

DEPLOY DELL CLIENT TOOLS WITH MICROSOFT INTUNE

Step by Step installing Dell Client Management Tools

Summary

This document is for informational purposes only and may contain typographical errors and technical inaccuracies. The content is provided as is, without express or implied warranties of any kind

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Dell Technologies – CSG Service Product Group

Author: Sven Riebe

| | |
|---------|-------------|
| Author | Sven Riebe |
| Version | Draft 1.2.5 |
| Editor | |

Contents

| | |
|--|----|
| Release notes | 4 |
| Summary | 5 |
| Deployment Best Practices..... | 5 |
| Dell Trusted Device Agent | 6 |
| Introduction..... | 6 |
| Prepare Dell Trusted Device Agent for Intune..... | 6 |
| Scripts for Install, Uninstall and Detection..... | 7 |
| Import and Deployment settings Dell Trusted Device Agent for Intune..... | 14 |
| Dell Power Manager | 22 |
| Introduction..... | 22 |
| Prepare Dell Power Manager for Intune | 22 |
| Scripts for Install, Uninstall and Detection..... | 23 |
| Import and Deployment settings Dell Power Manager for Intune | 30 |
| Dell Command Update | 38 |
| Introduction..... | 38 |
| Prepare Dell Command Update for Intune..... | 38 |
| Scripts for Install, Uninstall and Detection..... | 39 |
| Import and Deployment settings Dell Command Update for Intune..... | 45 |
| Dell Command Monitor..... | 53 |
| Introduction..... | 53 |
| Prepare Dell Command Monitor for Intune | 53 |
| Scripts for Install, Uninstall and Detection..... | 54 |
| Import and Deployment settings Dell Command Monitor for Intune | 60 |
| Dell Command Configure..... | 68 |
| Introduction..... | 68 |
| Prepare Dell Command Configure for Intune | 68 |
| Scripts for Install, Uninstall and Detection..... | 69 |
| Import and Deployment settings Dell Command Configure for Intune | 75 |

| | |
|---|-----|
| Dell Display Manager..... | 83 |
| Introduction..... | 83 |
| Prepare Dell Display Manager for Intune | 83 |
| Download Dell Display Manager V1.x..... | 84 |
| Download Dell Display Manager V2.x..... | 84 |
| Scripts for Install, Uninstall and Detection..... | 85 |
| Script set for Dell Display Manager V1.x..... | 86 |
| Script set for Dell Display Manager V2.x..... | 91 |
| Import and Deployment settings Dell Display Manager for Intune | 97 |
| Dell SupportAssist for Business PCs | 106 |
| Introduction..... | 106 |
| Prepare Dell SupportAssist for Business PCs for Intune | 107 |
| Folder for SupportAssist for Business PC | 116 |
| Folder for SupportAssist Dependency Application..... | 116 |
| Scripts for Install, Uninstall and Detection..... | 116 |
| Import and Deployment settings Dell SupportAssist for Business PCs for Intune | 124 |
| Part Dependency Application | 124 |
| Part SupportAssist for Business Application..... | 132 |
| Dell Optimizer | 142 |
| Introduction..... | 142 |
| Prepare Dell Optimizer for Intune | 142 |
| Download Dell Optimizer..... | 142 |
| Download Microsoft .net Runtime 6.x | 144 |
| Scripts for Install, Uninstall and Detection..... | 144 |
| Script for Dell Optimizer | 145 |
| Script for Microsoft .net Runtime..... | 155 |
| Import and Deployment settings Dell Optimizer for Intune | 157 |
| Part for Microsoft .net Runtime (Dependency) | 157 |
| Part for Dell Optimizer | 166 |
| Dell Peripheral Manager | 175 |
| Introduction..... | 175 |
| Prepare Dell Peripheral Manager for Intune | 175 |
| Scripts for Install, Uninstall and Detection..... | 176 |

| | |
|---|-----|
| Import and Deployment settings Dell Peripheral Manager for Intune | 182 |
| References..... | 190 |

Release notes

| Version | Changes |
|---------|---|
| 1.0.0 | Initial document |
| 1.0.1 | Change process of MSI extract of Dell Command Update and Dell Power Manager |
| 1.1.0 | <ul style="list-style-type: none">- Change from MSI to DUP (Dell Update Package) if possible.- New scripts for all Install/Uninstall/Detection- Dell SupportAssist for Business now with Dependency App |
| 1.1.1 | <ul style="list-style-type: none">- Add Download Information for Dell Display Manager 2.x and Support Matrix |
| 1.2.0 | <ul style="list-style-type: none">- Rebrand to Microsoft Intune- Update Install instruction Dell Optimizer, add .net 6.x dependency- Update Install instruction Dell Display Manager, add section for DDM 2.x- Correction Dell Command Update Install/Uninstall script |
| 1.2.1 | <ul style="list-style-type: none">- Correct Display Manager 2 Install/Uninstall to run with PowerShell 32 Bit |
| 1.2.2 | <ul style="list-style-type: none">- Update Install instruction Dell Support Assist, add .net 6.x dependency |
| 1.2.3 | <ul style="list-style-type: none">- Add Dell Peripheral Manager to this document |
| 1.2.4 | <ul style="list-style-type: none">- Update Dell Trusted Device section from MSI to DUP deployment- Add MS Event logging for install and uninstall scripts |
| 1.2.5 | <ul style="list-style-type: none">- Update on section Dell Support Assist for Business PC on prepare MSI and MST and update for the Install Script |

Summary

This document provides a step-by-step guide to deployment of the Dell Client Tools with Microsoft Intune (Intune) with Intune only option. Installation instructions are also supplied for the deployment of Dell tools with solutions like Microsoft SCCM. This document is designed for modern management systems without co-management.

