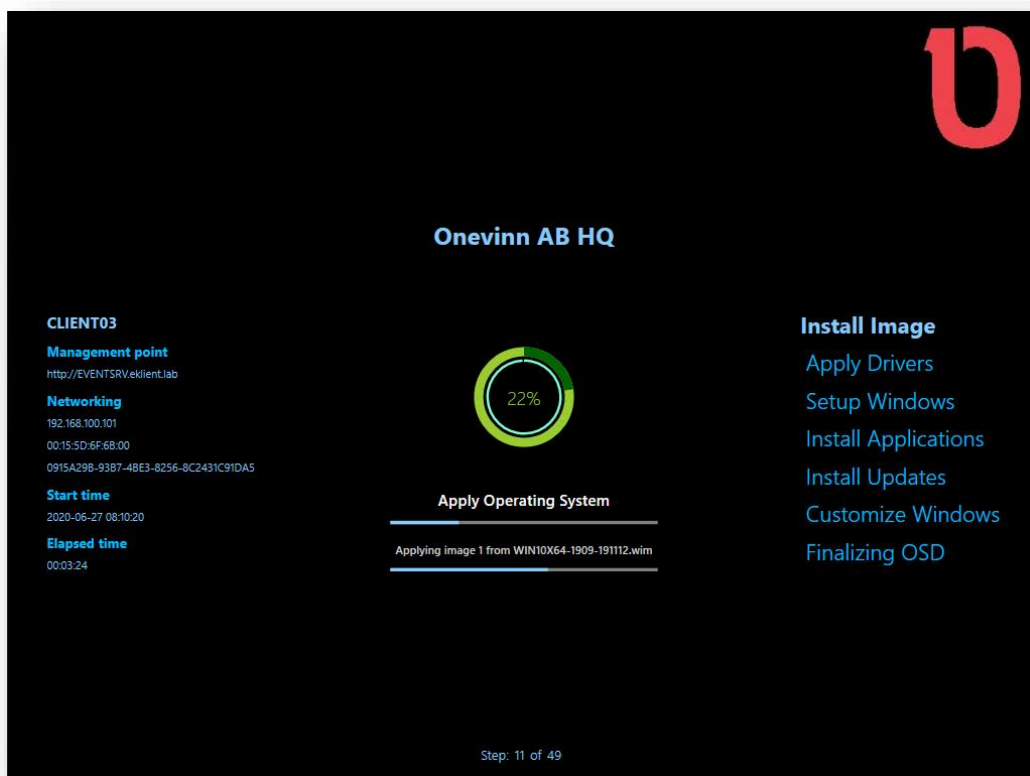


Onevinn AB

TSBackground for Configuration Manager

User's manual



CONTENTS

1.	Version	3
2.	Description and requirements	4
3.	Media	5
3.1.	TSBackground files and folders	6
4.	backgrounds	7
5.	layout	7
6.	Grid rules	9
7.	Design viewer	9
8.	Property binding variables	11
9.	custom variables	11
10.	Show and hide grids	12
11.	TS Steps Example	13
12.	Special Steps	14
12.1.	GatherLocal	14
12.2.	CopyTSBToHDD	14
12.3.	Dialogs	15
13.	Error Handling	16
14.	Boot image extra files	17
15.	Start in PE (winpeshl.ini)	19
16.	Backgrounds	20
17.	TSBackground.exe.config	21
17.1.	Update 2020-11-20	22
17.2.	Progress UI Colors	23
17.2.1.	Available colors:	23
18.	Debug	24
19.	initiate time and connection	25
	Remote control	29
20.	Issues	32
20.1.	Time	32
20.2.	VariablesToHide	33
20.3.	Custom variables in .xaml	33
20.4.	MDT UDI and serviceui.exe	34
20.5.	Dialog steps missing parameter	35
20.6.	deprecated Unattend keys	35

1. VERSION

Doc Version	Author	Date	Remark
1.0	Johan Schrewelius	2019-05-04	Document Created
1.1	Johan Schrewelius	2019-05-13	Updated
1.2	Johan Schrewelius	2019-05-26	Mark II released
1.3	Johan Schrewelius	2019-06-07	Improved start in full OS, simplified clean-up
1.4	Johan Schrewelius	2019-06-20	Bugfixes and F8 capture, auto copy to disk
1.5	Johan Schrewelius	2019-07-18	Added Remote Control
1.6	Johan Schrewelius	2020-07-02	New version, TSBackground MK III
1.7	Johan Schrewelius	2020-08-28	Updated Issues section
1.8	Johan Schrewelius	2021-05-23	New config setting: "WaitSpinnerColor"

2. DESCRIPTION AND REQUIREMENTS

TSBackground replaces BGINfo as well as its predecessor OSDBackground as background (Wallpaper) generator during **OS Deployment with Configuration Manager**. And yes, it works in full OS.

TSBackground.exe is a **DotNet 4.6.2 WPF Application** and requires PowerShell and DotNet being added to Boot Images as “Optional Component”.



TSBackground has built-in debug features intended as replacement for the “F8 Command support” in ConfigMgr, access is password protected.

TSBackground has been designed to allow extensive customization; the entire layout is loaded at runtime and all built-in as well as custom Task sequence variable can be used. An exception from this is the centrally positioned circular progress bar that can only be customized with regards to colors or turned off. Any WPF Xaml file present in the Layout folder will be loaded at application launch.

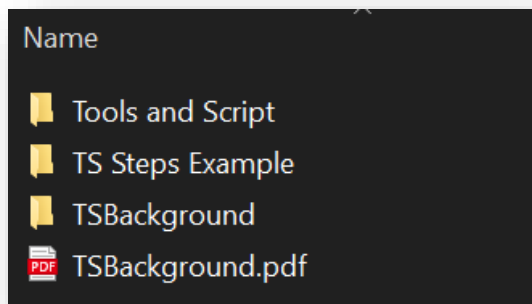
This guide will focus on our recommended usage and configuration. We will include the necessary files in the boot image and perform the initial launch there. No other methods are encouraged.

It is not necessary but preferable to have some knowledge in xaml (wpf) to gain full usage of this utility.

MDT Standalone is not supported.

3. MEDIA

Unpack the downloadable zip archive, this should result in a folder like this:



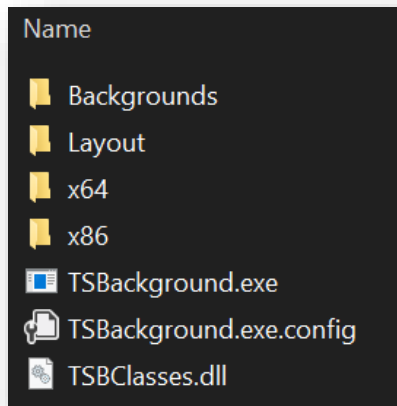
There are three subfolders:

“Tools and Script” contains a couple of utilities and a script to be used during design and deployment, we will get back to them as we need them during this guide.

“TS Steps Example” contains an exported Task Sequence with an assortment of relevant steps, some of them necessary.

“TSBackground” contains the actual application to be invoked in your Task sequence. See next section for details.

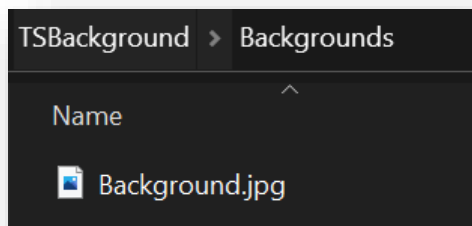
3.1. TSBackground files and folders



Type	Name	Explanation
Folder	Backgrounds	Contains a minimum of one background image (Background.jpg)
Folder	Layout	Contains at least one wpf xaml file describing the layout of the dynamic information presented on screen during OS deployment.
Folder	x64	DLL's missing in Windows PE x64
Folder	x86	DLL's missing in Windows PE x86
Application	TSBackground.exe	The application itself
Library	TSBClasses.dll	Dynamic link library (Classes)
Configuration	TSBackground.exe.config	Basic configuration including encrypted Debug password.

4. BACKGROUNDS

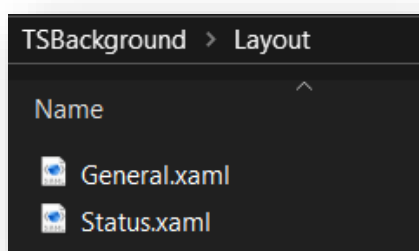
TSBackground comes with one sample background, **make sure to keep the name**. It is possible to dynamically change the background during deployment but “Background.jpg” will always be loaded on first start.



