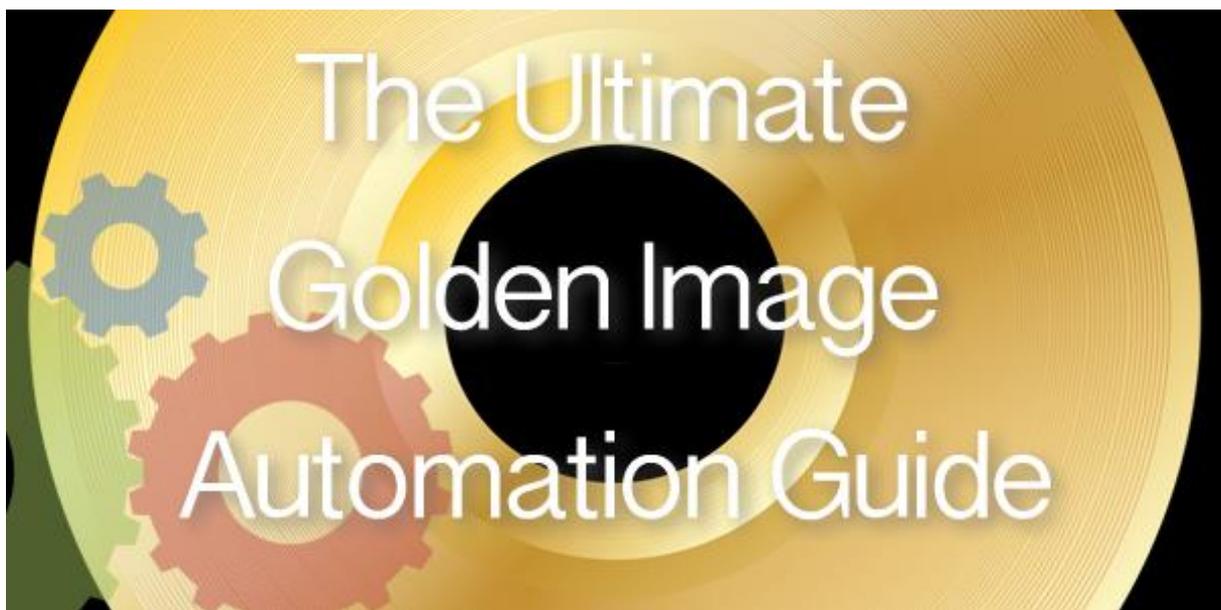




Workspace Guru

The Ultimate Golden Image Automation Guide

V2.0



Chris Twiest

Index

About:.....	4
Author:	4
VERSION 2.0	4
Disclaimer:.....	4
What is a Golden Image and why should I create one automatically?.....	5
Before we begin; what do we need?	5
Target VM specifications.....	5
Automating Citrix Provisioning Services (PVS).....	6
Automate Target VM snapshot.....	7
Automate Target VM XenServer	8
Deploy Ivanti/RES ONE Automation Agent	9
Prepare tasks on Target VM.....	10
Add to AD	10
Install Windows Features RDP Host.....	11
Activate Windows	11
Install WSUS Patches (Recursive).....	11
Create Snapshot (Optionally)	11
Optionally snapshot on XenServer	11
Add Modules to Projects and Projects to Deployment Runbook	12
Create Deployment Runbook.....	13
Deploy Middleware	14
Install .NET 3.....	14
Install .NET 4.7.....	14
Install Adobe Flash	15
Install Java	16
Install Silverlight	17
Deploy Applications.....	18
Deploy Citrix / Ivanti Workspace.....	18
Install Ivanti Identity Directory Client	19
Install Ivanti Workspace Workspace Control	20
Install Citrix VDA.....	20
Install Citrix PVS Target Device Software	22
Create Snapshot (Optionally)	24
Optional Snapshot on XenServer	25
Repeat Install WSUS Patches (Recursive)	25

Add to Deployment Run book.....	25
Seal Disk	26
Optimize Disk	26
VMware OS Optimization Tool.....	26
Citrix Optimizer	28
Install Base Image Script Framework (BIS-F).....	28
Extra Optimizations	30
Delete Snapshot (Optionally)	32
Optionally Delete the snapshots XenServer	33
Start Imaging	34
Close Disk, Assign it to Collection Citrix PVS	34
Send Finish Email.....	36
Finish the Deployment Run Book.....	36

About:

As you may know I am a huge fan of automating as much as possible and recently I have been really busy with completely automating the Golden Image process with ivanti RES ONE Automation. The goal is to create a new Golden Image by scheduling just one deployment Run book without extra manual tasks.

Author:

Chris Twiest, Senior consultant at Detron in the Netherlands. With more than 10 years of experience in managing and creating workspace environments. Add me on LinkedIn or email me at chris.twiest@dtncomputers.nl

Please visit my blog:

<https://workspace-guru.com>

VERSION 2.0

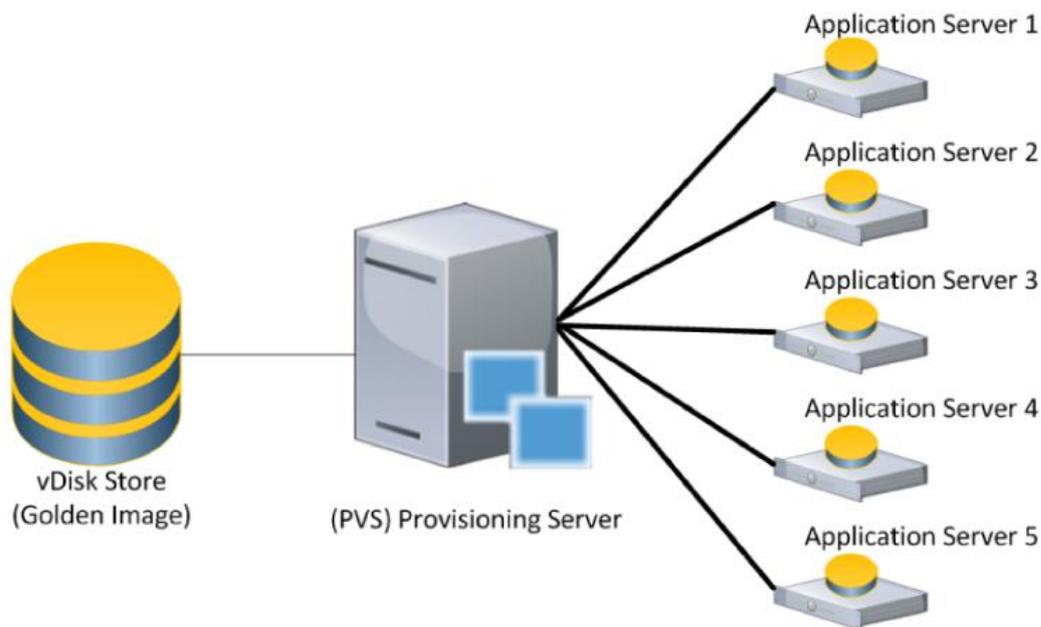
In the last year I have implemented this runbook at multiple customers multiple times. During this period I tweaked the runbook a lot. So Version 2.0 now includes all these tweaks and even more automation by doing more with PowerShell. Since I now also have automated the process for XenServer I have included a lot of XenServer scripts as well.

Disclaimer:

The Ultimate Golden Image Automation Guide or other publications are my own personal opinions and are by no means associated with my employer.

What is a Golden Image and why should I create one automatically?

A Golden Image is a template for a virtual machine (VM), virtual desktop, server or hard disk drive. A Golden Image may also be referred to as a clone image, master image or base image. In this blog we will use the Golden Image as our vDisk in Citrix Provisioning Services (PVS). Citrix PVS lets you boot multiple devices from one vDisk. This way all your VDI's or XenApp servers are the same and after a reboot they return to settings in the Golden Image. Because your whole VDI or XenApp environment will run from this vDisk, it's really important that this disk is created as clean and unspoiled as possible. That's why automating the creation of the Golden Image (vDisk) is important. Another good reason for automating the process is the fact that you don't need to use Citrix Versioning anymore. If something needs to be changed in the disk you don't create a new version but you edit it in the automation process and create a completely new Golden Image (vDisk) again. This way you will always know what's in your disk and how it's build up.



Before we begin; what do we need?

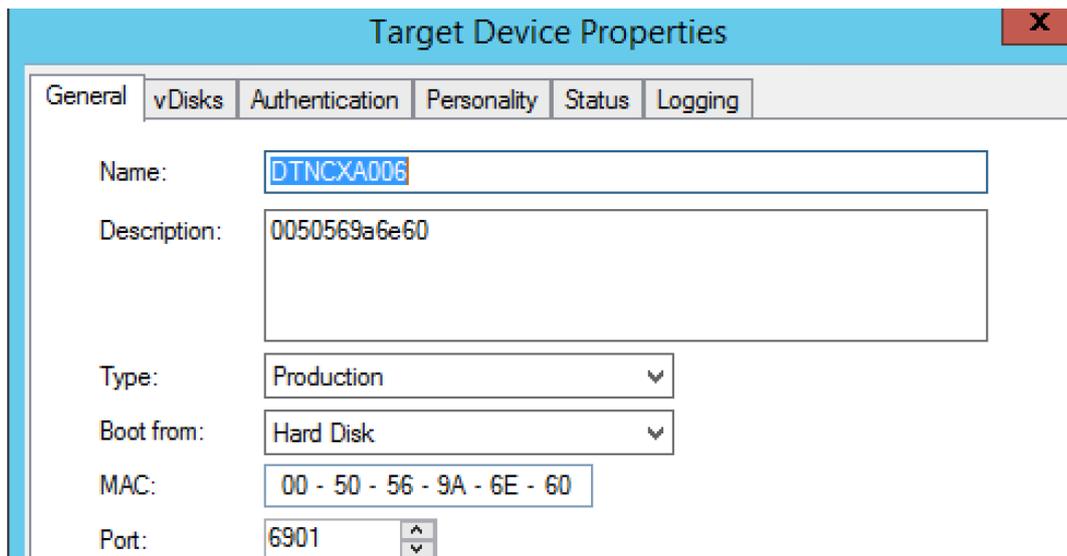
Before we begin with creating the Golden Image we need a few things, these are:

- Citrix XenDesktop/XenApp environment, this guide will be using 7.15LTSR
- Citrix Provisioning Services environment, this guide will be using 7.15LTSR
- Ivanti Automation environment, this guide will be using v2018.1, Of course you can use the provided scripts in SCCM or any other automator.
- VMware vSphere is used in this guide but you can change this to Hyper-V or XenServer. I will add XenServer PowerShell examples.
- Target VM

Target VM specifications

First, we need to create a Target VM. This VM will be the machine on which we will create the Golden Image. Create a VM with at least 2 vCPU's and 4 GB of RAM. Add 2 hard drives to it, one of 100GB (This will be the C: System partition) and one of 40 GB (this will be the D: Cache partition). Now make sure that your target VM can reach the VM network and the PVS network. If you have split this network then you have to add a extra network adapter. Make sure to create reservations for the MAC addresses of the adapters in your DHCP server. This way the target VM will always have

the same IP so that DNS always works. When the hardware is done install your OS on the 100 GB Disk. This guide will use Windows Server 2016 as the Golden Image OS but you can of course change that. After installing Windows make sure to install your Hypervisor Tools like VMware tools. Don't forget to format the second disk and make it the D: Cache Partition and turn the Windows firewall off. If you use computer domain name identification in automation you need to change the workgroup name to the same name as your domain. See [here on the success center](#). If you have all these things in place we need to edit the target VM. Make sure the target VM boots from the PVS network adapter. Then create a new Device in Citrix PVS with the name and MAC address of the target device (which is in this guide DTNCXA006) and set the boot from to Harddisk.



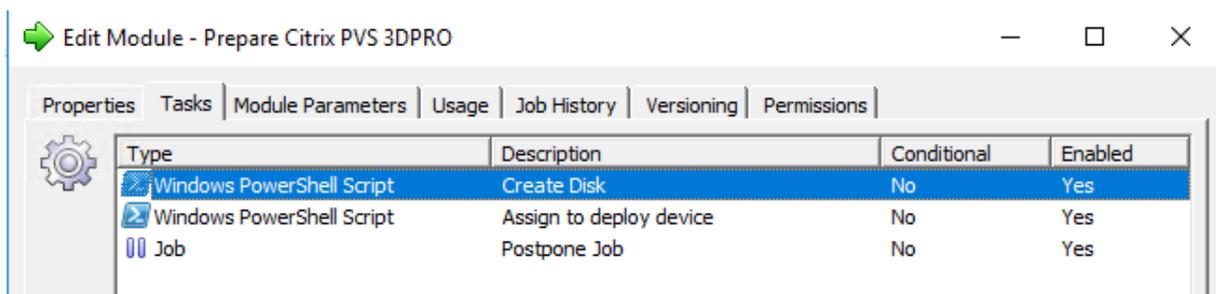
Shut down the target VM and create a snapshot of it, called Before_Runbook. We will revert back to this state every time we start the Golden Image process.

TIP: You could integrate Microsoft MDT into the automation process to also deploy the OS automatically.

Automating Citrix Provisioning Services (PVS)

Because we're going to make a new vDisk from our Golden Image, we first need to create a new empty vDisk in Citrix Provisioning Services and assign it to the Target VM. These are the first tasks we need to automate. Luckily Citrix PVS comes with a PowerShell SDK to create scripts. You can read all about it [here](#). With PowerShell we need to create an Automation task that will do the following:

- Get the date of today just simple PowerShell command (*Get-Date -format dd-MM-yyyy*) and export results to an Automation parameter called `$(date)`
- Create a new empty disk with *Start-PVSCreateDisk*
- Assign the disk to the Target VM



The module must be run on the agent on which your PVS store and PVS server are located. In the guide this is the DTNXd003 . The PowerShell Tasks in the module are the following:

To Add the new vDisk to the PVS library:

```
Import-Module "C:\Program Files\Citrix\Provisioning Services Console\Citrix.PVS.SnapIn.dll"
```

```
$date = Get-Date -format yyyy-MM-dd  
$diskname = "$[Base Name]"+"$Date"
```

```
Start-PVSCreateDisk -Name "$Diskname" -Size "$[Size]" -StoreName "$[Store Name]" -SiteName  
"$[Site Name]" -VHDx
```