Recommendation:

This guide will assume that you are using a clean Dell Ready Image. In case you are using the standard preinstallation of Windows 10, please uninstall all existing Dell tools. This will avoid any conflicts between the standard versions that are preinstalled and their corresponding Business version (e.g., Dell SupportAssist vs. Dell SupportAssist for Business PCs) you want to install on your device by following this documentation. Alternatively, you can use a clean Windows 10 image as well but remember that the idea of modern provisioning is not to spend time on creating an OS Image.

Notice: For your security all installations scripts include a check if older versions are installed and start a uninstall first of these Applications.



Deployment Best Practices

In this guide we follow the best practices for modern management with Microsoft Intune / Microsoft Intune. We also take into consideration that most of our customers will leverage modern deployment options like Windows Autopilot for pre-provisioned deployment (also known as Autopilot white glove). That's why we choose to document all examples based on Microsoft Intune Win32 app (.intunewin format) and not by using Line-of-business app (.msi, .appx, .appxbundle, .msix, and .msixbundle formats). Win32 app provides the best experience with its management capabilities and is also preferred formats for all pre-provisioning services.

Dell Trusted Device Agent

Introduction

The Dell Trusted Device agent is part of the Dell SafeBIOS product portfolio. The Trusted Device agent supports these security and identity functions:

- BIOS Verification
- BIOS Events & Indicators of Attack
- Image Capture
- Intel ME Verification
- Security Risk Protection Score
- Dell Event Repository and SIEM integration

Reference: Dell Trusted Device Installation and Administrator Guide

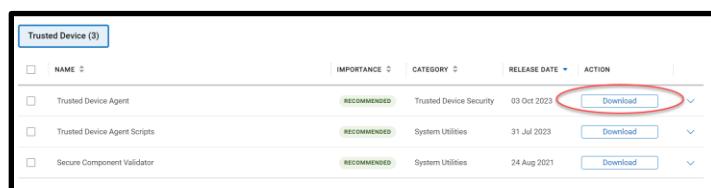
<https://www.dell.com/support/home/en-us/product-support/product/trusted-device/docs>

Prepare Dell Trusted Device Agent for Intune

Download of the newest Version of Dell Trusted Device Agent from the Dell website.

<https://www.dell.com/support/home/en-us/product-support/product/trusted-device/drivers>

Click Download for the Application Trusted Device Agent



If you have downloaded the file from <https://www.dell.com/support/home/en-us/product-support/product/trusted-device/drivers>, copy file to your software repository for the next step.



Scripts for Install, Uninstall and Detection

It is possible to use the native file for installation. In this document we are using PowerShell scripts to cover different scenarios of Install new, Update, and uninstall for a later automation of uploading applications by API.

All script could be download on Github Repository:

<https://github.com/svenriebedell/Dell-Tools-Intune-Install>

Install Script

The script makes a precheck if older versions are installed and starting an uninstall first to have a clean environment.

```
#####
##### Function section #####
#####

function Get-installedcheck
{
    param
    (
        [Parameter(mandatory=$true)][string] $AppSearchString
    )

    $AppCheck = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearchString'"

    If ($null -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        return $false
    }
}

#####
##### variable section #####
#####

$InstallerName = Get-ChildItem *.exe | Select-Object -ExpandProperty Name
$ProgramPath = "." + $InstallerName
[Version]$ProgramVersion_target = (Get-Command $ProgramPath).FileVersionInfo.ProductVersion
$AppSearch = "%Dell%Trusted%Device%" #Parameter to search in registry
$UninstallApp = Get-CimInstance -ClassName Win32_Product | Where-Object {$_.Name -like "Dell*Trusted*Device*"}
$SoftwareName = "Dell Trusted Device Agent"
[Version]$ProgramVersion_current = $UninstallApp.Version

#####
##### program section #####
#####

#####
##### generate Logging Resources #####
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

#####
##### Checking if older Version is installed and uninstall this Version #####
#####

If ($null -ne $ProgramVersion_current)
{
    if ($ProgramVersion_target -gt $ProgramVersion_current)
    {
        #####
        # uninstall Software old #
        #####
        $Argument = "/x " + $UninstallApp.PackageCache + " /qn REBOOT=ReallySuppress"
        Start-Process -FilePath msieexec.exe -ArgumentList "$Argument" -Wait
```

```

#####
# uninstall success check      #
#####
$UninstallResult = Get-installedcheck -AppSearchString $AppSearch

If ($UninstallResult -eq $true)
{
    Write-Host "uninstall is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_current).ToString()
        Uninstall = $false
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message
$UninstallData

}

Else
{
    Write-Host "uninstall is successful" -BackgroundColor Green

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_current).ToString()
        Uninstall = $true
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message
$UninstallData

}

}

Else
{
    Write-Host "same version is installed"

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $false
        Reason = "same version is installed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData

}

}

Exit 0
}

#####
#Install new Software          #
#####

Start-Process -FilePath "$ProgramPath" -ArgumentList "/s" -Wait

#####
# install success check      #
#####