5. LAYOUT

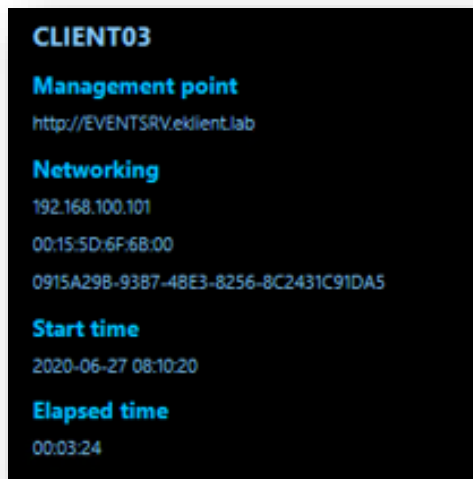
Since OSDBackground was released a couple of years ago there has been a lot of requests for more possibilities to customize the layout. TSBackground should solve all such problems as it loads the layout at runtime, whatever standard wpf xaml code that is placed in the **subfolder “Layout”** will be loaded into the UI at start time. Complex components cannot be used but the possibilities are immense.

Included in the folder are two files, “**General.xaml**” and “**Status.xaml**”:



Please, study the content of these files to gain understanding of the functionality.

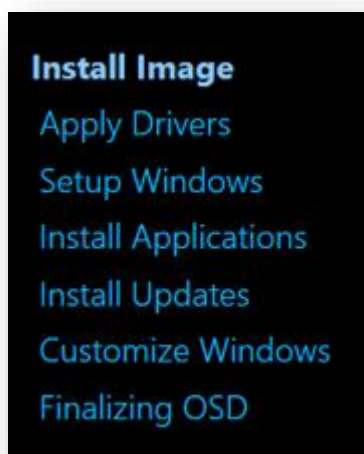
“General.xaml” contains the xaml code for showing system information and elapsed time to the left in the screenshot.



The file contains a **parent Grid named “General”** which **owns several nameless child grids, any named grid can be turned off and on dynamically during deployment**. As opposite a nameless grid will always be visible or follow its parent with regards to visibility.

```
<Grid Name="General"
```

“Status.xaml” holds several status grids, these are numbered Status01 – Status07 and intended to be highlighted one at the time depending on the present status of the deployment, for example “Install Image” or “Finalizing OSD”.



```
<Grid Name="Status01"
```


6. GRID RULES

1. Any top or parent grid must include the standard Namespace name, like:

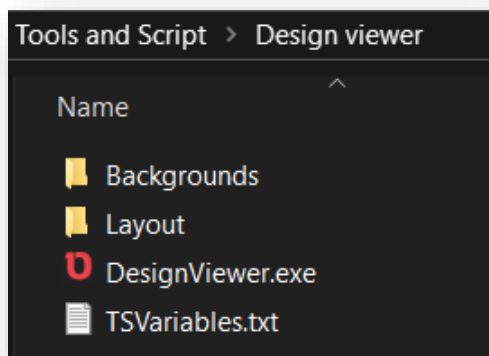
```
"xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
<Grid xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
```

2. Named Grids must have unique names.

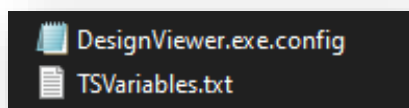
7. DESIGN VIEWER

To help designing your “layout” a viewer is included in the Tools folder:



Put you background and layout (xaml) files in the corresponding folders, when DesignViewer.exe is launched it will load and show the result of these files in the same way as TSBackground will during deployment. You can then show and hide each named Grid, modify your code and reload until the result is satisfactory. Each named Grid will be represented by a checkbox.

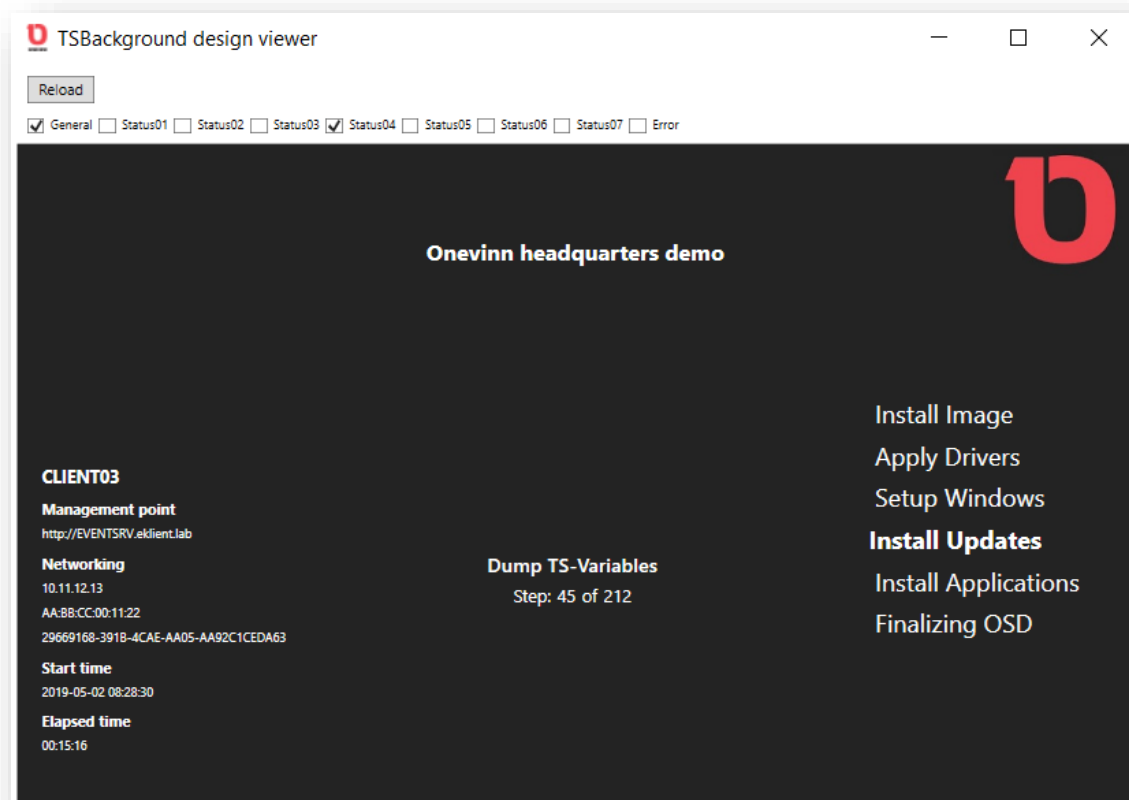
The folder also contains files with dummy data.



Viewer

Use the checkboxes to simulate the different phases of the Task sequence

When done, copy the files to corresponding folders in the TSBackground structure.



8. PROPERTY BINDING VARIABLES

Any Task sequence variable can be bound to for example a TextBox. Example:

Text="{Binding [_smstsmachinename]}"

The variable name is not case sensitive but make sure to surround the variable with brackets. For a complete (almost) list of available variables refer to the official documentation:

<https://docs.microsoft.com/en-us/sccm/osd/understand/task-sequence-variables>

In addition to this any custom variable will be available for binding once it has been created and set to a value.

9. CUSTOM VARIABLES

There are two types of variables, control variables and data variables. Control variables can be set from TS environment, Data variables are intended for usage in .xml files.

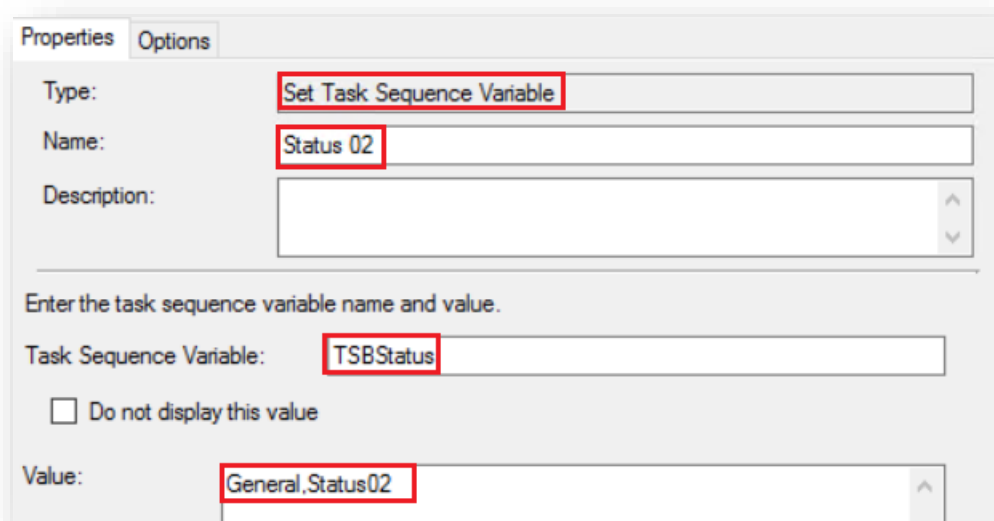
Type	Name	Explanation
Control	TSBStatus	Tells TSBackground which Grids should be visible, e.g. "General,Status01"
Control	TSBBackImage	If more than one image (jpg) is present in the Backgrounds folder, switch background by setting this variable to the filename, e.g. "SecondBackground.jpg"
Data	TSBIpAddress	IPv4 address of machine
Data	TSBMacAddress	Mac address of connected adapter
Data	TSBStartTime	The time the deployment was started
Data	TSBStartTimeUTC	The time the deployment was started converted to UTC, if your deploying cross time zones use this for displaying "Start time"
Data	TSBElapsedTime	Time (HH:mm:ss) since the deployment started
Data	TSBTotalelapsedTime	Same as above but written to TS Environment
Data	TSBCurrentAction	The TS steep number currently executed
Data	TSBFinalAction	The total number of steps in TS (Corrected with number of steps in error group).