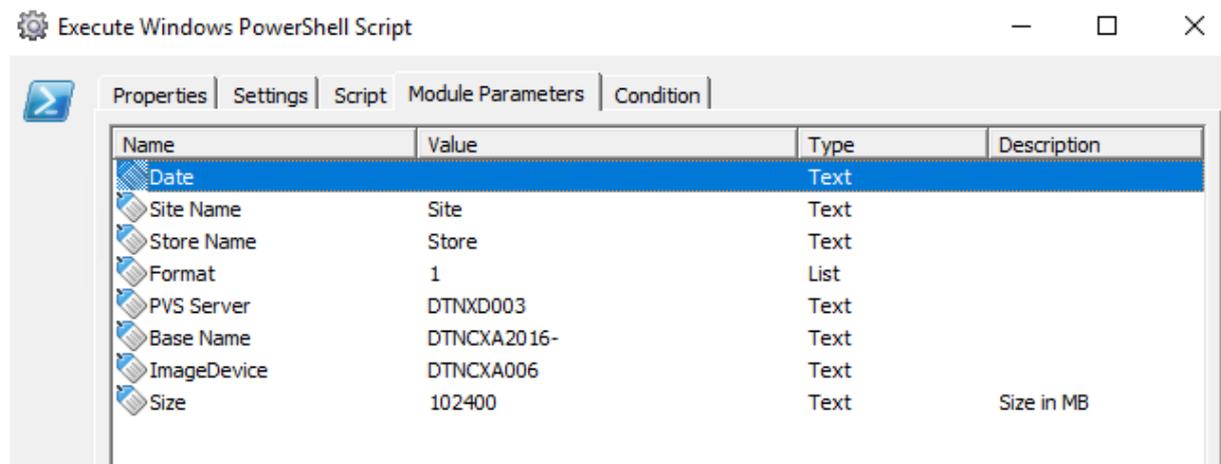
To assign the vDisk to the Target Device:

```
Import-Module 'C:\Program Files\Citrix\Provisioning Services  
Console\Citrix.PVS.SnapIn.dll'
```

```
$date = Get-Date -format yyyy-MM-dd  
$diskname = "$[Base Name]"+"$Date"
```

```
Add-PvsDiskLocatorToDevice -SiteName "$[Site Name]" -StoreName "$[Store Name]" -  
DiskLocatorName $Diskname -DeviceName "$[ImageDevice]" -RemoveExisting
```

The scripts above use the following Automation Parameters:



Automate Target VM snapshot

We need a Target VM with a clean installation of Windows and with an extra disk/Partition (in this guide it will be the D: partition) for PVS cache and it needs to boot from LAN. This is the machine that will become our Golden Image vDisk. In this guide I'm going to use Windows Server 2016 and we're going to turn it into a Citrix XenApp 7.15 server. So, it's important that every time we start the deployment Runbook the target VM is reverted back to its original state. So, we need to create a snapshot of the VM to which we can revert. In this guide I created a snapshot called

Before_Runbook, after the clean Windows 2016 and VMware tools installation and the configuration of the Cache partition. Now we need to create an Automation task that will do the following:

- Revert back to the Before_Runbook snapshot
- Boot the Target VM

We can do this by using the Ivanti Automation VMware vCenter connector which can be downloaded [here on the Ivanti Market Place](#). The module you need to create:

Type	Description
 Prepare VM VMware	
 vCenter Virtual Machine Snapshot	Applies a snapshot state to a virtual machine.
 vCenter Virtual Machine	Changes a virtual machine power status.

Task settings	
vCenter URL	^[vCenter URL]
vCenter administrator	^[vCenter administrator]
vCenter administrator password	^[vCenter administrator password]
VM name	DTNCXA006
Snapshot name	Before_Runbook

Task settings	
vCenter URL	^[vCenter URL]
vCenter administrator	^[vCenter administrator]
vCenter administrator password	^[vCenter administrator password]
VM Name	DTNCXA006
Force power status	no
VM power status	start

You must run these tasks on an agent with VMware PowerCLI installed. In my guide this is the DTNMGT01 my management server.

Automate Target VM XenServer

If you use XenServer you can use PowerShell to do the same process. To get PowerShell working with Citrix XenServer you need to download and install the XenServer SDK from [here under the Development Components section](#).

Development Components

SDK (Software Development Kit) 7.2.0

May 25, 2017
7MB - (.zip) [Download File](#)

Checksums
SHA-256 - 75557e31918d2dda7148d791e365a1c649820a2282fc6a6df1f937161ebf3ea0

To install the PowerShell SDK you need to copy the PowerShell Module folder from the zip to `$env:windir\system32\WindowsPowerShell\v1.0\Modules` To enable the module use the following command: `Import-Module XenServerPSModule`

```
$Xenhost = "XENSERVER"  
$Username = "ROOT"  
$Password = "P@ssw0rd"
```

```
Connect-XenServer -Url https://$Xenhost -UserName $username -Password $password -  
NoWarnCertificates -SetDefaultSession
```

```
$VMname = "DTNCXA006"  
$Snapshotname = "Before_RunBook"
```

```
$VM = Get-XenVM -Name $VMName  
$Snapshot = Get-XenVM -Name $Snapshotname
```

```
Invoke-XenVM -VM $VM -XenAction Revert -Snapshot $snapshot
```

```
start-sleep -Seconds 30
```

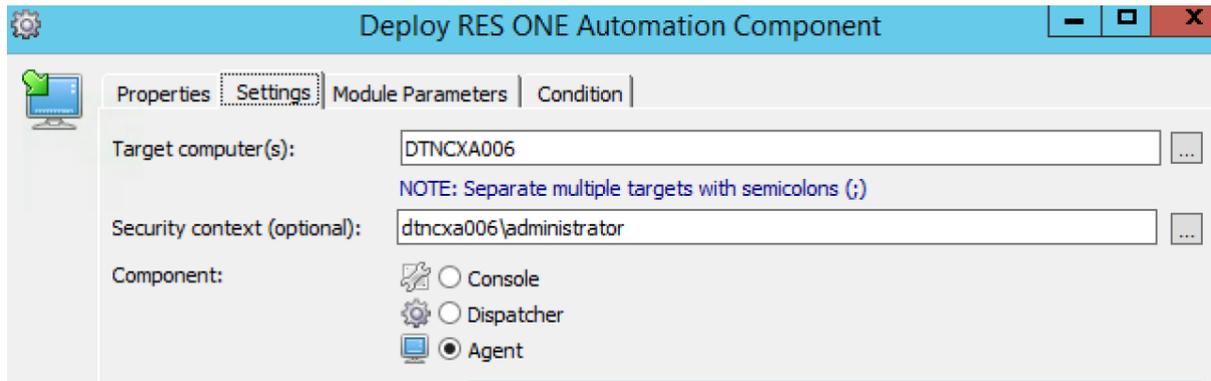
```
Invoke-XenVM -Name $VMname -XenAction Start
```

```
start-sleep -Seconds 60
```

Deploy Ivanti/RES ONE Automation Agent

After the target VM is booted we're going to start running tasks on the machine. This means that we're going to need an Automation Agent on it. We can do this with the Deploy RES ONE Automation task and configure it as followed.

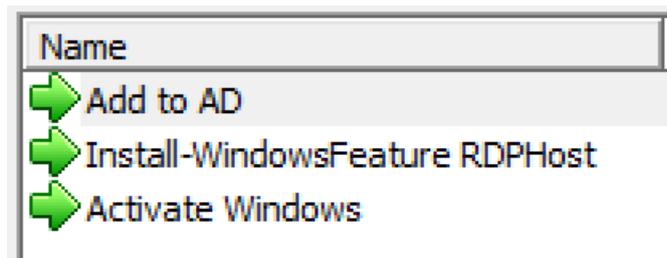
Type	Description
 Job	Postpone Job
 RES ONE Automation Component	Deploy Agent to "DTNCXA006"
 Job	Postpone Job



As you can see I also added two Postpone Jobs. The first one is 60 seconds to make sure that the Target VM is booted. The last postpone job is there because after this task I want to run tasks on the agent and it can take some time before the agent comes online in Automation, so I added another 60 seconds postpone.

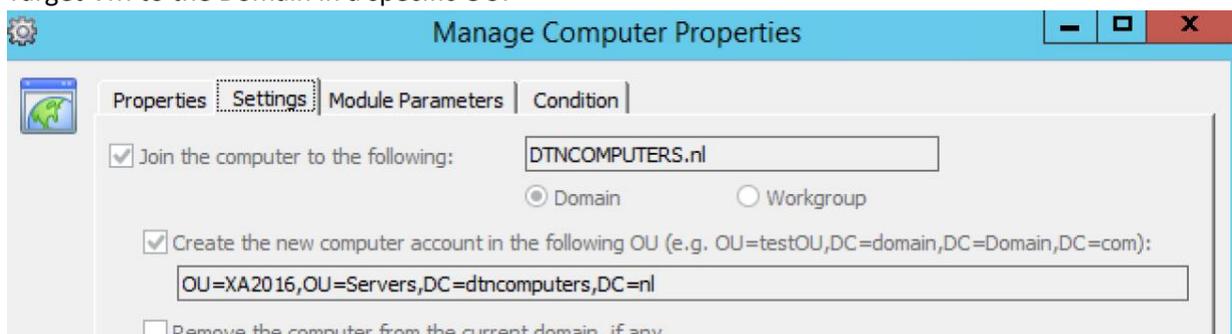
Prepare tasks on Target VM

Now that we have an agent on the target VM we can run some tasks before deploying the software. The tasks that we need to run are the following:

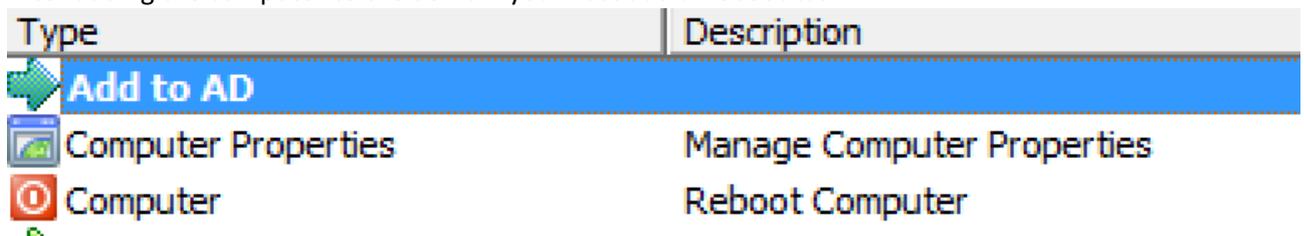


Add to AD

Add to AD is a simple Automation Task called Manage Computer Properties. With this task I add the Target VM to the Domain in a specific OU.



After adding the computer to the domain you must add a Reboot task.



Install Windows Features RDP Host

After that we need to run the module Install-WindowsFeature RDPHost. This is a great module to install Windows features and roles. Download it from the [Ivanti Market place created by James Szivos here](#). The feature we want to install is called RDS-RD-Server. You can add that to the FeatureName parameter. This will effectively run the following PowerShell command: `Install-WindowsFeature -Name "RDS-RD-Server" -IncludeManagementTools` After the script has run the module will reboot your machine. Now all we need to do is add the Domain Users to the local Remote Desktop Users group we do this with the add to local group task in Automation.

Activate Windows

The last module activates Windows to the KMS server by performing the `cscrip //B "%windir%\system32\slmgr.vbs" /ato`

Install WSUS Patches (Recursive)

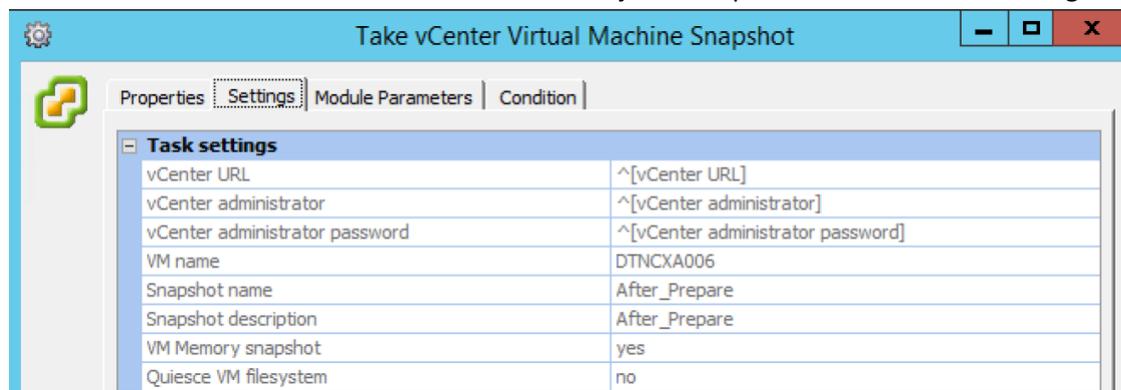
This project will install the Windows Updates from the internet or your WSUS server. The project is created by Virtual Engine's WSUS Integration Management Pack (WiMP) and you can download it [here](#). A great thing about this project is that it will keep repeating itself until all patches are installed.



For the deployment Runbook I copied the project that the WSUS integration pack from Virtual Engine made and added an extra reboot and extra scan.

Create Snapshot (Optionally)

When you run the deployment Runbook it can take some time before it completes, especially with Windows Updates which can take a long time. But what if your Runbook fails? It's a shame to start completely over but you don't want to fix things manually. What can you do? Well, snapshots of course. The same way we revert to the base snapshot in the beginning, that's how we can also take a snapshot with Automation. So, we need to add a snapshot after every stage and in the last stage we need to delete the snapshots. These snapshots are also really handy when testing your deployment Runbook. The module we need to create for this is just a snapshot task with the following setting:



Optionally snapshot on XenServer

The PowerShell command for the snapshot on XenServer is:

```
$Xenhost = "XENSERVER"  
$Username = "ROOT"  
$Password = "P@ssw0rd"
```

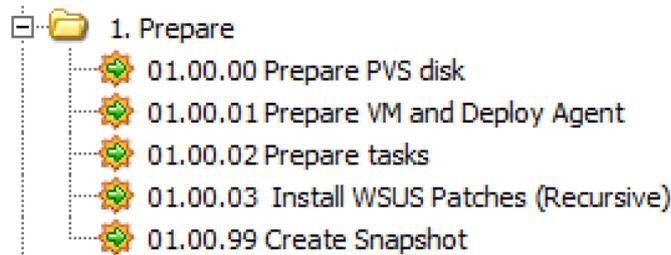
```
Connect-XenServer -Url https://$Xenhost -UserName $username -Password $password -  
NoWarnCertificates -SetDefaultSession
```

```
$VMname = "DTNCXA006"  
$Snapshotname = "After_PrePare"
```

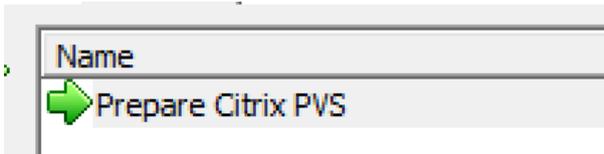
```
$VM = Get-XenVM -Name $VMname
```

```
Invoke-XenVM -Uuid $VM.uuid -XenAction Snapshot -NewName $Snapshotname
```

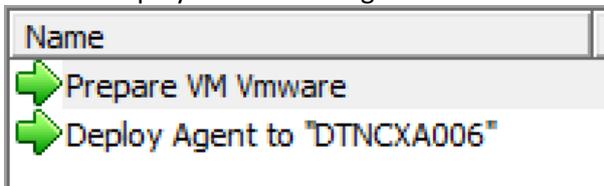
Add Modules to Projects and Projects to Deployment Runbook



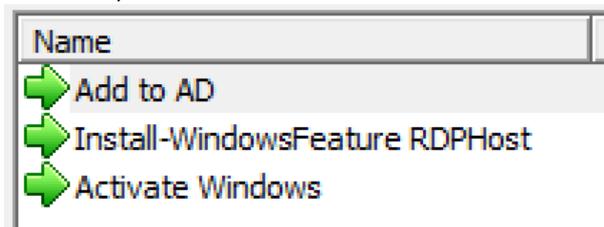
Now that we created all the preparation modules we need to add them into projects and then add the projects to the Deployment Runbook. We need to use a numbering system to keep the projects in the Runbook organized. I use 1 for preparation, 2 for deployment and 3 for Sealing, Optimizing and Imaging the vDisk. The first project we're going to make is the 01.00.00 Prepare PVS project. In this project we will add the module PVS tasks. This project needs to be run on an agent with PVS console and server installed.



