## Migration Optimization



Content Matrix PowerShell Introduction

## Publication Date: June 9, 2015

## Current Content Matrix Version: 7.2.0.16

**Copyright**

© 2015 Copyright Metalogix Corporation International GmbH.

All rights reserved. No part or section of the contents of this material may be reproduced or transmitted in any form or by any means without the written permission of Metalogix Software Corporation.

Metalogix Content Matrix Consoles™ are trademarks of Metalogix Software Corporation.

Windows SharePoint Services is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries. Other product and company names mentioned herein may be the trademarks of their respective owners.

**Technical Support**

For information about Metalogix Technical Support, please visit <http://metalogix.com/support>.

Technical support specialists can be reached by email at [support@metalogix.com](mailto:support@metalogix.com) or by phone at 1.202.609.9100.

We want to hear from you. Please send any comments regarding this document to [support@metalogix.com](mailto:support@metalogix.com).

The level of technical support provided depends upon the support package that you have purchased. Contact us to discuss your support requirements.

Contents

[Migration Optimization 0](file:///C:\Users\torbach\OneDrive\Documents\Hotfixes%20and%20Releases\7.2.0.16\Content%20Matrix%20PowerShell%20Introduction%20Guide.docx#_Toc421633491)

[Publication Date: June 9, 2015 0](file:///C:\Users\torbach\OneDrive\Documents\Hotfixes%20and%20Releases\7.2.0.16\Content%20Matrix%20PowerShell%20Introduction%20Guide.docx#_Toc421633492)

[Current Content Matrix Version: 7.2.0.16 0](file:///C:\Users\torbach\OneDrive\Documents\Hotfixes%20and%20Releases\7.2.0.16\Content%20Matrix%20PowerShell%20Introduction%20Guide.docx#_Toc421633493)

[Introduction 3](#_Toc421633494)

[What is PowerShell 4](#_Toc421633495)

[PowerShell object types 4](#_Toc421633496)

[PowerShell scripting objects 4](#_Toc421633497)

[Running PowerShell 6](#_Toc421633498)

[Using Metalogix Content Matrix with PowerShell 8](#_Toc421633499)

[Snapins 8](#_Toc421633500)

[Adding Snapins 8](#_Toc421633501)

[Using Metalogix Cmdlets 9](#_Toc421633502)

## Introduction

This guide provides an introduction to Microsoft PowerShell, and how to use Metalogix Content Matrix with it.

## What is PowerShell

Microsoft PowerShell is a task automation and configuration management framework, consisting of a scripting language and ISE built on the .NET framework. Metalogix Content Matrix uses PowerShell as a framework to automate and configure actions.

## PowerShell object types

PowerShell object types that will be used in this document:

#### DLL (module)

Dynamic link library (also known as a module). A library of code that must be installed on a machine before snapins from the library can be used in PowerShell. Metalogix will automatically register and install the Content Matrix DLLs during the Content Matrix console installation process. Microsoft SharePoint PowerShell dlls (non-Metalogix) are available on any server with a SharePoint hive.

**Tip:** To repair the Content Matrix PowerShell DLL installation, run the Content Matrix installation repair wizard.

**Tip:** Metalogix PowerShell cmdlets can be run anywhere Content Matrix is installed, but Microsoft SharePoint cmdlets cannot. Ensure that if you are directly accessing SharePoint cmdlets that you are running PowerShell from a machine with SharePoint installed.

#### Snapin

Compiled reference of cmdlets that can be added to a PowerShell execution process. Once registered, all entries within that cmdlet can be used in scripts during the lifetime of that process.

**Syntax**: add-pssnapin “name”

**Tip:** Every time you open the PowerShell ISE the Metalogix snapins must be added again, so try configuring a profile to automatically load them upon execution of the ISE process.

#### Cmdlet

A function that can be executed to perform a specific action. Cmdlet execution will encompass the majority of work done in PowerShell, as both Metalogix and Microsoft SharePoint use cmdlets to automate the majority of work.