10. SHOW AND HIDE GRIDS

The control variable “**TSBStatus**” should throughout the OSD be modified / set to the grids desired for visibility:

“TSBStatus”=“General,Status02”

This will cause TSBackground to show the two Grids, named “General” and “Status02”, all other named grids are hidden.



The screenshot shows the 'Properties' tab of a task configuration window. The 'Type' is set to 'Set Task Sequence Variable'. The 'Name' is 'Status 02'. The 'Description' field is empty. Below the description, there is a section titled 'Enter the task sequence variable name and value.' with a 'Task Sequence Variable' field set to 'TSBStatus'. There is an unchecked checkbox labeled 'Do not display this value'. The 'Value' field is set to 'General,Status02'.

If you use the original layout you must honor this.

End of successful OSD:

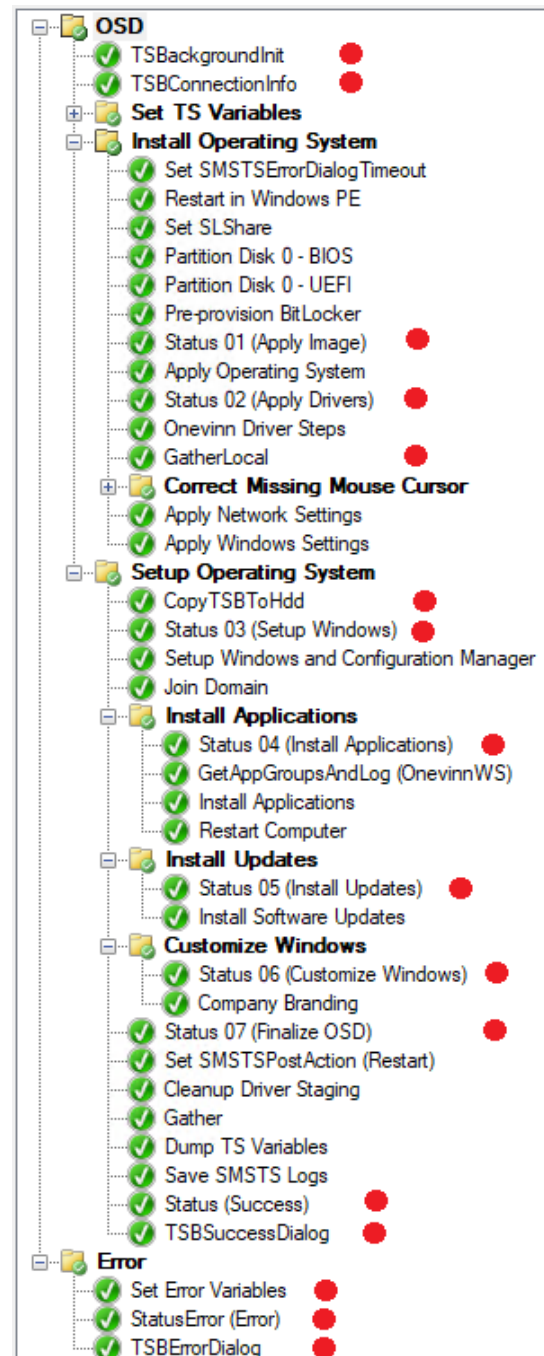
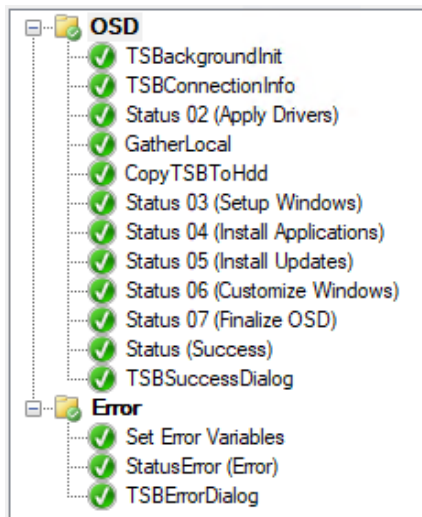
“TSBStatus”=“**Success**”

Set TSBStatus to Error in case of failure:

“TSBStatus”=“**Error**”

11. TS STEPS EXAMPLE

Included in the downloaded zip archive you will find a small Task sequence (Left) with all steps necessary. Import this TS into your environment and copy / paste the various steps to the appropriate location in your own TS. To the right, a working test TS from a lab.



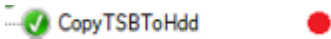
12. SPECIAL STEPS

12.1. GatherLocal



This step can be run multiple times during the deployment and will create and (re)populate the same set of custom TS variables as the “Gather.ps1” script available at the GitHub download page.

12.2. CopyTSBToHDD

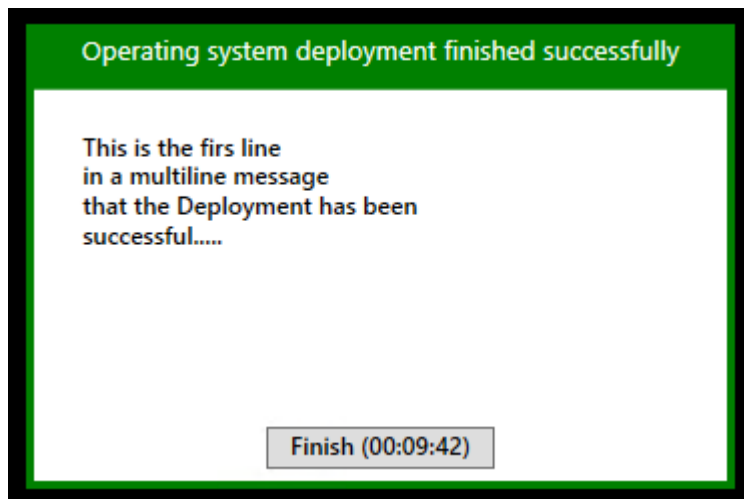


This step **must** be run after the image has been laid down to disk and before the “Setup Windows and ConfigMgr” step.

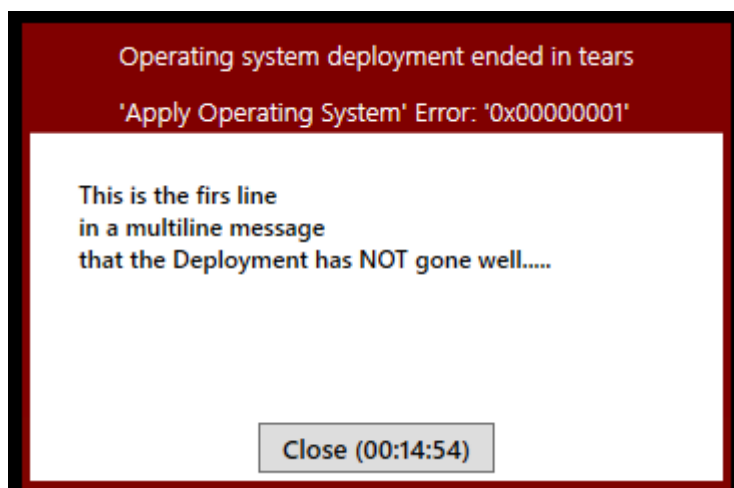
12.3. Dialogs

TSBackground can show two types of dialogs, study and edit the dialog steps (inline scripts) to gain understanding and modify the text as needed.

✓ TSBSuccessDialog ●

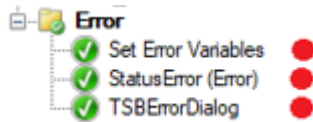


✓ TSBErrorDialog ●



13. ERROR HANDLING

The error section must as a minimum include the two first steps....