After that we create a 01.00.01 Prepare VM module in this project. We add the revert snapshot, boot VM and Deploy Automation Agent module.

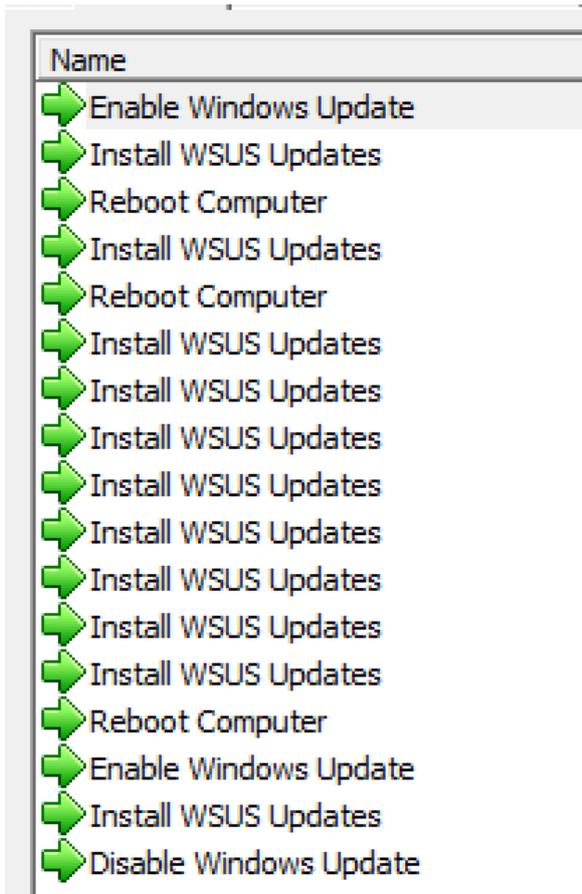


Then we create a third project named 01.00.02 Prepare Task. Here we add the preparation modules, add to ad, install-windowsFeature and Windows Activation.

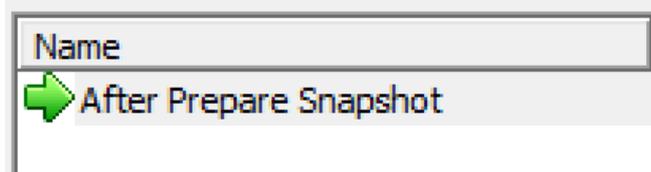


After that we create the 01.00.03 Install WSUS patches recursive project, which is a clone from the default virtual engine wsus integration project but with an extra Reboot added and Install Updates

task added.



The last project is 01.00.99 Create Snapshot. Here we add the After Prepare Snapshot task.



Create Deployment Runbook

Now we need to add the projects to the deployment Runbook. Make sure to set the right Agents with the right projects. Because the target VM doesn't have an Automation Agent yet, set the task that we are going to run on the target VM on Use Run Book Parameter and then select RunBookWho. After that, go to the Runbook parameter and enter your Target VM name. In my Runbook its DTNCXA006.

Job	Who
--> ---- Prepare	
01.00.00 Prepare PVS disk	DTNXD003
01.00.01 Prepare VM and Deploy Agent	DTNMGT01
01.00.02 Prepare tasks	\${RunBookWho}
01.00.02 Install WSUS Patches (Recursive)	\${RunBookWho}
01.00.99 Create Snapshot	DTNMGT01

01.00.02 Prepare tasks

01.00.02 Prepare tasks

Ask during Run Book scheduling

Run on one of the specified agents

Use Run Book Parameter

Name	Value
RunBookWho	DTNCXA006

Deploy Middleware

Edit Project - 02.00.00 Deploy Middleware

Properties | **Modules** | Current Resulting Tasks | Project Parameters | Usage | Job History

Name	Description
Install ".net3"	
Install ".net4.7"	
Install "Adobe Flash 26.0.0.151"	
install "Java Runtime Environment ..."	
Install "Silverlight_x64.exe"	
Reboot Computer	

The middleware we're going to put in the Golden Image will be .NET 3, .NET 4.7, Adobe Flash, Java and Silverlight. We then finish the project with a Reboot.

Install .NET 3

.NET 3 will be deployed with a PowerShell script. The script is:

`Install-WindowsFeature Net-Framework-Core -source "NETWORKPATH\Microsoft\Win2016\sxs"`
 The folder we specify after source needs to have the "microsoft-windows-netfx3-ondemand-package.cab" file. You can find this file in the ISO of the Windows installation media in the folder Sources\sxs.

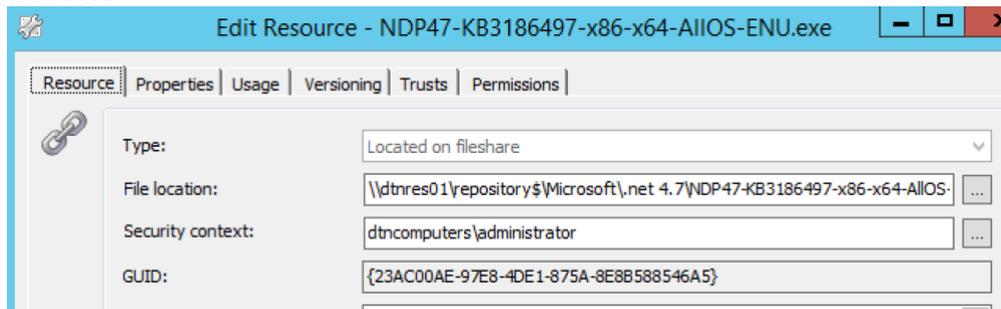
Install ".net3"

Windows PowerShell Script Execute Windows PowerShell Script

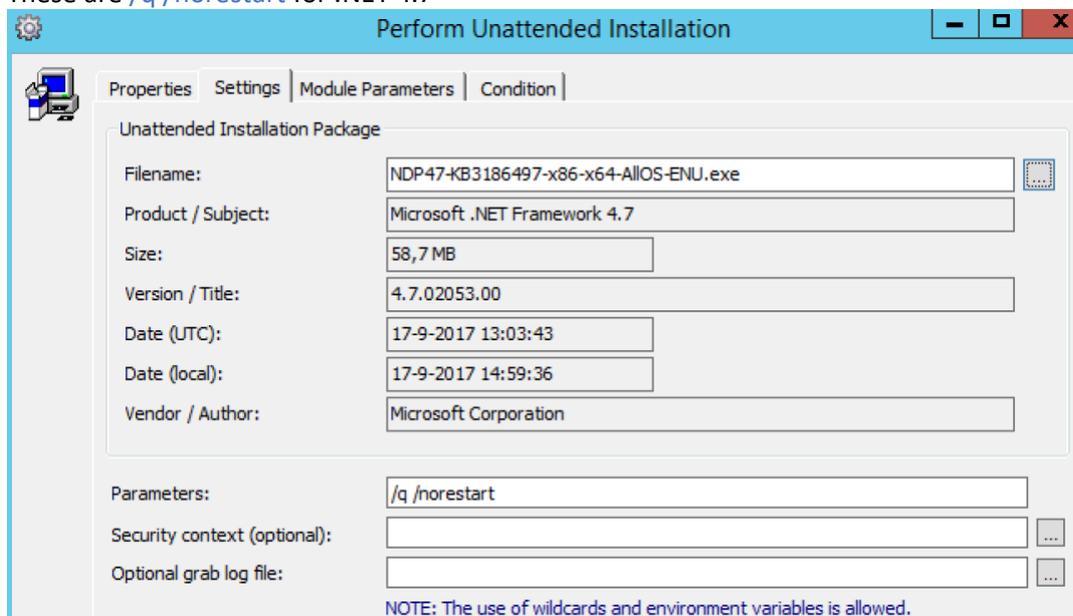
Install .NET 4.7

We're going to deploy .NET4.7 with a Unattended Installation Task. First we're going to download the .net 4.7 offline installation file from [Microsoft here](#). Now we need to add the installer to the

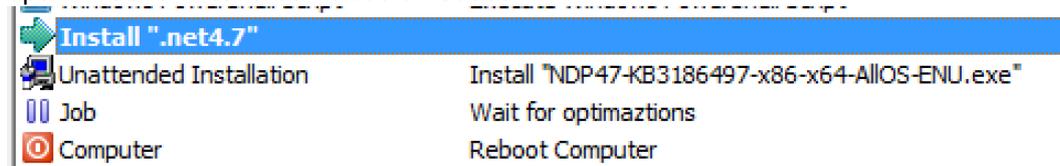
Automation Resources. I always use a linked resource to a File share, this saves space in the database.



Now we need to create the Unattended Installation task and enter the right installation parameters. These are `/q /norestart` for .NET 4.7

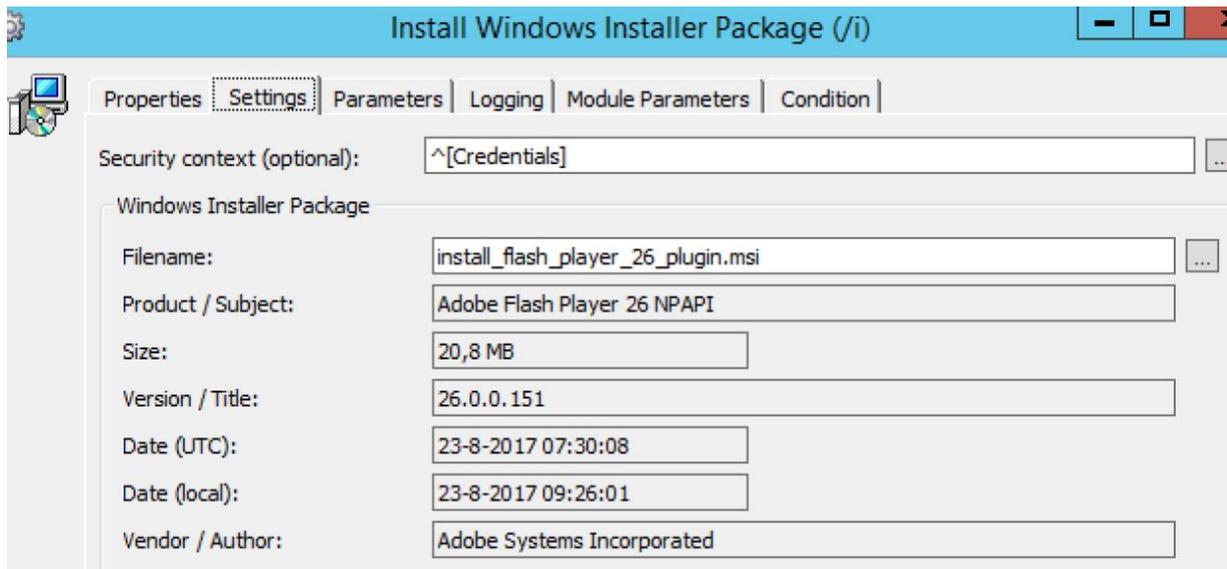


After the installation we add a Postpone job for 3 minutes so that the target VM can finish the .NET optimizations. After that we add a Reboot.



Install Adobe Flash

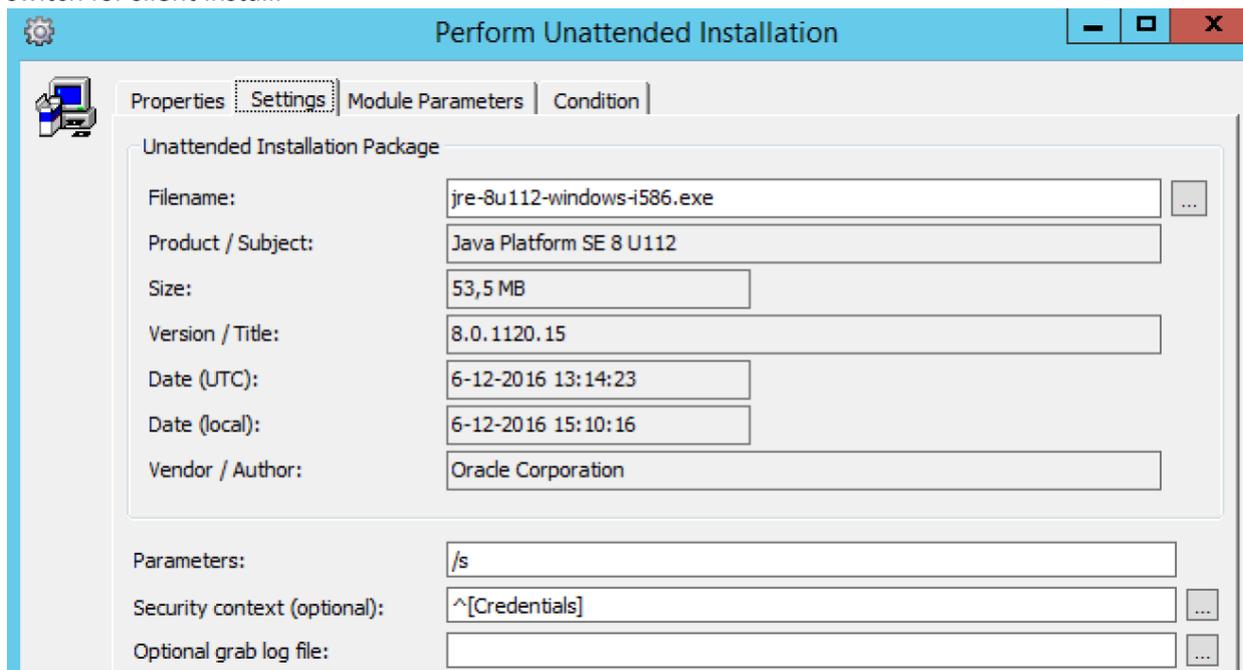
Because we use Windows 2016 we only need to Install the Adobe Flash plugin and not the Active-X component. We need to download the offline flash installer from [Adobe here](#). Now we need to add the installer to an Automation Resource. After that we need to create a Windows Installer Package task and select the Adobe Flash resource.