**Syntax:** get-mlsharepointsite –url “URL”

### PowerShell scripting objects

An in depth explanation of scripting and the corresponding objects is outside the scope of this document, but the following objects will be relevant:

#### Variable

A variable in PowerShell is a storage container for a value. Variables always have the prefix “$”

**Syntax:** $variable = “Value”

#### Conditional statement

A conditional statement in PowerShell is a logical gate which will only execute if the condition is met.

**Syntax:** if($condition -eq $true) { #Run }

#### Loop

A loop in PowerShell is a conditional logic block that will run as many times as the condition evaluates to true. There are many types of loops, but this document will only reference the Foreach loop. The foreach loop is a special type of loop that takes in a collection of objects (a variable), and will run once for every object in the collection.

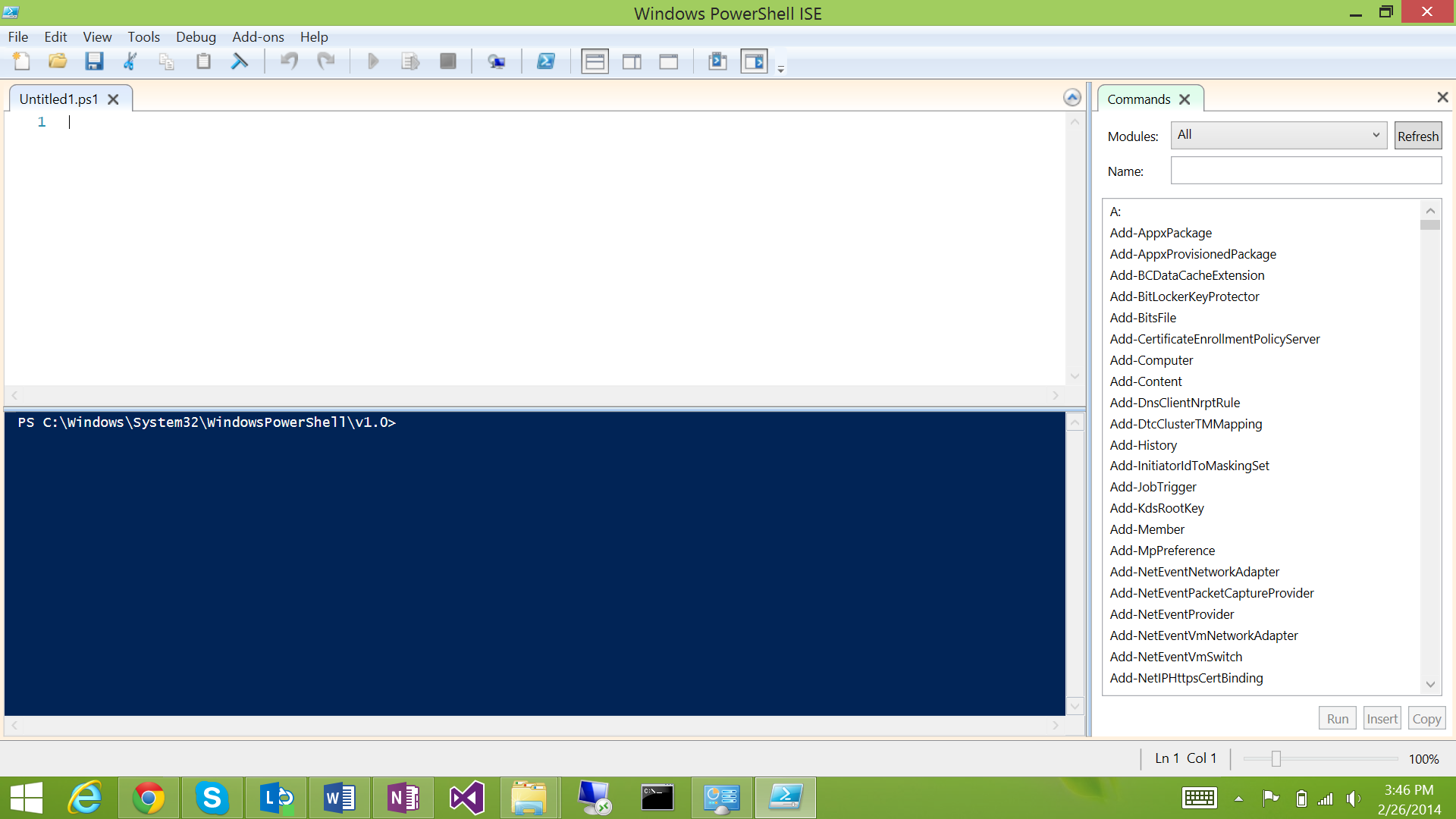
**Syntax:** Foreach($site in $siteCollection) { #Run }