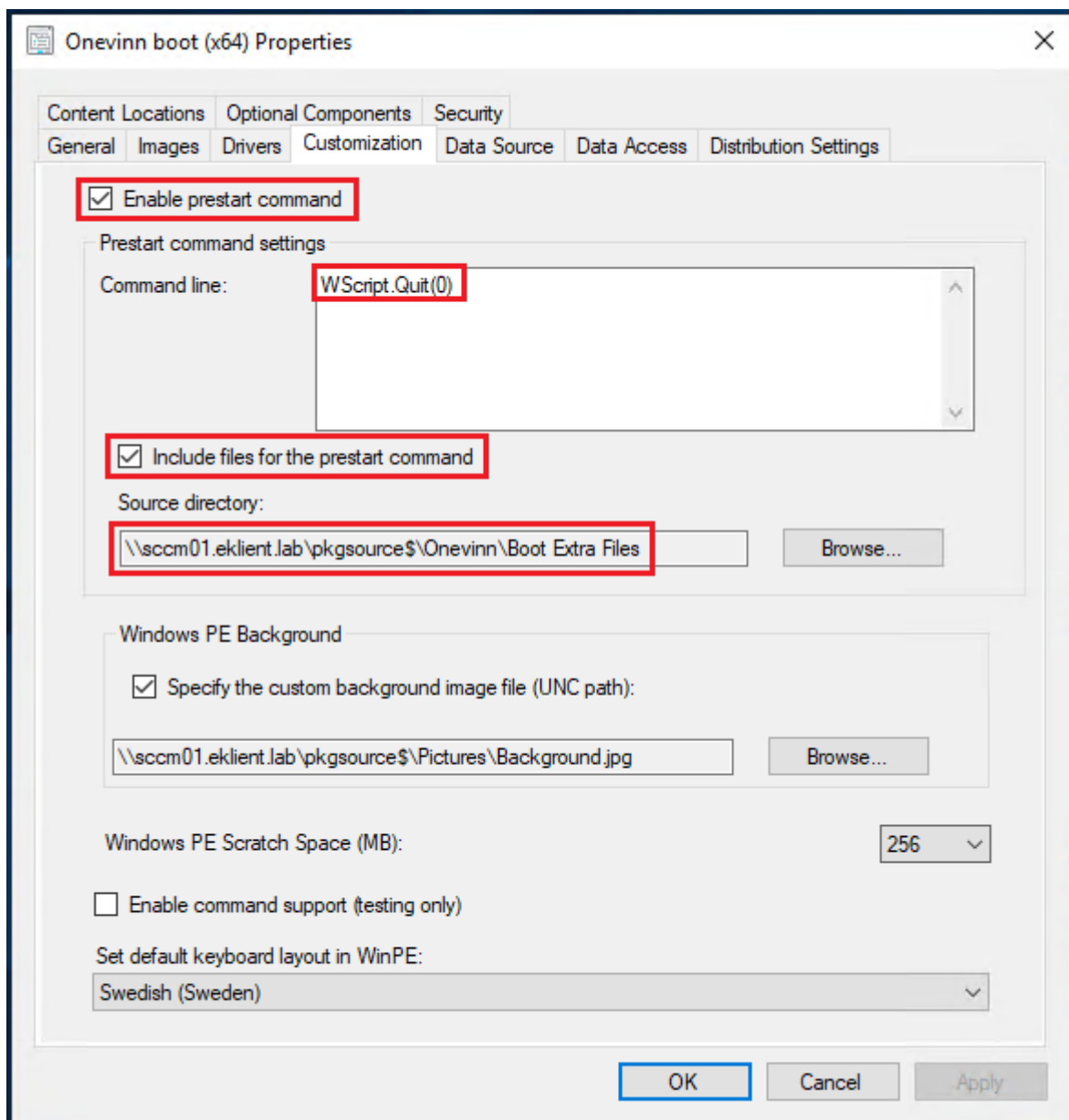


More about error handling here: <http://ccmexec.com/2016/12/error-handling-in-ts-without-mdt-using-osdbackground/>

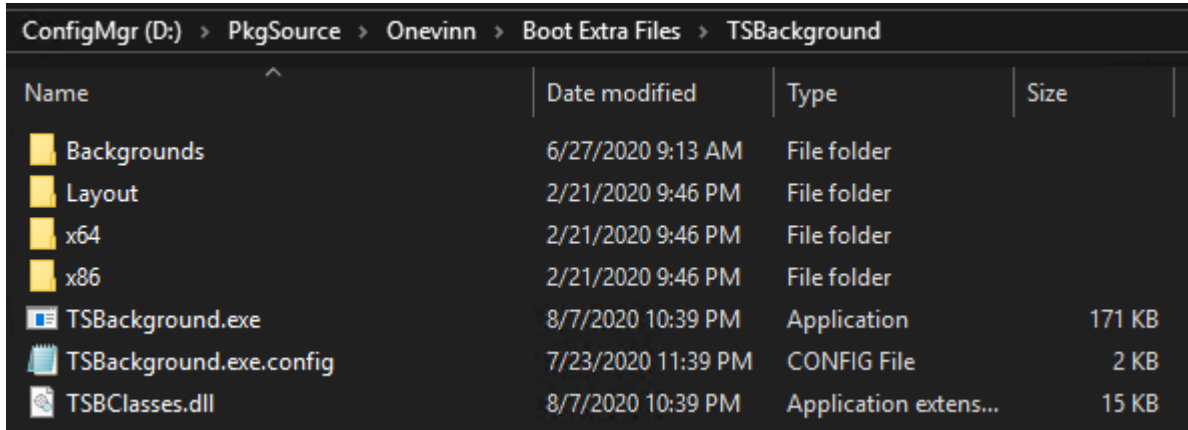
14. BOOT IMAGE EXTRA FILES

If you are not familiar with including extra files in boot images it is highly recommended to read up on it before continuing. We are doing it the easy way by using the built-in functionality on the “Customization” tab of the image properties.

So, simply right click your boot image, go to the Customization tab and fill in the following.



In this example the included folder is named “Boot Extra Files”. In this folder we have, as a sub folder, placed the entire folder TSBackground.



Name	Date modified	Type	Size
Backgrounds	6/27/2020 9:13 AM	File folder	
Layout	2/21/2020 9:46 PM	File folder	
x64	2/21/2020 9:46 PM	File folder	
x86	2/21/2020 9:46 PM	File folder	
TSBackground.exe	8/7/2020 10:39 PM	Application	171 KB
TSBackground.exe.config	7/23/2020 11:39 PM	CONFIG File	2 KB
TSBClasses.dll	8/7/2020 10:39 PM	Application extens...	15 KB

ConfigMgr does not allow adding extra files if not a Prestart command is present. TSBackground does not need it, so we put in a dummy command:

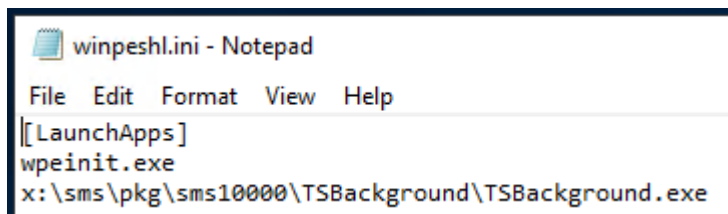
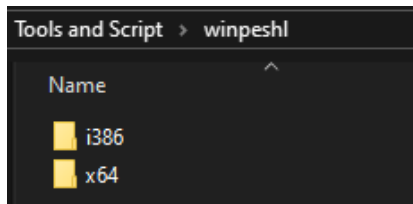
WScript.Quit(0)



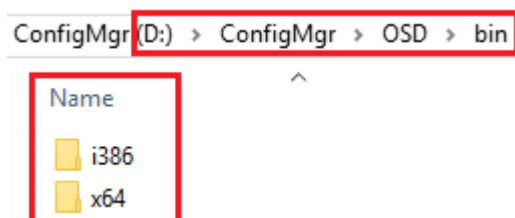
This command does not do anything, it just returns “Zero”, allowing the deployment to run.

15. START IN PE (WINPESHL.INI)

The only recommended way to start TSBackground.exe during Windows PE is to use OSDInjection, this means adding a file called “winpeshl.ini” to your WinPE boot image. The process is rather simple and **prepared INI files are included under “Tools and script”**.



Locate your Configuration manager installation folder and browse to the subfolder containing the osd-binaries. In my lab ConfigMgr is installed in folder “ConfigMgr” on the D-drive:



Now **copy the corresponding “winpeshl.ini”** from “Tools and script” into the x64- and i386-folders. Next time you upgrade your boot images the files will be included under “X:\Windows\System32” and executed prior to anything else (very much like an old Dos autoexec.bat), and launch TSBackground.exe.

