

Dism-Temporary-Directory.md

 github.com/nasbench/Misc-Research/blob/main/Other/Dism-Temporary-Directory.md

DismHost Temporary Directory

You'll notice that sometimes when DISM or a related PowerShell Cmdlet that uses the DISM module such as `Get-WindowsOptionalFeature` is run. A child DismHost process will be spawned with from a temporary directory like this

- `C:\Users\<user>\AppData\Local\Temp\<GUID>\DismHost.exe`
- `C:\Windows\Temp\<GUID>\DismHost.exe`

This occurs if DISM selects to run in a Sandbox.

Note

We know that sandbox mode is enabled for `offline` use. I still don't know what's the condition to trigger it for `online` mode. Feel free to add the information if you have it For further investigation inspect the functions `CreateImageSessionFromLocation` and `CreateImageSession` which are the ones responsible for creating the temporary directory and selecting the sandbox mode usage. They are defined in `dismcore.dll`

If we investigated the code We can see that DISM checks a variable passed on by the `CreateImageSession` function to determine if a sandbox needs to be created.

A sandbox in this context means using a temporary directory for staging purposes and so forth. If this variable is false we can see calls to `CreateGuidFolderEx` and `CopyDirectoryEx2` that are responsible for creating the temp directory and copying the some files.

```

if ( a3 )
{
    v134 = ((__int64 (__fastcall *) (void **))ATL::g_strmgr[3])(&ATL::g_strmgr) + 24;
    ATL::CStringT<unsigned short,ATL::StrTraitATL<unsigned short,ATL::ChTraitsCRT<unsigned
short>>>::Format(
        &v134,
        L"No Sandbox was created, DISM running in-place.");
    ...
    ...
    ...
}
else
{
    appended = IsProcessElevated(&v131);
    if ( appended < 0 )
    {
        ...
        ...
        ...
    }
    ...
    ...
    appended = CGuidFolders::CreateGuidFolderEx(this + 51, a5, (unsigned int)v131);
    if ( appended < 0 )
    {
        ...
        ...
        ...
    }
    ...
    ...
    ...
    if ( !(unsigned int)CopyDirectoryEx2(v159) )
    {
        v68 = GetLastError();
        ...
        ...
        ...
    }
    ...
    ...
    ...
}

```



According to Microsoft's documentation the directory called **ScratchDir** is used when extracting files for temporary use during servicing. And is the one used by default as the sandbox location.

/ScratchDir:<path_to_scratchdirectory>

Specifies a temporary directory that will be used when extracting files for temporary use during servicing. The directory must exist locally. If not specified, the \Windows%Temp% directory will be used, with a subdirectory name of randomly generated hexadecimal value for each run of DISM. Items in the scratch directory are deleted after each operation.

You should not use a network share location as a scratch directory to expand a package (.cab or .msu file) for installation. The directory used for extracting files for temporary usage during servicing should be a local directory.

Example:

```
Dism /image:C:\test\offline  
/ScratchDir:C:\Scratch /Add-Package  
/PackagePath:C:\packages\package.cab
```

To test this we can enforce this mode and test that this behavior is actually influenced by the sandbox. We do so by using the following command with the undocumented **/sandbox** flag

```
dism /online /get-packages /ScratchDir:[path-to-scratch-dir] /sandbox
```



You can monitor the **C:\Windows\Logs\DISM\dism.log** file for information about the mode used.