## Running PowerShell

Microsoft PowerShell can be run in multiple ways (in order of usefulness):

1. PowerShell ISE
2. Dell’s PowerGUI
3. Metalogix PowerShell Console
4. Microsoft PowerShell Console
5. Run a .ps1 file

#### PowerShell ISE

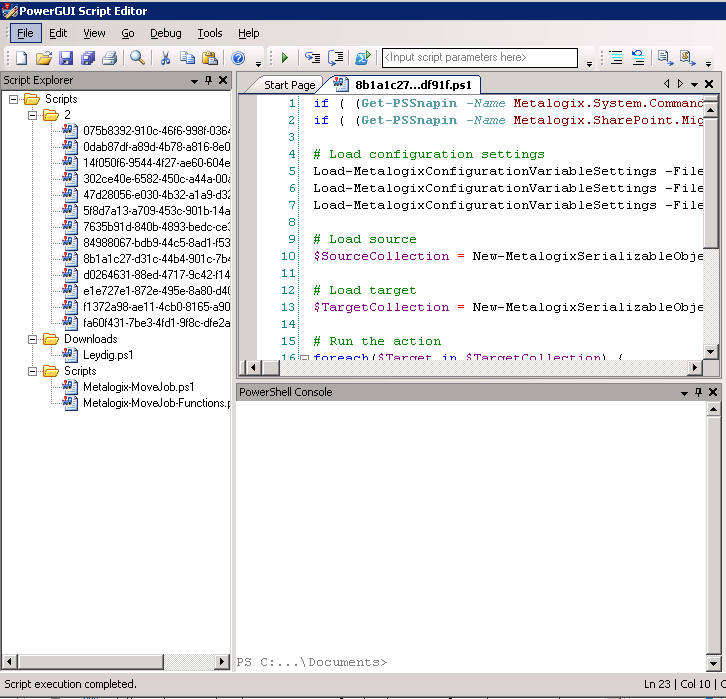


Available here: C:\Windows\System32\WindowsPowerShell\v1.0\powershell\_ise.exe

The PowerShell ISE is an integrated scripting environment designed for executing and debugging PowerShell scripts. Whenever possible the ISE should be utilized as it provides the most power and capability. Use the PowerShell ISE whenever possible.

**Tip:** Use the PowerShell ISE to register your PowerShell snapins, and it will auto-complete cmdlet calls and provide suggestions while debugging.

### Dell’s PowerGUI

PowerGUI is a more powerful PowerShell script editor that is available from Dell for free from [here](http://software.dell.com/products/powergui-freeware/).  


### Metalogix PowerShell Console

Available here: **Metalogix install directory**

The Metalogix PowerShell Console is a scripting console able to run individual PowerShell commands and scripts. The Metalogix PowerShell Console comes equipped with the Metalogix snapins predefined, and will be able to execute Metalogix cmdlets immediately.

### Microsoft PowerShell Console

Available here: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

The Microsoft PowerShell Console is identical to the Metalogix PowerShell Console, except that the Metalogix snapins are not equipped. To run Metalogix PowerShell scripts in the default console, the [snapins must be added](#_Adding_Snapins).

### Run a .ps1 file

Running .ps1 files will automatically open the default scripting console for the .ps1 file format and run the script in that context. If this is the Microsoft PowerShell Console, the Metalogix snapins must be added before the script will run successfully.

## Using Metalogix Content Matrix with PowerShell

Metalogix installs PowerShell DLLs during the Metalogix Content Matrix Console installation.

**Tip:** If Content Matrix is not installed on the computer that is running the PowerShell scripts, then they will not be able to access the Metalogix snapins without manual installation.

