How to manage the SQL Server Desktop Engine (MSDE 2000) or SQL Server 2005 Express Edition by using the osql utility

This article was previously published under Q325003

Article ID : 325003
Last Review : December 14, 2005
Revision : 2.0

On This Page

SUMMARY

The SQL Server Desktop Engine (also known as MSDE 2000) does not have its own user interface because it is primarily designed to run in the background. Users interact with MSDE 2000 through the program in which it is embedded. The only tool that is provided with MSDE 2000 is the **osql** utility. The executable file, Sql.exe, is located in the MSSQL\Binn folder for a default instance of MSDE 2000. This article focuses on how to manage MSDE 2000 by using the **osql** utility.

If you are using SQL Server 2005, you can also use the **osql** utility to manage SQL Server 2005 Express Edition. However, this feature will be removed in a future version of Microsoft SQL Server 2005. We recommend that you do not use this feature in new development work and plan to modify applications that currently use the feature. Use the Sqlcmd utility instead. For more information about how to use the Sqlcmd utility, visit the following Microsoft Developer Network (MSDN) Web site:

http://msdn2.microsoft.com/en-us/library/ms170207.aspx (http://msdn2.microsoft.com/en-us/library/ms170207.aspx)

What is Osql?

The **osql** utility is a Microsoft Windows 32 command prompt utility that you can use to run Transact-SQL statements and script files. The **osql** utility uses the ODBC database application programming interface (API) to communicate with the server.

How Do You Use Osql?

Typically, you use the **osql** utility these ways:

- Users interactively enter Transact-SQL statements in a manner similar to working with the command prompt.
- Users submit an **osql** job either by:
  - Specifying a single Transact-SQL statement to run. -or-
  - -or-By pointing the utility to a script file that contains Transact-SQL statements to run.

Interactively Enter Transact-SQL Statements

To display a list of the case-sensitive options of the **osql** utility, type the following at a command prompt, and then press ENTER:

osql -?

For more information about each option of the **osql** utility, see the "osql Utility" topic in SQL Server Books Online.

To interactively enter Transact-SQL statements, follow these steps:

1. Verify that MSDE 2000 is running.
2. Connect to MSDE 2000 (for more information, see the section titled "Connect to SQL Server Desktop Engine (MSDE 2000)").
3. At the **osql** prompt, type the Transact-SQL statements, and then press ENTER. When you press ENTER, at the end of each input line, **osql** caches the statements on that line.

   - To run the currently cached statements, type "Go", and then press ENTER.

   - To run a batch of Transact-SQL statements, enter each Transact-SQL command on separate lines. Then, type "Go" on the last line to signal the end of the batch and to run the currently cached statements.

   The results appear in the console window.

4. To exit from **osql**, type QUIT, or EXIT, and then press ENTER.

Submit an Osql Job

Typically, you submit an **osql** job one of two ways. You can either:

- Specify a single Transact-SQL statement.

  -or-

- -or-Point the utility to a script file.

Here is more information about each method.

Specify a Single Transact-SQL Statement

To run a Transact-SQL statement against the local default instance of MSDE 2000, type a command similar to this one

osql -E -q "Transact-SQL statement"

where

- **-E** uses Microsoft Windows NT authentication.

  -and-

- -and-**-q** runs the Transact-SQL statement but does not exit **osql** when the query completes.

To run the Transact-SQL statement and exit **osql**, use the **-Q** argument instead of **-q**.

Point the Utility to a Script File

To point the utility to a script file, follow these steps:

1. Create a script file that contains a batch of Transact-SQL statements (such as myQueries.sql).
2. Open a command prompt, type a command similar to the following, and then press ENTER

   osql -E -i input_file

   where

   input_file is the full path of the script file. For example, if the script file myQueries.sql is located in the C:\Queries folder, replace the parameter *input_file* with C:\Queries\myQueries.sql.

   The results of the script file appear in the console window. If you want to direct the results to a file, add the **-**

o*output_file* argument to the command shown earlier. For example:

osql -E -i input_file -o output_file

where

output_file is the full path of the output file.

To remove the numbering and prompt symbols in the output, add the **-n** option to the command shown earlier. For example:

osql -E -i input_file -o output_file -n

Connect to SQL Server Desktop Engine (MSDE 2000)

To connect to MSDE 2000, follow these steps:

1. Verify that MSDE 2000 is running.
2. Open a command prompt on the computer that is hosting the instance of MSDE 2000 to which you want to connect.
3. Type the following command, and then press ENTER:

   osql -E

   This connects you to the local, default instance of MSDE 2000 by using Windows Authentication.

