chunk command (the server needs this
// to know that the file has been completely sent)
db.SendEndChunk

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

FILE DELETE

Syntax
FILE DELETE fileName

Privileges
WRITE

Description
FILE DELETE deletes fileName on server side (associated to the current in use database).

Example
This example deletes file”image.png".

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute(“FILE DELETE image.png;”)
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

FILE RENAME

Syntax
FILE RENAME oldName TO newName

Privileges
WRITE

Description
FILE RENAME renames file named oldName to newName on server side (associated to the current in use database).

Example
This example renames file"foo.png" TO "bar.png".

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("FILE RENAME foo.png TO bar.png;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

SHOW FILE INFO

Syntax
SHOW FILE INFO fileName
SHOW FILE EXISTS fileName

Privileges
NONE

Description
SHOW FILE INFO collect detailed information about file specified in fileName. The EXISTS variant just collect a boolean value that indicates in the file exists or not.

Example
This example just collect information about a foo.png file.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs As RecordSet = db.SQLSelect("SHOW FILE INFO foo.png;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
DisplayRecordSet(rs)
```

SHOW FILES

Syntax
SHOW FILES

Privileges
NONE

Description
SHOW FILES returns a RecordSet with all the files associated to the currently in use database.

Example
This example just list all the files associated with current myDatabase db.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"
db.DatabaseName = "myDatabase"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs As RecordSet = db.SQLSelect("SHOW FILES;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
DisplayRecordSet(rs)
```

PLUGINS

DISABLE PLUGIN

Syntax
DISABLE PLUGIN pluginName

Privileges
PLUGIN

Description
DISABLE PLUGIN disables the passed pluginName.

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("DISABLE PLUGIN fts1;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

ENABLE PLUGIN

Syntax
ENABLE PLUGIN pluginName

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 CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("ENABLE PLUGIN fts1;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

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 name, version, copyright, description and exename.

Example
The following example returns the characteristics of the installed plugins.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW PLUGINS;")
DisplayRecordSet (rs)
```

OTHERS

CLOSE CONNECTION

Syntax
CLOSE CONNECTION connectionID

Privileges
SERVER

Description
CLOSE CONNECTION closes the connection for the passed connectionID. 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 connectionID.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("CLOSE CONNECTION 10;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

COLLECT STATS

Syntax
COLLECT STATS

Privileges
ADMIN

Description
COLLECT STATS causes an immediate update to server statistics.
They can be viewed by calling SHOW SERVER STATS.

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("COLLECT STATS;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

Dim rs as RecordSet = db.SQLSelect("SHOW SERVER STATS")
DisplayRecordSet (rs)
```

PING

Syntax
PING

Privileges
NONE

Description
Used to determine if the server is alive. It either completes successfully, or returns an error. The PING command also reset the PING TIMEOUT timer.

Example
The following pings a remote server.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("PING;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

QUIT SERVER

Syntax
QUIT SERVER

Privileges
ADMIN

Description
QUIT SERVER quits the server you are connected to.

Examples

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("QUIT SERVER;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

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 command, context, and privilege fields. Commands registered by plugins are also returned.

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

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

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

SHOW CONNECTIONS

Syntax
SHOW CONNECTIONS

Privileges
SERVER

Description
SHOW CONNECTIONS returns 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 returned by the RecordSet are id, address, username, connection_date, last_activity, database, client_version, client_type, language.

Example
The following example returns information about all the connected users.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

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

SHOW CHANGES

Syntax
SHOW CHANGES

Privileges
NONE

Description
SHOW CHANGES returns the number of rows modified, inserted or deleted by the most recently completed INSERT, UPDATE or DELETE statement on the database connection. Executing any other type of SQL statement does not modify the value returned by this function. Only changes made directly by the INSERT, UPDATE or DELETE statement are considered - auxiliary changes caused by triggers, foreign key actions or REPLACE constraint resolution are not counted..

Example
The following example count the latest changes.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("USE DATABASE Invoice;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

db.SQLExecute("INSERT INTO data (col1) VALUES ('test');")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

Dim rs as RecordSet = db.SQLSelect("SHOW CHANGES;")
DisplayRecordSet (rs)
```

SHOW INFO

Syntax
SHOW INFO

Privileges
NONE

Description
SHOW INFO returns server information as a RecordSet. A lot of useful information are returned by this command in a key/value RecordSet.

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

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

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

SHOW LASTROWID

Syntax
SHOW LASTROWID

Privileges
NONE

Description
SHOW LASTROWID returns the last rowid added to the current database as a RecordSet. This command is equivalent to the sqlite3_last_insert_rowid API, so it returns the returns the rowid of the most recent successful INSERT into the database from the current client connection. If no successful INSERTs have ever occurred on that database connection, zero is returned.

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

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("USE DATABASE Images;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

db.SQLExecute("INSERT INTO data (col1) VALUES ('test');")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage

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

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 key/values rows in a RecordSet.

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

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW MY INFO")
DisplayRecordSet (rs)
```

SHOW SERVER STATS

Syntax
SHOW SERVER STATS
SHOW SERVER STATS FROM date1 TO date2

Privileges
NONE

Description
SHOW SERVER STATS returns a RecordSet with statistics on a variety of server parameters as key/values rows. Informations includes memory usage, number of concurrent connections, commands counter and much more. Statistics are automatically updated every 10 minutes and if you need to show stats just from a give date range then use the FROM date1 TO date2 parameters. Both dates are in SQL format.

Example

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430

db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

Dim rs as RecordSet = db.SQLSelect("SHOW SERVER STATS")
DisplayRecordSet (rs)
```