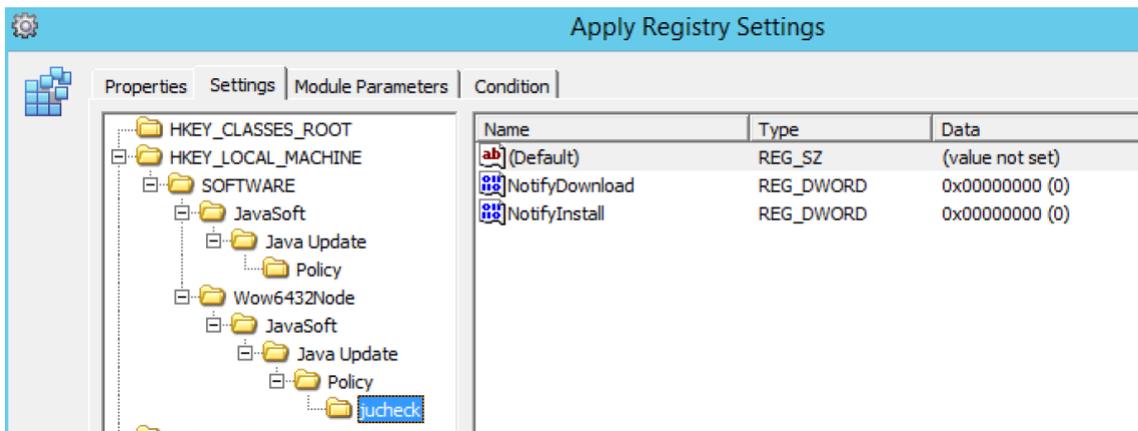
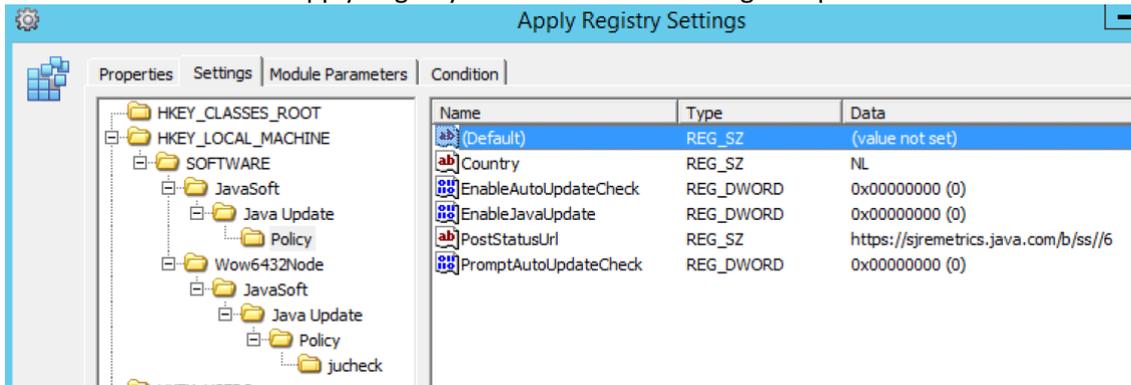
We don't need to set extra parameters.

Install Java

We are going to download the latest Java release from [Oracle here](#). After that we add the installer .Exe to the Automation resources. Now we create an Unattended Installation Task and we set the `/s` switch for silent install.

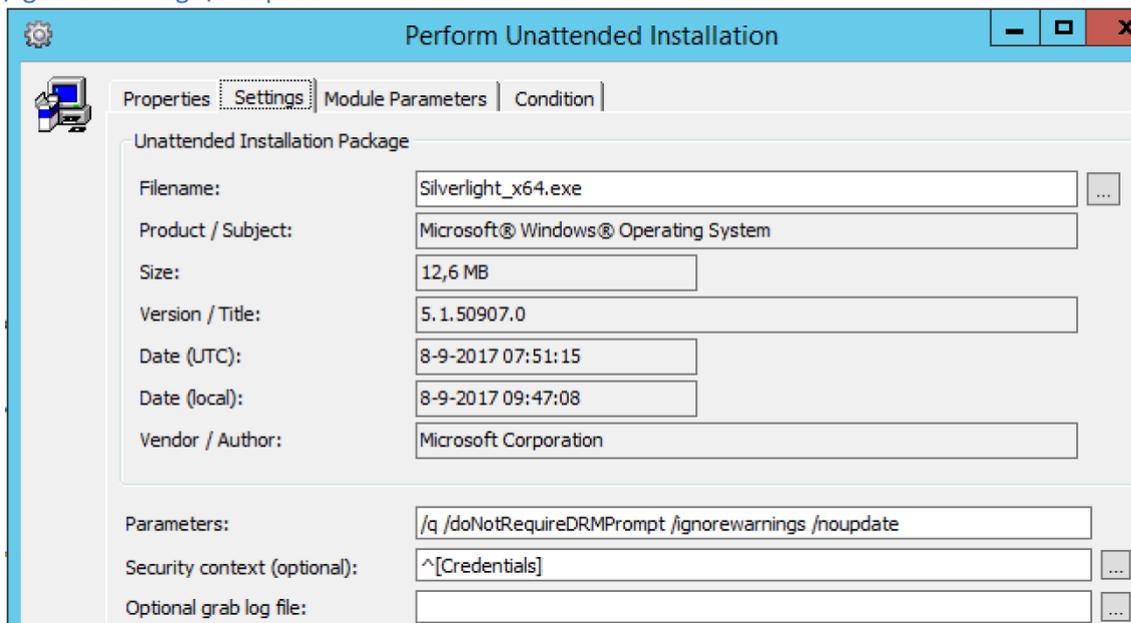


After that we create an Apply Registry task to Disable checking for updates like this:



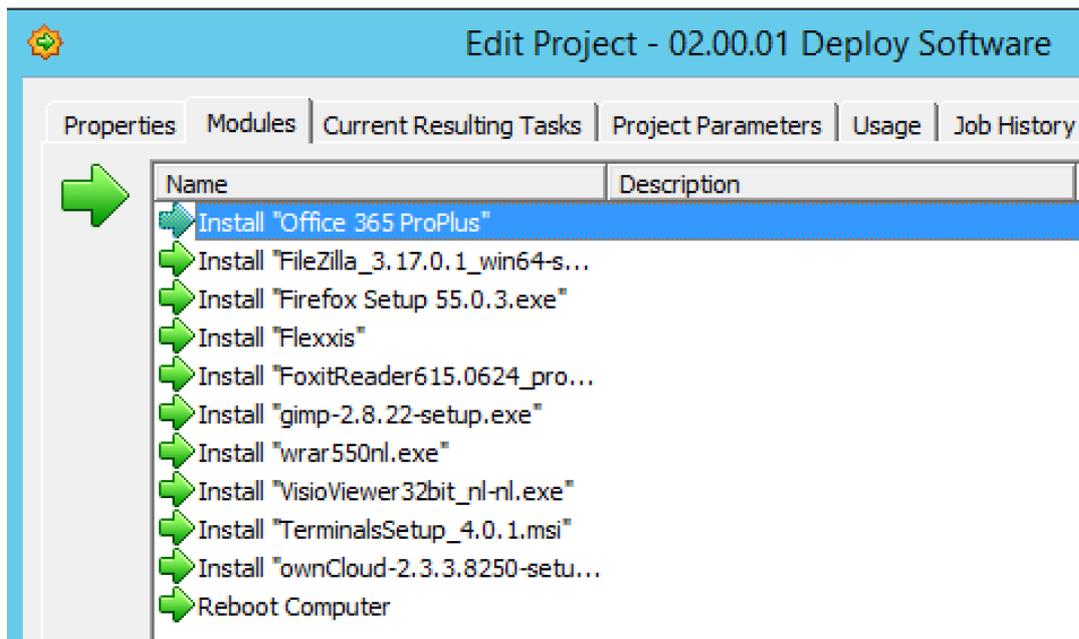
Install Silverlight

We're going to download the latest silverlight plugin from [Microsoft here](#). After we add the .exe to an Automation Resource. Now we're going to create an Unattended Installation Task. Select the Silverlight Resource and enter the following Installation parameters: `/q /doNotRequireDRMPrompt /ignorewarnings /noudate`



Now we just need to add all these modules to a new Project, named 02.00.00 Deploy Middleware.

Deploy Applications



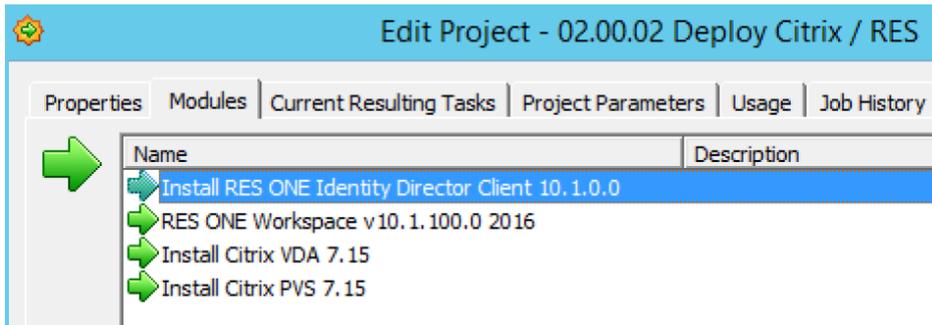
Now we're going to create a project for the user applications. In this project we're going to add all the applications that are needed for the users to work on the system. So applications like Microsoft Office, Exact Financing, PDF reader, GIMP, Firefox, etc etc etc.. This project will be different for every environment based on the environment needs. But to help you out here are some of the installation parameters for some common applications:

- Microsoft Office 365, /Configure "install.xml" create your XML with this awesome [Click-To-Run Configuration XML editor on GitHub](#).
- FileZilla, /S
- Mozilla Firefox, -ms
- Foxit PDF Reader, /verysilent /quiet /norestart
- GIMP 2.8, /silent /norestart /SUPPRESSMSGBOXES
- WinRAR, /S
- Microsoft VisioViewer, /quiet
- OwnCloud Client, /S /LocalSyncFolder=FOLDER /DestinationURL=URL

Now add all the modules to a new Project and name it 02.00.01 Deploy Software. We're also going to add a Reboot task as a last task.

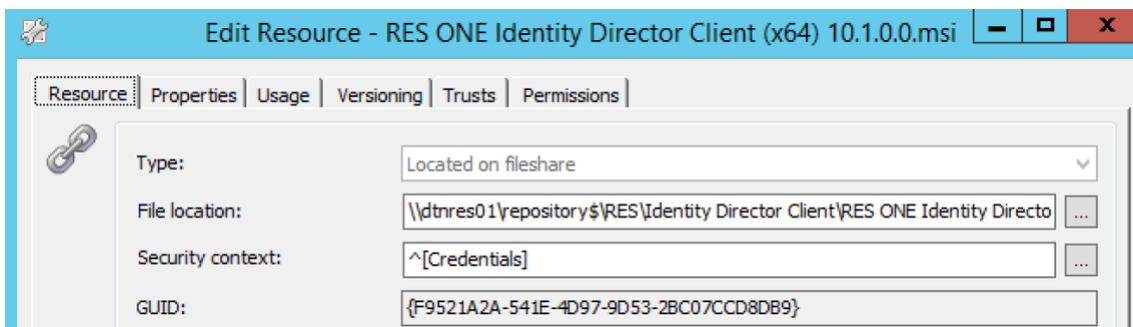
Deploy Citrix / Ivanti Workspace

Now we're going to create modules and a Project for the last category of applications. These are Management Tools, applications we need to lock down the desktop and make the desktop available to the user.

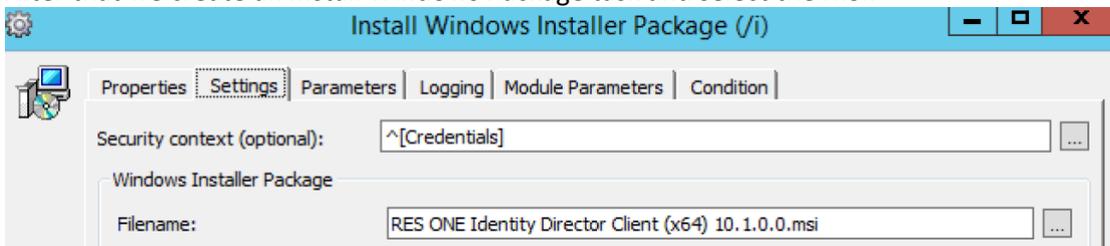


Install Ivanti Identity Directory Client

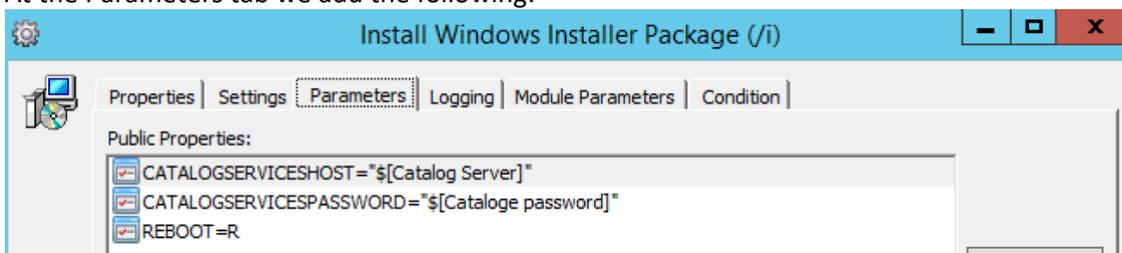
We're going to install the Identity Directory Client from the [RES Success center here](#). After that we are going to add the .MSI to an Automation Resource.



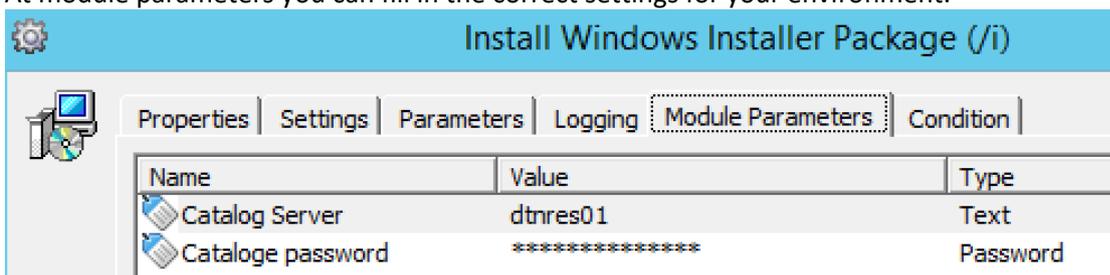
After that we create an Install Windows Package task and select the MSI.



At the Parameters tab we add the following:

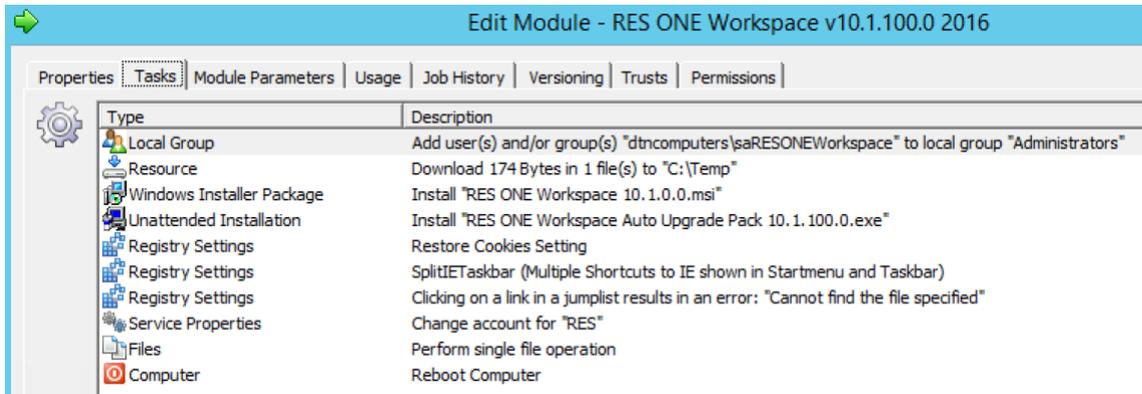


At module parameters you can fill in the correct settings for your environment.



Install Ivanti Workspace Workspace Control

You can download a complete RES ONE Workspace Installation Module from the [Ivanti Market Place here this module is created by Rob Aarts](#). The module comes with handy parameters and registry settings. Because I use a service account for my RES ONE Workspace Agent I added a task to add my service account to the local administrators. After the installation I added a task to change the RES Workspace Agent Service to run with the service account. You can find all [silent installation parameters here](#).