```

```

#####
$UninstallResult = Get-installedcheck -AppSearchString $AppSearch

If ($UninstallResult -ne $true)
{
    Write-Host "install is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $false
        Reason = "Installation failed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Error -EventId 11 -Message $UninstallData
}

Else
{
    [Version]$ProgramVersion_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearch'" | Select-Object -ExpandProperty Version

    If ($ProgramVersion_current -ge $ProgramVersion_target)
    {
        Write-Host "install is successful" -BackgroundColor Green

        $UninstallData = [PSCustomObject]@{
            Software = $SoftwareName
            Version = ($ProgramVersion_target).ToString()
            Install = $true
            Reason = "Update/Install/Newer Version"
        } | ConvertTo-Json

        Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message
$UninstallData
    }

    Else
    {
        Write-Host "install is unsuccessful" -BackgroundColor red

        $UninstallData = [PSCustomObject]@{
            Software = $SoftwareName
            Version = ($ProgramVersion_target).ToString()
            Install = $false
            Reason = "Older Version installed $ProgramVersion_current"
        } | ConvertTo-Json

        Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message
$UninstallData
    }
}
}

```

Uninstall Script

The script starts the uninstalling of application. The Dell Trusted Device requests an immediate reboot. This Reboot will suppress by parameter REBOOT=R

```
#####
##### Function section #####
#####

function Get-installedcheck
{
    param
    (
        [Parameter(mandatory=$true)][string] $AppSearchString
    )

    $AppCheck = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearchString'"

    If ($null -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        return $false
    }
}

#####
##### variable section #####
#####

$AppSearch = "%Dell%Trusted%Device%" #Parameter to search in registry
$Program_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearch'"
$ArgumentString = '/i "'+$ProgramPath + '" /qn REBOOT=ReallySuppress'
$SoftwareName = $Program_current.Name
$ApplicationID_current = $Program_current.IdentifyingNumber

#####
##### program section #####
#####

#### generate Logging Resources
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

#####
##### #uninstall Software #####
#####

# check if Software is installed

if ($null -eq $ApplicationID_current)
{
    Write-Host "no software is installed"
    exit 0
}
else
{
    Start-Process -FilePath msiexec.exe -ArgumentList "/x $ApplicationID_current /qn REBOOT=R" -Wait

#####
# uninstall success check #
#####

$UninstallResult = Get-installedcheck -AppSearchString $AppSearch

If ($UninstallResult -eq $true)
{
    Write-Host "uninstall is unsuccessful" -BackgroundColor Red
}
```

```

$UninstallData = [PSCustomObject]@{
    Software = $SoftwareName
    Version = $Program_current.Version
    Uninstall = $false
} | ConvertTo-Json

Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData

}

Else
{

    Write-Host "uninstall is successful" -BackgroundColor Green

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = $Program_current.Version
        Uninstall = $true
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
}
}
}

```

Detection Script

The detection script showing success of installation. It could be done as well without a script but again it is helpful for later automation of upload processes in the future.

The **yellow marked** version must be adjusted with each new version.

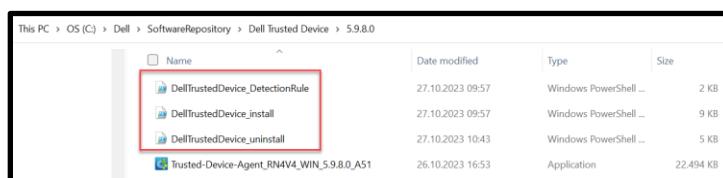
```

#####
# Program with target Version
#####
$ProgramVersion_target = [5.9.8.0] # need to be the same like the msi file
$ProgramVersion_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '%Trusted Device%'" | Select-Object -ExpandProperty Version

if($ProgramVersion_current -ge $ProgramVersion_target)
{
    Write-Host "Found it!"
}

```

Please, copy these files in the same folder as your EXE Installer File.

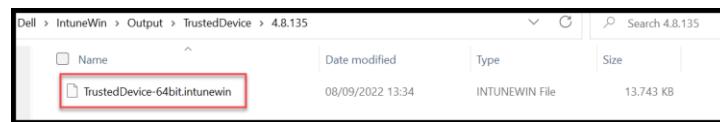


Start the Microsoft Win32 Content Prep Tool (aka IntuneAppUtil.exe). In case you are not familiar with this tool, you will find documentation here: <https://docs.microsoft.com/en-us/mem/intune/apps/apps-win32-prepare>

| Argument | Value |
|---------------|---|
| Source Folder | folder where you have stored the unzipped MSI |
| Setup file | Main installer file like msi/exe/ps1, etc. |
| Output Folder | where you want to store the IntuneWin |

```
Please specify the source folder: C:\Dell\SoftwareRepository\Dell Trusted Device\5.9.8.0
Please specify the setup file: Trusted-Device-Agent_RN4V4_WIN_5.9.8.0_A51.exe
Please specify the output folder: C:\Dell\IntuneWin\Output\TrustedDevice\5.9.8.0
Do you want to specify catalog folder (Y/N)?n
```

IntuneWin is now prepared and ready for installation by Intune.