Note:

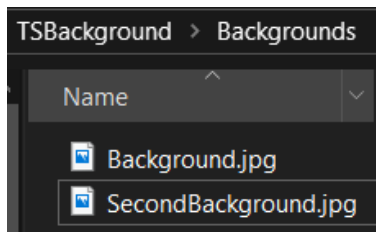
As long as the files are present in the folders they will be included in all boot images when you run “Update Distribution Points”, so make sure to remove the files once your “TSBackground-images” are built, and put them back if you need to rebuild.

With this approach you do not have to think about restarting TSBackground after a restart in PE, it is automatically taken care of. “winpeshl.ini” is executed on every reboot, as opposite of prestart commands that only executes on the initial boot.

A positive side effect of using this alternative is that restarts in PE are blocked while in Debug mode.

16. BACKGROUNDS

If you wish to change background during OSD, just set “TSBBackImage”=“SecondBackground.jpg”

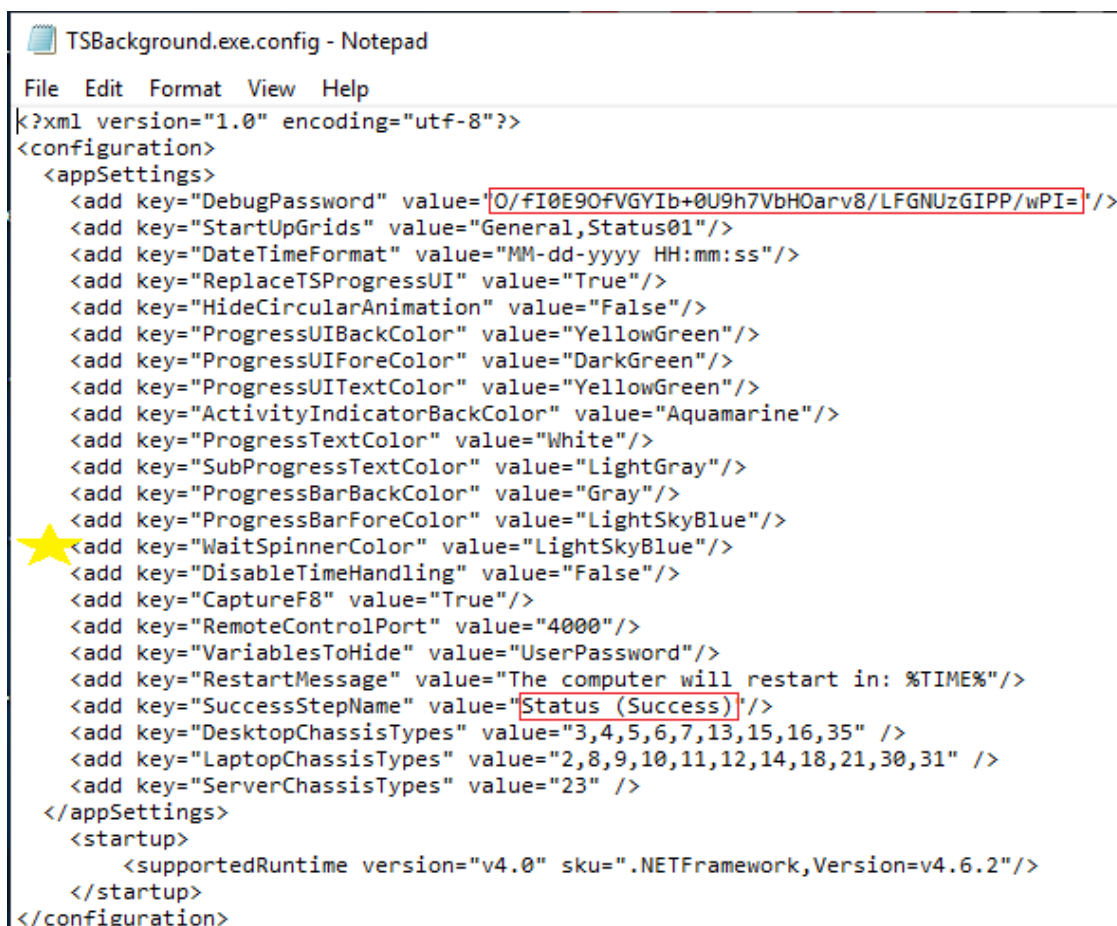
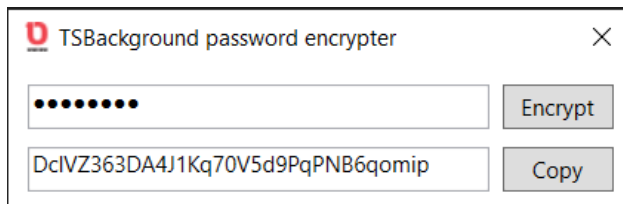


Of course, this should be done by a Set TS Variable step in your Task sequence.

17. TSBACKGROUND.EXE.CONFIG

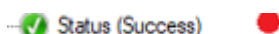
Open “TSBackground.exe.config” and make appropriate changes to reflect your environment.

To be able to use the Debug features of TSBackground you need to create an encrypted password. Look up the “Password Encrypter” in the “Tools and Script” folder, start it and create your password:



“WaitSpinnerColor”: new in version 1.0.21143.01, TSB now starts much earlier and has to wait for the TS, in the meantime we show a “ProgressRing”, use this key to set the color.

Make sure the “SuccessStepName” is correct, we need it to accurately determine when the TS has succeeded.



Additional config keys:

Name	Explanation
StartUpGrids	Tells TSBackground which Grids should be visible at start up before set by TSStatus, e.g. "General,Status01"
DateTimeFormat	In what format time should be displayed, e.g. "G" or "yyyy-MM-dd HH:mm:ss"
ShowSpinnerAnimation	True to show the center screen animation, false to hide.
ReplaceTSProgressUI	Tells TSBackground to replace the build-in TSProgressUI.
DisableTimeHandling	TSBackground automatically handle system time during the deployment, if this functionality must be turned off set this key to false.
CaptureF8	Whether the F8-key should be captured and used to active debug mode.
VariablesToHide	If you have sensitive data (passwords) in TS variables, you can hide them from the Debugger. Use a semi-colon separated list. See Issues
RemoteControlPort	Port for remote connection (TSBackground Remote control). Leave blank to disable the functionality.
RestartMessage	Message shown during countdown to restart during TS.
3x ChassisTypes	Definitions used to set IsDesktop, IsLaptop, IsServer variables.

17.1. Update 2020-11-20

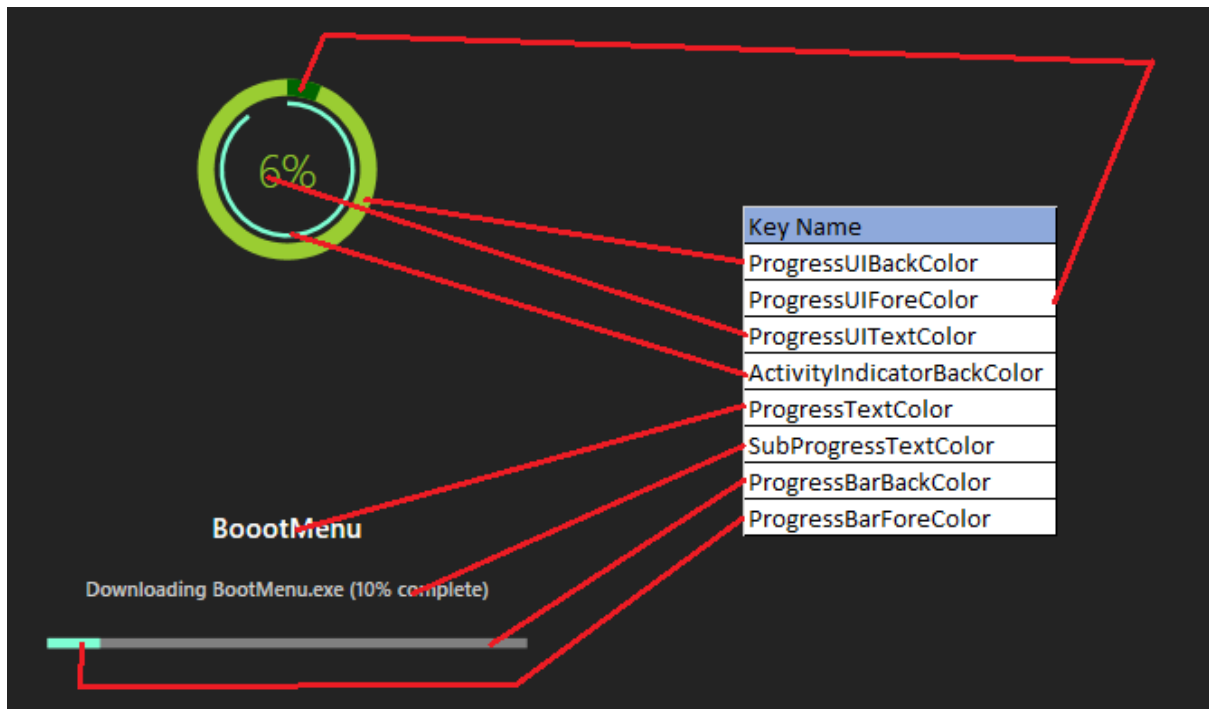
It is now possible to use Debug with AD authentication as alternative to encrypted password.

```
<add key="DebugPassword" value=""/>
<add key="UserGroupDN" value="CN=TSBackground Debug Users,OU=User Groups,OU=eKlient,DC=eklient,DC=lab"/>
<add key="LogonDomain" value="EKLIENT"/>
<add key="StartUpGrids" value="General,Status01"/>
<add key="DateTimeFormat" value="yyyy-MM-dd HH:mm:ss"/>

<add key="UserGroupDN" value="CN=TSBackground Debug Users,OU=User Groups,OU=eKlient,DC=eklient,DC=lab"/>
<add key="LogonDomain" value="EKLIENT"/>
```