Install Citrix VDA

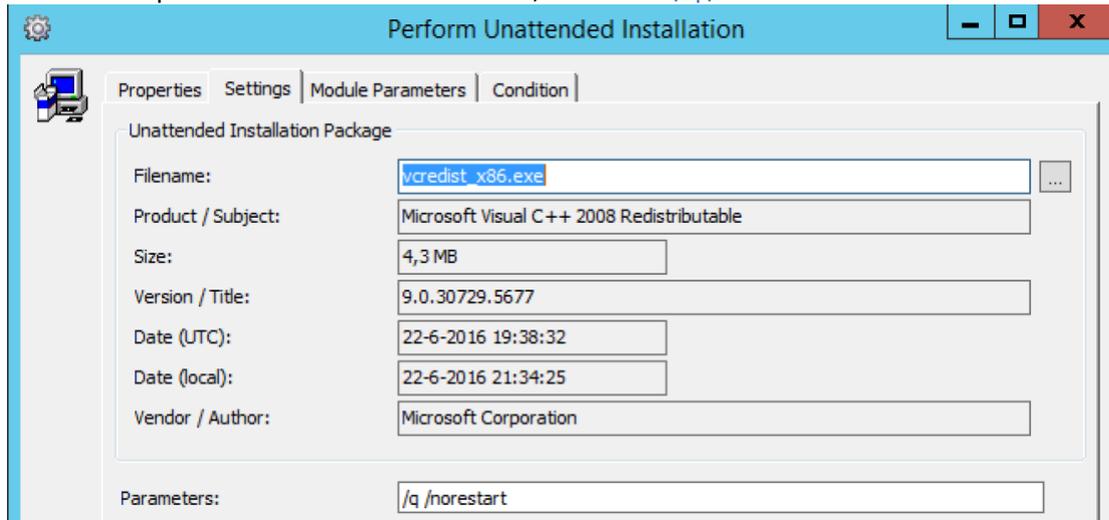
Now we need to create a module to install the Citrix VDA. As we said in part 1 we are using Citrix 7.15. To install the Citrix VDA silent we first need to install all prerequisites. These are Microsoft Visual C++ 2008, 2013, 2015 (x86), (x64) . All these setups are in the Citrix XenDesktop ISO under Support:

Citrix > _Citrix 7.15 > XenApp_and_XenDesktop_7.15 > Support >

Name	Date modified	Type	Size
AdDeploy	4-8-2017 21:18	File folder	
Configuration	4-8-2017 21:18	File folder	
DirectorWMIProvider	4-8-2017 21:07	File folder	
DotNet452	4-8-2017 21:11	File folder	
Installer4.5	4-8-2017 21:11	File folder	
OpenGL_Software_Accelerator	4-8-2017 21:11	File folder	
SharedManagementObjects	4-8-2017 21:11	File folder	
SQLExpress	4-8-2017 21:18	File folder	
SQLLocalDB2014	4-8-2017 21:18	File folder	
Tools	4-8-2017 21:17	File folder	
UnRegisterVDA	4-8-2017 21:18	File folder	
VcRedist_2008_SP1	4-8-2017 21:11	File folder	
VcRedist_2013_RTM	4-8-2017 21:18	File folder	
VcRedist_2015	4-8-2017 21:11	File folder	

Now we're going to add all the vc_redist_x86.exe and vc_redist_x64.exe of these folders to Automation Resources. We need to create an Unattended Installation Task for all these vc_redist_XXX.exe. The

silent install parameters are the same for all, these are: `/q /norestart`



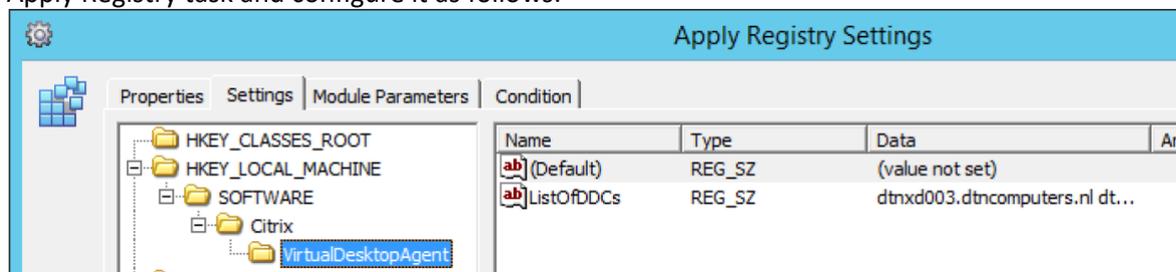
After every vcredist installation we add a 10 Second postpone job and we finish with a Reboot. Now we're ready to start the installation of the XenDesktopVDA.exe installation.

Here we run into an problem with the installation with Ivanti Automation. I and more people have tried to install the VDA correctly with a normal Unattended installation tasks. But although it finishes in Automation the VDA is not completely installed. But If you use a Execute Command task and then at a script action like this:

```
"\\Repository\repository$\Citrix\7.15\XD\x64\XenDesktop Setup\XenDesktopVDASetup.exe" /quiet /components VDA /controllers "DELIVERYCONTROLLER.FABRIKAM.COM" /masterimage /noreboot /optimize /disableexperiencemetrics /enable_hdx_ports /enable_hdx_udp_ports /enable_remote_assistance /exclude "Personal vDisk","Machine Identity Service","Citrix User Profile Manager","Citrix User Profile Manager WMI Plugin","Citrix Universal Print Client","Citrix Telemetry Service","Citrix Personalization for App-V - VDA"
```

It will work!

After the installation task we add another postpone task of 60 seconds to make sure the setup is correctly finished. Because it can sometimes happen that the controller parameter doesn't work correctly, we're going to add the controllers again through the registry. We're going to create an Apply Registry task and configure it as follows:



Of course you must add your own Delivery Controllers as the value of the `ListOFDDCs` . If you have multiple leave a blank space between them.

Now we add another reboot and the module is done.

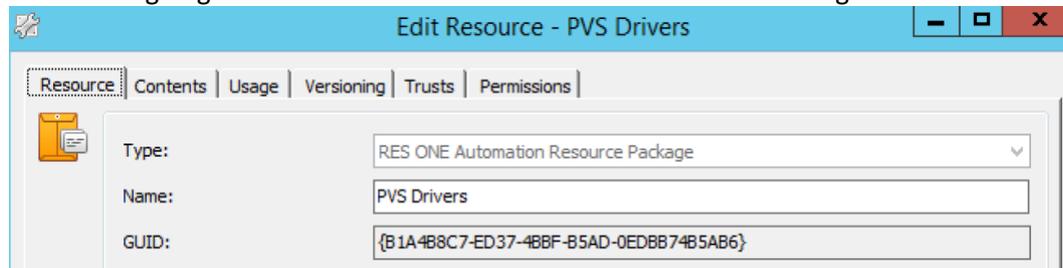
Type	Description	Conditional	Enabled
Unattended Installation	Microsoft Visual C++ 2008 (x86)	No	Yes
Job	Postpone Job	No	Yes
Unattended Installation	Microsoft Visual C++ 2008	No	Yes
Job	Postpone Job	No	Yes
Unattended Installation	Microsoft Visual C++ 2013 (x86)	No	Yes
Job	Postpone Job	No	Yes
Unattended Installation	Microsoft Visual C++ 2013	No	Yes
Job	Postpone Job	No	Yes
Unattended Installation	Microsoft Visual C++ 2015 (x86)	No	Yes
Job	Postpone Job	No	Yes
Unattended Installation	Microsoft Visual C++ 2015	No	Yes
Computer	Reboot Computer	No	Yes
Command	Execute "[SCRIPT]"	No	Yes
Job	Postpone Job	No	Yes
Registry Settings	Apply Registry Settings	No	Yes
Computer	Reboot Computer	No	Yes

Install Citrix PVS Target Device Software

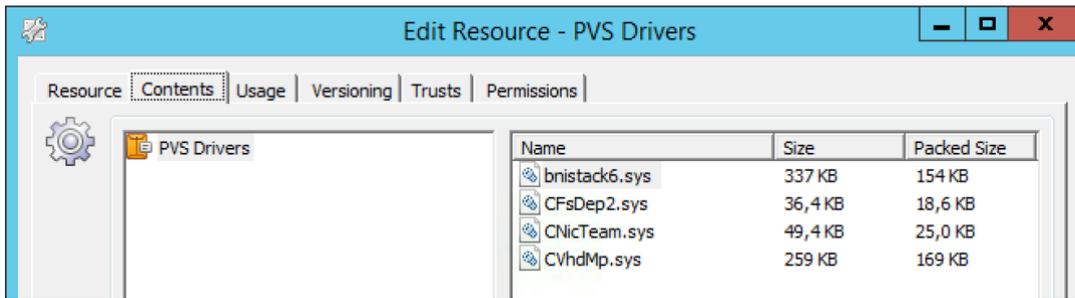
So the last application we're going to install is the Citrix PVS Target Device Software. And this is a tricky one! Because the installation registers a few drivers .sys files in the system32 folder and this doesn't work when silent installing the PVS target device software. To fix this we need to manually copy the drivers, but first we need to get the drivers. We do this by installing Citrix PVS Target Device manually for one single time. Logon to Target VM and install the target device software. You can find this on the ProvisioningServices715 ISO, in the folder Device. It's called PVS_Device_64.exe. After the one-time manual install we go to C:\Windows\System32 and copy to following files to a network share:

-  bnistack6.sys
-  CFsDep2.sys
-  CNicTeam.sys
-  CVhdMp.sys

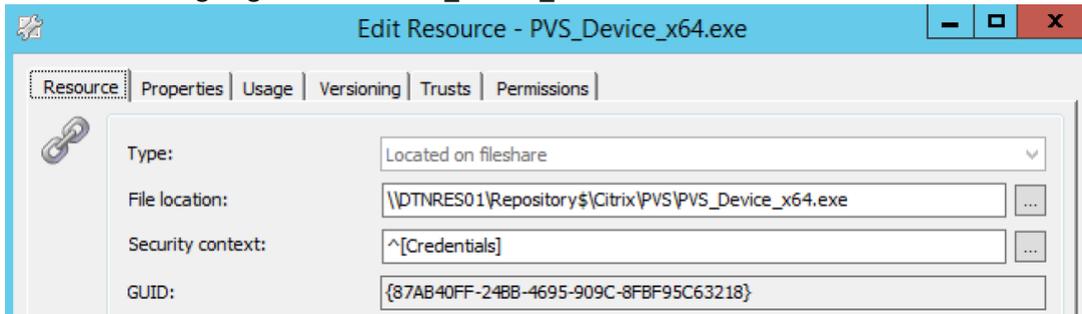
Now we're going to add these files to an Automation Resource Package:



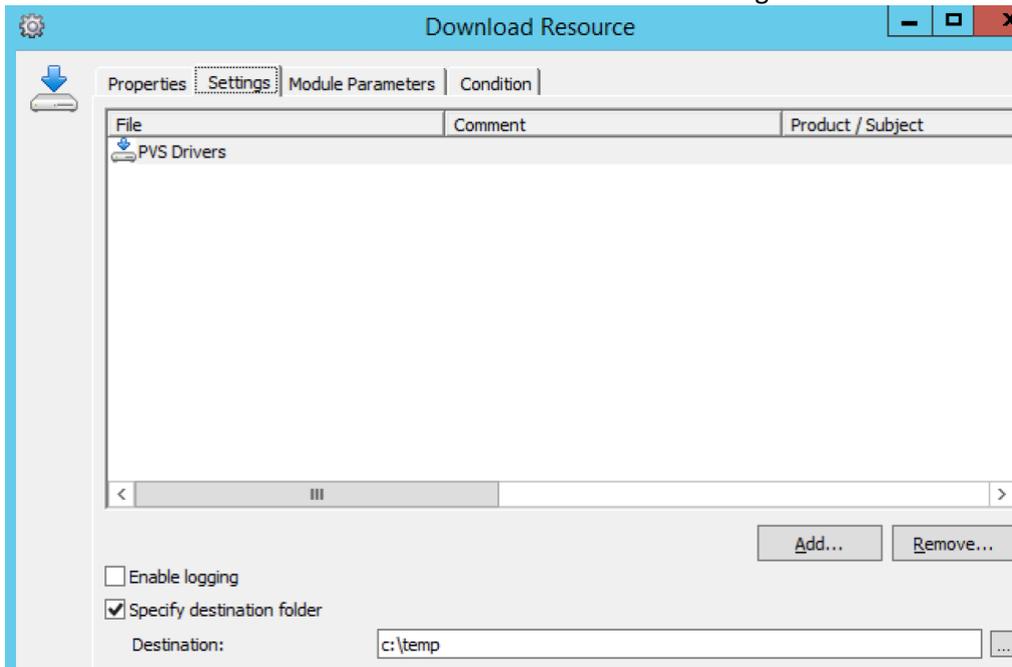
Add the drive files to the Contents tab:



Now we're also going to add the PVS_Device_x64.exe to an Automation Resource:



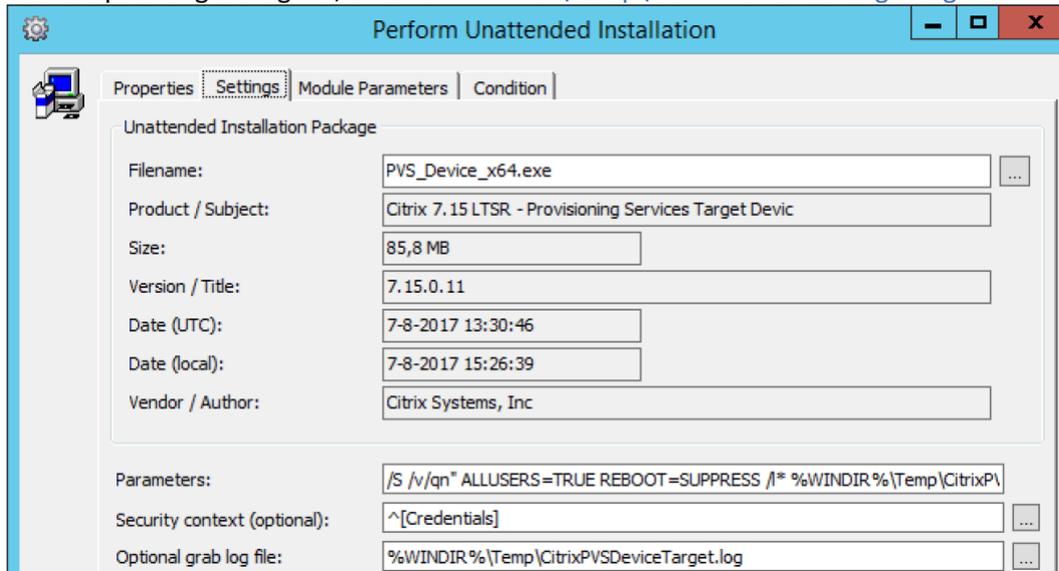
Create a new module with a resource Download task and configure it as follows:



Create an Unattended Installation task. Select the PVS_Device_x64.exe and add the following parameters:

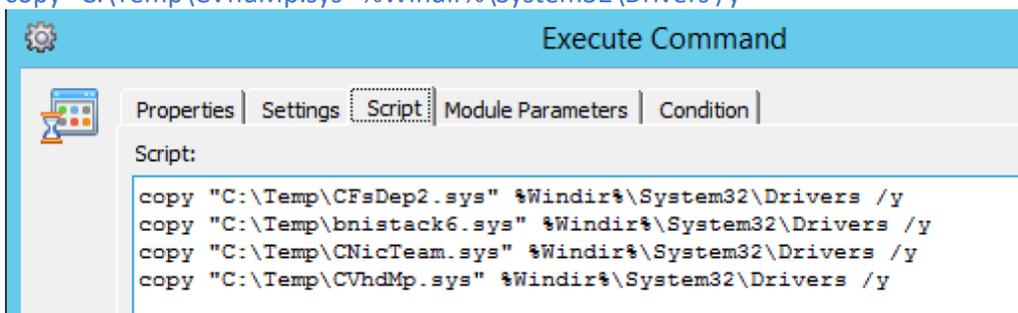
`/S /v/qn" ALLUSERS=TRUE REBOOT=SUPPRESS /!* %WINDIR%\Temp\CitrixPVSDDeviceTarget.log"`

At the Optional grab log file, enter %WINDIR%\Temp\CitrixPVSDDeviceTarget.log

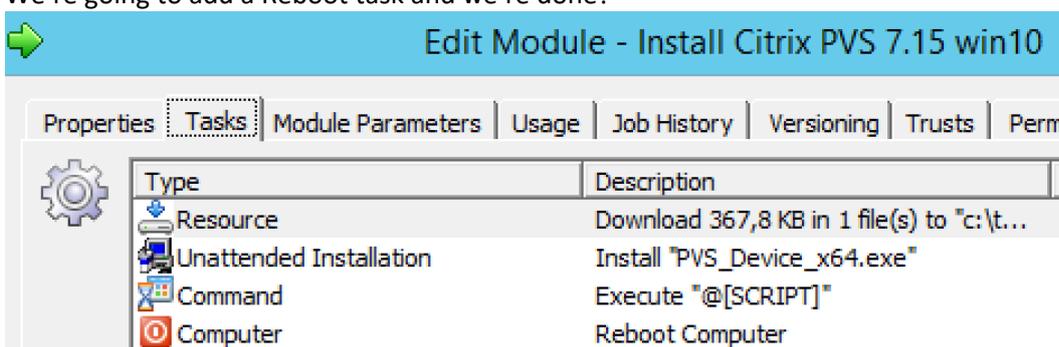


Now we're going to add an Execute command task and add this command:

```
copy "C:\Temp\CFsDep2.sys" %Windir%\System32\Drivers /y
copy "C:\Temp\bnistack6.sys" %Windir%\System32\Drivers /y
copy "C:\Temp\CNicTeam.sys" %Windir%\System32\Drivers /y
copy "C:\Temp\CVhdMp.sys" %Windir%\System32\Drivers /y
```



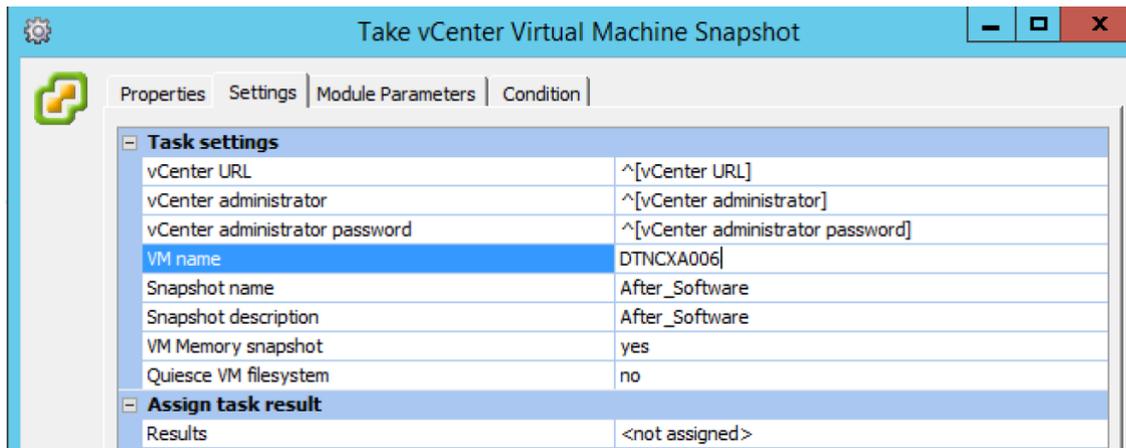
We're going to add a Reboot task and we're done!



Add all the modules to a new Project and name it 02.00.02 Deploy Citrix RES.

Create Snapshot (Optionally)

For the same reasons as in part 1 we can add a extra snapshot of the targetVM after installing the software. Create a new snapshot task and configure it as follows:



Add the snapshot task to a Project named 02.00.03 After Software Snapshot

Optional Snapshot on XenServer

Use the following PowerShell script for a After Software snapshot on XenServer.

```
$Xenhost = "^[XenServer]"
$username = "^[Root Account]"
$password = "^[Root Password]"
```

```
Connect-XenServer -Url https://$Xenhost -UserName $username -Password $password -
NoWarnCertificates -SetDefaultSession
```

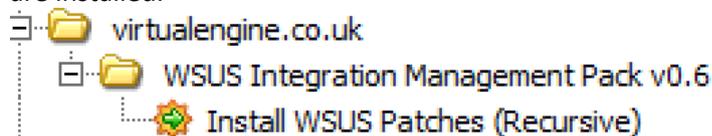
```
$VMname = "$[DeploymentAgent]"
$Snapshotname = "After_Software"
```

```
$VM = Get-XenVM -Name $VMname
```

```
Invoke-XenVM -Uuid $VM.uuid -XenAction Snapshot -NewName $Snapshotname
```

Repeat Install WSUS Patches (Recursive)

We already run this project but you can add it again to make sure the latest office patches are also included. This project will install the Windows Updates from the internet or your WSUS server. The project is created by Virtual Engine's WSUS Integration Management Pack (WiMP) and you can download it [here](#). A great thing about this project is that it will keep repeating itself until all patches are installed.



For the deployment Runbook I copied the project that the WSUS integration pack from Virtual Engine made and added an extra reboot and extra scan.

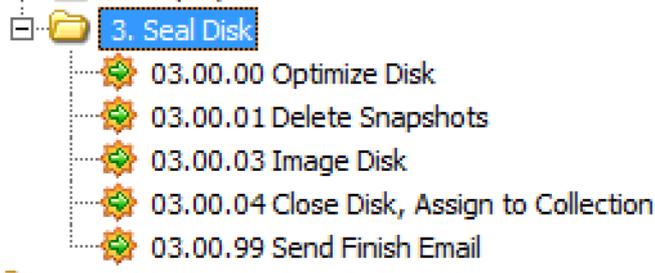
Add to Deployment Run book

Now add the projects to the deployment Run book. Make sure that all the software projects use the RunBookWho parameter except the Snapshot project that needs to be run on an agent with PowerCLI installed. In this guide this is the DTNMGTO1.

--> ---- Deploy Software			
02.00.00	Deploy Middleware	\$_[RunBookWho]	Yes
02.00.01	Deploy Software	\$_[RunBookWho]	Yes
02.00.02	Deploy Citrix / RES	\$_[RunBookWho]	Yes
02.00.03	After Software Snapshot	DTNMGT01	Yes
02.00.04	Install WSUS Patches (Recursive)	\$_[RunBookWho]	Yes

Seal Disk

In this last part of the blog we're going to optimize and seal the golden image and create a vDisk through the Citrix Imaging process. Of course we will automate all these steps.



Optimize Disk

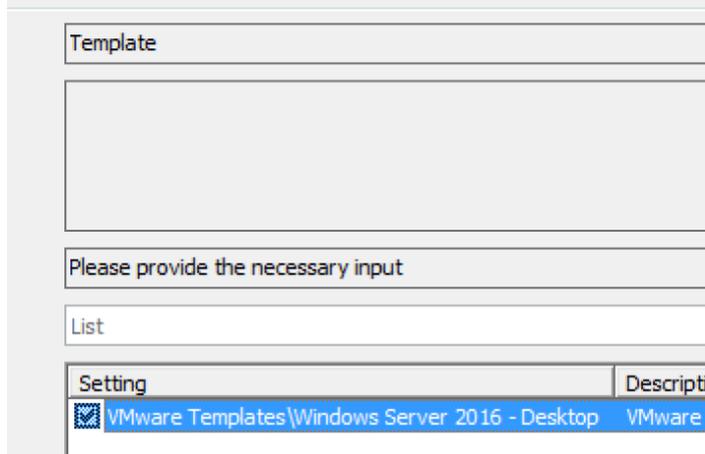
VMware OS Optimization Tool

Before we're going to image the vDisk we need to optimize the OS. OS optimization is needed to create an as lean as possible golden image. Because Windows 2016 can be used for many roles and can be virtual or physical, installed Microsoft adds a lot of services and register settings to make it as flexible as possible. But since we're going to use the OS virtual and as XenApp host there are a lot of services and settings we don't need. By disabling all these services and settings we create a faster and leaner Golden Image. Just think about all the wasted resources if you don't optimize your OS and then deploy 200 XenApp server from your golden image! To optimize the OS we're going to use the [VMware OS optimization tool which can be downloaded here](#). We can use two ways to automate the VMware OS optimization tool. The first one is created by [Rob Aarts and is part of his Windows 10 rollout startup package which can be downloaded here](#). You can download the package and use the VMware Optimization tool Module from it. The module will do the following:

Type	Description	Conditional
Resource	Download 12,1 MB in 1 file(s) to "%temp%"	No
Command	Start the OSOT tool (needs to started once before a template can...	No
Job	Wait 30 sec.	No
Command	Stop the VMware OSOT tool	No
Command	Analyse the client	No
Job	Wait 30 sec.	No
Command	Stop the VMware OSOT tool	No
Command	Apply OSOT Template to the client	No
Files	Perform single file operation	No
Service Properties	Start Windows Firewall	No
Service Properties	Start Windows Audio	No
Computer	Reboot Computer	No

We need to add 2 things to the module because after running the VMware OS Optimization tool the Windows Firewall, Audio and Audio Endpoint Builder services are turned off. Because we're using the Golden Image as XenApp server the user probably wants to have sound enabled. And if we disable the Firewall services here, Windows Search Spotlight won't work anymore so using the spotlight in the start menu will not pop-up the search window. This is not desirable for users. The last thing we

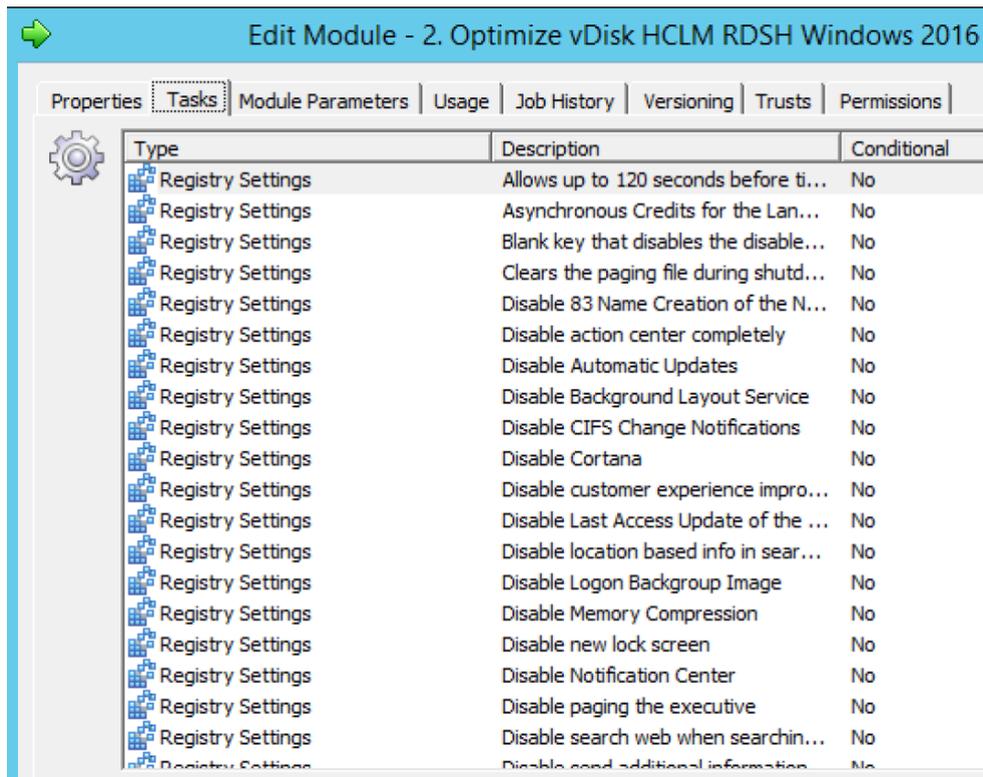
need to do is change the Template variable to [VMware Templates\Windows Server 2016 – Desktop](#)



The screenshot shows a configuration window with a 'Template' field at the top. Below it is a large empty text area. A message says 'Please provide the necessary input'. There is a 'List' button. At the bottom, a table is visible with two columns: 'Setting' and 'Description'. One row is highlighted in blue, containing the text 'VMware Templates\Windows Server 2016 - Desktop' under 'Setting' and 'VMware' under 'Description'.

The VMware OS Optimization Tool turns off services and changes registry but of course RES / Ivanti ONE Automation can also do it. So the second way to run the VMware OS Optimization Tool is to not run it but take all tasks it does and create tasks in RES / Ivanti ONE Automation instead. This is of course a lot of work but luckily [Arjan van t'Hoff has already done this and you can download it from the Ivanti Market Place here.](#) If you want to use those modules instead, don't forget to remove the services mentioned above. One of the great things about doing it this way is that you can see the specific settings that are being configured and turned off. Then again you won't be benefitting from new settings in updates of the VMware OS Optimization Tool.

- 
- The screenshot shows a task list in a software interface. A folder icon is followed by the text 'Optimize vDisk Windows 2016 RDSH'. Below this folder, there are five tasks, each with a green arrow icon pointing to the right:
- 1. Optimize vDisk Disable Scheduled Tasks RDSH Windows 2016
 - 2. Optimize vDisk HCLM RDSH Windows 2016
 - 3. Optimize vDisk Services RDSH Windows 2016
 - 4. Optimize vDisk Improve Login Time RDSH Windows 2016
 - 9. Reboot vDisk RDSH Windows 2016

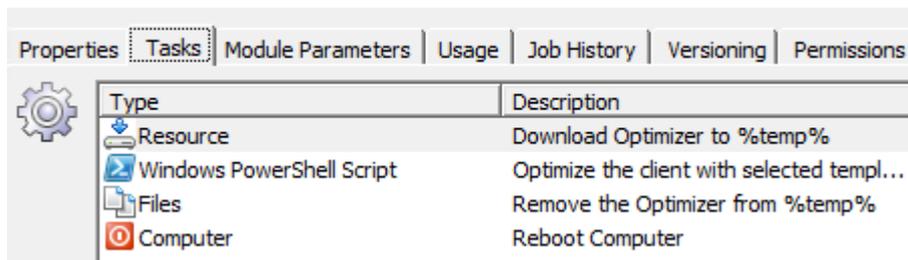


And of course we finish the VMware OS Optimization off with a Reboot.

Citrix Optimizer

You can [download the Citrix Optimizer](#) from Citrix and run this in your image. Read about the differences between Citrix and VMware Optimizer [here on my blog](#). You can download the module to run the Citrix Optimizer from the [Ivanti Market Place here](#).

