Useful undocumented extended stored procedures

Alexander Chigrik
chigrik@mssqlcity.com

An extended stored procedure (xp) is a dynamic link library that runs directly in the address space of SQL Server and is programmed using the SQL Server Open Data Services API. You can run extended stored procedures from the Query Analyzer, for example, just as you would normal stored procedures. Extended stored procedures are used to extend the capabilities of SQL Server. You can take advantage of the many extended stored procedures that come with SQL Server, or you can write your own in a programming language such as C or C++.

In this article, I want to tell you about some useful undocumented extended stored procedures. These extended stored procedures work with SQL Server 7.0, as well as with SQL Server 2000.

sp_MSgetversion

This extended stored procedure can be used to get the current version of Microsoft SQL Server. To get the current SQL Server version, run

EXEC master..sp_MSgetversion

Note. A more common way to retrieve the current SQL Server version (this way provides more information) is to use following SELECT statement:

SELECT @@version

xp_dirtree

This extended stored procedure can be used to get a list of all the folders for the folder named in the xp. To get a list of all the folders in the C:\MSSQL7 folder, run:

EXEC master..xp_dirtree 'C:\MSSQL7'

xp_enum_oledb_providers

This extended stored procedure is used to list of all the available OLE DB providers. It returns Provider Name, Parse Name and Provider Description. To get a list of all OLE DB providers for your SQL Server, run:

EXEC master..xp_enum_oledb_providers

xp_enumcodepages

This extended stored procedure can be used to list of all code pages, character sets and their description for your SQL Server. To get a list of all code pages and character sets, run:

EXEC master..xp_enumcodepages

xp_enumdsn

This extended stored procedure returns a list of all System DSNs and their description. To get the list of System DSNs, run:

EXEC master..xp_enumdsn

xp_enumerrorlogs

This extended stored procedure returns the list of all error logs with their last change date. To get the list of error logs, run:

EXEC master..xp_enumerrorlogs

xp_enumgroups

This extended stored procedure returns the list of Windows NT groups and their description. To get the list of the Windows NT groups, run:

EXEC master..xp_enumgroups

xp_fileexist

You can use this extended stored procedure to determine whether a particular file exists on the disk or not.

Syntax:

EXECUTE xp_fileexist filename [, file_exists INT OUTPUT]

For example, to check whether the file boot.ini exists on disk c: or not, run:

EXEC master..xp_fileexist 'c:\boot.ini'

xp_fixeddrives

This very useful extended stored procedure returns the list of all hard drives and the amount of free space in Mb for each hard drive.

To see the list of drives, run:

EXEC master..xp_fixeddrives

xp_getnetname

This extended stored procedure returns the WINS name of the SQL Server that you're connected to.

To view the name, run:

EXEC master..xp_getnetname

xp_readerrorlog

This extended stored procedure returns the content of the errorlog file. You can find the errorlog file in the C:\MSSQL7\Log directory, by default for SQL Server 7.0.

To see the text of the errorlog file, run:

EXEC master..xp_readerrorlog

xp_regdeletekey

This extended stored procedure will delete an entire key from the registry. You should use it very carefully.

Syntax:

```
EXECUTE xp_regdeletekey [@rootkey=]'rootkey',
                        [@key=]'key'
```

For example, to delete the key 'SOFTWARE\Test' from 'HKEY_LOCAL_MACHINE', run:

```
EXEC master..xp_regdeletekey
     @rootkey='HKEY_LOCAL_MACHINE',
     @key='SOFTWARE\Test'
```

xp_regdeletevalue

This extended stored procedure will delete a particular value for a key in the registry. You should use it very carefully.

Syntax:

```
EXECUTE xp_regdeletevalue [@rootkey=]'rootkey',
                          [@key=]'key',
                          [@value_name=]'value_name'
```

For example, to delete the value 'TestValue' for the key 'SOFTWARE\Test' from 'HKEY_LOCAL_MACHINE', run:

```
EXEC master..xp_regdeletevalue
     @rootkey='HKEY_LOCAL_MACHINE',
     @key='SOFTWARE\Test',
     @value_name='TestValue'
```

xp_regread

This extended stored procedure is used to read from the registry.

Syntax:

```
EXECUTE xp_regread [@rootkey=]'rootkey',
                   [@key=]'key'
                   [, [@value_name=]'value_name']
                   [, [@value=]@value OUTPUT]
```

For example, to read into the variable @test from the value 'TestValue' from the key 'SOFTWARE\Test' from the 'HKEY_LOCAL_MACHINE', run:

```
DECLARE @test varchar(20)
EXEC master..xp_regread @rootkey='HKEY_LOCAL_MACHINE',
  @key='SOFTWARE\Test',
  @value_name='TestValue',
  @value=@test OUTPUT
SELECT @test
```

xp_regwrite

This extended stored procedure is used to write to the registry.

Syntax:

```
EXECUTE xp_regwrite [@rootkey=]'rootkey',
                    [@key=]'key',
                    [@value_name=]'value_name',
```

```
[@type=]'type',
[@value=]'value'
```

For example, to write the variable 'Test' to the 'TestValue' value, key 'SOFTWARE\Test', 'HKEY_LOCAL_MACHINE', run:

```
EXEC master..xp_regwrite
     @rootkey='HKEY_LOCAL_MACHINE',
     @key='SOFTWARE\Test',
     @value_name='TestValue',
     @type='REG_SZ',
     @value='Test'
```

xp_subdirs

This extended stored procedure is used to get the list of folders for the folder named in the xp. In comparison with **xp_dirtree**, **xp_subdirs** returns only those directories whose depth = 1.

This is the example:

EXEC master..xp_subdirs 'C:\MSSQL7'

Note. Keep in mind that these undocumented extended stored procedures are not officially supported by Microsoft, and that they may not be found in the next version of SQL Server.