Just leave the DebugPassword blank and put in the full DN of the AD group containing allowed users.

17.2. Progress UI Colors

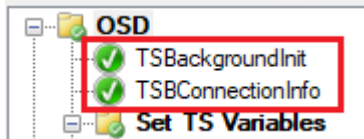


17.2.1. Available colors:

<https://docs.microsoft.com/en-us/dotnet/api/system.windows.media.colors?view=netframework-4.8>

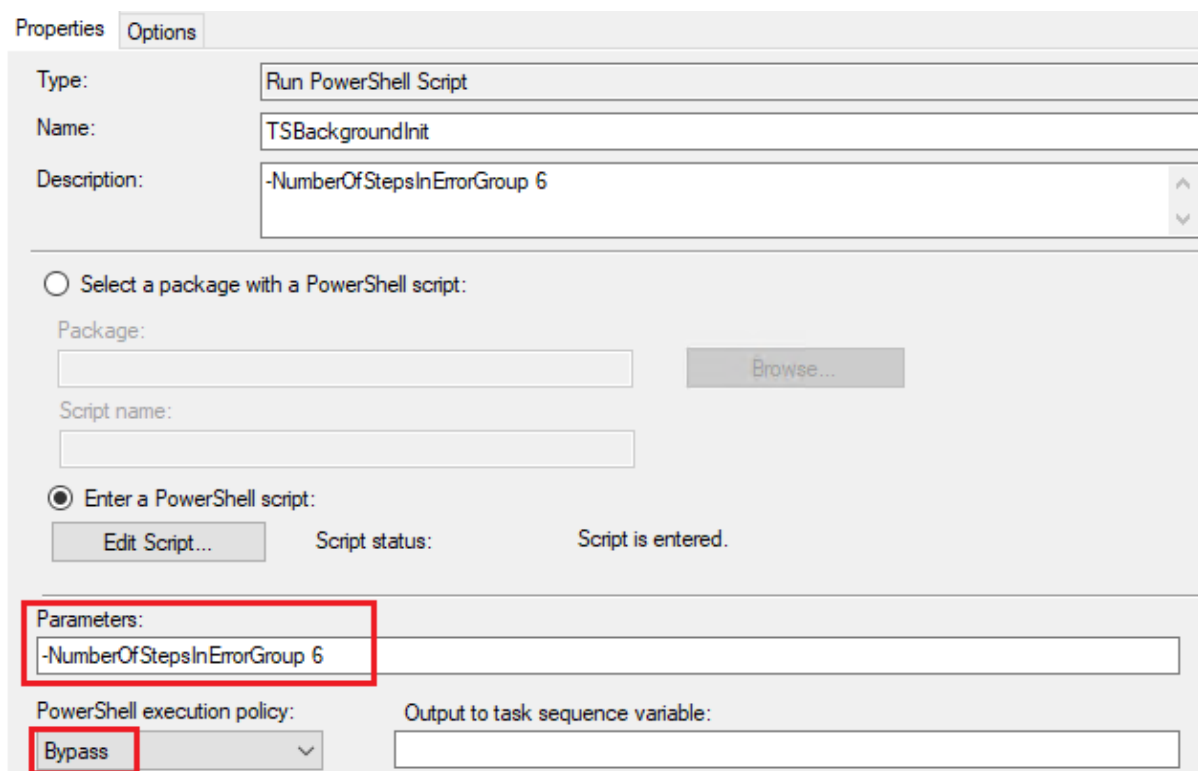
19. INITIATE TIME AND CONNECTION

Do not miss this!



The first step will, apart from other things, save the TS Start time if the OSD is started from full OS. The second step will “post” connection information (for remote control) on the status message queue. Without these steps you will not gain full usage of the application.

If you have an error group in your TS the “percentage done” calculation will never reach 100 % unless we correct the number of steps in TS by reducing it with the length of the error group.



Properties Options

Type: Run PowerShell Script

Name: TSBackgroundInit

Description: -NumberOfStepsInErrorGroup 6

☐ Select a package with a PowerShell script:

Package: Browse...

Script name:

☒ Enter a PowerShell script:

Edit Script... Script status: Script is entered.

Parameters: -NumberOfStepsInErrorGroup 6

PowerShell execution policy: Bypass

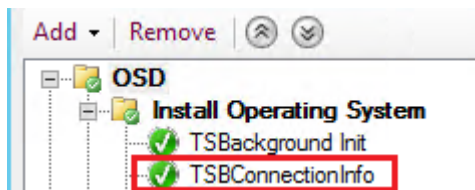
Output to task sequence variable:

Parameters:

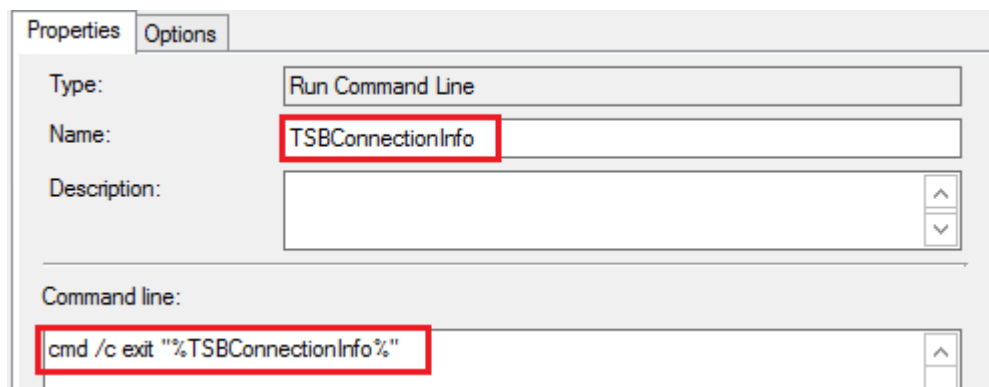
-NumberOfStepsInErrorGroup 6 (Optionally **-DateTimeFormat “G”**)

TSBConnectionInfo

This step is very important, without it we will not be able to present a list of running deployments in the Remote connection viewer (TSBackground Remote control).



The step is an ordinary Run command step like:



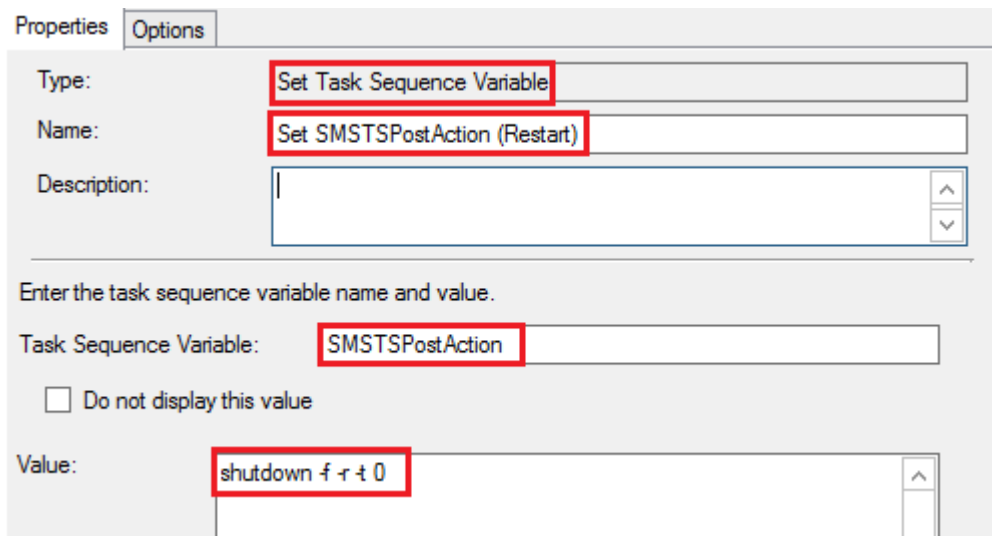
```
cmd /c exit "%TSBConnectionInfo%"
```

Make sure to use the exact name “TSBConnectionInfo”.

