**Automate SQL Server Log Monitoring**

**By** [**Willem Gossink**](http://www.sqlservercentral.com/Authors/Articles/Willem__Gossink/426028/)**, 2015/10/30 (first published: 2013/09/16)**

**FROM:** [**http://www.sqlservercentral.com/articles/SQL+Server+error+log/101955/**](http://www.sqlservercentral.com/articles/SQL+Server+error+log/101955/)

**Introduction**

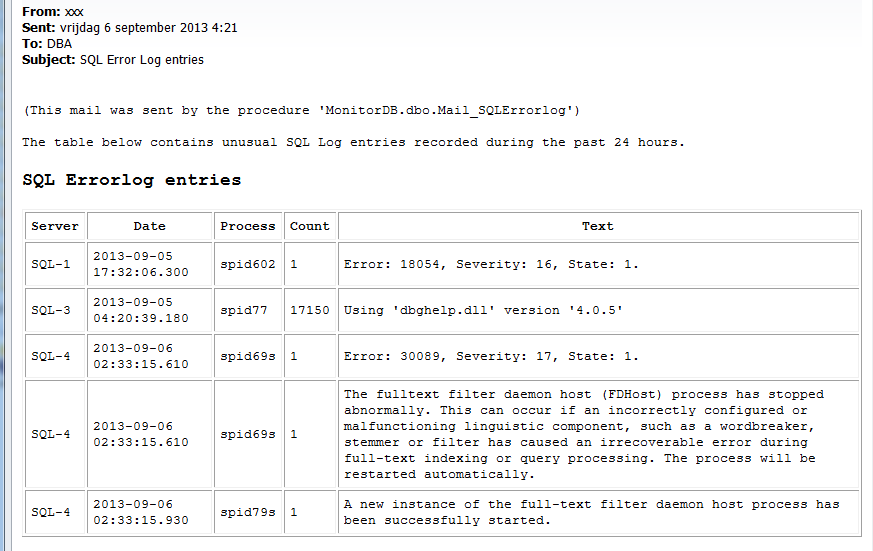
Do you scan the SQL Error Logs of all your SQL production servers on a daily basis? Do you monitor the logs for failed logins? Without tools, this can be a challenge because the bulk of the log messages can usually be ignored. The solution described below requires no external tooling and may help you not to miss out on important error messages and failed logins. It is designed to run from a monitor server and poll linked servers but can also be used ‘locally’ and run on a per-server basis.

**Setup**

The core of the solution is a stored procedure which uses XP\_READERRORLOG to retrieve the last 24 hours of messages from a given SQL server. It can do this for a number of SQL servers and will check older versions of the server log if the current version contains less than 24 hours of messages (this can happen when SQL is restarted or the log is reinitialized). In case of identical messages, the message is logged only once, but the number of occurrences is recorded.

Many error messages are informational and others may be ‘expected’ in your environment. You probably want to be warned only about the unexpected ones, so a stored procedure is included which filters the results in the holding table. The basis of the filter is an ‘exclusion table’: a table with (parts of) messages you want to exclude. You can define general exclusions (all servers, any time) or specific ones (only server X, or only server Y between 02:15 and 04:30).

Finally, all remaining log entries for all servers combined are emailed in an easy-to-read tabular layout:



http://www.sqlservercentral.com/Resources/Images/zoom.gif[Zoom in](javascript:;)  |  [Open in new window](javascript:;)

**Setup in More Detail**

Below, setup is described for a multi-server situation. This implies having/choosing a monitor server with linked server access to all SQL servers whose error logs you want to monitor. The linked servers need to have the option 'RPC out' set to 'True'. Having a separate monitor server is not technically necessary. If you do not have the resources, you can use one of your regular SQL servers as monitor server.

On your monitor server (SQL 2005/2008(R2)):

* Create a small (e.g. 10 MB) database called 'MonitorDB'. Create one or more linked servers on the monitor server and make sure you can access them with the account used for the check(s). It is probably easiest to start with a sysadmin account and narrow down as needed if you find the solution useful.
* Make sure you have SQL Database Mail configured; record the name of a mail profile you can use to email the results of the check.
* Create the holding table to contain the error log records of all servers:

USE [Monitordb]

GO

CREATE TABLE [dbo].[SQLErrorLog]

(

[ServerName] [SYSNAME],

[LogDate] [DATETIME] NULL,

[ProcessInfo] [VARCHAR](50) NULL,

[Text] [VARCHAR](2000) NULL,

[Count] [INT] NULL

);

* Create the exclusion table:

USE [Monitordb]

GO

CREATE TABLE [dbo].[SQLErrorLog\_Exclude]

(

[Text] [VARCHAR](200) NOT NULL,

[ServerName] [SYSNAME] NULL,

[StartHour] [SMALLINT] NULL,

[EndHour] [SMALLINT] NULL

);

* Populate the exclusion table to avoid warnings about ‘expected’ log entries. You may want to check out this script for some ideas. Tailor to your needs

USE [Monitordb]

TRUNCATE TABLE [SQLErrorLog\_exclude]

;

/\*

Columns and order:

1) Error Text 2) Server name 3) Start hour 4) End hour

Examples:

Exclude all occurrences of 'error text' on any server, any time

INSERT INTO [SQLErrorLog\_exclude]

SELECT 'error text', null, null, null

Only exclude an error if it occurs on ServerA

INSERT INTO [SQLErrorLog\_exclude]

SELECT 'error text', 'ServerA', null, null

Only exclude an error if it occurs on ServerA between 7PM and 9PM

INSERT INTO [SQLErrorLog\_exclude]

SELECT 'error text', 'ServerA', 19, 21

\*/

INSERT INTO [SQLErrorLog\_exclude]

SELECT '(c)', null,null,null

UNION ALL SELECT 'Microsoft SQL Server', null,null,null

UNION ALL SELECT 'A self-generated certificate was successfully loaded for encryption', null,null,null

UNION ALL SELECT 'All rights reserved', null,null,null

UNION ALL SELECT 'Authentication mode is MIXED', null,null,null

UNION ALL SELECT 'backed up', null,null,null

UNION ALL SELECT 'changed from 0 to 0', null,null,null

UNION ALL SELECT 'Clearing tempdb database', null,null,null

UNION ALL SELECT 'Database mirroring has been enabled on this instance of SQL Server', null,null,null

UNION ALL SELECT 'Dedicated admin connection support was established', null,null,null

UNION ALL SELECT 'FILESTREAM: effective level = 0, configured level = 0', null,null,null

UNION ALL SELECT 'found 0 errors and repaired 0 errors', null,null,null

UNION ALL SELECT 'informational message', null,null,null

UNION ALL SELECT 'Log was restored', null,null,null

UNION ALL SELECT 'Logging SQL Server messages in file', null,null,null

UNION ALL SELECT 'provider is ready to accept connection', null,null,null

UNION ALL SELECT 'Registry startup parameters', null,null,null

UNION ALL SELECT 'Service Broker manager has started', null,null,null

UNION ALL SELECT 'Server is listening on', null,null,null

UNION ALL SELECT 'Server process ID is', null,null,null

UNION ALL SELECT 'Starting up database', null,null,null

UNION ALL SELECT 'System Manufacturer: ''VMware, Inc.''', null,null,null

UNION ALL SELECT 'The Database Mirroring protocol transport is now listening', null,null,null

UNION ALL SELECT 'The error log has been reinitialized', null,null,null

UNION ALL SELECT 'The Service Broker protocol transport is disabled or not configured', null,null,null

UNION ALL SELECT 'Using locked pages for buffer pool', null,null,null

UNION ALL SELECT 'dbname:rtc: RtcNightlyMaint executed', null,null,null

UNION ALL SELECT 'Online CPU addition is not supported ', 'scom-sql',null,null

UNION ALL SELECT 'Warning: Failure to calculate super-latch promotion threshold', null,null,null

UNION ALL SELECT 'Using ''dbghelp.dll'' version ''4.0.5''', null,null,null

;

* Create the main stored procedure to read the error logs:

USE [MonitorDB]

GO

CREATE PROCEDURE [dbo].[Retrieve\_LinkedServer\_ErrorLog]

@Server\_Name SYSNAME = @@SERVERNAME

, @Start\_Time DATETIME = NULL

, @End\_Time DATETIME = NULL

AS

/\* Procedure to retrieve SQL error log entries from a linked server for central monitoring purposes.

This procedure is run from a central monitor server.

Parameters:

@Server\_Name: name of a linked server as data source

@Start\_Time : date/time of first Error Log record to be retrieved. Default = current date/time - 1 day

@End\_Time : date/time of last Error Log record to be retrieved. Default = current date/time

Steps:

1) Create table in tempdb on linked server to hold records or truncate table if it already exists.

2) Execute XP\_READERRORLOG on linked server with a start and end date. The current log file and 6 previous versions are scanned,

in order not to miss records in case of a new log file / new log files.

3) Copy the records to a holding table on the monitor server, while doing some aggregation

(in case of records with identical 'text', the oldest date/time and SPID value are retained).

\*/

IF @Start\_Time IS NULL

SELECT @Start\_Time = GETDATE()-1

IF @End\_Time IS NULL

SELECT @End\_Time = GETDATE()

DECLARE @cmd NVARCHAR(MAX)

SELECT @cmd = '

DECLARE @a NVARCHAR(MAX);

SELECT @a = ''SET NOCOUNT ON;

IF NOT EXISTS (SELECT 1 FROM tempdb.sys.objects WHERE name = ''''TempSQLErrorLog'''' AND type = ''''U'''')

BEGIN

CREATE TABLE tempdb.dbo.TempSQLErrorLog (

[LogDate] DATETIME,

[ProcessInfo] VARCHAR(50),

[Text] VARCHAR(1000) )

END

ELSE

BEGIN

TRUNCATE TABLE tempdb.dbo.TempSQLErrorLog;

END

DECLARE @no SMALLINT

SELECT @no = 0

WHILE (@no < 7)

BEGIN

INSERT INTO tempdb.dbo.TempSQLErrorLog

EXEC master.SYS.XP\_READERRORLOG @no, 1, '''''''' , '''''''' , '

+ '''''' + CONVERT(VARCHAR(19),@start\_time,120) + '''''' + ', '

+ '''''' + CONVERT(VARCHAR(19),@end\_time,120) + '''''' + '

SELECT @no = @no + 1

END

''

EXEC [' + @server\_name + '].master.dbo.sp\_executesql @statement = @a -- execute command stored in @a on linked server

'

-- PRINT @cmd -- for debug purposes

EXEC (@cmd) -- exec the nested exec statement

-- Copy remote records to local holding table and add server name.

-- For entries with identical value in Text field, only the record with lowest Logdate and Processinfo value is copied, and Count field is incremented.

SELECT @cmd = '

SET NOCOUNT ON;

INSERT INTO Monitordb.dbo.SQLErrorLog

SELECT ''' + @Server\_Name + '''

, MIN(Logdate)

, MIN(Processinfo)

, Text

, COUNT(\*)

FROM [' + @Server\_Name + '].tempdb.dbo.TempSQLErrorLog

GROUP BY [Text]

'

-- PRINT @cmd -- for debug purposes

EXEC master.dbo.sp\_executesql @cmd -- execute command stored in @cmd

* Create the procedure to filter the error logs:

USE [MonitorDB]

GO

CREATE PROCEDURE [dbo].[Cleanup\_Errorlog\_Holding\_Table]

AS

/\*\*\* Script to clean up the SQL Errorlog holding table, based on a table containing entries to be excluded.

The exclude table contains entries that need not be reported, so these can be deleted from the

holding table. After this 'cleanup', all remaining entries in the holding table will be mailed as a

notification.

An exclusion MUST contain

- a text string. Records in the errorlog holding table containing this text

(+ any other text surrounding it) will be deleted.

An exclusion MAY contain

- a 'starthour' and 'endhour'. All entries recorded between starthour and endhour

and containing the specified text will be deleted.

- a server name. All entries containing the specified text, coming from this server

and recorded between starthour and endhour will be deleted

\*/

SET NOCOUNT ON

DECLARE cleanup CURSOR FOR

SELECT [Text]

, ServerName

, StartHour

, EndHour

FROM dbo.SQLErrorLog\_Exclude

DECLARE @text VARCHAR(200)

, @servername SYSNAME

, @starthour SMALLINT

, @endhour SMALLINT

OPEN cleanup

FETCH NEXT FROM cleanup INTO @text, @servername, @starthour, @endhour

WHILE @@FETCH\_STATUS = 0

BEGIN

IF @starthour IS NULL AND @endhour IS NULL

-- Delete all records with this text, regardless of time reported

DELETE dbo.SQLErrorLog

WHERE CHARINDEX(@text,[Text]) > 0

AND (servername = @servername

OR @servername IS NULL)

ELSE IF (@starthour = @endhour) OR (@starthour IS NULL AND @endhour IS NOT NULL)

-- Delete records on the basis of @endhour

DELETE dbo.SQLErrorLog

WHERE CHARINDEX(@text,[Text]) > 0

AND (servername = @servername

OR @servername IS NULL)

AND DATEPART(hh, [LogDate]) = @endhour

ELSE IF (@starthour IS NOT NULL

AND @endhour IS NULL)

-- Delete records on the basis of @starthour

DELETE dbo.SQLErrorLog

WHERE CHARINDEX(@text,[Text]) > 0

AND (servername = @servername

OR @servername IS NULL)

AND DATEPART(hh, [LogDate]) = @starthour

ELSE IF @starthour < @endhour

-- Delete records >= @starthour and <= @endhourur

DELETE dbo.SQLErrorLog

WHERE CHARINDEX(@text,[Text]) > 0

AND (servername = @servername

OR @servername IS NULL)

AND DATEPART(hh, [LogDate]) >= @starthour

AND DATEPART(hh, [LogDate]) <= @endhour

ELSE IF @starthour > @endhour

-- Delete records >= @starthour or <= @endhour

DELETE dbo.SQLErrorLog

WHERE CHARINDEX(@text,[Text]) > 0

AND (servername = @servername

OR @servername IS NULL)

AND (DATEPART(hh, [LogDate]) >= @starthour

OR DATEPART(hh, [LogDate]) <= @endhour)

ELSE -- undefined condition: log error

BEGIN

DECLARE @proc VARCHAR(500)

SELECT @proc = DB\_NAME() + '.' + OBJECT\_SCHEMA\_NAME (@@PROCID) + '.' + OBJECT\_NAME(@@PROCID)

RAISERROR ( 'Proc: %s: Undefined condition found in exclusion table, please contact DBA. Offending line: %s, %s, %u, %u', 10, 1

, @proc, @servername, @text, @starthour ,@endhour) WITH LOG

END

FETCH NEXT FROM cleanup INTO @text, @servername, @starthour, @endhour

END

CLOSE cleanup

DEALLOCATE cleanup

* Create the procedure to mail the results:

USE [MonitorDB]

GO

CREATE PROC [dbo].[Mail\_SQLErrorlog]

@recipients VARCHAR(MAX) = 'someone@company.com'

, @profile\_name SYSNAME = '<your\_mail\_profile>'

AS

/\* Script to mail 'unusual' SQL Error Log entries, after collection in a holding table on the monitor server.

This script should be run after retrieval of linked server error logs (procedure 'Retrieve\_LinkedServer\_ErrorLog')

and removal of trivial messages (procedure 'Cleanup\_Errorlog\_Holding\_Table')

\*/

-- Send message if the error log holding table contains entries other than failed logins

IF EXISTS (SELECT 1 FROM dbo.SQLErrorlog WHERE CHARINDEX('Login failed for user', [text]) = 0)

BEGIN

DECLARE @subj VARCHAR(200)

, @xml NVARCHAR(MAX)

, @body NVARCHAR(MAX)

SELECT @subj = 'SQL Error Log entries'

SET @body = '<html><body>

<p><font size="2" face="monaco">

(This mail was sent by the procedure ''' + DB\_NAME() + '.' + OBJECT\_SCHEMA\_NAME (@@PROCID) + '.' + OBJECT\_NAME(@@PROCID) + ''') <BR><BR>

The table below contains unusual SQL Log entries recorded during the past 24 hours.

<H3>SQL Errorlog entries</H3>

<table border="1" cellpadding="5">

<p><font size="2" face="monaco">

<tr>

<th> Server </th> <th> Date </th> <th> Process </th> <th> Count </th> <th> Text </th> </tr>'

SET @xml = CAST(

(SELECT [ServerName] AS 'td',''

, CONVERT(CHAR(30),[LogDate],21) AS 'td',''

, [Processinfo] AS 'td',''

, [Count] AS 'td',''

, [Text] AS 'td',''

FROM dbo.SQLErrorlog

WHERE CHARINDEX('Login failed for user', [text]) = 0 -- no regular failed login warnings

ORDER BY [ServerName], [Count] DESC, [LogDate] DESC

FOR XML PATH('tr'), ELEMENTS)

AS NVARCHAR(MAX))

SET @body = @body + @xml +'</table></body></html>'

EXEC msdb.dbo.sp\_send\_dbmail -- Send email

@profile\_name = @profile\_name

, @recipients = @recipients

, @subject = @subj

, @body = @body

, @body\_format ='HTML'

END

You are now ready to run a first check. As a test, from a query window, issue the commands below:

USE MonitorDB

EXEC [Retrieve\_LinkedServer\_ErrorLog]

The message ‘Command completed successfully’ should appear. You have now captured the error log info of your monitor server.

Next, try one of your linked servers. From your monitor server, issue the same command but add a server name, e.g. ‘ServerA’:

EXEC [Retrieve\_LinkedServer\_ErrorLog] 'ServerA'

(If you use a Windows login and run into error 18456: ‘Login failed for user 'NT AUTHORITY\ANONYMOUS LOGON’’, you can either try to fix the related SPN and/or ‘double hop’ problem or use a SQL login with sufficient privileges on all servers to avoid any Kerberos problems).

If this is successful, the error log records of both servers have been collected on the monitor server. Now, remove the records you do not want to show up:

EXEC Cleanup\_Errorlog\_Holding\_Table

Finally, send the mail

EXEC [Mail\_SQLErrorlog]

@recipients = '<email adress>' ,

@profile\_name = '<your sql mail profile>'

Unless the filter has removed all messages you should now see ‘Mail queued’.

Please note that failed logins are not included in the mail. This is because in our company we report on these separately, in a modified format including a column for source IP address. If you do want failed logins included in the mail, simply remove the two occurrences of the ‘where’ statement from the code: WHERE CHARINDEX('Login failed for user', [text]) = 0

To separately report on failed logins, you can use this procedure:

USE [MonitorDB]

GO

CREATE PROC [dbo].[Mail\_Failed\_Logins]

@recipients VARCHAR(MAX) = 'someone@company.com'

, @profile\_name SYSNAME = '<your\_mail\_profile>'

AS

/\* Script to mail failed login attempts recorded in SQL Error Logs, after collection in a holding table on the monitor server.

This script should be run after retrieval of linked server error logs (procedure 'Retrieve\_LinkedServer\_ErrorLog')

and removal of trivial messages (procedure 'Cleanup\_Errorlog\_Holding\_Table')

Messages other than failed login attempts are excluded; these are mailed by a separate procedure ('Mail\_SQLErrorlog').

\*/

IF EXISTS (SELECT 1 FROM dbo.SQLErrorlog WHERE CHARINDEX('Login failed for user', [Text]) > 0)

BEGIN

DECLARE @subj VARCHAR(200)

, @body NVARCHAR(MAX)

, @xml NVARCHAR(MAX)

SELECT @subj = 'Failed SQL logins'

SET @body = '<html><body>

<p><font size="2" face="monaco">

(This mail was sent by the procedure ''' + DB\_NAME() + '.' + OBJECT\_SCHEMA\_NAME (@@PROCID) + '.' + OBJECT\_NAME(@@PROCID) + ''') <BR><BR>

The table below contains SQL failed logins recorded during the past 24 hours.

<H3>SQL Failed Logins</H3>

<table border="1" cellpadding="5">

<p><font size="2" face="monaco">

<tr>

<th> Server </th> <th> Date </th> <th> Login Name </th> <th> Count </th> <th> Client Name </th> <th> Extra Info </th> </tr>'

SELECT @xml = CAST(

(SELECT LEFT([ServerName],25) AS 'td','' -- Server

, CONVERT(CHAR(30),[LogDate],21) AS 'td','' -- Date

, CASE

WHEN CHARINDEX('''''' , [Text]) > 0

THEN ''

ELSE LEFT(SUBSTRING([Text], CHARINDEX('''' , [Text]) + 1, CHARINDEX('''', [Text], CHARINDEX('''' , [Text]) + 1) - CHARINDEX('''' , [Text]) - 1),30)

END AS 'td','' -- Login Name

, [COUNT] AS 'td','' -- Count

, LEFT(SUBSTRING([Text], CHARINDEX('[CLIENT', [Text]) + 9, CHARINDEX(']', [Text]) - CHARINDEX('[CLIENT', [Text]) - 9),17)

AS 'td','' -- Client Name

, CASE

WHEN CHARINDEX('''. [',[Text]) = 0

-- print extra info if any is found between user name (terminated by single quote, full stop and space) and CLIENT info (starting with left square bracket

THEN LEFT(SUBSTRING([Text], CHARINDEX('''. ',[Text]) + 3, CHARINDEX('. [',[Text]) - CHARINDEX('''. ',[Text]) - 1),100)

-- print the info between the user name and client IP address

ELSE ''

END AS 'td','' -- Extra Info

FROM dbo.SQLErrorlog

WHERE CHARINDEX('Login failed for user', [Text]) > 0

ORDER BY 1, 2, 3, 5

FOR XML PATH('tr'), ELEMENTS )

AS NVARCHAR(MAX))

SET @body = @body + @xml +'</table></body></html>'

EXEC msdb.dbo.sp\_send\_dbmail -- Send email

@profile\_name = @profile\_name

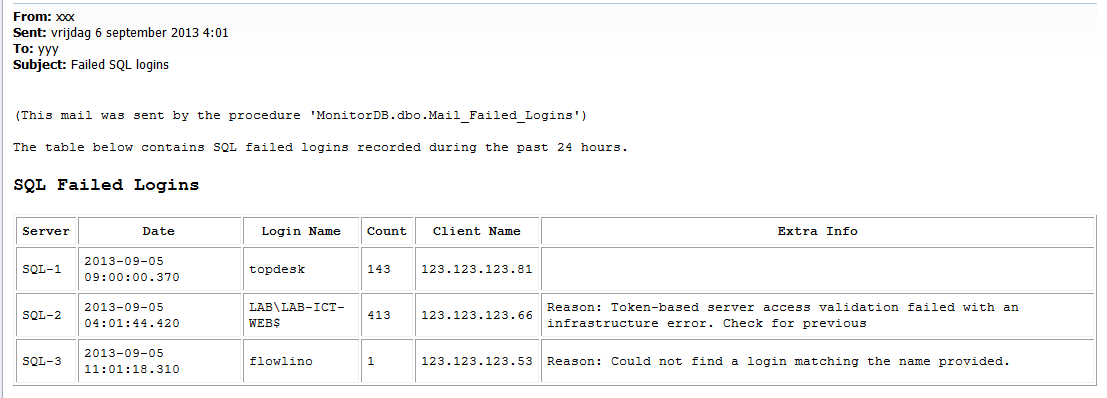
, @recipients = @recipients

, @subject = @subj

, @body = @body

, @body\_format ='HTML'

END

An example of its output:

How do you make the procedures in Listing 4 run against all your linked servers? You can either write a FOR loop, or use a wrapper procedure like this, which I use to run a bunch of other SP’s against the servers defined on my monitor server:

USE [MonitorDB]

GO

CREATE PROC [dbo].[Check\_Remote\_Servers]

@proc SYSNAME

AS

/\* Execute a stored procedure against all linked servers defined on this server.

\*\* Params: @proc = name of stored procedure to be executed

\*\* Example: EXEC Check\_Remote\_Servers @proc = 'Retrieve\_LinkedServer\_ErrorLog'

\*\* This will execute the procedure 'Retrieve\_LinkedServer\_ErrorLog' against all linked servers

\*/

SET NOCOUNT ON

DECLARE @server\_name SYSNAME

, @cmd NVARCHAR(MAX)

DECLARE @linked\_servers TABLE ([server\_name] SYSNAME)

INSERT @linked\_servers

SELECT name

FROM master.sys.servers

SELECT @server\_name = MIN([server\_name]) FROM @linked\_servers

WHILE @server\_name IS NOT NULL

BEGIN

PRINT 'Starting proc ''' + @proc + ''' on server ''' + @server\_name + ''''

SET @cmd = 'EXEC [' + @proc + '] @Server\_Name = [' + @server\_name + '] '

EXEC sp\_executesql @cmd

DELETE @linked\_servers WHERE [server\_name] = @server\_name

SELECT @server\_name = MIN([server\_name]) FROM @linked\_servers

END

If you have this in place, you can combine the commands in a SQL Agent job, to run each night from MonitorDB:

-- empty holding table to allow for new content

TRUNCATE TABLE SQLErrorlog

-- Collect remote error log entries

EXEC Check\_Remote\_Servers

@proc = 'Retrieve\_LinkedServer\_ErrorLog'

-- remove unwanted entries from holding table

EXEC Cleanup\_Errorlog\_Holding\_Table

-- mail error log records

EXEC [Mail\_SQLErrorlog]

@recipients = '<email adress>'

, @profile\_name = '<your sql mail profile>'

-- mail failed logins

EXEC [Mail\_Failed\_Logins]

@recipients = '<email adress>'

, @profile\_name = '<your sql mail profile>'