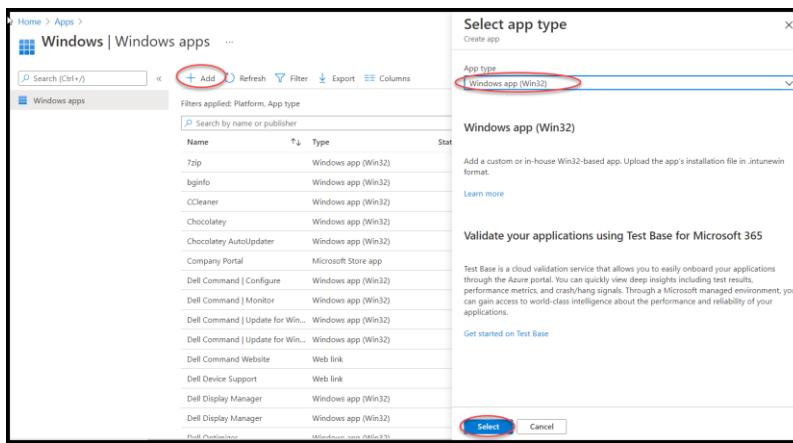
Import and Deployment settings Dell Trusted Device Agent for Intune

Click 'Add'

| Field | Value |
|---------------|---------------------|
| Source Folder | Windows app (Win32) |

Click 'Select'

Select Windows app (Win32) as application.



Section 'App information'

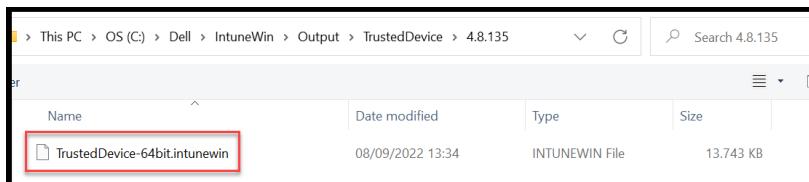


Click 'Select app package file'

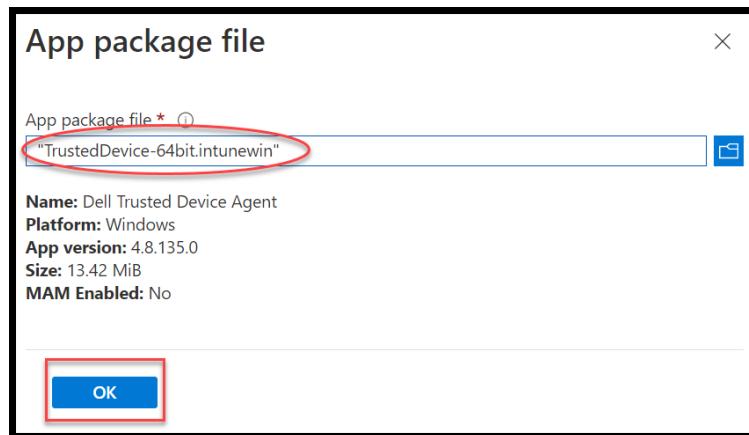
Click 'Folder'



Select 'TrustedDevice-64bit.intunewin'



Click 'OK'



| Field | Value |
|---|--|
| Name | Dell Trusted Device Agent |
| Publisher | Dell Inc. |
| App Version | 5.9.8.0 Note: Use version of the Dell Trusted Device Agent |
| Show this as a featured app in the Company Portal | No (default security app by device) |

Click 'Next'

The screenshot shows the "Add App" wizard. On the first step, the "Name" field is filled with "Dell Trusted Device Agent". The "Description" field contains the path "Trusted-Device-Agent_RN4V4_WIN_5.9.8.0_A51.exe". The "Publisher" field is set to "Dell Inc.". The "App Version" field is set to "5.9.8.0". The "Category" dropdown shows "2 selected". Under the "Show this as a featured app in the Company Portal" section, the "No" option is selected. At the bottom of the screen, there are "Previous" and "Next" buttons, with the "Next" button highlighted with a red box.

Section 'Program'

The screenshot shows the 'Program' tab selected in the navigation bar of an application management interface. The tab is highlighted with a red circle. Other tabs visible include 'App information', 'Requirements', 'Detection rules', 'Dependencies', 'Supersedence (preview)', 'Assignments', and 'Review + create'. The main content area displays configuration options for an application.

| Field | Value |
|-------------------|--|
| Install command | powershell.exe -executionpolicy bypass .\\DellTrustedDevice_install.ps1 |
| Uninstall command | powershell.exe -executionpolicy bypass .\\DellTrustedDevice_uninstall.ps1 |

Click 'Next'

The screenshot shows the 'Program' configuration page of an application management interface. The 'Install command' field contains the PowerShell script: 'powershell.exe -executionpolicy bypass .\\DellTrustedDevice_install.ps1'. The 'Uninstall command' field also contains a similar PowerShell script. The 'Next' button at the bottom is highlighted with a red box, indicating the next step in the process.