## Snapins

Metalogix snapins must be added to PowerShell before any cmdlets can be run. There are three Content Matrix PowerShell snapins available after installation of the Metalogix Content Matrix Console:

1. Metalogix.System.Commands
2. Metalogix.SharePoint.Commands
3. Metalogix.SharePoint.Migration.Commands

**Tip:** To find all of the available commands in a snapin, you can add the snapin to the PowerShell ISE and refresh the cmdlet list on the commands window. Once it refreshes with the new snapin, you can examine each cmdlet available in the snapin by choosing it from the modules dropdown. If you aren’t using the PoweShell ISE, you can call “Get-Command -pssnapin metalogix.\*” to print the commands available in all Metalogix snapins.

**Tip:** To get the parameters on a specific cmdlet, you can call “Get-Help Get-MLSharePointSite”

#### Metalogix.System.Commands

This snapin is responsible for interpreting the Content Matrix auto-generated script retrieval objects and filters, and should very rarely be used manually.

#### Metalogix.SharePoint.Commands

This snapin contains cmdlets that perform Add/Get/Remove/Compare actions. It is most commonly used to retrieve the source and target objects in a SharePoint copy action. It can also be used to create, delete or compare objects in the live SharePoint environment.

#### Metalogix.SharePoint.Migration.Commands

This snapin contains cmdlets that perform migration actions.

### Adding Snapins

To use snapins in PowerShell, you must add them to the process. As mentioned above, the Metalogix PoweShell console comes with all snapins automatically added, others do not. To add the Metalogix snapins, you must execute the following statements:

Add-PSSnapin metalogix.system.commands

Add-PSSnapin metalogix.sharepoint.commands

Add-PSSnapin metalogix.sharepoint.migration.commands

These statements can be run manually when you open run your PowerShell management utility, or they can be added to each script at the beginning.

**Tip:** Trying to add a snapin when it is already added will throw an error, so try wrapping the snapin addition code at the beginning of your scripts with a check to see if they already exist.

### Using Metalogix Cmdlets

All Metalogix functions are executed via pre-built cmdlets in PowerShell that take a set of parameters corresponding to the action options.

**Tip:** An explanation of each cmdlet (and parameters) is outside the scope of this document, but can be retrieved via the Metalogix PowerShell Reference, as well as the built in product documentation under the “PowerShell” heading.

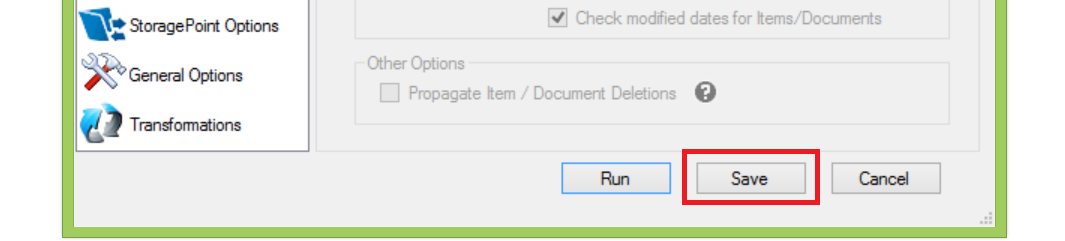
#### Generating scripts

Scripts using the Metalogix Content Matrix PowerShell cmdlets are usually created by generating a script from the UI. The action cmdlets can be called directly, but the parameters can be difficult to understand without configuring and generating them from the UI first.

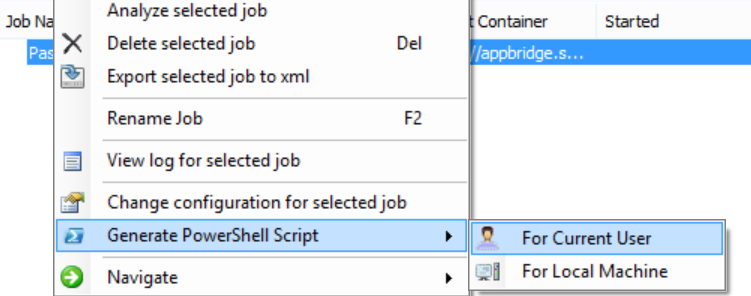
**Tip:** The majority of actions in Content Matrix can generate PowerShell scripts from their configuration, but there are some exceptions, like copying taxonomy term stores.

