Automatically Running a Process When SQL Server or SQL Agent Starts

By Gregory A. Larsen

Have you ever had a need to run a query or a process as soon as SQL Server starts? Or run some set of tasks when SQL Server Agent Starts? Possibly you want to run a cleanup routine, a copy process or have some task started each time SQL server or SQL Agent is started. Well if this is the case then this article will discuss a couple of options you might consider using to accomplish automatically running your process.

Executing a Stored Procedure When SQL Server Starts up

SQL Server has the capability to start a Stored Procedure (SP) when it starts up. This is done by setting the autoexecution option on your SP. To do this the SP must first exist. After creating your SP you can then use the system SP "sp_procoption" to set the autoexecution option for your SP. Once the autoexecution option is turned on for your SP, it will start every time SQL Server is started. This system SP is also used to turn off autoexecution.

The sp_procoption system SP uses the following syntax to turn on the autoexecution option:

```
sp_procoption @ProcName = 'YourSP',
              @OptionName = 'startup',
              @OptionValue = 'on'
```

Here you can see the "sp_procoption" SP is setting the autoexecution option by setting the @OptionName to "startup" and the @OptionValue to "on" for a @ProcName of "YourSP". After running the code above the SP "YourSP" will autoexecute whenever SQL Server starts. One of the requirements that SQL Server has is that the code for any SP that needs to autoexecute be stored in the master database. Therefore you much first place your SP in the master database before executing the "sp_procoption" system SP.

If you want to stop your code from autoexecuting you will need to turn off the autoexecution option for your SP. The following code can be used to turn off autoexecution for a stored procedure:

```
sp_procoption @ProcName = 'YourSP',
              @OptionName = 'startup',
              @OptionValue = 'off'
```

Here I passed a @ProcName value of "YourSP", a @OptionValue of "off" and a @OptionName of "startup" to the "sp_option" system SP. This will turn off the autoexecution property for the SP "YourSP".

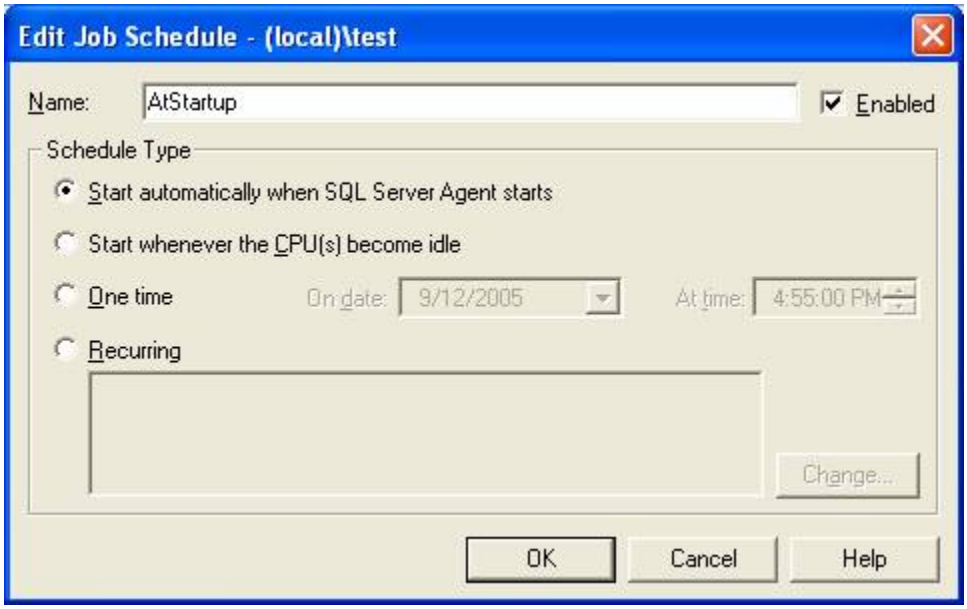
If you need to identify if an SP has been setup for autoexecution you can check the object properties. To do this you use the Meta Data OBJECTPROPERTY scaler function. By using this scaler function to check the "ExecIsStartup" property of an SP you can determine if the SP is set up for autoexecution. The OBJECTPROPERTY function will return a 1 if the autoexecution is set "on" for an SP or 0 (zero) if it is turned off. Here is some code that uses the OBJECTPROPERTY Meta Data scaler function to check to see if the object "usp_autostart" in the master database is setup for autoexecution.

```
use master
go
IF OBJECTPROPERTY ( object_id('usp_autostart'),'ExecIsStartup') = 1
   print 'usp_autostart will be executed at startup'
```

When the above code is run, if autoexecution is set "on" then the message "usp_autostart will be executed at startup" will be displayed.

