

Control Who Connects to SQL Server and Who Does Not

By Rick Obsitnik



Rick Obsitnik

About the Author

Rick, a specialist in database application development, who recently made the change to database administration, has worked in IT for 25 years in nearly all areas excluding sales! Quite an achievement! He currently works in a SQL Server 2005, ESRI ArcGIS SDE 9.3 environment with Visual Studio 2005. His appetite for knowledge in all things IT grows daily, although his first love is databases. His interest in humor and the absurd led to the creation of the [Sniglets](#) section on the SQL Server Club web site.

You can read more articles and blogs by Rick on his [Critical Status blog](#), which was designed to keep track of and access his database and development related thoughts, ideas and issues.

Everyday we Data Shepherds must deal with various mundane issues. This could be backing up a database due to changes to be made to production or restoring a database due to testing by a developer or killing processes due to a user's orphaned connection to a dropped application.

You may be working on one server but the block may occur on another server. You may need to perform some after hours processing but there could be users still connected. It would be nice to just call a stored proc, enter a server name and database name and have it drop all general user connections. Well I'm supplying you with a little more artillery in your DBA arsenal below.

The key to any cross server activity is [linked servers](#) as I've spoken of and given examples in previous articles. I like to work off of one of my servers. Generally, it's a production server but I do a bulk of my daily administration there. I also work off of a couple of Development servers.

So it's a good thing to be able to quickly run a script or command against one or more remote servers. Here I'm going to use a script for finding blocks which can be scheduled to run frequently to analyze possible problems before your phone rings from a baffled user or angry department manager. Below is some T-SQL code for finding blocks.

```
CREATE PROCEDURE [dbo].[spFIND_BLOCKS]
-- Add the parameters for the stored procedure here
@SERVER NVARCHAR(35),
@DBNAME NVARCHAR(35)
```

AS

```
BEGIN
```

```
DECLARE
```

```
    @COMMAND NVARCHAR(4000)
```

```
SELECT @COMMAND =
```

```
    ,
```

```
    SELECT
```

```
        D.NAME,  
        SP.*
```

```
    FROM
```

```
        [' + @SERVER + '].[MASTER].SYS.SYSPROCESSES SP,  
        [' + @SERVER + '].[MASTER].SYS.DATABASES D
```

```
    WHERE
```

```
        SP.DBID = D.DATABASE_ID AND  
        D.NAME = ' + CHAR(39) + @DBNAME + CHAR(39) + ' AND  
        [BLOCKED] <> 0
```

```
    ORDER BY
```

```
        SP.BLOCKED,  
        D.NAME,  
        SP.LOGINAME
```

```
    ;
```

```
EXEC(@COMMAND);
```

```
END
```

No rocket science here. This queries the system processes running against a particular database, based on the input parameters of server instance and specific database. You can use these values to manually kill processes in the instance's Activity Monitor found the instance Management object in the Object Explorer in SSMS. Or, you can pass these values on to another proc such as the below Kill Process proc.

```
CREATE PROCEDURE [dbo].[spKILL_PROCESSES]
```

```
    @SERVER NVARCHAR(35),  
    @DBNAME NVARCHAR(35),  
    @LOGIN NVARCHAR(100),  
    @STARTLOGINTIME NVARCHAR(22),  
    @ENDLOGINTIME NVARCHAR(22)
```

```
AS
```

```
BEGIN
```

```
DECLARE
```

```
    @LOGINCLAUSE NVARCHAR(500),  
    @BEGINDT NVARCHAR(500),  
    @ENDDT NVARCHAR(500),  
    @COMMAND NVARCHAR(4000),  
    @COMMAND2 NVARCHAR(4000),
```

```

        @SPID SMALLINT

    SELECT @LOGINCLAUSE =
        CASE
            WHEN @LOGIN IS NOT NULL THEN 'AND LOGINAME = ' + CHAR(39) +
@LOGIN + CHAR(39) + ''
            ELSE ''
        END

    PRINT '@LOGINCLAUSE = ' + @LOGINCLAUSE

    IF @STARTLOGINTIME IS NOT NULL

        BEGIN

            SELECT @BEGINDT =
                'AND LOGIN_TIME >= ' + CHAR(39) + @STARTLOGINTIME +
CHAR(39) + ''
            PRINT '@STARTLOGINTIME = ' + @STARTLOGINTIME

        END

    ELSE

        BEGIN

            SET @BEGINDT = ''

            IF @ENDLOGINTIME IS NOT NULL

                BEGIN

                    SELECT @ENDDT =
                        'AND LOGIN_TIME <= ' + CHAR(39) +
@ENDLOGINTIME + CHAR(39) + ''
                    PRINT '@ENDLOGINTIME = ' + @ENDLOGINTIME

                END

            ELSE

                SET @ENDDT = ''

        END

    SELECT @COMMAND =
        ,
        DECLARE c CURSOR FORWARD_ONLY STATIC FOR
        SELECT
            SPID
        FROM
            [' + @SERVER + '].MASTER.SYS.SYSPROCESSES
        WHERE

```

```

DBID =
    (SELECT
        DATABASE_ID
    FROM
        [' + @SERVER + '].MASTER.SYS.DATABASES
    WHERE
        [NAME] = ' + CHAR(39) + @DBNAME +
CHAR(39) + ') AND

CMD NOT IN
    (' + CHAR(39) + 'CHECKPOINT SLEEP' + CHAR(39) + ', '
    + CHAR(39) + 'LAZY WRITER' + CHAR(39) + ', '
    + CHAR(39) + 'LOCK MONITOR' + CHAR(39) + ', '
    + CHAR(39) + 'SIGNAL HANDLER' + CHAR(39) + ') ' +
@LOGINCLAUSE +
@BEGINDT +
@ENDDT +
';'

PRINT @COMMAND
EXEC(@COMMAND);

OPEN c
FETCH NEXT FROM
    c
INTO
    @SPID
SELECT @COMMAND = ''

WHILE @@FETCH_STATUS = 0
    BEGIN

        SELECT @COMMAND = 'KILL ' + CAST(@SPID AS NVARCHAR(5))
        PRINT @COMMAND
        PRINT ''
        PRINT ''
        SELECT @COMMAND2 =
            'EXEC (' + CHAR(39) + @COMMAND + CHAR(39) + ') AT [' +
@SERVER + '];'

        PRINT @COMMAND2
        PRINT ''
        PRINT ''
        EXEC(@COMMAND2);
        FETCH NEXT FROM
            c
        INTO
            @SPID

    END

CLOSE c
DEALLOCATE c

```

END

This particular proc can kill processes base on input parameters of an instance, database, a particular login and a login start and/or end time frame. Combined, you can create a script that searches servers and kill offending processes based on some analysis for valid blocking. The code is pretty straight forward in that you search based on the above input parameters for offending process ids. You can then loop through the result set to run the KILL command for each Process ID.

Again, this is not rocket science but a little bit of preventative measures can reduce headaches. This can also quickly defuse issue the daily phone calls of faulty processes you cannot control. You CAN control who connects or not... ;-)