Section 'Requirements'

The screenshot shows the 'Add App' interface with the 'Requirements' tab selected. The URL in the address bar is 'Home > Apps > Windows > Add App'. Below the address bar, it says 'Windows app (Win32)'. The tabs at the top are: App information (green checkmark), Program (green checkmark), Requirements (blue circle with a red border, indicating it's selected), Detection rules (grey), Dependencies (grey), Supersedence (preview) (grey), Assignments (grey), and Review + create (grey). The main content area displays two requirements:

| Field | Value |
|-------------------------------|--|
| Operating system architecture | 64-Bit |
| Minimum operating system | 2004 (Please note Dell support drivers and software only with the latest Win version + N -2) |

Click 'Next'

The screenshot shows the 'Add App' interface on the 'Requirements' step. The URL in the address bar is 'Home > Apps | Windows > Windows | Windows apps > Add App'. Below the address bar, it says 'Windows app (Win32)'. The tabs at the top are: App information (green checkmark), Program (green checkmark), Requirements (blue circle with a red border, indicating it's selected), Detection rules (grey), Dependencies (grey). The main content area has a heading 'Specify the requirements that devices must meet before the app is installed:' followed by several input fields:

- Operating system architecture * (grey info icon): 64-bit (highlighted with a red box)
- Minimum operating system * (grey info icon): Windows 10 2004 (highlighted with a blue box)
- Disk space required (MB): (empty)
- Physical memory required (MB): (empty)
- Minimum number of logical processors required: (empty)
- Minimum CPU speed required (MHz): (empty)

At the bottom are 'Previous' and 'Next' buttons, with 'Next' being highlighted.

Section 'Detection rules'

The screenshot shows the 'Add App' interface with the 'Detection rules' tab selected. Other tabs like 'App information', 'Program', 'Requirements', 'Dependencies', 'Supersedeance (preview)', 'Assignments', and 'Review + create' are visible but not selected.

| Field | Value |
|--------------|------------------------|
| Rules format | Use a custom detection |

Click 'Folder'

The screenshot shows the 'Add App' interface under the 'Detection rules' tab. A red box highlights the 'Script file' input field, which contains 'Select a file'. Other fields include 'Rules format' set to 'Use a custom detection script', 'Run script as 32-bit process on 64-bit clients' (set to No), and 'Enforce script signature check and run script silently' (set to No).

Select 'DellTrustedDevice_DetectionRule.ps1'

The screenshot shows a file explorer window displaying files in the path 'OS (C) > Dell > SoftwareRepository > Dell Trusted Device > 5.9.8.0'. A red box highlights the file 'DellTrustedDevice_DetectionRule.ps1', which is a Windows PowerShell script (PS1 file).

Click 'Next'

The screenshot shows the 'Add App' interface under the 'Detection rules' tab. The 'Script file' field now contains the path 'DellTrustedDevice_DetectionRule.ps1', which is highlighted with a red box. The 'Next' button at the bottom is also highlighted with a red box.

Section 'Dependencies'



No changes

Section 'Supersedence'

A screenshot of the 'Add App' interface with the 'Supersedence (preview)' tab selected. The top navigation bar and tabs are identical to the 'Dependencies' section. The main content area has a note about superseding applications and references to 'Learn more'. Below this is a table titled 'Apps that this app will supersede'. The table lists three entries for 'Dell Trusted Device Agent' by Dell Inc. with versions 4.2.111.0, 4.0.102.0, and 3.8.94.0. To the right of each entry is a column for 'Uninstall previous version' with two buttons: 'Yes' and 'No'. A red circle highlights this column. A red annotation above the table reads: 'If you select "Yes" please check if the detection of old app does not identify the new app'. At the bottom of the screen are 'Previous' and 'Next' buttons, with 'Next' being highlighted with a red border.

No changes

Note: You can also uninstall old software via Supersede, but make sure that the detection of the old application does not also detect the new one, otherwise you will create an installation loop.

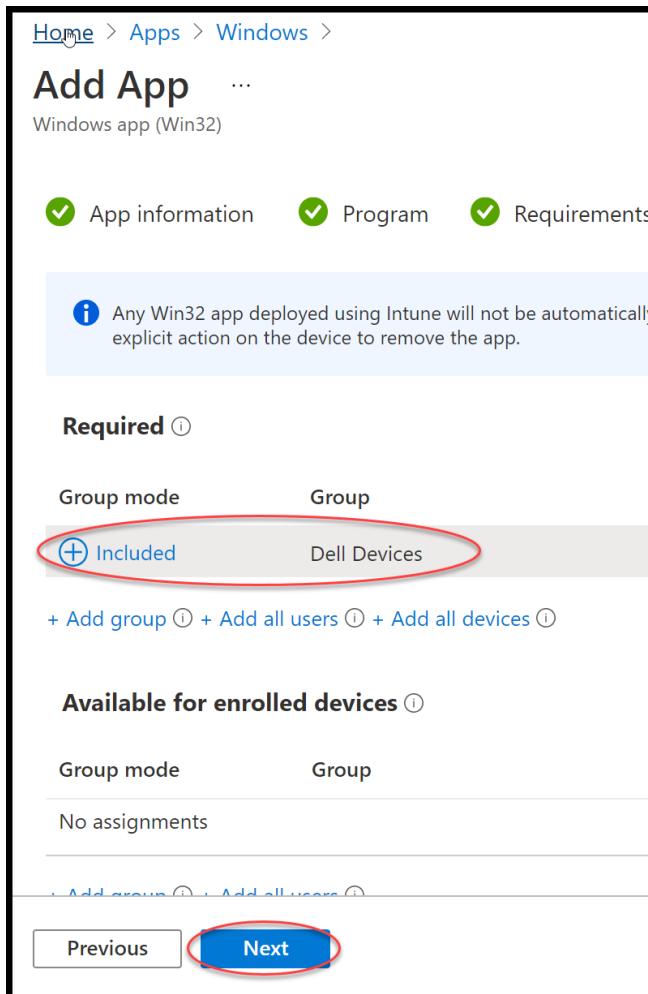
Section 'Assignments'



