How to receive Deadlock information automatically via email

By Geoff Albin, 2010/10/08

Receive Deadlock detail via email.

I had spent many hours on Google trying to find the best way to send a notification on deadlock occurrences. While it was a fairly straight forward process on how we get notified a deadlock has occurred, I wanted a bit more. I not only wanted to be told when the Deadlock occurred, I wanted to also be emailed the actual Deadlock information.

Every time a deadlock occurs in SQL Server, the detailed info about things like the SPID, the statement that was running, who the victim was, and so on does get logged. But getting the generic alert that SQL Server can create simply will tell you "something" has occurred. It would be the dutiful DBA's job to log into the server in question and dig into the Error Log to get the Deadlock details.

Since capturing Deadlock info is not turned on by default. We do have to do two things in order to make the scripts I have written work properly.

Requirement number one

The first requirement is to turn on the appropriate Trace Flags. We do that by running the following command.

```
DBCC TRACEON (3605,1204,1222,-1)
```

A brief overview to what those flags do is listed below.

- 3605 = write what we want to the error log.
- 1204 = Capture Deadlock Events.
- 1222 = Capture Deadlock Events with more info (SQL 2005 and higher)

It is important to note that setting trace flags this way will only work until the next time SQL Server is restarted. If you want to make sure your trace flags are always set this way, you can edit the startup options of SQL Server by using the -T switch or creating a SQL Agent job to run that code when the Agent starts.

Requirement number two

The second requirement is to ensure you have DB Mail setup and working. I will provide no detail on how to accomplish that. It will just be assumed that you have a working DB Mail profile.

Lets get started.

So now we have trace flags set and DB Mail is working we are ready to get into how we send the Deadlock information to an email address when a Deadlock occurs.

Since the structure of the error log changed in SQL 2005, we have two ways of doing this. Each method is basically the same, however, pay attention to where you deploy this script. I have included detailed comments so you can follow along.

For SQL Server 2000

```
--== This is for SQL 2000. ==--
--== We will create a temporary table to hold the error log detail. ==--
--== Before we create the temporary table, we make sure it does not already exist. ==--
 IF OBJECT_ID('tempdb.dbo.ErrorLog') IS Not Null
 BEGIN
 DROP TABLE tempdb.dbo.ErrorLog
 END
 --== We have checked for the existence of the temporary table and dropped it if it was there. ==--
 --== Now, we can create the table called tempdb.dbo.ErrorLog ==--
 CREATE TABLE tempdb.dbo.ErrorLog (Id int IDENTITY (1, 1) NOT NULL,
 ERRORLOG VARCHAR(4000), ContRow int)
--== We create a 3 column table to hold the contents of the SQL Server Error log. ==--
--== Then we insert the actual data from the Error log into our newly created table. ==--
 INSERT INTO tempdb.dbo.ErrorLog
 EXEC master.dbo.sp_readerrorlog
--== With our table created and populated, we can now use the info inside of it. ==--
 BEGIN
--== Set a variable to get our instance name. ==--
--== We do this so the email we receive makes more sense. ==--
 declare @servername nvarchar(150)
 set @servername = @@servername
--== We set another variable to create a subject line for the email. ==--
 declare @mysubject nvarchar(200)
 set @mysubject = 'Deadlock event notification on server '+@servername+'.'
 --== Now we will prepare and send the email. Change the email address to suite your environment. ==--
 exec master.dbo.xp_sendmail @recipients = 'DBA_Super_Hero@email.com',
 @subject = @mysubject,
 @message = 'Deadlock has occurred. View attachment to see the deadlock info',
 @query = 'select ERRORLOG from tempdb.dbo.ErrorLog where Id >= (select TOP 1 Id from tempdb.dbo.ErrorLog WHERE ERRORLOG Like ''%Deadl
 @width = 600,
 @attach_results = 'True',
 @no_header = 'True'
 END
 --== Clean up our process by dropping our temporary table. ==--
 DROP TABLE tempdb.dbo.ErrorLog
```