Clean up

To remove TSBackground at the end of a successful deployment a post action command is **required**, add a **SMSTSPostAction** command:

shutdown -f -r -t 0



Properties Options

Type: Set Task Sequence Variable

Name: Set SMSTSPostAction (Restart)

Description:

Enter the task sequence variable name and value.

Task Sequence Variable: SMSTSPostAction

☐ Do not display this value

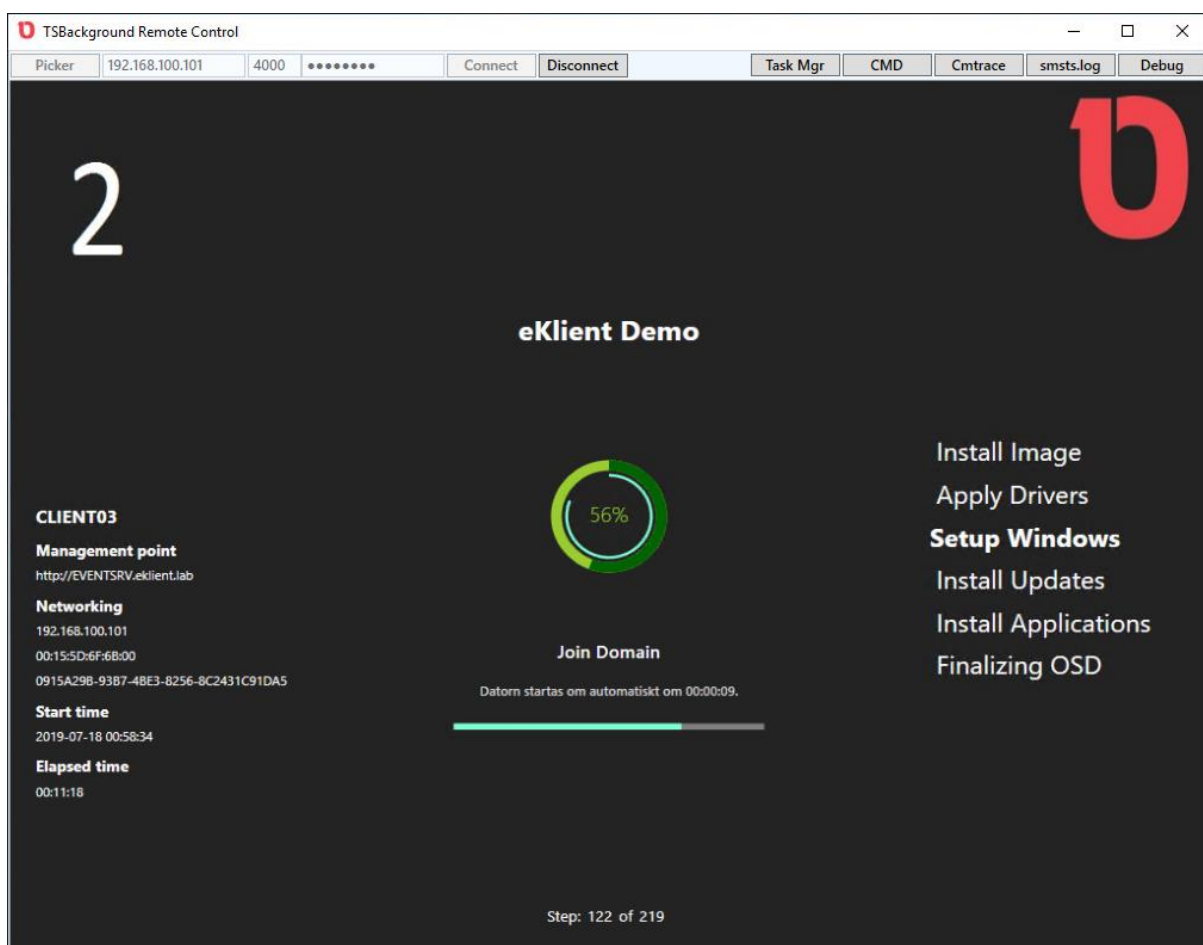
Value: shutdown -f -r -t 0

This step should be placed close to the end of the TS before your error section. TSBackground needed a lot more “help” in a previous version, the command has now been simplified as more functionality have been built in.



REMOTE CONTROL

TSBackground supports remote control. The functionality is suitable for debugging, not so much monitoring. The resolution is limited, and it does consume quite some resources (CPU and RAM) when a connection is active. It is intended as replacement for DaRT with the addition that it works not only during the PE phase of the deployment but also during the full OS phase, after “Setup Windows and ConfigMgr”.

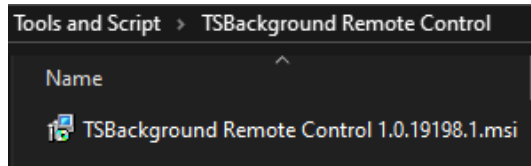


Settings and buttons are available at the top.

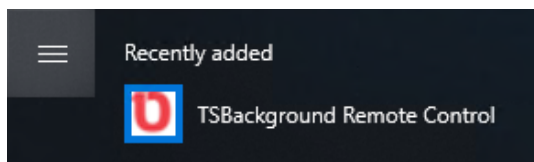
Connection related information, such as client name or IP, and remote port number must be filled in. To the right there are quick launch buttons for commonly used tools.

Installation

“TSBackground Remote Control” is installed as a single msi which can be found under the Tools and Script folder.

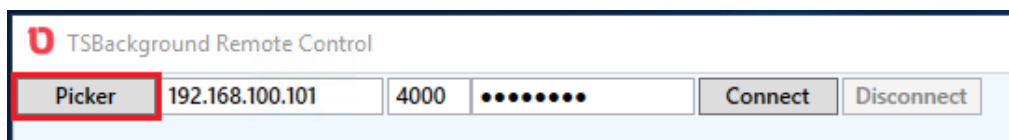


The installer adds a shortcut to the Start menu.

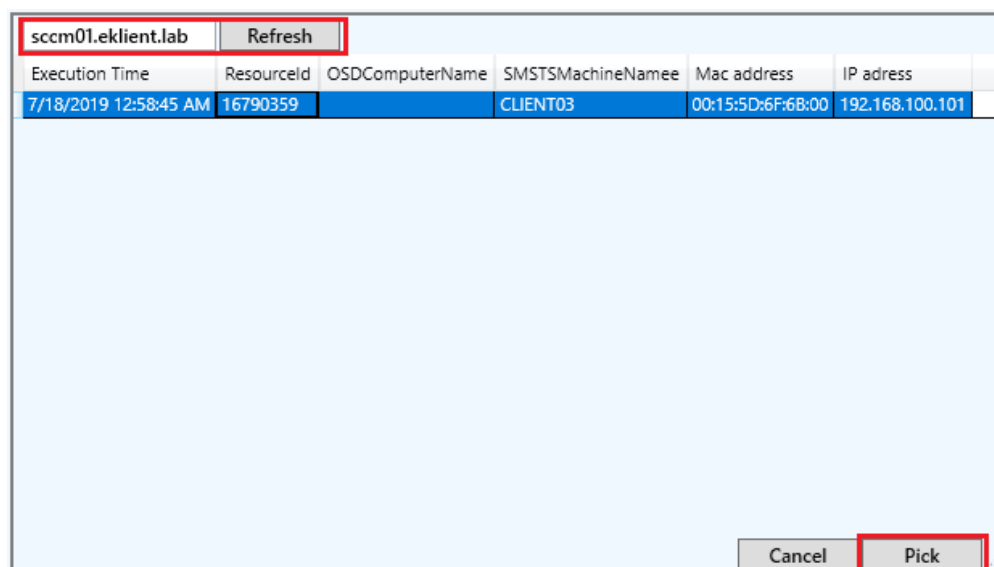


Connection

The application accepts IP address or client name. Default port is 4000, change according to settings in the TSBackground.exe.config described above. You also need to put in the debug password before connecting.







Approximately 30-90 seconds after the TS has started the machine should be available in the “Picker”. Here you need to fill in the name of your Site server.