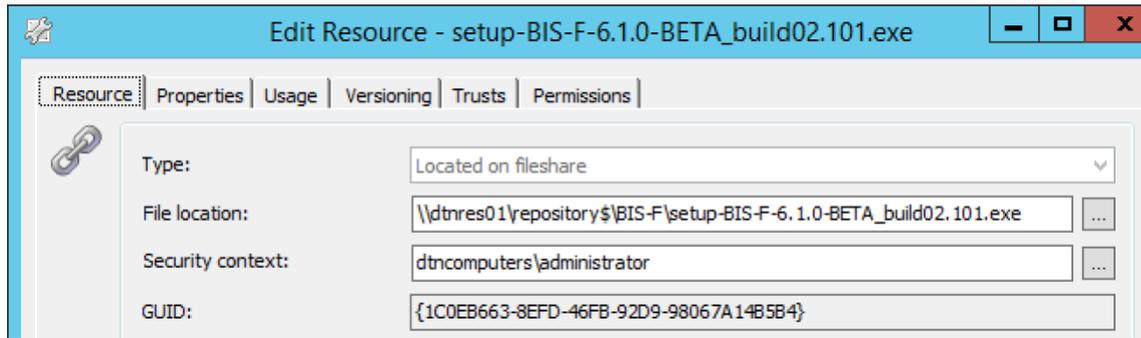
👉 Edit Module - Citrix Optimizer Execute



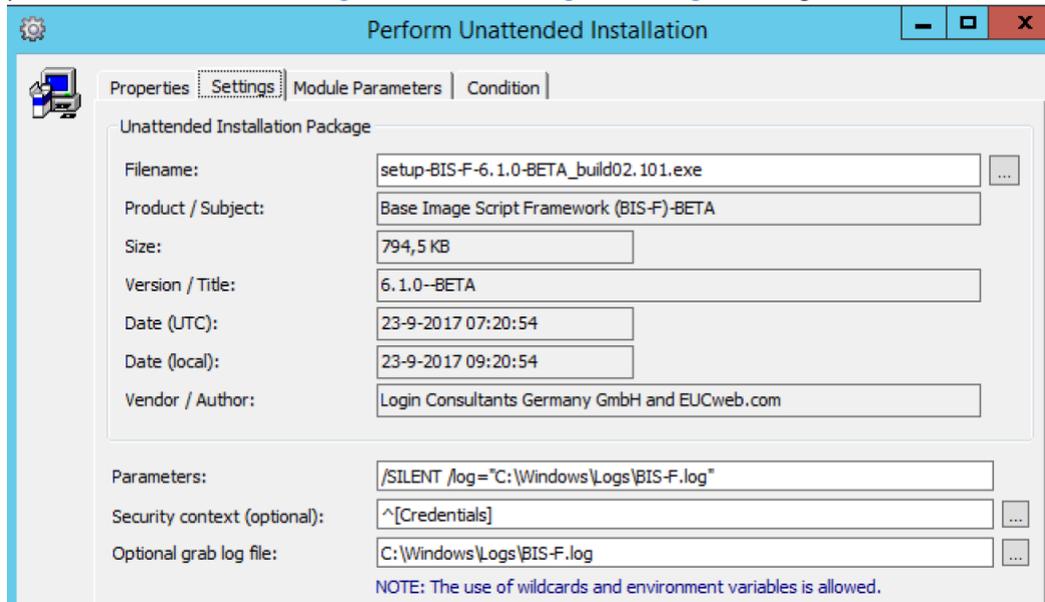
Install Base Image Script Framework (BIS-F)

We have now Optimized the Windows 2016 OS to be run as a XenApp Server. But we're not only going to run the Golden Image as a XenApp server, we're going to use it as a vDisk and provide the disk to multiple XenApp servers through Citrix Provisioning Services. So what we now need to do is make the disk generic and move items like Eventlogs, Print Spooler, Citrix Cache and RES cache which are server specific to the Persistent D: disk. To do this we can use the Base Image Script Framework because it does all these things and much, much more! One of the great things it does is image the disk to a vDisk but more on that later on. First we need to install BIS-F and configure it. [Download BIS-F 6.1.0 from login consultants here](#). After downloading the .EXE add it to RES / Ivanti ONE

Automation Resources:



Now we're going to create a new module with an unattended installation task, the installation parameters are: `/SILENT /log="c:\Windows\Logs\BIS-F.log"`. Configure the task as follows:



BIS-F is configured through Microsoft GPO the ADMX file to configure BIS-F is located in `c:\Program Files (x86)\Base Image Script Framework (BIS-F)\ADMX`

This means that we first have to run the installer on a management server to get the ADMX file. After that we can of course remove the software. We need to copy the content from that folder to the domain policy store, this is normally `\\DOMAIN\Sysvol\Policies\PolicyDefinitions`. After doing that we can create a new GPO object and browse to `COMPUTER CONFIGURATION > Administrative Templates > Base Image Script Framework (BIS-F)`. Of course the configuration will be different from environment to environment but there are a few mandatory settings we need to set.

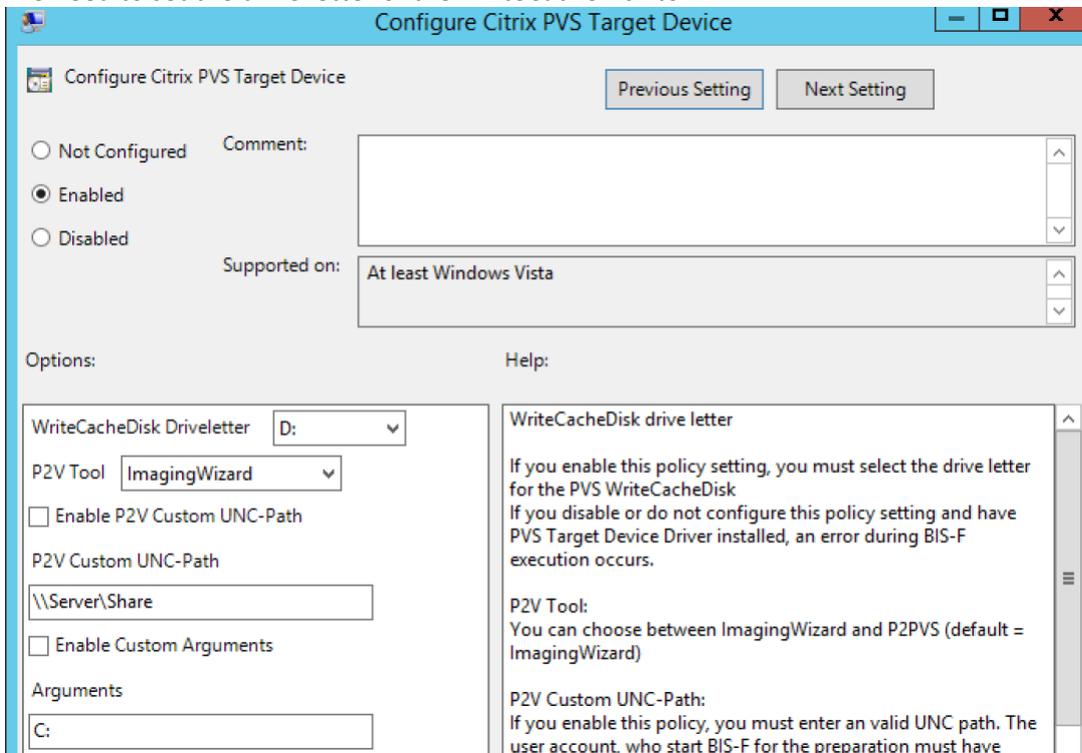
In Global, set the Configure VerySilent and Shutdown Base Image after sealing to Enabled. The rest in Global is up to you.

Setting	State	Comment
Configure VerySilent	Enabled	No
Shared Configuration	Disabled	No
Configure Folders	Enabled	No
Configure Logging	Disabled	No
Configure Personalization	Disabled	No
Suppress Pending Reboot	Disabled	No
Shutdown Base Image after sealing	Enabled	No

Then go to RES folder and set both policies to Disabled or else the RES ONE Automation imaging task will fail.

Setting	State	Comment
Disable RES ONE Automation Agent	Disabled	No
Disable RES ONE Automation Agent Service	Disabled	No

The last thing we must configure is in the Citrix folder, the policy Configure Citrix PVS Target Device. We need to set the drive letter of the WriteCacheDisk to D:

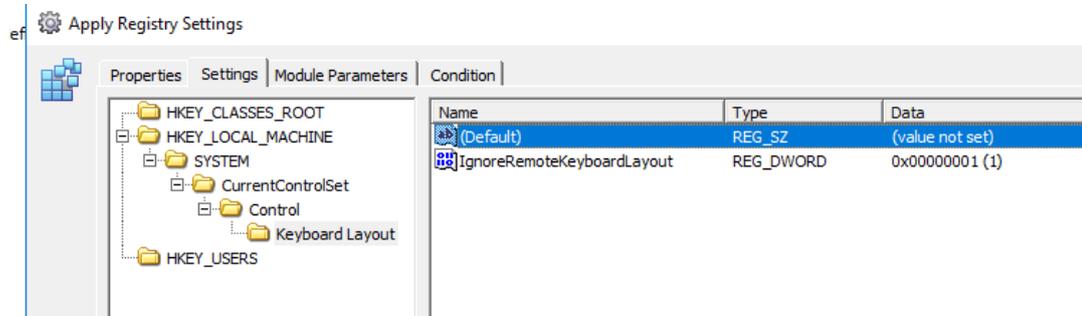


Now we need to make sure that our target VM gets these settings by setting the GPO on the right OU. I would advise you to check out all the policy settings in the BIS-F policy. Add the VMware OS Optimization Tool and Install BIS-F modules to a new Project and name it 03.00.00 Optimize Disk

Extra Optimizations

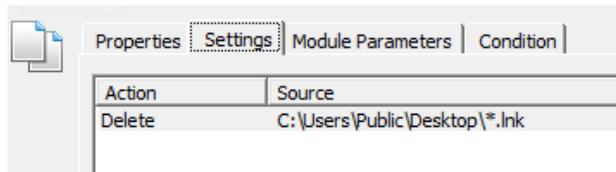
There are some little extra optimizations you can do in your image. Here are a few:

- Set the ignore remote keyboard layout registry. This will fix issues when your client OS has a different languages keyboard installed.



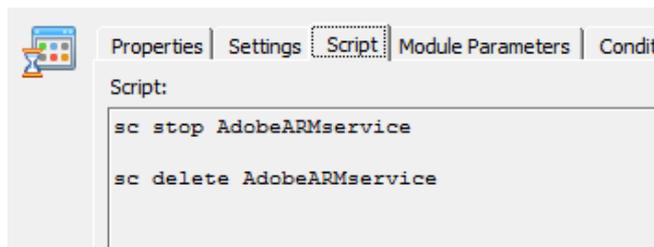
- Empty public desktop, this will delete all the shortcuts programs have made on the default desktop.

Perform File Operations



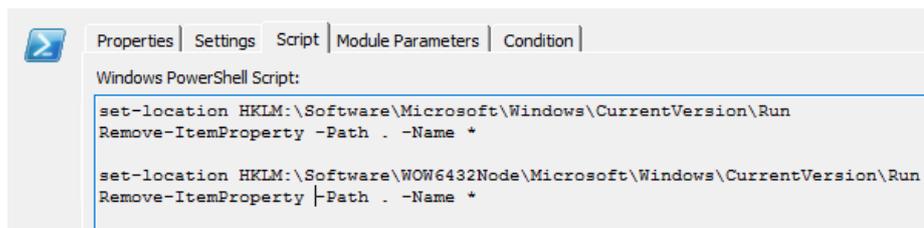
- Delete the adobe updater services. This will make sure that your user won't get update notifications about adobe.

Execute Command



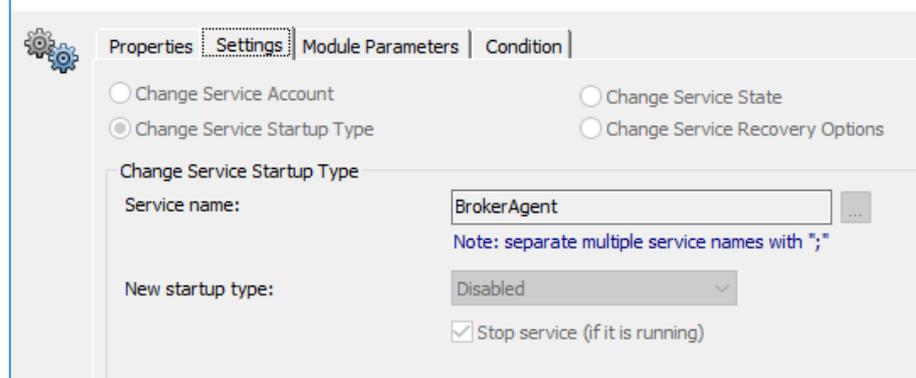
- Empty Run, this will make sure that users login faster and wont start up unnecessary programs. This is a PowerShell script:

Execute Windows PowerShell Script



- (optionally) Turn of the Citrix Broker Agent, this way the target devices that run from this vDisk won't register with the delivery controller automatically and users aren't allowed to sign in. This gives you time to load in App-V or scripts after a reboot.

Manage Service Properties



Delete Snapshot (Optionally)

If you have chosen to create snapshot during the deployment we need to delete them now before we start imaging. Create a new module named Delete Snapshots and add the following tasks to it.

The screenshot shows the 'Edit Project - 03.00.01 Delete Snapshots' window. The 'Modules' tab is active, displaying a table of tasks:

Type	Description
Deletes a snapshots of a virtual ...	
vCenter Virtual Machine	Changes a virtual machine power status.
vCenter Virtual Machine Snapshot	Deletes a snapshot of a virtual machine.
vCenter Virtual Machine Snapshot	Deletes a snapshot of a virtual machine.
vCenter Virtual Machine	Changes a virtual machine power status.

First stop the target VM.

The screenshot shows the 'Power vCenter Virtual Machine' task configuration window. The 'Settings' tab is active, displaying the following configuration:

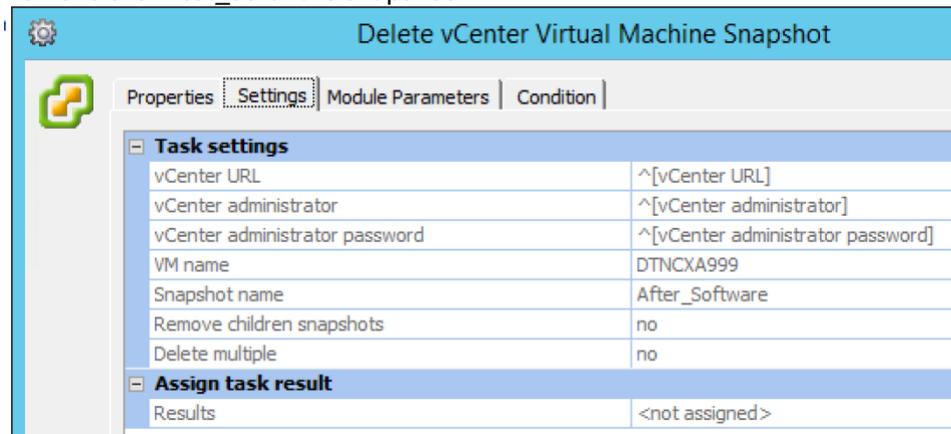
Task settings	
vCenter URL	^[vCenter URL]
vCenter administrator	^[vCenter administrator]
vCenter administrator password	^[vCenter administrator password]
VM Name	DTNCXA999
Force power status	no
VM power status	stop
Assign task result	
Results	<not assigned>

Remove the After_Prep snapshot.

