Server Reboot Order Script

**Table of Contents**

[1 General purpose of this script 2](#_Toc371685388)

[2 Quick start guide 2](#_Toc371685389)

[2.1 Reading the log file 3](#_Toc371685390)

[2.1.1 Common errors 4](#_Toc371685391)

[3 Manual 5](#_Toc371685392)

[3.1 Add new servers 5](#_Toc371685393)

[3.2 Add new service to stop 5](#_Toc371685394)

[3.3 Add new boot groups 5](#_Toc371685395)

[3.4 General important variables 6](#_Toc371685396)

[3.5 Technical description 6](#_Toc371685397)

|  |  |  |
| --- | --- | --- |
| Author | Date | Content |
| Patrick Mangold (App. BE-SCC) | 8.11.2013 | Initial documentation |
| Patrick Mangold (App. BE-SCC) | 26.11.2013 | Added folder deletion and file move |
|  |  | Reviewed document |
|  |  |  |

# General purpose of this script

This script has the function that servers (only windows) with dependencies reboot in a specified order after the patch procedure. It has three main functions:

* Notify any user which is logged on to the system that the server will reboot in xx minutes to prevent data loss
* Stop specified services on the server before rebooting it
* Restart the server

With that we can ensure that servers with dependencies reboot in the right order after the patching was done. For example a DB-server needs to be available when a related Web-server is starting up.

Since release v5\_4 it is possible to do some tasks with files and folders after the services were stopped:

-Delete a specified directory (provide a network path)

-Copy all files in the root of a provided folder to a new folder in the same directory with the name copy\_\*DATE\*

# Quick start guide

Connect to the server shhwsr0025 and browse to the following directory:

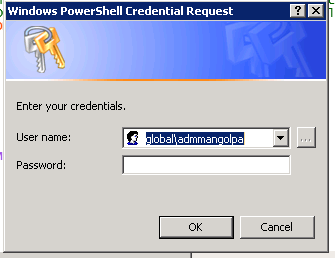
D:\Scripts\Schindler\Microsoft\Patching\reboot

There you can find:

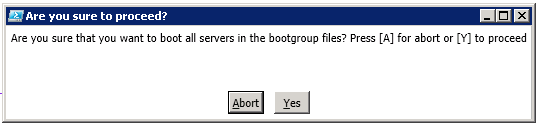
* The script
* The boot group files containing the servers of this group
* The server\_services files containing a server with its service to stop on each line
* The server\_delete\_folders files containing a list of servers with directories to delete
* The server\_files\_in\_folder\_to\_copy files containing a list of servers with directories, in which we have to copy all files in the root directory away to a new folder
* The log file with the date of creation
* The change log file with the major version changes

Select the newest version and “Run with PowerShell”

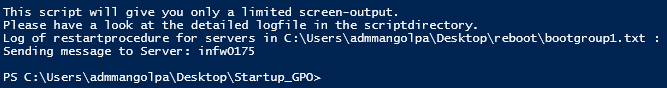




Provide your ADM Account from GLOBAL Domain with rights to restart all servers. Your current login account should be suggested by default.



Click “Yes” if you want to reboot all servers in the list. Be aware that you should only perform this script during the time window of the patch weekend. **If you do this outside this time window you will reboot a lot of business critical servers!**

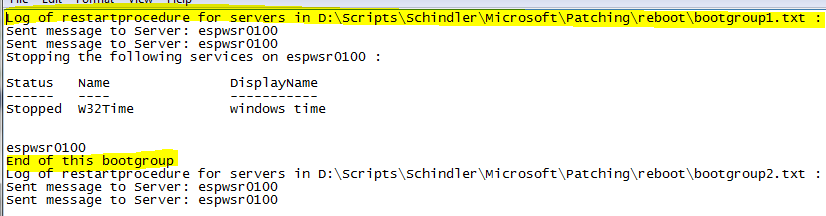


The screen-output is very limited. You should see a counter which indicates how long it takes till the script sends the next message to the servers of this boot group. In addition you should get a feedback which boot group is currently processed and the action that is performed with which servers. Therefore I recommend having a look at the log file which is in the same directory as the script. Errors regarding the reboot process will be logged there.

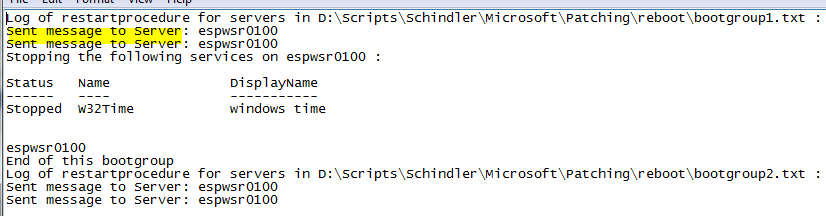
## Reading the log file

Here is an example how a log should look like:

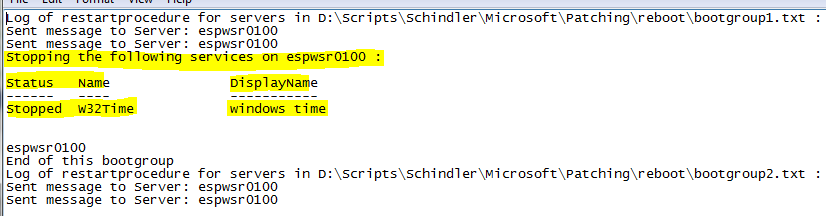




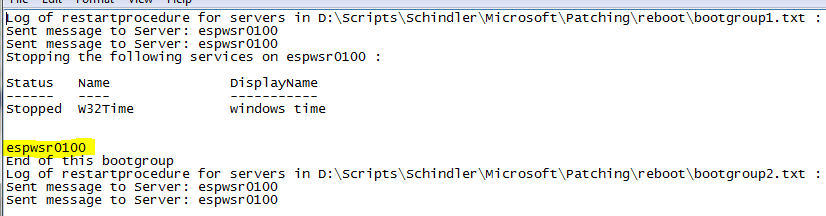
The text, which is marked yellow for demonstration, indicates during which boot group this messages have been created.



You can see “Sent message to Server” for each message the user got. The user gets a notification every minute. Per default the first message will be sent two minutes before the server reboots.



You can specify which services have to be stopped before the server restarts. When the service stopped successfully you should see a message similar to this.



When the target system successfully received the restart commando, you should see only the name of the. In case of errors you will see the server name and the error message.

### Common errors



In this case the service could not be stopped. This is most likely a result of providing a wrong service name. Another case is that the server is not reachable or you don’t have sufficient rights to stop the service.



In this case the server server4 could not be contacted. This is most likely because the server was down or you didn’t have enough rights to see who was logged on on this server at that time.



If you get an error in the part where rebooted servers are logged similar to this, the server is most likely already uninstalled and no more needed.

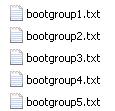


You can also get a message like this in that part of the log. It means exactly what it tells you: “You don’t have enough rights to reboot this machine”.

# Manual

In this part of the manual you will get a description how you can add new servers, services or boot groups.

## Add new servers



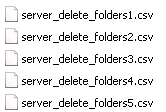
Simply add your new server to the file with the right group name according to the patch document. Please notice that there should be only one server per line. GLOBAL Server should work without FQDN but I recommend always using FQDN.

## Add new service to stop

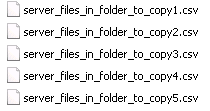
Add your server and service to the corresponding server\_service file (right click, edit). The number represents the bootgroup. The same as under 4.1 counts here. Use FQDN and one server + service per line. Separate them with a comma without unnecessary spaces. If a server requires multiple services to be stopped, simply add a new line with the same servername but the next service.

## Add a new directory to be deleted



Please note, that the servername is only for informational purposes and doesn’t affect source or destination path. Add the path to the directory with FQDN of the server in the path. The script will delete the folder and all subdirectories.

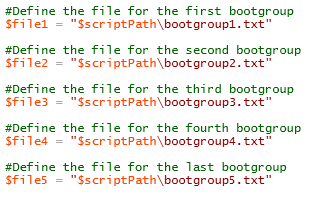
## Add a new directory with files to be copied



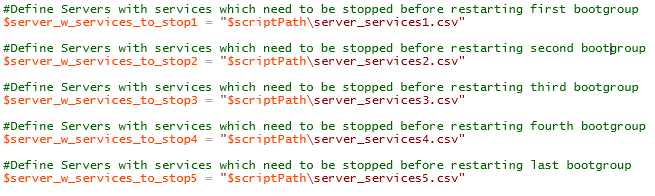
Please note, that the servername is only for informational purposes and doesn’t affect source or destination path. Add the path to the directory with FQDN of the server in the path. The script will copy all files in the root of this directory to a newly created folder named copy\_\*DATE\*.

## Add new boot groups

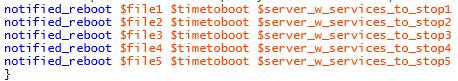
This is a bit more complicated than the previous tasks but simple too. Edit the PowerShell script and add these three entries:



Add a new line and copy the last two lines. Adjust the numbers to the number of your new boot group and adjust the description (green comments) to the boot logic. Create a new file in the directory of the script with the name you just provided.



Do the same with the services file.



Add a new line at the end of this part of the script. Call the function notified\_reboot and provide the variable names you just created in the two previous steps. $timetoboot represents how many minutes before the server boots the user will be notified. You can use the $timetoboot variable and use the 2 min default or provide your own variable (or you can just provide the number of minutes directly but that is not so nice because the sources and times are provided at the top of the script).

## General important variables



This variable specifies how many minutes before the server reboots every user should be notified. (Increasing this number will result in a longer pause between the boot groups.



This variable specifies how long the script should wait before it continues, after it finished rebooting the last server of the previous boot group. During this time the user won’t get any messages. After this time passed, the first notification will be sent to the user.

## Technical description

Have a look at the script itself. Most of it has comments to provide you some help.



There is only one thing that looks a bit weird when you read the script for the first time. Why the hell is this guy invoking a script block on a “remote computer” which is in fact the localhost? Because every thing is simplified as much as possible (only one super function for the whole work), the standard errortrapping with possible errorlogging and special error actions in combination with the special wish to continue the script after that inside multiple loops is not possible. By default the erroraction stop with custom trapping can only continue after the loop. If you want to continue inside the loop you must give up the option with custom errorlogging and set the erroraction to continue. By storing the important command inside a variable which can hold scriptblocks, you cann specify an errorhandling function and call it with the parameter scriptblock. Now this scriptblock runns inside your function which has a custom errorhandler. As a result of using a scriptblock, you have to send the data to a remote host. To process it on our own server we specify our server. If an error occurs powershell can’t get the link to the loop because you started the script in a kind of “separate environment”. If an error occurs now, we can handle it customized and the next executable command is, as you might already got, just the next command inside the loop.