Starting a Process When SQL Server Agent Starts Up

If you have a need to run one or more processes every time SQL Server Agent starts then SQL Server agent scheduling can accomplish that. A SQL Server Agent job can be automatically scheduled to run when SQL Server Agent starts up. To automatically run a job, when SQL Server Agent starts, create a new schedule and then select the "Start Automatically when SQL Server Agent Starts" radio button option for "schedule type". The screen below shows an example of a schedule named "AtStartup" that will run my job "Cleanup" every time SQL Server Agent Starts:



Any job that uses the schedule type of "Start Automatically when SQL Server Agent starts" will be run every time SQL Server Agent Starts. Remember SQL Server Agent can be stopped and restarted multiple times without bringing down SQL Server. So keep in mind that this method is not exactly the same as using the autoexecution option for a stored procedure.

Conclusion

As you can see, there are a couple of different ways to autmatically start a process when SQL Server or SQL Agent starts. Keep in mind that the SQL Agent method is slightly different than using the sp_procoption to automatically start a process. If you have a need to automatically start a process whenever SQL Server or SQL Agent starts, then the two options I discussed should be able to help you out.

Automatically Running Stored Procedures at SQL Server Startup

By: Armando Prato | Read Comments (3) | Related Tips: More > Stored Procedures

Problem

I have a stored procedure I want to run when SQL Server starts. Is there a way to execute this procedure automatically each time the SQL Server service is started?

Solution

SQL Server offers the system stored procedure **sp_procoption** which can be used to designate one or more stored procedures to automatically execute when the SQL Server service is started. This is a handy option that can be leveraged for a variety of uses. For instance, you may have an expensive query in your database which takes some time to run at first execution. Using sp_procoption, you could run this query at server startup to pre-compile the execution plan so one of your users does not become the unfortunate soul of being first to run this particular query. I've used this feature to set up the automatic execution of a Profiler server side trace which I've scripted. The scripted trace was made part of a stored procedure that was set to auto execute at server start up.

sp_procoption Parameters

```
exec sp_procoption @ProcName = ['stored procedure name'],
@OptionName = 'STARTUP',
@OptionValue = [on|off]
```

Here is an explanation of its parameters:

- Parameter **@ProcName** is self explanatory; it's the name of the procedure marked for auto-execution
- Parameter **@OptionName** is the option to use. The only valid option is **STARTUP**
- Parameter **@OptionValue** toggles the auto-execution on and off

Using sp_procoption comes with certain restrictions:

- You must be logged in as a sysadmin to use sp_procoption

- You can only designate standard stored procedures, extended stored procedures, or CLR stored procedures for startup
- The stored procedure must be located in the **master** database
- The stored procedure must not require any input parameters or return any output parameters

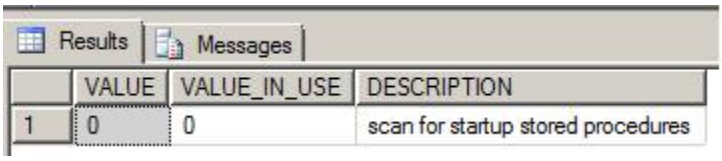
In the following example, I create a stored procedure that will be automatically run everytime my SQL Server instance starts. The purpose of this procedure is to write a row to a database table that logs the service start-up time. Using this table, I can get an idea of server up-time. The following script creates a new database that stores a metric table called SERVER_STARTUP_LOG. This table will hold the date and time the server was last started up. Once this infrastructure is built, I create the stored procedure that will be used to INSERT into this table at server startup. Note that the procedure is created in the **master** database.

```
USE MASTER
GO
CREATE DATABASE SERVER_METRICS
GO
USE SERVER_METRICS
GO
CREATE TABLE DBO.SERVER_STARTUP_LOG
(
LOGID INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
START_TIME DATETIME NOT NULL
CONSTRAINT DF_START_TIME DEFAULT GETDATE()
)
GO
USE MASTER
GO
CREATE PROCEDURE DBO.LOG_SERVER_START
AS
SET NOCOUNT ON
PRINT '*** LOGGING SERVER STARTUP TIME ***'
INSERT INTO SERVER_METRICS.DBO.SERVER_STARTUP_LOG
DEFAULT VALUES
GO
```

Now that the necessary objects have been built, we need to mark the created procedure to automatically start up when the server starts up. Running the following query, we can see that the sp_configure advanced option 'scan for startup procs' needs to be set. There is no need to do it manually; running sp_procoption will automatically set it for you.

```
USE MASTER
GO
SELECT VALUE, VALUE_IN_USE, DESCRIPTION
FROM SYS.CONFIGURATIONS
WHERE NAME = 'scan for startup procs'
```

```
GO
```