And for all other version, (2005, 2008, 2008 R2)

```
--== This is for SQL 2005 and higher. ==--
--== We will create a temporary table to hold the error log detail. ==--
--== Before we create the temporary table, we make sure it does not already exist. ==--
 IF OBJECT_ID('tempdb.dbo.ErrorLog') IS Not Null
 BEGIN
 DROP TABLE tempdb.dbo.ErrorLog
 END
 --== We have checked for the existence of the temporary table and dropped it if it was there. ==--
 --== Now, we can create the table called tempdb.dbo.ErrorLog ==--
CREATE TABLE tempdb.dbo.ErrorLog (Id int IDENTITY (1, 1) NOT NULL,
logdate DATETIME, procInfo VARCHAR(10), ERRORLOG VARCHAR(MAX))
--== We create a 3 column table to hold the contents of the SQL Server Error log. ==--
--== Then we insert the actual data from the Error log into our newly created table. ==--
 INSERT INTO tempdb.dbo.ErrorLog
 EXEC master.dbo.sp_readerrorlog
--== With our table created and populated, we can now use the info inside of it. ==--
 BEGIN
--== Set a variable to get our instance name. ==--
--== We do this so the email we receive makes more sense. ==--
 declare @servername nvarchar(150)
 set @servername = @@servername
--== We set another variable to create a subject line for the email. ==--
 declare @mysubject nvarchar(200)
 set @mysubject = 'Deadlock event notification on server '+@servername+'.'
 --== Now we will prepare and send the email. Change the email address to suite your environment. ==--
 EXEC msdb.dbo.sp_send_dbmail @recipients='DBA_Super_Hero@email.com',
 @subject = @mysubject,
 @body = 'Deadlock has occurred. View attachment to see the deadlock info',
 @query = 'select logdate, procInfo, ERRORLOG from tempdb.dbo.ErrorLog where Id >= (select TOP 1 Id from tempdb.dbo.ErrorLog WHERE ERR
 @query_result_width = 600,
 @attach_query_result_as_file = 1
 END
 --== Clean up our process by dropping our temporary table. ==--
 DROP TABLE tempdb.dbo.ErrorLog
```

Next Steps

In order to get those to work every time SQL Server encounters a Deadlock, we have to create a SQL Server Agent Job and a SQL Server Agent Alert. The basic approach is to create a Job that is called by the Alert service.

Let's first create a SQL Agent Job. The job will have no schedule. Note that the script below needs to be edited. If you do not run your SQL Server in mixed mode, you will need to change **@owner_login_name=N'sa'** to a user that can run the job. Also, note that the script contains an email address. You will have to enter a **valid email address**. This would be the email address of the person that will be troubleshooting the Deadlock occurrences. You will create this SQL Agent job on every instance you want to receive Deadlock info for.

SQL Server 2000 Job

```
USE [msdb]
GO
BEGIN TRANSACTION
DECLARE @ReturnCode INT
SELECT @ReturnCode = 0
IF NOT EXISTS (SELECT name FROM msdb.dbo.syscategories WHERE name=N'[Uncategorized (Local)]' AND category_class=1)
BEGIN
EXEC @ReturnCode = msdb.dbo.sp_add_category @class=N'JOB', @type=N'LOCAL', @name=N'[Uncategorized (Local)]'
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
END
DECLARE @jobId BINARY(16)
EXEC @ReturnCode = msdb.dbo.sp_add_job @job_name=N'Deadlock Job',
 @enabled=1,
 @notify_level_eventlog=0,
 @notify_level_email=0,
 @notify_level_netsend=0,
 @notify_level_page=0,
 @delete_level=0,
 @description=N'No description available.',
 @category_name=N'[Uncategorized (Local)]',
 @owner_login_name=N'sa', @job_id = @jobId OUTPUT
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
EXEC @ReturnCode = msdb.dbo.sp_add_jobstep @job_id=@jobId, @step_name=N'Deadlock has occurred.',
 @step_id=1,
 @cmdexec_success_code=0,
 @on_success_action=1,
 @on_success_step_id=0,
 @on_fail_action=2,
 @on_fail_step_id=0,
 @retry_attempts=0,
 @retry_interval=0,
 @os_run_priority=0, @subsystem=N'TSQL',
 @command=N'---- This is for SQL 2000. ==--
--== We will create a temporary table to hold the error log detail. ==--
--== Before we create the temporary table, we make sure it does not already exist. ==--
 IF OBJECT_ID(''tempdb.dbo.ErrorLog'') IS Not Null
 BEGIN
 DROP TABLE tempdb.dbo.ErrorLog
 END
 --== We have checked for the existence of the temporary table and dropped it if it was there. ==--
 --== Now, we can create the table called tempdb.dbo.ErrorLog ==--
 CREATE TABLE tempdb.dbo.ErrorLog (Id int IDENTITY (1, 1) NOT NULL,
 ERRORLOG VARCHAR(4000), ContRow int)
--== We create a 3 column table to hold the contents of the SQL Server Error log. ==--
--== Then we insert the actual data from the Error log into our newly created table. ==--
 INSERT INTO tempdb.dbo.ErrorLog
 EXEC master.dbo.sp_readerrorlog
```

```
--== With our table created and populated, we can now use the info inside of it. ==--
 BEGIN
--== Set a variable to get our instance name. ==--
--== We do this so the email we receive makes more sense. ==--
 declare @servername nvarchar(150)
 set @servername = @@servername
--== We set another variable to create a subject line for the email. ==--
 declare @mysubject nvarchar(200)
 set @mysubject = ''Deadlock event notification on server ''+@servername+''.''
 --== Now we will prepare and send the email. Change the email address to suite your environment. ==--
 exec master.dbo.xp_sendmail @recipients = ''DBA_Super_Hero@email.com'',
 @subject = @mysubject,
 @message = ''Deadlock has occurred. View attachment to see the deadlock info'',
 @query = ''select ERRORLOG from tempdb.dbo.ErrorLog where Id >= (select TOP 1 Id from tempdb.dbo.ErrorLog WHERE ERRORLOG Like ''''%De
 @width = 600,
 @attach_results = ''True'',
 @no_header = ''True''
 END
 --== Clean up our process by dropping our temporary table. ==--
 DROP TABLE tempdb.dbo.ErrorLog
',
 @database_name=N'master',
 @flags=0
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
EXEC @ReturnCode = msdb.dbo.sp_update_job @job_id = @jobId, @start_step_id = 1
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
EXEC @ReturnCode = msdb.dbo.sp_add_jobserver @job_id = @jobId, @server_name = N'(local)'
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
COMMIT TRANSACTION
GOTO EndSave
QuitWithRollback:
 IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION
EndSave:
GO
```

