Update multiple servers in a single bound

Working with replication, specifically merge replication, I often have the need to run a sql script on each server. I have 1 main publisher server, 2 re-publishers running SQL Standard Edition and approximately 20 subscriber servers running SQL Express that are on user laptops. A “pet peeve” of mine is repetitive work, so if I can automate it I will. In this article, I have put together a process to automate running a script in a single bound or rather from a stored procedure called by a job.

I have created a database that I use for my DBA activities that I call DBA\_Reports; I use it over and over in my articles. I’ve included the pertinent portions in this article so you can make this process work. However if you’ve already set up my other processes, then you can skip to step 2. If you are starting from scratch, please make sure that you have the following before you begin:

* DBMail enabled on your server - this is absolutely key to this process
* Enable xp\_cmdShell on each remote server, so you will be able to see what servers are on line and load them into a table (this will be discussed and used in steps 2 and 3). This is also important to this working properly.
* Depending how you are set up and what servers you are watching, you may need to create a linked server for each server. In my case, since my users are laptops with SQL Express installed on them, I had to create a linked server to each.
* Create yourself as an operator so you can receive notifications if your job fails. Personally I like to add a notification on every job I create in case it fails. This is optional.

I'm not going to go into details on how to set these up as they are readily available from other sources, however, just consider them as prerequisites.

**Step One: Create the Table and Get Servers Job**

Once you set up the above, it's time to create the tables we will use.

The “ServerList” table below will house the servers name and email address associated to it. This table will be used later for email notifications and also for verifying what servers are currently online. The server names will be populated by a job, however the email addresses will need to be manually added. Although this takes some manual setup initially, once setup, only minimal maintenance will be required if new servers come online. I have a job that runs 3 times a day that will populate this table. If a new server is inserted, I receive an email notification so I know to go and add the email address for that server. The email address in my case belongs to the owner of the server however it could be anyone supporting a server. This is the only manual intervention I have after initial setup. You will see this job later in this article. Copy/paste/Run the script below to create the ServerList table.

use DBA\_Reports

go

CREATE TABLE [dbo].[ServerList](

[Server] [varchar](128) NOT NULL,

[Server\_email] [varchar](50) NULL,

CONSTRAINT [PK\_ServerList] PRIMARY KEY CLUSTERED

( [Server] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, FILLFACTOR = 99) ON [PRIMARY])

ON [PRIMARY]

The “DBA\_Reports - Get New Servers” job will create and populate the ServerList\_Stage and populate ServerList tables. This job uses xp\_cmdshell to call the ''sqlcmd /Lc'' command. This command searches the network looking for servers where SQL Server is installed. Once it finds a server, it will insert the server name into a temp table called #Server, then as long as the server name is not null, it will create and insert into the ServerList\_Stage table. The ServerList\_Stage is just a staging table where we will use it to compare to our ServerList table later, so it is, as it says just a “staging” table. If the record does not already exist, then it will insert it. If a new record is inserted, it will send an email listing out the newly added records with the subject of ''SQL Server - New Servers Found Today”. When I received these emails, I immediately go to the ServerList table and add in an email address that is associated to that newly added server. By doing this, I will be prepared prior to my other jobs running.

After you create this job, schedule it to run every day. I have my job running 3 times a day because I have users in multiple time zones as well as logging in at all different times. Once this job is setup and an initial run has occurred edit the ServerList table and enter an email address for each server. Copy/Paste/Run the script below and make sure you change <ADD YOUR INFO HERE> to your information.

USE [msdb]

GO

BEGIN TRANSACTION

DECLARE @ReturnCode INT

SELECT @ReturnCode = 0

IF NOT EXISTS (SELECT name FROM msdb.dbo.syscategories WHERE name=N'DBA Monitoring' AND category\_class=1)

BEGIN

EXEC @ReturnCode = msdb.dbo.sp\_add\_category @class=N'JOB', @type=N'LOCAL', @name=N'DBA Monitoring'

IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

END

DECLARE @jobId BINARY(16)

EXEC @ReturnCode = msdb.dbo.sp\_add\_job @job\_name=N'DBA\_Reports - Get New Servers',

@enabled=1,

@notify\_level\_eventlog=2,

@notify\_level\_email=2,

@notify\_level\_netsend=0,

@notify\_level\_page=0,

@delete\_level=0,

@description=N'Gets the new servers that are not in the table and sends out an email for new servers',

@category\_name=N'DBA Monitoring',

@owner\_login\_name=N'sa',

@notify\_email\_operator\_name=N'<ADD YOUR INFO HERE>', @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'step1',