**To generate a script:**

1. Configure an action in the standard fashion, except instead of running the action once configuration is complete, choose the “Save” button at the bottom of the configuration dialog instead.



1. Right click the entry in the job listing that corresponds to the saved configuration and choose the “Generate PowerShell Script->For Current User” option.



**Tip:** Scripts generated by the application are automatically saved as <Unique GUID>.ps1 in the running user %temp% directory.

1. The generated script can either be saved with a permanent name in a different location, or components of it can be copied and pasted into a different script.

#### Editing scripts

Scripts generated by Content Matrix have the following steps:

1. **$SourceCollection = New-MetalogixSerializableObjectCollection …**
   * This block initializes the source collection for the action
2. **$TargetCollection = New-MetalogixSerializableObjectCollection …**
   * This block initializes the target collection for the action
3. **foreach($Target in $TargetCollection) …**
   * This block iterates through each object in the target collection, in case there are multiple. Note that the CMDLET itself will iterate through each object in the source collection if it supports multiple source objects.
4. **$SourceCollection | Copy-MLSharePointSite -Target $Target …**
   * This block runs once per iteration of the foreach loop in step 3, and pipes the source collection into the cmdlet execution. The RUN CMDLET

**Tip:** If the script will be run more than once, or modified in any fashion, a best practice is to replace step 1 and 2 with the Get-Object cmdlets from the Metalogix.SharePoint.Commands snapin.

Editing a Metalogix generated PowerShell script can be done quickly and easily. Let’s create the same script below using Metalogix cmdlets:

1. **$source = Get-MlSharePointSiteFromDatabase …**
   1. This block retrieves a source database site and stores it in the $source variable
2. **$target = Get-MlSharePointSite …**
   1. This block retrieves a target SharePoint site and stores it in the $target variable
3. **Copy-MLSharePointSite –source $source –target $target …**
   1. This block copies the source site into the target site by running the cmdlet with the same options, replacing the –source and –target parameters with the variables we retrieved in step 1 and 2

This is a much simpler script that performs the same action. The auto-generated information from Content Matrix can be confusing, so it is recommended to replace it like the above before editing any more of the script.

**Tip:** Any parameters in the Metalogix cmdlets can be replaced by variables. Try replacing parameters to see what effects it has on the copy (in test migrations…).

#### Advanced editing

PowerShell supports many advanced operations. Some examples of what you can use PowerShell to achieve:

* Reading data from a CSV file or SharePoint list and using it to change cmdlet parameters
* Reading XML files and copying specified objects from them
* Reading entire SharePoint object hierarchies and migrating them using Content Matrix cmdlets sequentially. Example, copying an entire SharePoint database or web application in a single script
* Scaling out Content Matrix migrations

#### Running scripts

Once the script has been generated, edited and saved, it can be run from any of the [previously mentioned programs.](#_Running_PowerShell)

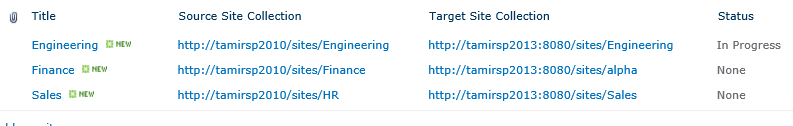
**Tip:** Before you can run PowerShell scripts on a machine, you’ll need to set the execution policy of that machine. By default, a server will deny scripts permission to run, so you need to run the “Set-ExecutionPolicy -ExecutionPolicy <Policy type>” cmdlet.

## Sample Script to Demote Site Collections based on values in a SharePoint list

To illustrate an example of the PowerShell capabilities of Content Matrix, and help you understand more of how this works, we have included a sample PowerShell script in this document below. Before using it, please change its name to remove the .txt component. This script will need to be modified in your environment to work.