   To connect to a named instance of MSDE 2000, type:

   osql -E -S servername\instancename

   If you receive the following error message, MSDE 2000 may not be running or you may have provided an incorrect name for the named instance of MSDE 2000 that is installed:

   > **[Shared Memory]SQL Server does not exist or access denied.**
   > **[Shared Memory]ConnectionOpen (Connect()).**

   If you are successfully connected to the server, the following prompt appears:

   ```
   1>
   ```

   This prompt indicates that **osql** is started. Now, you can interactively enter Transact-SQL statements and the results appear on the command prompt line.

Manage MSDE 2000

The remaining sections of this article introduce you briefly to the Transact-SQL commands most frequently used to manage MSDE 2000.

Create a New Login

A user cannot connect to SQL Server without providing a valid login id. The **sp_grantlogin** stored procedure is used to authorize a Microsoft Windows network account (either a group or a user account) for use as a SQL Server login for connecting to an instance of SQL Server by using Windows Authentication. The following example permits a Windows NT user named Corporate\Test to connect to a SQL Server instance:

```
EXEC sp_grantlogin 'Corporate\Test'
```

Only members of the **sysadmin** or the **securityadmin** fixed server roles can run the **sp_grantlogin** stored procedure. For more information about roles, see the "Roles, SQL Server Architecture" topic in SQL Server Books Online.

For more information about the **sp_grantlogin** stored procedure, see the "sp_grantlogin, Transact-SQL Reference" topic in SQL Server Books Online.

You use the **sp_addlogin** stored procedure to create a new login account for SQL Server connections by using SQL

Server Authentication. The following example creates a SQL Server login for a user named "test" with a password of "hello":

```
EXEC sp_addlogin 'test','hello'
```

Only members of the **sysadmin** and the **securityadmin** fixed server roles can run the **sp_addlogin** stored procedure. For more information about the **sp_addlogin** stored procedure, see the "sp_addlogin, Transact-SQL Reference" topic in SQL Server Books Online.

Access a Database

After a users connects to an instance of SQL Server, they cannot perform activities in a database until the **dbo** grants them access to the database. You can use the **sp_grantdbaccess** stored procedure to add a security account for a new user to the current database. The following example adds an account for a Microsoft Windows NT user named Corporate\BobJ to the current database and names it "Bob":

```
EXEC sp_grantdbaccess 'Corporate\BobJ', 'Bob'
```

The **sp_adduser** stored procedure performs the same function as the **sp_grantdbaccess** stored procedure. Because, the **sp_adduser** stored procedure is included for backward compatibility, Microsoft recommends that you use the **sp_grantdbacess** stored procedure.

Only members of the **sysadmin** fixed server role, the **db_accessadmin** and the **db_owner** fixed database roles can run the **sp_grantdbaccess** stored procedure. For more information about the **sp_grantdbaccess** stored procedure, see the "sp_grantdbaccess, Transact-SQL Reference" topic in SQL Server Books Online.

How to Change the Password for a Login

To change the password of a login, use the **sp_password** stored procedure. The following example changes the password for the login "test" from "ok" to "hello":

```
EXEC sp_password 'ok', 'hello','test'
```

Execute permissions default to the public role for a user that is changing the password for his or her own login. Only members of the **sysadmin** role can change the password for another user's login. For more information about the **sp_password** stored procedure, see the "sp_password, Transact-SQL Reference" topic in SQL Server Books Online

Create a Database

A MSDE 2000 database is made up of a collection of tables that contain data and other objects, such as views, indexes, stored procedures, and triggers, which are defined to support activities performed with the data. To create a MSDE 2000 database, use the "CREATE DATABASE" Transact-SQL command. For more information about creating a database, see the "Creating a Database" topic in SQL Server Books Online.

The following example creates a database named **Test**. Because no additional parameters are added to the command, the **Test** database will be the same size as the **model** database:

```
CREATE DATABASE Test
```

CREATE DATABASE permission defaults to members of the **sysadmin** and the **dbcreator** fixed server roles. For more information about the "CREATE DATABASE" command, see the "CREATE DATABASE, Transact-SQL Reference" topic in SQL Server Books Online.

To create a new database object, use the CREATE Transact-SQL command. For example, to create a new table, use the "CREATE TABLE" Transact-SQL command. For more information, refer to SQL Server Books Online.

Back Up and Restore Databases

The backup and restore component of SQL Server provides an important safeguard for protecting critical data stored in SQL Server databases.

With proper planning, you can recover from many failures, including:

- Media failure.
- User errors.
- Permanent loss of a server.

Additionally, backing up and restoring databases is useful for other purposes, such as copying a database from one server to another. By backing up a database from one computer and restoring the database to another, you quickly and easily make a copy of a database.

For more information about database backup and restore operations, see the "Backing Up and Restoring Databases" topic in SQL Server Books Online.

The following example performs a full database backup for a database named **mydb**, names the backup Mydb.bak, and then stores the backup in the C:\Msde\Backup folder:

```
BACKUP DATABASE mydb TO DISK = 'C:\MSDE\Backup\mydb.bak'
```

The following example performs a log backup for a database named **mydb**, names the backup Mydb_log.bak, and then stores it in the C:\Msde\Backup folder:

```
BACKUP LOG mydb TO DISK = 'C:\MSDE\Backup\mydb_log.bak'
```

BACKUP DATABASE and BACKUP LOG permissions default to members of the **sysadmin** fixed server role and the **db_owner** and **db_backupoperator** fixed database roles. For more information about the BACKUP statement, see the "BACKUP, Transact-SQL Reference" topic in SQL Server Books Online.

MSDE includes the **SQL Server Agent** Service for managing scheduled jobs. For example, you can create and schedule a Transact-SQL backup job. The SQL Server Agent Service manages the job scheduling. For sample code about how to use the various stored procedures with MSDE 2000 to perform a backup and schedule the backup, see the following article in the Microsoft Knowledge Base:

241397 (http://support.microsoft.com/kb/241397/EN-US/) How To Back Up a Microsoft Data Engine Database with Transact-SQL

For more information about the SQL Server Agent Service, see the "SQL Server Agent Service" topic in SQL Server Books Online.

Backing up a database is only half of the process. It is important to know how to restore the database from a backup. The following example restores a database that is named **mydb** from the backup file C:\Msde\Backup\Mydb.bak:

```
RESTORE DATABASE mydb FROM DISK ='C:\MSDE\Backup\mydb.bak'
```

If the database that is being restored does not exist, the user must have CREATE DATABASE permissions to run the RESTORE statement. If the database exists, RESTORE permissions default to members of the **sysadmin** and **dbcreator** fixed server roles and the owner (**dbo**) of the database. For more information about the RESTORE statement, see the "RESTORE, Transact-SQL Reference" topic in SQL Server Books Online .

Attach and Detach a Database

The data and transaction log files of a database can be detached and then reattached to another server, or even to the same server. Detaching a database removes the database from SQL Server but leaves the database intact in the data and transaction log files that compose the database. You can then use these data and transaction log files to attach the database to any instance of SQL Server, including the server from which the database was detached. This makes the database available in exactly the same state it was in when it was detached. For more information, see the "Attaching and Detaching a Database" topic in SQL Server Books Online.

The following example detaches a database named **mydb** from the current instance of SQL Server:

```
EXEC sp_detach_db 'mydb'
```

Only members of the **sysadmin** fixed server role can run the **sp_detach_db** stored procedure. For more information about the **sp_detach_db** stored procedure, see the "sp_detach_db, Transact-SQL Reference" topic in SQL Server Books Online.

The following example attaches two files from a database named **mydb** to the current instance of SQL Server:

```
EXEC sp_attach_db @dbname = N'mydb',
   @filename1 = N'C:\MSDE\Backup\mydb.mdf',
    @filename2 = N'C:\MSDE\Backup\mydb.ldf'
```

The capital letter N is used to prefix a Unicode String constant. The "N" prefix stands for National Language in the SQL-92 standard. For more information, see the following article in the Microsoft Knowledge Base:

> 239530 (http://support.microsoft.com/kb/239530/EN-US/) INF: Unicode String Constants in SQL Server Require N Prefix

Only members of the **sysadmin** and the **dbcreator** fixed server roles can run this procedure. For more information about the **sp_attach_db** stored procedure, see the "sp_attach_db, Transact-SQL Reference" topic in SQL Server Books Online.

The following information about the use of the **osql** utility also applies to all editions of Microsoft SQL Server 2000.

REFERENCES

To download an updated version of the SQL Server 2000 Books Online, visit the following Microsoft Web site:

> http://www.microsoft.com/sql/techinfo/productdoc/2000/books.asp
> (http://www.microsoft.com/sql/techinfo/productdoc/2000/books.asp)

To download the SQL Server 7.0 version of SQL Server Books Online, visit the following Microsoft Web site:

> http://download.microsoft.com/download/SQL70/File/2/Win98/En-US/SQLBOL.exe (http://download.microsoft.com/download/sql70/file/2/win98/en-us/sqlbol.exe)

For more information about MSDE 2000, see the following articles in the Microsoft Knowledge Base:

> 319930 (http://support.microsoft.com/kb/319930/EN-US/) How To Connect to Microsoft Desktop Engine

> 241397 (http://support.microsoft.com/kb/241397/EN-US/) How To Back Up a Microsoft Desktop Engine Database with Transact-SQL

APPLIES TO

- Microsoft SQL Server 2000 Desktop Engine (Windows)
- Microsoft SQL Server 2000 Desktop Engine

Keywords: kbdownload kbhowtomaster KB325003