@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'USE [DBA\_Reports]

GO

IF EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N''[dbo].[ServerList\_Stage]'') AND type in (N''U''))

DROP TABLE [dbo].[ServerList\_Stage]

GO

USE [DBA\_Reports]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[ServerList\_Stage](

[Server] [varchar](128) NOT NULL,

CONSTRAINT [PK\_ServerList\_Stage] PRIMARY KEY CLUSTERED

(

[Server] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, FILLFACTOR = 99) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

-- Load table with server names

-- This will only work if XP\_CMDSHELL is Enabled

USE DBA\_Reports

GO

Create table #Server ( [Server] [varchar](128) )

Insert Into #Server

Exec xp\_cmdshell ''sqlcmd /Lc''

Insert Into ServerList\_Stage ([Server])

select [Server] from #Server where [Server] is not null

DROP Table #Server

GO

--Send email to with new servers found

SET nocount ON

--

DECLARE @Subject VARCHAR (100)

SET @Subject=''SQL Server - New Servers Found Today''

DECLARE @Count AS INT

SELECT @Count=COUNT(\*) FROM DBA\_Reports.dbo.ServerList\_Stage where [Server] not in (select [Server] from [ServerList])

PRINT @Count

IF @Count > 0

BEGIN

DECLARE @tableHTML NVARCHAR(MAX) ;

SET @tableHTML =

N''<table border="1">'' +

N''<tr>'' +

N''<th>Server</th>'' +

N''</tr>'' +

CAST ( ( SELECT td=[server],''''

FROM [ServerList\_Stage]

where [Server] not in (select [Server] from [ServerList])

FOR XML PATH(''tr''), TYPE

) AS NVARCHAR(MAX) ) +

N''</table>'' ;

EXEC msdb.dbo.sp\_send\_dbmail

@profile\_name = ''<ADD YOUR INFO HERE>'',

@recipients = ''<ADD YOUR INFO HERE>'',

@subject = @Subject,

@body = @tableHTML,

@body\_format = ''HTML'' ;

END

INSERT INTO [DBA\_Reports].[dbo].[ServerList]

([Server])

SELECT [Server]

FROM [DBA\_Reports].[dbo].[ServerList\_Stage]

where [Server] not in (select [Server] from [DBA\_Reports].[dbo].[ServerList])

GO

',

@database\_name=N'DBA\_Reports',

@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\_jobschedule @job\_id=@jobId, @name=N'Daily 3 times',

@enabled=1,

@freq\_type=4,

@freq\_interval=1,

@freq\_subday\_type=8,

@freq\_subday\_interval=3,

@freq\_relative\_interval=0,

@freq\_recurrence\_factor=0,

@active\_start\_date=20110512,

@active\_end\_date=99991231,

@active\_start\_time=63000,

@active\_end\_time=220000

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

**Step Two: Create the Stored Procedure - push\_changes\_to\_servers**

This stored procedure will do the following:

* Check to see which servers are on line then. Thes portion is similar to step one of this article however it creates an additional temp table called ServerList\_update which collects who is currently on line, then uses that list to compare against the ServerList table to grab the owner email address.
* For each server
  + Send an email out to the server owner that you put in step one asking users to remain on the network for maintenance
  + Run a query as defined by you by running sqlcmd. Before you run the sproc, you will need to add your script below for the variable @query. The @query is a varchar(8000) and you will need to place your script between the single quotes. If your script contains a parameter or something that is in single quotes, then you will need to use an extra set of single quotes around that portion. For example if you are inserting using a where clause like WHERE something = ‘something’ then you will need to use this WHERE something = ‘’something’’. These are two single quotes not a double quote.
  + Send an “all clear” message to server owners where the update was applied.
* Finally it will send an email to me or rather the DBA showing what servers were updated and the script that was run.

This job could be ran from a job, however since you will be changing the @query variable with a new query, it is just as easy to run it from the command line using exec push\_changes\_to\_servers after you recompiled the stored procedure.

Copy/Paste/Run the script below to create the stored procedure. Please change the areas in <> that state to add your info.