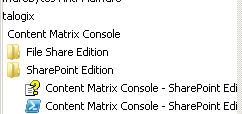
The purpose of the script is to loop through a SharePoint 2010 list that contains items referencing source Site Collections, Target Site Collections onto which you’d like to demote (Paste as Subsite) the source Site Collection, and Status (see image below):

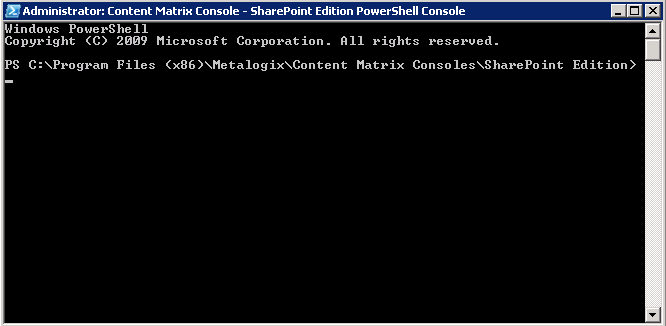




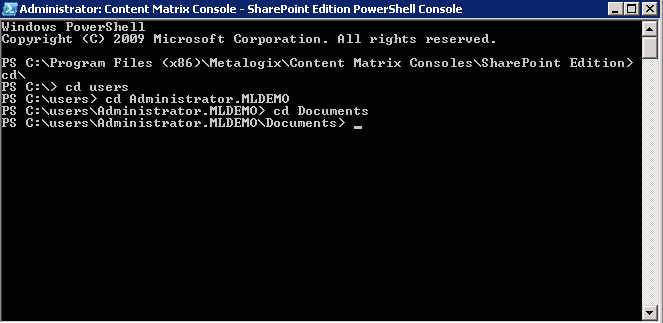
The script is well commented to further explain what it does. Once you have modified the script to suit your environment, save it on the machine in which you have Content Matrix SharePoint Edition installed to your documents folder. Then follow these steps:

1. Go to Start/All Programs/ Metalogix/Content Matrix Console/SharePoint Edition/ and select Content Matrix Console – SharePoint Edition PowerShell Console.

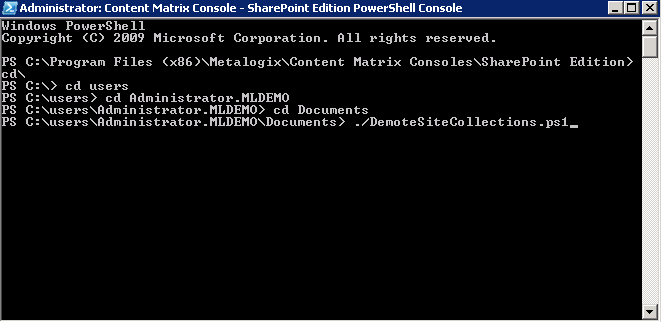




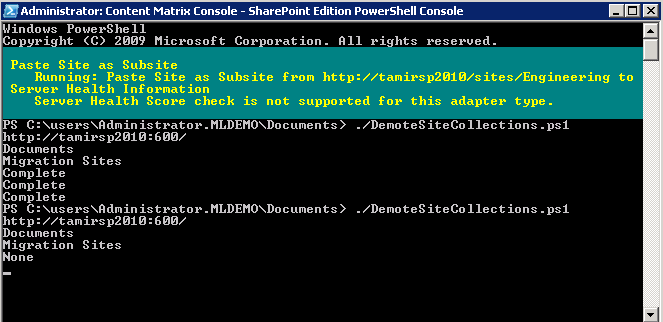
1. In the console, change directory to your Documents directory to your documents directory:



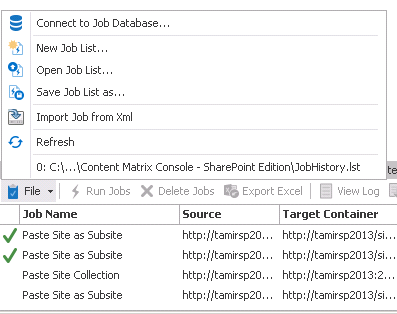
1. Now run the script with the format ./DemoteSiteCollections.ps1



1. You should start seeing progress in the migration



1. Once the script completes, use the Content Matrix console to confirm that the Site Collections were indeed demoted/pasted as subsites to your target.
2. In the Job list, go to the File Menu, and select Refresh to see the jobs in the job log:



1. Confirm in the list that the Status for each item has now changed to complete:

