**Downgrading Your SQL Server Edition**

<https://sqljana.wordpress.com/2017/09/22/powershell-save-millions-by-downgrading-your-sql-server-edition-partially-automate-it/>

## Why downgrade edition?

There are two reasons –

1. Save money
2. You mostly only need Standard Edition!
3. There is practically no difference between Standard & Enterprise

* Standard edition is limited to lesser of 4 sockets or 24 cores with a maximum memory of 128 GB plus a few truly Enterprise level features like Compression, Availability Groups, Partitioning, ColumnStore indexes etc are off limits.
* I would say most places would fall under this threshold for “Standard” but feel inferior to say they run “Standard”! I don’t, especially when money matters.
* But, all kidding aside, most shops don’t even realize that they do not use any Enterprise features on 90% of their instances but pay Enterprise price anyway! If you don’t trust me, go check for yourself at your place – we did, on hundred’s of SQL Server instances! I painfully built the infrastructure to do this type of thing using PowerShell in seconds  if not a few minutes, for scanning hundreds of servers/instances.
* You could easily check each database on your instances by running this query to see if any enterprise features are in use and if it returned nothing, it is a perfect candidate for a downgrade!

**SELECT feature\_name FROM sys.dm\_db\_persisted\_sku\_features**Microsoft makes everything easy but when it comes to downgrading your SQL Server edition from say Enterprise to Standard, there isn’t a switch.

For Microsoft that would have been easy but they made it hard for a reason! They don’t want you to downgrade your edition. You will understand why when you look at the [pricing differences](https://www.microsoft.com/en-us/sql-server/sql-server-2016-pricing)

|  |  |  |
| --- | --- | --- |
| **SQL Server 2016 Enterprise Edition (per core)** | **SQL Server 2016 Standard Edition (per core)** | **SQL Server 2016 Standard Edition (CAL)** |
| $ 14,256 | $3,717 | $931 |

This is for volume licensing.

## Savings on 50 instances with 4 cores each

Standard is almost 1/4 th the cost of Enterprise edition!

|  |  |  |
| --- | --- | --- |
| **SQL Server 2016 Enterprise Edition** (50 instances with 4 cores each) | **SQL Server 2016 Standard Edition** (50 instances with 4 cores each) | **Savings** |
| $ 28,51,200 | $ 7,43,400 | $ 21,07,800 |

## Hard vs. Easy way to downgrade:

Obviously, the hard way to downgrade is to make backups of [E V E R Y T H I N G] imaginable on SQL Server, SSRS, SSAS and SSIS in addition to logins, jobs, linked servers, credentials, security, proxies etc., and restore them perfectly. This is in addition to the numerous instance level settings/configuration. The very thought of missing something is the reason enough for most people to not attempt this and continue to waste money.

Fortunately, there is an easier way and it turns out to be really simple – The Jonathan Kehayias method. Thanks Jonathan.

* <https://www.sqlskills.com/blogs/jonathan/downgrading-sql-server-editions/>

**The over-simplified steps are:**

* Make backups of everything (optional but is good for safety)
* Shutdown
* Make a physical file backup of the system databases master, msdb, model
* Uninstall the enterprise edition
* Remove system database files if they are still left over (user db’s will remain as is)
* Install standard edition using the same folder locations/settings
* Get up to the same service pack level as before
* Shutdown standard edition
* Copy back the system database files
* You are done…the user databases, security settings, jobs and everything will reappear!

### \*\*WARNING\*\*

This method DOES NOT seem to work for SQL Server versions 2014 and above. However, I can personally vouch for versions 2008, 2008R2 and 2012 where I know that it works.

## Why automate such a simple process?

It is so simple that you could do it in your sleep. However, there are reasons you want to at least partially automate this

1. You have a ton of instances where you need to downgrade edition
2. You don’t want to check all the details for every instance
3. You want your commands ready to go and not make stupid mistakes
4. You want a clean and consistent record of your activity/backups
5. Save time and be efficient
6. You don’t want to forget a step and lose everything in the process

I chose PowerShell to generate the steps as PowerShell commands with text comments/instructions so that I can run through them to perform the downgrade.

**Note**: This only generates a set of helpful commands to do the downgrade and not produce a script that does the whole thing for you! i.e., semi-automate.

## Are there any prerequisites to automate?

Besides PowerShell, you really don’t need much but life is easier with some of the modules below. It makes the commands very precise. They don’t have to be on the host being downgraded. It could be on a jump-box where you usually do admin stuff.

### PowerShell SQL Server related modules:

* dbatools
* SQLTranscriptase
* SQLPS or SQLSERVER PowerShell module

Yes. I have started to fall in love with [dbatools](https://dbatools.io/" \t "_blank) and use it quite a bit. Thank you dbatools team. Also, I love the PowerShell documentor for SQL instances – [SQLTranscriptase.](https://github.com/gwalkey/SQLTranscriptase" \t "_blank) Thanks to George Walkey. It is very thorough with its documentation and script generation.

dbatools is a general purpose swiss army knife for the DBA and SQLTranscriptase is a must have documentation module for documenting your instance. Both of these are used. The latter is optional. SQLPS is the native PowerShell module for older versions of SQL Server and SQLSERVER is the PowerShell module for newer versions but is backwards compatible. You should be able to google and find out how to install these.

### You also need

Again, this could be on a network share somewhere so that you are not having to deal with constantly copying it around.

* The software installation files location to the edition of SQL Server installed (need this to uninstall via command line)
* The software installation files location which has the Standard edition setup.exe
* The software installation location to the service pack equal to the service pack that was in place before switching editions.