xp_cmdshell (Transact-SQL) - SQL Server

By markingmyname

Archived: 2026-04-06 01:58:16 UTC

Applies to: SQL Server

Spawns a Windows command shell and passes in a string for execution. Any output is returned as rows of text.

Transact-SQL syntax conventions

```
xp_cmdshell { 'command_string' } [ , NO_OUTPUT ]
```

Important

Arguments for extended stored procedures must be entered in the specific order as described in the Syntax section. If the parameters are entered out of order, an error message occurs.

The string that contains a command to be passed to the operating system. *command_string* is **varchar(8000)** or **nvarchar(4000)**, with no default. *command_string* can't contain more than one set of double quotation marks. A single pair of quotation marks is required if any spaces are present in the file paths or program names referenced in *command_string*. If you have trouble with embedded spaces, consider using FAT 8.3 file names as a workaround.

An optional parameter, specifying that no output should be returned to the client.

`0` (success) or `1` (failure).

Executing the following `xp_cmdshell` statement returns a directory listing of the current directory.

```
 EXECUTE xp_cmdshell 'dir *.exe';
GO
```

The rows are returned in an **nvarchar(255)** column. If the `NO_OUTPUT` option is used, only the following output is returned:

```
The command(s) completed successfully.
```

The Windows process spawned by `xp_cmdshell` has the same security rights as the SQL Server service account.

`xp_cmdshell` operates synchronously. Control isn't returned to the caller until the command-shell command is completed. If `xp_cmdshell` is executed within a batch and returns an error, the batch will fail.

When it's called by a user that isn't a member of the **sysadmin** fixed server role, `xp_cmdshell` connects to Windows by using the account name and password stored in the credential named `##xp_cmdshell_proxy_account##`. If this proxy credential doesn't exist, `xp_cmdshell` fails.

The proxy account credential can be created by executing `sp_xp_cmdshell_proxy_account`. As arguments, this stored procedure takes a Windows user name and password. For example, the following command creates a proxy credential for Windows domain user `SHIPPING\KobeR`. Replace `<password>` with a strong password.

```
EXECUTE sp_xp_cmdshell_proxy_account 'SHIPPING\KobeR', '<password>';
```

For more information, see sp_xp_cmdshell_proxy_account.

Because malicious users sometimes attempt to elevate their privileges by using `xp_cmdshell`, `xp_cmdshell` is disabled by default. Use `sp_configure` or **Policy Based Management** to enable it. For more information, see xp_cmdshell Server Configuration Option.

When first enabled, `xp_cmdshell` requires CONTROL SERVER permission to execute and the Windows process created by `xp_cmdshell` has the same security context as the SQL Server service account. The SQL Server service account often has more permissions than are necessary for the work performed by the process created by `xp_cmdshell`. To enhance security, access to `xp_cmdshell` should be restricted to highly privileged users.

To allow non-administrators to use `xp_cmdshell`, and allow SQL Server to create child processes with the security token of a less-privileged account, follow these steps:

1. Create and customize a Windows local user account or a domain account with the least privileges that your processes require.
2. Use the `sp_xp_cmdshell_proxy_account` system procedure to configure `xp_cmdshell` to use that least-privileged account.

   Note

   You can also configure this proxy account using SQL Server Management Studio by right-clicking **Properties** on your server name in Object Explorer, and looking on the **Security** tab for the **Server proxy account** section.
3. In Management Studio, using the `master` database, execute the following Transact-SQL statement to give specific non-**sysadmin** users the ability to execute `xp_cmdshell`. The specified user must exist in the `master` database.

   ```
   GRANT exec ON xp_cmdshell TO N'<some_user>';
   ```

Now non-administrators can launch operating system processes with `xp_cmdshell` and those processes run with the permissions of the proxy account that you configured. Users with CONTROL SERVER permission (members

of the **sysadmin** fixed server role) continue to receive the permissions of the SQL Server service account for child processes that are launched by `xp_cmdshell`.

To determine the Windows account being used by `xp_cmdshell` when launching operating system processes, execute the following statement:

```
EXECUTE xp_cmdshell 'whoami.exe';
```

To determine the security context for another login, execute the following Transact-SQL code:

```
EXECUTE AS LOGIN = '<other_login>';
GO

EXECUTE xp_cmdshell 'whoami.exe';

REVERT;
```

The following example shows the `xp_cmdshell` extended stored procedure executing a directory command.

```
EXECUTE master..xp_cmdshell 'dir *.exe';
```

The following example uses `xp_cmdshell` to execute a command string without returning the output to the client.

```
USE master;

EXECUTE xp_cmdshell 'copy c:\SQLbcks\AdvWorks.bck
    \\server2\backups\SQLbcks', NO_OUTPUT;
GO
```

In the following example, the `xp_cmdshell` extended stored procedure also suggests return status. The return code value is stored in the variable `@result`.

```
DECLARE @result AS INT;

EXECUTE @result = xp_cmdshell 'dir *.exe';

IF (@result = 0)
    PRINT 'Success';
ELSE
    PRINT 'Failure';
```

The following example writes the contents of the `@var` variable to a file named `var_out.txt` in the current server directory.

```
 DECLARE @cmd AS SYSNAME, @var AS SYSNAME;
SET @var = 'Hello world';
SET @cmd = 'echo ' + @var + ' > var_out.txt';

EXECUTE master..xp_cmdshell @cmd;
```

The following example writes the contents of the current directory to a file named `dir_out.txt` in the current server directory.

```
 DECLARE @cmd AS SYSNAME, @var AS SYSNAME;
SET @var = 'dir /p';
SET @cmd = @var + ' > dir_out.txt';

EXECUTE master..xp_cmdshell @cmd;
```

- General extended stored procedures (Transact-SQL)
- xp_cmdshell (server configuration option)
- Surface area configuration
- sp_xp_cmdshell_proxy_account (Transact-SQL)