SQL Server 2005 and higher Job

Note that the script below needs to be edited. If you do not run your SQL Server in mixed mode, you will need to change **@owner_login_name=N'sa'** to a user that can run the job. Also, note that the script contains an email address. You will have to enter a **valid email address**. This would be the email address of the person that will be troubleshooting the Deadlock occurrences. You will create this SQL Agent job on every instance you want to receive Deadlock info for.

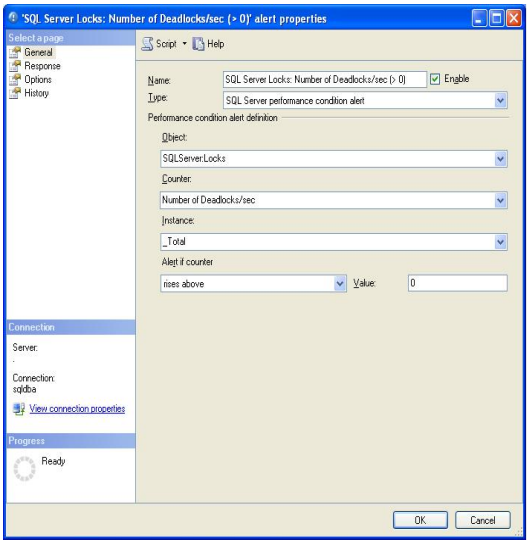
```
USE [msdb]
GO
BEGIN TRANSACTION
DECLARE @ReturnCode INT
SELECT @ReturnCode = 0
IF NOT EXISTS (SELECT name FROM msdb.dbo.syscategories WHERE name=N'[Uncategorized (Local)]' AND category_class=1)
BEGIN
EXEC @ReturnCode = msdb.dbo.sp_add_category @class=N'JOB', @type=N'LOCAL', @name=N'[Uncategorized (Local)]'
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
END
DECLARE @jobId BINARY(16)
EXEC @ReturnCode = msdb.dbo.sp_add_job @job_name=N'Deadlock Job',
 @enabled=1,
 @notify_level_eventlog=0,
 @notify_level_email=0,
 @notify_level_netsend=0,
 @notify_level_page=0,
 @delete_level=0,
 @description=N'No description available.',
 @category_name=N'[Uncategorized (Local)]',
@owner_login_name=N'sa', @job_id = @jobId OUTPUT
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
EXEC @ReturnCode = msdb.dbo.sp_add_jobstep @job_id=@jobId, @step_name=N'Deadlock has occurred.',
 @step_id=1,
 @cmdexec_success_code=0,
 @on_success_action=1,
 @on_success_step_id=0,
 @on_fail_action=2,
 @on_fail_step_id=0,
 @retry_attempts=0,
 @retry_interval=0,
 @os_run_priority=0, @subsystem=N'TSQL',
 @command=N'---- This is for SQL 2005 and higher. ==--
--== We will create a temporary table to hold the error log detail. ==--
--== Before we create the temporary table, we make sure it does not already exist. ==--
 IF OBJECT_ID(''tempdb.dbo.ErrorLog'') IS Not Null
 BEGIN
 DROP TABLE tempdb.dbo.ErrorLog
 END
 --== We have checked for the existence of the temporary table and dropped it if it was there. ==--
 --== Now, we can create the table called tempdb.dbo.ErrorLog ==--
CREATE TABLE tempdb.dbo.ErrorLog (Id int IDENTITY (1, 1) NOT NULL,
logdate DATETIME, procInfo VARCHAR(10), ERRORLOG VARCHAR(MAX))
--== We create a 3 column table to hold the contents of the SQL Server Error log. ==--
--== Then we insert the actual data from the Error log into our newly created table. ==--
 INSERT INTO tempdb.dbo.ErrorLog
 EXEC master.dbo.sp_readerrorlog
--== With our table created and populated, we can now use the info inside of it. ==--
 BEGIN
--== Set a variable to get our instance name. ==--
--== We do this so the email we receive makes more sense. ==--
 declare @servername nvarchar(150)
 set @servername = @@servername
--== We set another variable to create a subject line for the email. ==--
 declare @mysubject nvarchar(200)
```

```
 set @mysubject = ''Deadlock event notification on server ''+@servername+''.''
 --== Now we will prepare and send the email. Change the email address to suite your environment. ==--
 EXEC msdb.dbo.sp_send_dbmail @recipients=''DBA_Super_Hero@email.com'',
 @subject = @mysubject,
 @body = ''Deadlock has occurred. View attachment to see the deadlock info'',
 @query = ''select logdate, procInfo, ERRORLOG from tempdb.dbo.ErrorLog where Id >= (select TOP 1 Id from tempdb.dbo.ErrorLog WHERE EF
 @query_result_width = 600,
 @attach_query_result_as_file = 1
 END
 --== Clean up our process by dropping our temporary table. ==--
 DROP TABLE tempdb.dbo.ErrorLog
',
 @database_name=N'master',
 @flags=0
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
EXEC @ReturnCode = msdb.dbo.sp_update_job @job_id = @jobId, @start_step_id = 1
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
EXEC @ReturnCode = msdb.dbo.sp_add_jobserver @job_id = @jobId, @server_name = N'(local)'
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
COMMIT TRANSACTION
GOTO EndSave
QuitWithRollback:
 IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION
EndSave:
GO
```

