**Allow SQL Server Agent jobs management for non sysadmins**

[Cludio-Silva](https://www.sqlservercentral.com/author/Cludio-Silva), 2019-08-22 (first published: 2019-08-13)

FROM: <https://www.sqlservercentral.com/blogs/allow-sql-server-agent-jobs-management-for-non-sysadmins>

I don’t know about your experience when it comes to SQL Server Agent jobs but often I receive requests to grant permission so that clients can handle their jobs in an autonomous way.

Any problem with that?

It depends 

If you’re not familiarized with the difficulty this can bring, let me share with you a different way to accomplish the task without adding the login to the sysadmin instance role. I’m sure that you don’t want to do that on a production instance.

The next possible way is to make the login as the owner of the job and it needs to belong to one of the [msdb fixed database roles](https://docs.microsoft.com/en-us/sql/ssms/agent/sql-server-agent-fixed-database-roles?view=sql-server-2017)! You can take a look at this documentation and see the matrix there for each role.

This means that if we are talking about a single login, you won’t have any problem at all.

Using service accounts instead

It is common having services accounts that are job owners so they can run within the proper context.

In my humble opinion, this starts to be a little strange when it comes to modifying the agent job. It means that the service account needs permissions on agent roles and someone will need to connect to SQL Server using the service account (run as) so they can manage the agent job. It works, but not practical.

AD Groups

Many of the scenarios I get from clients is that they have an AD Group that contains all of all maintainers or support people.

Being able to put groups as job owner would be awesome, but unfortunately, it’s not possible.

I can hear the sad trombone!

My suggestion/recommendation/approach to this:

This may seem to be too much work, but at the end of the day, I feel it’s the best balance between security and the ability for the client to manage their agent jobs as they wish within their context.

Wrappers

I suggest that you create a wrapper for each system stored procedures that client need within msdb. It can be just the sp\_update\_job\_step. You may also get a request to be able to change the schedules of the job and you need to create another wrapper also for the sp\_update\_job\_schedule system stored procedure.

Security context

Use the [EXECUTE AS OWNER](https://docs.microsoft.com/en-us/sql/t-sql/statements/execute-as-clause-transact-sql?view=sql-server-2017#arguments) so they can impersonate the sysadmins permissions and call the system procedure.

Be more granular (HIGHLY RECOMMENDED!!)

Say that you have your administration jobs and the jobs from the client. To narrow down the scope for the client you may want to add an extra validation using the job name prefix.

If they don’t have a naming convention, you can always ask if they can start using one and update the agent jobs accordingly.

Ask them to help you to help them! 

The base principle here is simple if the name starts with the specified prefix the execution can proceed, otherwise, it will stop running and will return an error message saying they can’t update that specific job.

Here is a code example of a wrapper for the sp\_update\_job system stored procedure:

USE [msdb]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROC [dbo].[sp\_update\_job\_for\_non\_admins]

@job\_id UNIQUEIDENTIFIER = NULL, -- Must provide this or current\_name

@job\_name sysname = NULL, -- Must provide this or job\_id

@new\_name sysname = NULL,

@enabled TINYINT = NULL,

@description NVARCHAR(512) = NULL,

@start\_step\_id INT = NULL,

@category\_name sysname = NULL,

@owner\_login\_name sysname = NULL,

@notify\_level\_eventlog INT = NULL,

@notify\_level\_email INT = NULL,

@notify\_level\_netsend INT = NULL,

@notify\_level\_page INT = NULL,

@notify\_email\_operator\_name sysname = NULL,

@notify\_netsend\_operator\_name sysname = NULL,

@notify\_page\_operator\_name sysname = NULL,

@delete\_level INT = NULL,

@automatic\_post BIT = 1 -- Flag for SEM use only

WITH EXECUTE AS OWNER

AS

BEGIN

IF EXISTS (SELECT [name]

FROM msdb.dbo.sysjobs

WHERE job\_id = @job\_id

AND [name] LIKE '<customer prefix>%'

)

BEGIN

EXEC msdb.dbo.sp\_update\_job

@job\_id

,@job\_name

,@new\_name

,@enabled

,@description

,@start\_step\_id

,@category\_name

,@owner\_login\_name

,@notify\_level\_eventlog

,@notify\_level\_email

,@notify\_level\_netsend

,@notify\_level\_page

,@notify\_email\_operator\_name

,@notify\_netsend\_operator\_name

,@notify\_page\_operator\_name

,@delete\_level

,@automatic\_post

END

ELSE

BEGIN

RAISERROR ('The job\_id used does not belong to an <customer prefix> job.', 16, 1);

END

END

GO

More examples on my GitHub

If you want to leverage on the stored procedures that I have already created you can download them from [ManageAgentJobsNonsysAdmin folder on my GitHub repository](https://github.com/ClaudioESSilva/SQLServer-PowerShell/tree/master/ManageAgentJobsNonsysAdmin)

Giving permissions to the wrapper objects

Create a role (will be easy to manage) on the msdb database and add the logins (nominal or groups) to it.

Grant EXECUTE permissions for all of your compiled \*\_for\_non\_admins stored procedures.

Other options are to grant the EXECUTE permissions to the existing database fixed role where the client login/group is already member.

T-SQL code example:

use [msdb]

GO

GRANT EXECUTE ON [dbo].[sp\_update\_job\_for\_non\_admins] TO [SQLAgentOperatorRole]

GO

You can be more creative but be aware of the maintenance costs

Another option that comes into my mind is to include the agent jobs names (totally or partially) on a control table.

However, this can bring more work when talking about maintaining this process. If the client has somehow a fixed number of jobs maybe it is not too much work otherwise it can be a nightmare.

Wrap up

We have seen how we can provide more control to clients so they can manage their jobs without compromising security.

I have used this method several times with success. By success I mean, I was able to explain to the client the limitation on SQL Server side and, on the other hand I present to them a possible solution.

It is not easy to change the code within a step using the stored procedure instead of using the interface? Sure it is not! But at least you provide a working solution that makes the client autonomous! In order to help them show how they can script out the changes, add the suffix for *NonAdmins* and they are good to go.

Thanks for reading.