The Dell Trusted Device Agent supports only Dell (Latitude, Optiplex, Precision and mobile XPS) it makes sense to have a dynamic group which only incl. these systems.

| Option | Value |
|--------------------------------|--------------------------|
| Required | Add group 'Dell Devices' |
| Available for enrolled devices | |
| Uninstall | |

Click 'Next'



The app is now finished

Click 'Create'

Home > Windows | Windows apps >

Add App

Windows app (Win32)

App information Program Requirements Detection rules Dependencies

Summary

App information

| | |
|---|--|
| App package file | Trusted-Device-Agent_RN4V4_WIN_5.9.8.0_A51.intunewin |
| Name | Dell Trusted Device Agent |
| Description | Trusted-Device-Agent_RN4V4_WIN_5.9.8.0_A51.exe |
| Publisher | Dell Inc. |
| App Version | 5.9.8.0 |
| Category | Dell Tools Computer management |
| Show this as a featured app in the Company Portal | No |
| Information URL | No Information URL |
| Privacy URL | No Privacy URL |

Previous Create

Ready to work.

| Name | Type | Status | Version | Assigned |
|---------------------------|---------------------|---------|---------|----------|
| Dell Trusted Device Agent | Windows app (Win32) | 5.9.8.0 | Yes | |

Dell Power Manager

Introduction

Dell Power Manager software provides simplified and efficient power management capabilities for Dell notebooks and tablets running Windows 7, Windows 8, and Windows 10 operating systems.

Dell Power Manager Version User Guide

<https://www.dell.com/support/home/en-us/product-support/product/dell-command-power-manager/docs>

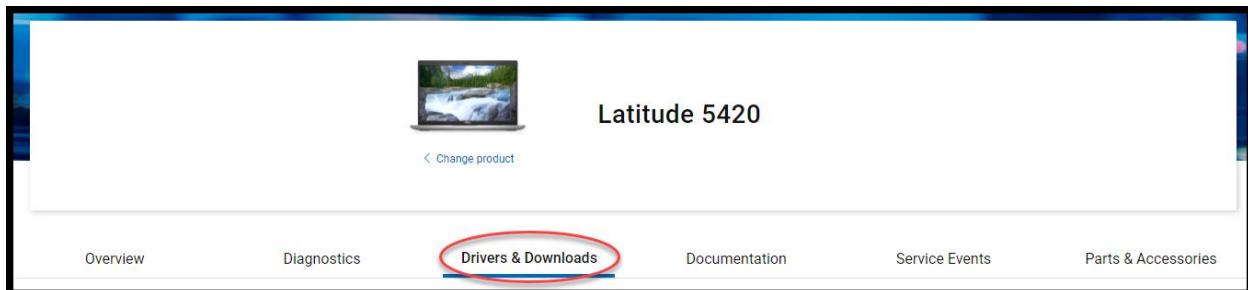
NOTICE: If you have installed Dell Optimizer 4.0.310.0 or later you cannot install Dell Power Manager on the same device separately.

Prepare Dell Power Manager for Intune

Download the newest Version of Dell Power Manager from our website.

<https://www.dell.com/support/home/en-us>

Note: Please select a device platform like Latitude 5420 and move to section 'Drivers & Downloads'



Choose Manually to find a specific driver

| Field | Value |
|---------|---------------|
| Keyword | Power Manager |

Find 'Dell Power Manager Service' and download the installer file.

The screenshot shows a search interface for finding drivers. The 'Keyword' field contains 'power manager'. The 'Operating system' dropdown is set to 'Windows 10, 64-bit'. The search results table has one entry: 'Dell Power Manager Service' (RECOMMENDED, Systems Management, 16 Aug 2022). The 'Download' button is highlighted.

If you have downloaded the file from dell.com/support, copy file to your software repository for the next step.



Scripts for Install, Uninstall and Detection

It is possible to use the native file for installation. In this document we are using PowerShell scripts to cover different scenarios of Install new, Update, and uninstall for a later automation of uploading applications by API.

All script could be download on Github Repository:

<https://github.com/svenriebedell/Dell-Tools-Intune-Install>

Install Script

The script makes a precheck if older versions are installed and starting an uninstall first to have a clean environment.

```
#####
#### Function section      #####
#####

function Get-installedcheck
{
    param
    (
        [Parameter(mandatory=$true)][string] $AppSearchString
    )

    $AppCheck = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearchString'"

    If ($null -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        return $false
    }
}

#####
#### variable section      #####
#####

$InstallerName = Get-ChildItem .\*.exe | Select-Object -ExpandProperty Name
$ProgramPath = ".\" + $InstallerName
[Version]$ProgramVersion_target = (Get-Command $ProgramPath).FileVersionInfo.ProductVersion
$AppSearch = "%Dell%Power%Manager%" #Parameter to search in registry
$Program_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearch'"
[Version]$ProgramVersion_current = $Program_current.Version
$ApplicationID_current = $Program_current.IdentifyingNumber
$SoftwareName = $Program_current.Name

$Program_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearch'"
$SoftwareName = $Program_current.Name
$ApplicationID_current = $Program_current.IdentifyingNumber

#####
#### program section      #####
#####

#### generate Logging Resources
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

#####
#Checking if older Version is installed and uninstall this Version#
#####

If ($null -ne $ProgramVersion_current)
{
    if ($ProgramVersion_target -gt $ProgramVersion_current)
    {
        #####
        # uninstall Software old #
        #####
        Start-Process -FilePath msiexec.exe -ArgumentList "/x $ApplicationID_current /qn" -Wait

        #####
        # uninstall success check #
        #####
        $UninstallResult = Get-installedcheck -AppSearchString $AppSearch
    }
}
```

```

If ($UninstallResult -eq $true)
{
    Write-Host "uninstall is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_current).ToString()
        Uninstall = $false
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData
}

Else
{
    Write-Host "uninstall is successful" -BackgroundColor Green

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_current).ToString()
        Uninstall = $true
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
}

}
Else
{
    Write-Host "same version is installed"

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $false
        Reason = "same version is installed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData

    Exit 0
}
}

#####
#Install new Software
#####
# install success check
#####
$Program_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearch'"
$SoftwareName = $Program_current.Name
$UninstallResult = Get-installedcheck -AppSearchString $AppSearch

If ($UninstallResult -ne $true)
{
    Write-Host "install is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $false
        Reason = "Installation failed, please check if a version of Dell Optimizer is installed on the machine"
}

```

```

        } | ConvertTo-Json

        Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Error -EventId 11 -Message $UninstallData

    }

Else
{
    [Version]$ProgramVersion_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearch'" | Select-Object -
ExpandProperty Version

    If ($ProgramVersion_current -ge $ProgramVersion_target)
    {

        Write-Host "install is successful" -BackgroundColor Green

        $UninstallData = [PSCustomObject]@{
            Software = $SoftwareName
            Version = ($ProgramVersion_target).ToString()
            Install = $true
            Reason = "Update/Install/Newer Version"
        } | ConvertTo-Json

        Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData

    }
    Else
    {
        Write-Host "install is unsuccessful" -BackgroundColor red

        $UninstallData = [PSCustomObject]@{
            Software = $SoftwareName
            Version = ($ProgramVersion_target).ToString()
            Install = $false
            Reason = "Older Version installed $ProgramVersion_current"
        } | ConvertTo-Json

        Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData

    }
}

```

Uninstall Script

The script starts the uninstalling of application.

```
#####
##### Function section #####
#####

function Get-installedcheck
{
    param
    (
        [Parameter(mandatory=$true)][string] $AppSearchString
    )

    $AppCheck = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearchString'"

    If ($null -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        return $false
    }
}

#####
##### variable section #####
#####

$AppSearch = "%Dell%Power%Manager%" #Parameter to search in registry
$Program_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearch'"
$SoftwareName = $Program_current.Name
$ApplicationID_current = $Program_current.IdentifyingNumber

#####
##### program section #####
#####

##### generate Logging Resources
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

#####
#uninstall Software
#####

Start-Process -FilePath msiexec.exe -ArgumentList "/x $ApplicationID_current /qn" -Wait

#####
# uninstall success check #
#####

$UninstallResult = Get-installedcheck -AppSearchString $AppSearch

If ($UninstallResult -eq $true)
{
    Write-Host "uninstall is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = $Program_current.Version
        Uninstall = $false
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData
}

Else
{
    Write-Host "uninstall is successful" -BackgroundColor Green
}
```

```

$UninstallData = [PSCustomObject]@{
    Software = $SoftwareName
    Version = $Program_current.Version
    Uninstall = $true
} | ConvertTo-Json

Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
}

```

Detection Script

The detection script showing success of installation. It could be done as well without a script but again it is helpful for later automation of upload processes in the future.

The **yellow marked** version must be adjusted with each new version.

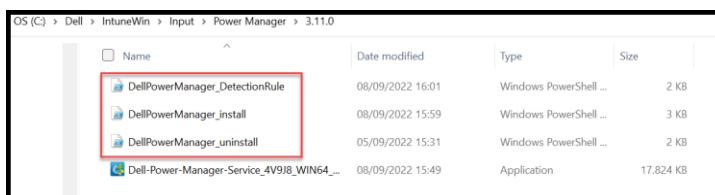
```

#####
# Program with target Version
#####
$ProgramVersion_target = '3.11.0' # need to be the same like the exe file
$ProgramVersion_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '%Dell%Power%Manager%'" | select -ExpandProperty Version

if($ProgramVersion_current -eq $ProgramVersion_target)
{
    Write-Host "Found it!"
}

```

Please, copy these files in the same folder as your EXE Installer File.