Integration

Simple integration is possible. Copy the installed files to you preferred location and launch the remote tool with argument /Client:<ComputerName>

C:\Program Files (x86)\Onevinn\TSBackground Remote Control	
Name	Date modified
 AdminUI.WqlQueryEngine.dll	4/10/2019 2:39 PM
 Microsoft.ConfigurationManagement.M...	3/21/2019 1:29 PM
 TSBClasses.dll	7/18/2019 9:19 AM
 TSBClient	7/18/2019 9:19 AM

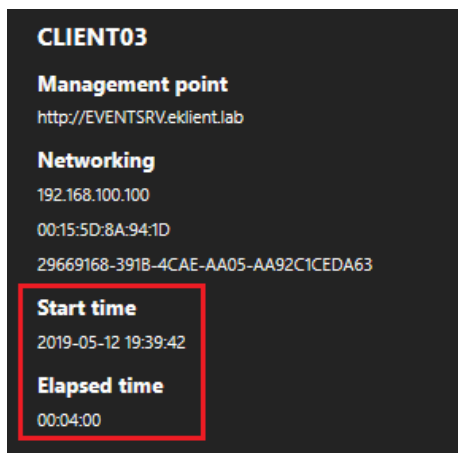
Example:

\\NetworkPath\TSBClient.exe /Client:<ComputerName>

20. ISSUES

20.1. Time

Time is always an issue and a ConfigMgr Task sequence is no exception. The shipped setup shows Start time and elapsed time; keeping track of these is not trivial. Depending on how the computer is booted different time and time zone apply. It is beyond possible to guarantee a correct result in every possible scenario and eventually you might not be able to show a correct elapsed time.



It seems that it uses different variables with different deviations depending on how the WinPE/OSD was launched. The table below shows how time is set in WinPE depending on how the TS was launched and which action TSBackground will take to correct the time.

Boot type	Variable	Timezone	Time	Action
PXE	_SMSTSMPTTime	OK	OK	Hands off
PXE + Restart	_SMSTSMPTTime	OK	PST	Adjust time
Valid Media	_SMTSTimezone	OK	OK	Hands Off
Expired Media	_SMSTSMPTTime	OK	PST	Adjust time
Media + Restart	_SMSTSMPTTime	PST	PST	Adjust timezon and time
Client / Full OS	_SMTSTimezone	OK	PST	Adjust time
Client + Restart	_SMTSTimezone	PST	PST	Adjust timezon and time

It has been suggested to “just use UTC”, that will only work in certain scenarios and is inadequate in others. Once the time and time zone have been corrected it’s possible to convert to UTC to bridge the gap between MP time in PE and local time in full OS, should there be one. TSBackground attempts that maneuver.

To get a correct result it is essential that TSBackground is the only utility, script or other means that manipulate time and time zone during the TS. If you were previously using for example OSDBackground make sure to remove any SetTime-script that was part of that implementation.

TSBackground has been tested on:

Configuration manager current branch **1810, 1902, 1909, 2004 and 2006** in all combinations with **ADK 1809, 1903 and 2004**.

If you have secured the above and time is still not handled correctly you might have to hide or remove that part of the general information, this is done in the "General.xaml" file. After all, aesthetics is important, but security is even more important, and the main purpose is to get rid of the F8-command support.

20.2. VariablesToHide

The configuration file key "**VariablesToHide**" cannot be empty. Leaving it empty will cause all variables to show "Hidden value" in the debugger. If you do not have any variables that needs hiding you can put in a dummy, like:

```
<add key="VariablesToHide" value="UserPassword"/>
```

20.3. Custom variables in .xaml

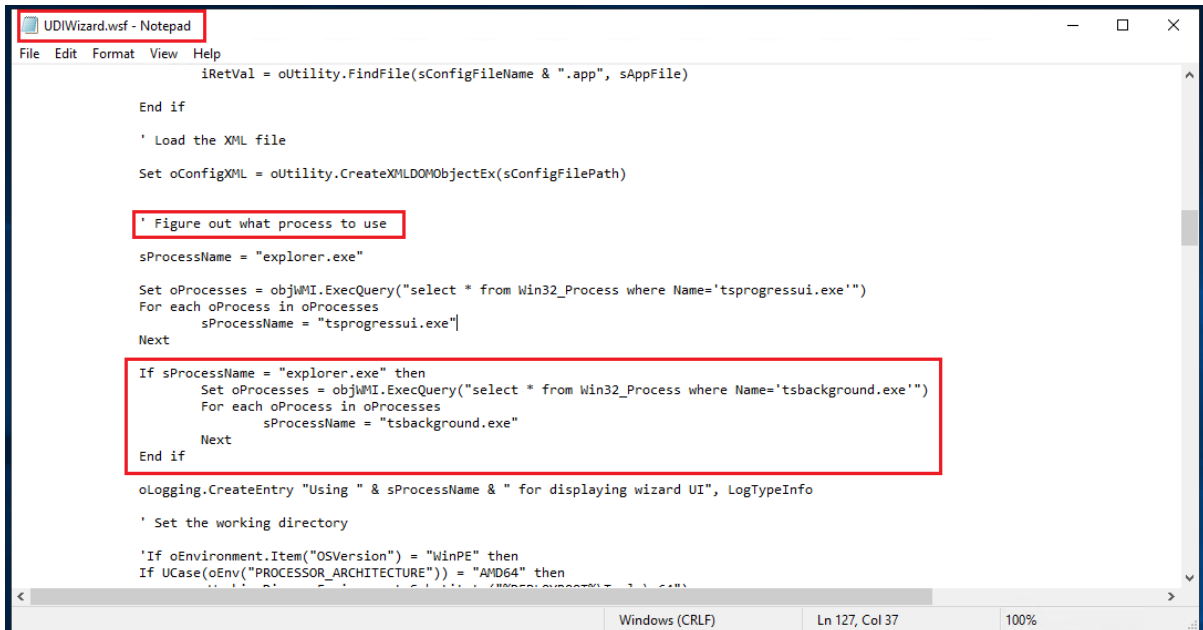
These variables are case sensitive. If you for example use OSDCOMPUTERNAME in you setup you must use the same casing in .xaml

```
Text="{Binding [OSDCOMPUTERNAME]}"
```

20.4. MDT UDI and serviceui.exe

By default, these utilities attach to the TSProgressUI.exe process to gain access to the desktop. Since TSBackground replaces TSProgressUI you might have to do some changes.

For UDI its necessary to edit the “UDIWizard.wsf” located in MDT Toolkit.



```

iRetVal = oUtility.FindFile(sConfigFileName & ".app", sAppFile)

End if

' Load the XML file
Set oConfigXML = oUtility.CreateXMLDOMObjectEx(sConfigFilePath)

' Figure out what process to use
sProcessName = "explorer.exe"

Set oProcesses = objWMI.ExecQuery("select * from Win32_Process where Name='tsprogressui.exe'")
For each oProcess in oProcesses
    sProcessName = "tsprogressui.exe"
Next

If sProcessName = "explorer.exe" then
    Set oProcesses = objWMI.ExecQuery("select * from Win32_Process where Name='tsbackground.exe'")
    For each oProcess in oProcesses
        sProcessName = "tsbackground.exe"
    Next
End if

oLogging.CreateEntry "Using " & sProcessName & " for displaying wizard UI", LogTypeInfo

' Set the working directory

'If oEnvironment.Item("OSVersion") = "WinPE" then
If UCase(oEnv("PROCESSOR_ARCHITECTURE")) = "AMD64" then

```

```

If sProcessName = "explorer.exe" then
    Set oProcesses = objWMI.ExecQuery("select * from Win32_Process where Name='tsbackground.exe'")
    For each oProcess in oProcesses
        sProcessName = "tsbackground.exe"
    Next
End if

```

serviceui.exe

Change the argument **Process:tsprogressui.exe** to **Process:tsbackground.exe**

20.5. Dialog steps missing parameter

A parameter was missing, add \$CountDown as indicated on the pictures:

```
$Caption = "Operating system deployment ended in tears"

$Text = @"
This is the first line
in a multiline message
that the Deployment has NOT gone well.....
"@

$ButtonText = "Close"

# 1 = Success, 1 = Error
$DialogType = 1
$CountDown = 900

$tsUI = New-Object -COMObject Microsoft.SMS.TsProgressUI
$tsUI.TSBDialo($Text, $Caption, $ButtonText, $DialogType, $CountDown)
```

This applies to both TS dialog steps.

20.6. deprecated Unattend keys

Usage of deprecated Unattend keys, such as **"SkipMachineOOBE"** can cause problems in Full OS – i.e., the application won't start after "Setup Windows and ConfigMgr".