USE [DBA\_Reports]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- ======================================================================

-- Author: Kimberly Killian

-- Create date: 7/29/11

-- Description: Push scripts to servers

=========================================================================

CREATE PROCEDURE [dbo].[push\_changes\_to\_servers]

AS

SET NOCOUNT ON;

Declare @DatabaseName varchar(128),

@ServerName varchar(128),

@cmdmail varchar(50)

IF EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[ServerList\_update]') AND type in (N'U'))

DROP TABLE [dbo].[ServerList\_update]

CREATE TABLE [dbo].[ ServerList\_update](

[Server] [varchar](128) NOT NULL,

CONSTRAINT [PK\_ServerList\_update] PRIMARY KEY CLUSTERED

( [Server] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, FILLFACTOR = 99) ON [PRIMARY]

) ON [PRIMARY]

-- Load table with server names

-- This will only work if XP\_CMDSHELL is Enabled

Create table #Server ( [Server] [varchar](128) )

Insert Into #Server

Exec xp\_cmdshell 'sqlcmd /Lc'

Insert Into ServerList\_update ([Server])

select [Server] from #Server where SERVER in (select [SERVER] from dbo.ServerList)

DROP Table #Server

-- Loop on Server list and run the update

Declare SrvCurs CURSOR FORWARD\_ONLY FOR

select Server from dbo.ServerList\_update

open SrvCurs

fetch SrvCurs into @ServerName

while @@FETCH\_STATUS = 0

BEGIN

--send warning email to remote servers

Declare @email varchar(25)

select @email = Server\_email from dbo.SSIS\_ServerList where Server = @ServerName

--send the email

Declare @body varchar (1000)

SET @body = '<html><body><font face=verdana size=2>

<p><strong>Database Maintenance has started on ' + @ServerName + '. Please do not disconnect your machine from the network. This may take some time. </strong><br>

An email will be sent upon completion.</p>

<p><i>We apologize for any inconvenience this may cause. Please contact me if you experience any issues or need to disconnect.</i></p>

<p>Thank you,<br><br>

<strong><Add your name here> <br>

<add your title><br>

<Add your company here> </strong><br>

<add your phone # here><br/>

<add your email here>

</p>

</body></html>'

--send warning email do not remove from network

EXEC msdb.dbo.sp\_send\_dbmail

@profile\_name = '<add your profile here>',

@recipients =@email,

@copy\_recipients = '<add your email here>',

@subject ='AUTOMATED SEND: Database Maintenance Started Please Do not disconnect',

@body =@body,

@body\_format ='HTML';

--Set the query to be used to update each server

Declare @query varchar(max)

SET @query = '<put your query here>'

--Run the query on each server

Declare @OpenServerRunQuery varchar(8000)

SET @OpenServerRunQuery = 'sqlcmd -E -S "' + @ServerName + '" -q "' + @query + '"'

--exec queries for each server

exec master..xp\_cmdshell @OpenServerRunQuery

--send completion email to remote servers

SET @body = '<html><body><font face=verdana size=2>

<p><strong>Database Maintenance has completed on ' + @ServerName + '. </strong><br></p>

<p>Thank you,<br><br>

<strong><Add your name here> <br>

<add your title><br>

<Add your company here> </strong><br>

<add your phone # here><br/>

<add your email here>

</p>

</body></html>'

--send all clear email

EXEC msdb.dbo.sp\_send\_dbmail

@profile\_name = '<add your profile here>',

@recipients =@email,

@copy\_recipients = '<add your email here>',

@subject ='AUTOMATED SEND: Database Maintenance has completed',

@body =@body,

@body\_format ='HTML';

fetch SrvCurs into @ServerName

END

--Send email of who was updated

SET nocount ON

--

DECLARE @Subject VARCHAR (100)

SET @Subject='SQL Server - Servers Updated Today'

DECLARE @Count AS INT

SELECT @Count=COUNT(\*) FROM ServerList\_Update

PRINT @Count

IF @Count > 0

BEGIN

DECLARE @tableHTML NVARCHAR(MAX) ;

SET @tableHTML =

N'<table border="1">' +

N'<tr>' +

N'<th>Server</th>' +

N'</tr>' +

CAST ( ( SELECT td=[server],''

FROM [ServerList\_Update]

ORDER BY Server

FOR XML PATH('tr'), TYPE

) AS NVARCHAR(MAX) ) +

'<br><br><hr>

Patch installed: <br>' + @query + '<br><br><hr>' +

N'</table>' ;

EXEC msdb.dbo.sp\_send\_dbmail

@profile\_name = '<add your profile here>',

@recipients = '<Add your email here>',

@subject = @Subject,

@body = @tableHTML,

@body\_format = 'HTML';

END

--clean up

close SrvCurs

deallocate SrvCurs

GO

Conclusion

This is a very simple process to automate repetive updates or pushing out a script manually to many servers.

Included in this package, the following will be created:

Tables:

ServerList\_Stage

ServerList

ServerList\_Update

Stored Procedures:

push\_changes\_to\_servers

Jobs:

DBA\_Reports - Get New Servers