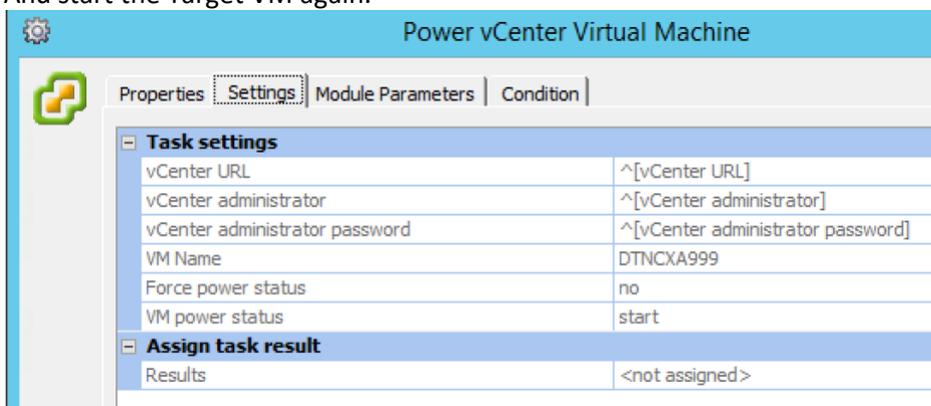
The screenshot shows the 'Delete vCenter Virtual Machine Snapshot' task configuration window. The 'Settings' tab is active, displaying the following configuration:

Task settings	
vCenter URL	^[vCenter URL]
vCenter administrator	^[vCenter administrator]
vCenter administrator password	^[vCenter administrator pas:
VM name	DTNCXA999
Snapshot name	After_Prep
Remove children snapshots	no
Delete multiple	no
Assign task result	
Results	<not assigned>

Remove the After_Software snapshot.



And start the Target VM again.



Add this module to a new Project and name it 03.00.01. Delete Snapshots in the Deployment Run Book. We need to make sure to run this project on the management server with PowerCLI installed.

Optionally Delete the snapshots XenServer

To delete the snapshots from XenServer use the following PowerShell script:

```
$Xenhost = "XenServer"  
$Username = "ROOT"  
$Password = "P@ssw0rd"
```

```
Connect-XenServer -Url https://$Xenhost -UserName $username -Password $password -  
NoWarnCertificates -SetDefaultSession
```

```
$VMname = "DTNCXA006"
```

```
Invoke-XenVM -Name $VMname -XenAction Shutdown
```

```
Start-Sleep -Seconds 30
```

```
$Snapshotname = "After_Software"
```

```
Remove-XenVM -Name $Snapshotname
```

Start-Sleep -Seconds 30

\$Snapshotname = "After_Prepere"

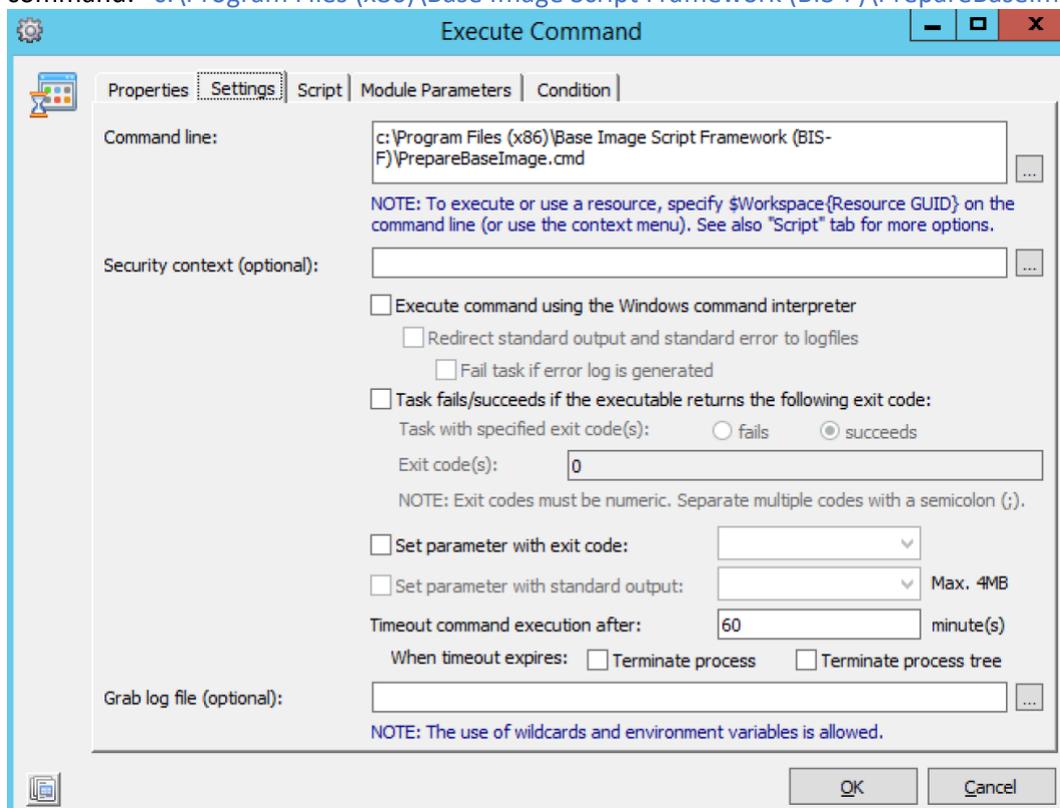
Remove-XenVM -Name \$Snapshotname

Start-Sleep -Seconds 30

Invoke-XenVM -Name \$VMname -XenAction Start

Start Imaging

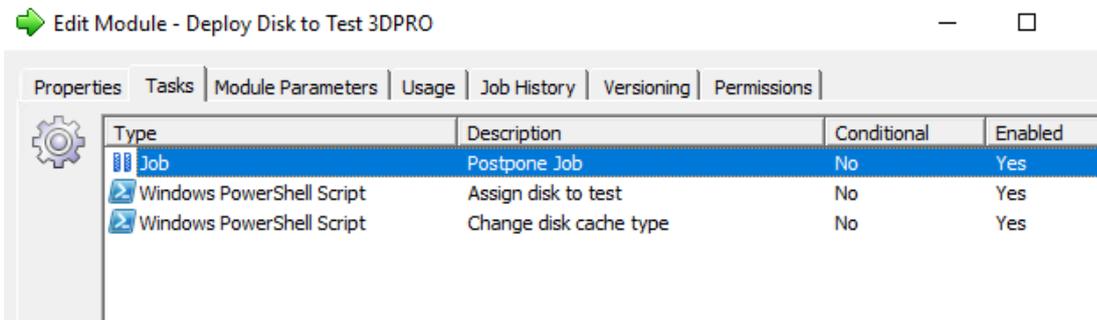
Now it's time to start imaging the disk. This process will copy the System disk of the Golden Image Target VM to the vDisk we created and assigned to the target VM in Part 1. Create a new module and name it Start Imaging. Add an Execute Command task and enter the following command: `"c:\Program Files (x86)\Base Image Script Framework (BIS-F)\PrepareBaseImage.cmd"`



This task can easily take up 30 minutes to complete so we need to set the time out to 60 minutes. This task will start the BIS-F tool and it will optimize the system to become a vDisk and it will start imaging the Disk. You can see the progress of the script through the Windows Explorer if you browse to `\\targetvm\Persistentdisk$\BISFLogs`, for example `\\DTNCXA006\D$\BISFLogs`. Add this module to a new Project and name it 03.00.02 Image Disk

Close Disk, Assign it to Collection Citrix PVS

After the vDisk is created it still is in Private mode and cannot be used with multiple XenApp servers. So we need to change the disk mode to standard. We are going to do this with PowerShell scripts. It is important to first add a Postpone Job to make sure that the TargetVM is shutdown or else the disk mode cannot be changed. I have added a postpone of 2 minutes.



Or use the following PowerShell scripts.

To change the vDisk to standard with Cache in RAM overflow to Harddisk:

```
Import-Module "C:\Program Files\Citrix\Provisioning Services Console\Citrix.PVS.SnapIn.dll"
```

```
$date = Get-Date -format yyyy-MM-dd
$basename = "$[Base Name]"+"$date"
```

```
Set-PvsDisk -Name $basename -StoreName "$[Store Name]" -SiteName "$[Site Name]" -
WriteCacheType "$[Cache Type]" -WriteCacheSize "$[Cache Size]"
```

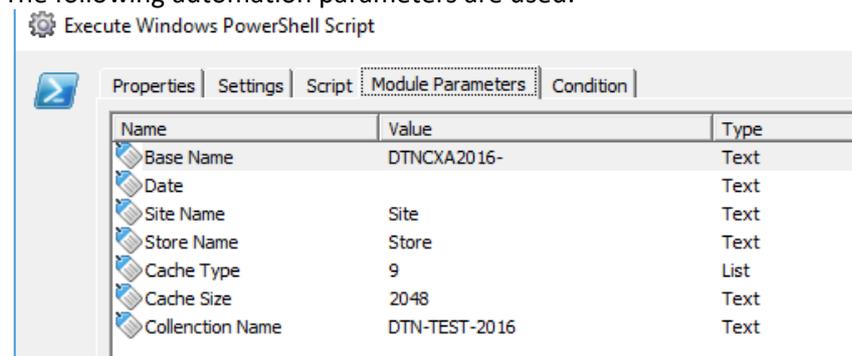
To assign the disk to a Collection for instance TEST-Collection:

```
Import-Module "C:\Program Files\Citrix\Provisioning Services Console\Citrix.PVS.SnapIn.dll"
```

```
$date = Get-Date -format yyyy-MM-dd
$basename = "$[Base Name]"+"$date"
```

```
Add-PvsDiskLocatorToDevice -SiteName "$[Site Name]" -StoreName "$[Store Name]" -
DiskLocatorName $basename -CollectionName "$[Collention Name]" -RemoveExisting
```

The following automation parameters are used:



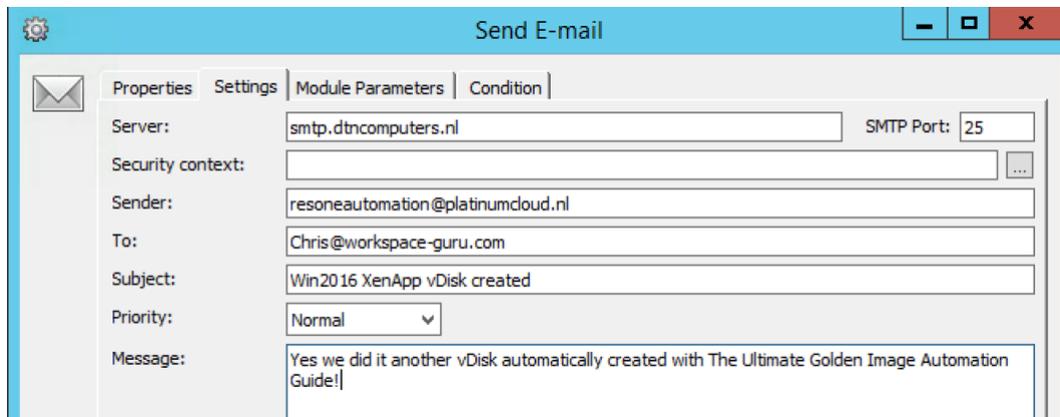
Cache type 9 stands for cache in RAM overflow to disk.

Add this module to a new Project and name it 03.00.03 Close Disk Assign to Collection.

Make sure the vDisk is available on all the PVS servers. [Check out my article on syncing here.](#) The article also includes a PowerShell script which you can use to sync your disks. You can use this in the

Send Finish Email

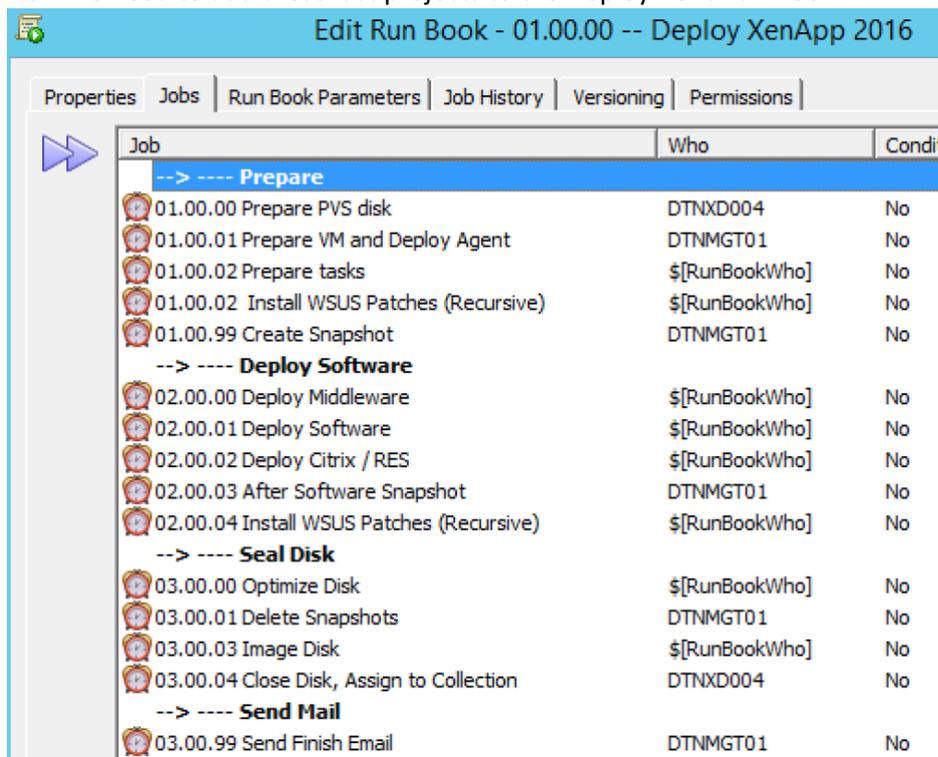
After the vDisk is created it is nice to receive an email especially if you schedule the deployment Run Book to run at night. So the last module we're going to create is a Send Email task. Create a new module and name it Send Email. Add an Email Send task to it, fill in your email info and add a text like: Yes we did it, another vDisk automatically created with The Ultimate Golden Image Automation Guide!



Add this module to a new Project and name it 03.00.99 Send Finish Email.

Finish the Deployment Run Book

Now we need to add these last projects to the Deployment Run Book.



- Add the 03.00.00 Optimize Disk Project and use the \${RunBookWho} parameter as Who.
- Add the 03.00.01 Delete Snapshots Project and select your management server which has PowerCLI installed as Who.
- Add the 03.00.02 Image Disk Project and use the \${RunBookWho} parameter as Who.
- Add the 03.00.03 Close Disk Assign to Collection Project and select your Citrix PVS server as Who.

- Add the 03.00.99 Send Finish Email Project and select any agent which is allowed to send emails as Who.

And that's it, we're done! Now you can deploy your vDisk automatically and there is no need any more to use Citrix PVS versioning. You just enter the changes like a new software application to the correct Projects and run the Run Book again!

If you have disabled the Citrix Broker services remember to create a After Reboot project that load in your last tasks like App-V applications and then starts your Citrix broker services again.