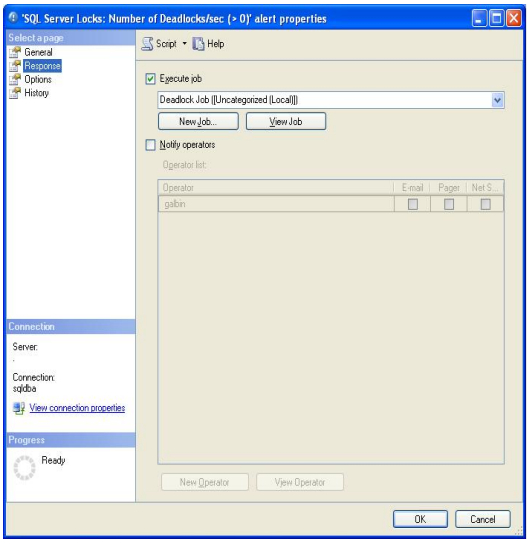
Final step

Now we have to create a SQL Agent alert to call the job we created. Open SSMS and log into the instance we are capturing Deadlock information for and expand the SQL Server Agent. Right click on the word Alerts and choose "New Alert..."

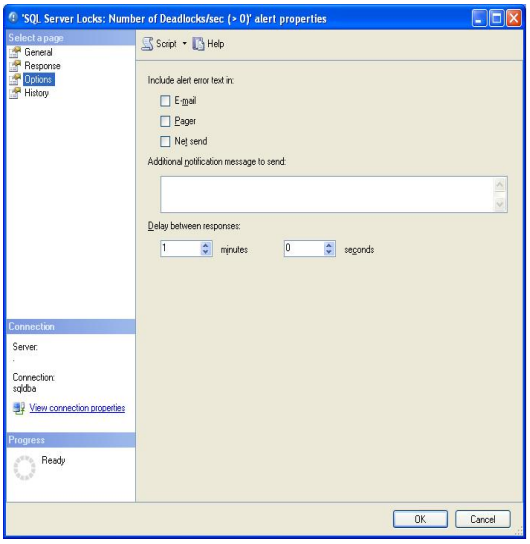
On the General page it should look like this;



On the Response page it should look like this:



On the Options page it should look like this:



That's all there is to it. The next time a Deadlock occurs on the instance you are monitoring you will you receive an email. The email you receive will have an attachment that will actually tell you what the deadlock was.