Sandbox Mode Not Used Example

```
2024-01-04 20:52:13, Info    DISM    DISM Manager: PID=5704 TID=26812 Copying DISM from
"C:\Windows\System32\Dism" - CDISMManager::CreateImageSessionFromLocation
2024-01-04 20:52:13, Info    DISM    DISM Manager: PID=5704 TID=26812 No Sandbox was created, DISM
running in-place. - CDISMManager::CreateImageSessionFromLocation
2024-01-04 20:52:14, Info    DISM    DISM Manager: PID=5704 TID=26812 Successfully loaded the
ImageSession at "C:\Windows\System32\Dism" - CDISMManager::LoadRemoteImageSession
```



Sandbox Mode Example

```
2024-01-04 20:49:12, Info    DISM    DISM Manager: PID=30216 TID=47688 Copying DISM from
"C:\Windows\System32\Dism" - CDISMManager::CreateImageSessionFromLocation
2024-01-04 20:49:15, Info    DISM    DISM Manager: PID=30216 TID=47688 Successfully loaded the
ImageSession at "C:\Users\xxx\AppData\Local\Temp\B473CFA4-799B-4FCE-B5DF-28C333DB7769" -
CDISMManager::LoadRemoteImageSession
...
2024-01-04 20:49:16, Info    DISM    DISM Manager: PID=30216 TID=47688 Image session successfully
loaded from location: C:\Users\xxx\AppData\Local\Temp\B473CFA4-799B-4FCE-B5DF-28C333DB7769 -
CDISMManager::CreateImageSession
```



Appendix

Undocumented Flag

The following flags are defined but no documented in both DISM Command-Line Options and DISM Global Options for Command-Line Syntax

Flag	Description
/external	Unknown
/provider	Unknown
/sandbox	Enables Sandbox Mode
/showtags	Unknown

PowerShell Cmdlets

The following PowerShell Cmdlet will use the Dism library and might trigger this behavior.

- Add-AppProvisionedPackage
- Add-AppProvisionedSharedPackageContainer
- Add-AppxProvisionedPackage
- Add-ProvisionedAppPackage
- Add-ProvisionedAppSharedPackageContainer
- Add-ProvisionedAppxPackage
- Add-WindowsCapability
- Add-WindowsDriver
- Add-WindowsImage
- Add-WindowsPackage
- Apply-WindowsUnattend
- Clear-WindowsCorruptMountPoint
- Disable-WindowsOptionalFeature
- Dismount-WindowsImage
- Enable-WindowsOptionalFeature
- Expand-WindowsCustomDataImage
- Expand-WindowsImage
- Export-WindowsCapabilitySource
- Export-WindowsDriver
- Export-WindowsImage
- Get-AppProvisionedPackage
- Get-AppProvisionedSharedPackageContainer
- Get-AppxProvisionedPackage
- Get-NonRemovableAppsPolicy
- Get-ProvisionedAppPackage
- Get-ProvisionedAppSharedPackageContainer
- Get-ProvisionedAppxPackage
- Get-WIMBootEntry
- Get-WindowsCapability
- Get-WindowsDriver
- Get-WindowsEdition
- Get-WindowsImage
- Get-WindowsImageConten
- Get-WindowsOptionalFeature
- Get-WindowsPackage
- Get-WindowsReservedStorageState
- Mount-WindowsImage
- New-WindowsCustomImage
- New-WindowsImage
- Optimize-AppProvisionedPackages
- Optimize-AppxProvisionedPackages
- Optimize-ProvisionedAppPackages
- Optimize-ProvisionedAppxPackages
- Optimize-WindowsImage
- Remove-AppProvisionedPackage
- Remove-AppProvisionedSharedPackageContainer
- Remove-AppxProvisionedPackage
- Remove-ProvisionedAppPackage
- Remove-ProvisionedAppSharedPackageContaine
- Remove-ProvisionedAppxPackage
- Remove-WindowsCapability

- Remove-WindowsDriver
- Remove-WindowsImage
- Remove-WindowsPackage
- Repair-WindowsImage
- Save-SoftwareInventory
- Save-WindowsImage
- Set-AppPackageProvisionedDataFile
- Set-AppXProvisionedDataFile
- Set-NonRemovableAppsPolicy
- Set-ProvisionedAppPackageDataFile
- Set-ProvisionedAppXDataFile
- Set-WindowsEdition
- Set-WindowsProductKey
- Set-WindowsReservedStorageState
- Split-WindowsImage
- Start-OSUninstall
- Update-WIMBootEntr
- Use-WindowsUnattend