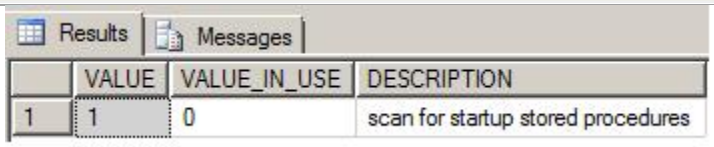
	VALUE	VALUE_IN_USE	DESCRIPTION
1	0	0	scan for startup stored procedures

We can now use sp_procoption to mark the procedure for auto-execution

```
USE MASTER
GO
EXEC SP_PROCOPTION LOG_SERVER_START, 'STARTUP', 'ON'
GO
```

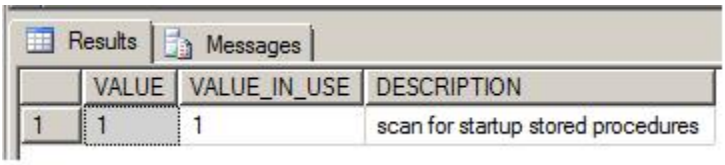
Re-running our configuration check, we now see that the server is configured to check for startup procedures (VALUE = 1) but the running value currently in effect is still set to not check for startup procedures (VALUE_IN_USE = 0). We'll need to re-start the SQL Server service to have the change take effect.

```
USE MASTER
GO
SELECT VALUE, VALUE_IN_USE, DESCRIPTION
FROM SYS.CONFIGURATIONS
WHERE NAME = 'scan for startup procs'
GO
```



	VALUE	VALUE_IN_USE	DESCRIPTION
1	1	0	scan for startup stored procedures

If we re-start the SQL Server service, we see that the configuration value now takes effect



	VALUE	VALUE_IN_USE	DESCRIPTION
1	1	1	scan for startup stored procedures

Furthermore, examining the previously created SERVER_STARTUP_LOG table, we see that the server startup time has been logged to the table

```
USE SERVER_METRICS
GO
SELECT * FROM SERVER_STARTUP_LOG
GO
```

Results | Messages

	LOGID	START_TIME
1	1	2008-08-25 21:36:07.547

Lastly, examining the SQL Server error log also verifies the procedure has been automatically run.

```
USE MASTER
GO
EXEC XP_READERRORLOG
GO
```

2008-08-25 21:36:07.510	spid5s	Launched startup procedure 'LOG_SERVER_START'.
2008-08-25 21:36:07.540	spid51s	*** LOGGING SERVER STARTUP TIME ***

Now let's turn the auto-execution off. Once set off, the procedure will not run the next time SQL Server starts.

```
USE MASTER
GO
EXEC SP_PROCOPTION LOG_SERVER_START, 'STARTUP', 'OFF'
GO
```

If you're unsure as to what procedures you've created have been marked to auto-execute, you can run the following query:

```
SELECT ROUTINE_NAME
FROM MASTER.INFORMATION_SCHEMA.ROUTINES
```

```
WHERE
OBJECTPROPERTY(OBJECT_ID(ROUTINE_NAME),'ExecIsStartup')
= 1
```

One thing you should be aware about: Dropping and re-creating marked stored procedures will require re-running sp_procoption. Dropping a procedure will cause the procedure to be "unmarked" for automatic execution. If you drop the procedure with no intent to re-create it, the system configuration setting 'scan for startup procs' will be left "on" until you manually set it to "off" using sp_configure or by turning off the procedure's auto-execution using sp_procoption. The process of turning procedure auto-execution on and off maintains this system configuration setting automatically.

From MSDN:

SYMPTOMS

Consider the following scenario. You use the SQL Native Client OLE DB provider (SQLNCLI) in SQL Server 2005 to create a linked server. You create a distributed transaction. The distributed transaction contains a query that uses the linked server to retrieve data from a table. When you commit the distributed transaction, you may receive the following error message:

```
Msg 1206, Level 18, State 167, Line 3
The Microsoft Distributed Transaction Coordinator (MS DTC) has cancelled
the distributed transaction.
```

Additionally, you may receive the following error message when you run a query after this behavior occurs:

```
Msg 8525, Level 16, State 1, Line 1
Distributed transaction completed. Either enlist this session in a new
transaction or the NULL transaction.
```

This problem occurs if the following conditions are true:

```
You use the SQLNCLI provider to create a linked server between two
instances of SQL Server 2005.

The XACT_ABORT option is set to ON.

In the distributed transaction, you try to release a rowset before
all rows in the rowset are processed.
```

Note **This problem may also occur if you call the ReleaseRows method in a distributed transaction to release a rowset before you commit a distributed transaction in an application.**

CAUSE

This problem occurs because the SQLNCLI provider incorrectly sends an attention signal to the linked server to roll back the distributed transaction.

WORKAROUND

To prevent the SQLNCLI provider from sending an attention signal to the server, use the SQLNCLI provider to consume fully any rowsets that the OLE DB consumer creates.