Start the Microsoft Win32 Content Prep Tool (aka IntuneAppUtil.exe). In case you are not familiar with this tool, you will find documentation here: <https://docs.microsoft.com/en-us/mem/intune/apps/apps-win32-prepare>

| Argument | Value |
|---------------|---|
| Source Folder | folder where you have stored the unzipped MSI |
| Setup file | Main installer file like msi/exe/ps1, etc. |
| Output Folder | where you want to store the IntuneWin |

```

Please specify the source folder: C:\Dell\IntuneWin\Input\Power Manager\3.11.0
Please specify the setup file: Dell-Power-Manager-Service_4V9J8_WIN64_3.11.0_A00.exe
Please specify the output folder: C:\Dell\IntuneWin\Output\Power Manager\3.11.0
Do you want to specify catalog folder (Y/N)?n

```

IntuneWin is now prepared and ready for installation by Intune.



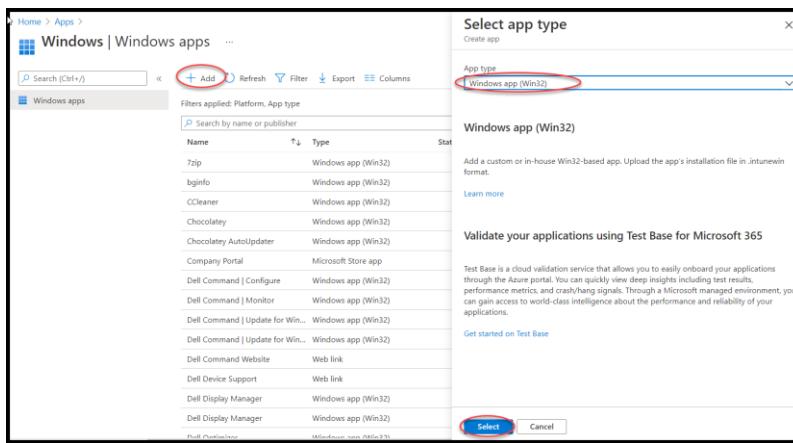
Import and Deployment settings Dell Power Manager for Intune

Click 'Add'

| Field | Value |
|---------------|---------------------|
| Source Folder | Windows app (Win32) |

Click 'Select'

Select Windows app (Win32) as application.

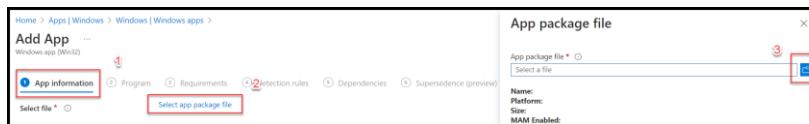


Section 'App information'

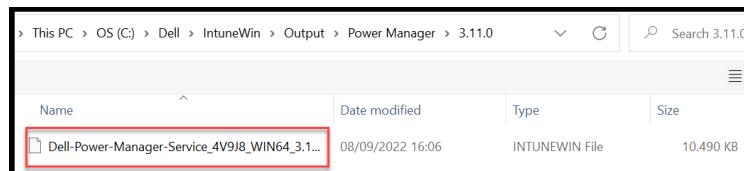


Click 'Select app package file'

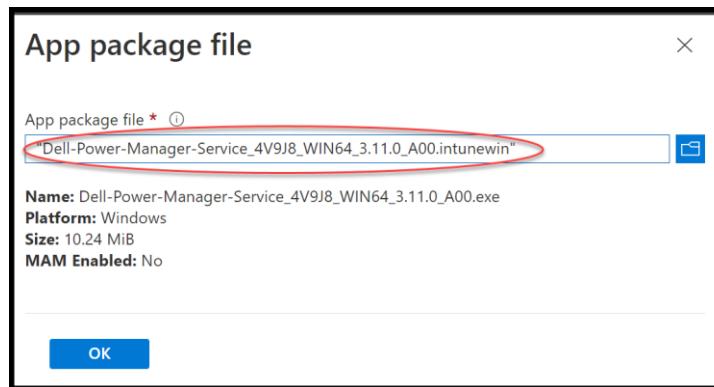
Click 'Folder'



Select 'Dell-Power-Manager-Service_4V9J8_WIN64_3.11.0_A00.intunewin'



Click 'OK'



| Field | Value |
|---|--|
| Name | Dell Power Manager |
| Publisher | Dell Inc. |
| App Version | 3.11.0 |
| | Note: Use version of the Dell Power Manager |
| Show this as a featured app in the Company Portal | Yes |

Click 'Next'

Home > Apps | Windows > Windows | Windows apps >

Add App ...

Windows app (Win32)

① App information ② Program ③ Requirements ④ Detection rules ⑤ Dependencies ⑥ Supersedence (preview)

Select file * ⓘ Dell-Power-Manager-Service_4V9J8_WIN64_3.11.0_A00.intunewin

Name * ⓘ Dell Power Manager

Description * ⓘ Dell-Power-Manager-Service_4V9J8_WIN64_3.11.0_A00.exe

Edit Description

Publisher * ⓘ Dell Inc.

App Version ⓘ 3.11.0

Category ⓘ 0 selected

Show this as a featured app in the Company Portal ⓘ Yes

Information URL ⓘ Enter a valid url

Previous Next

Section 'Program'

The screenshot shows the 'Program' tab selected in the 'Add App' interface. The tab bar includes 'App information', 'Program' (highlighted), 'Requirements', 'Detection rules', 'Dependencies', 'Supersedence (preview)', 'Assignments', and 'Review + create'. The main area displays the configuration for the 'Windows app (Win32)'.

| Field | Value |
|-------------------|---|
| Install command | powershell.exe -executionpolicy bypass .\\DellPowerManager_install.ps1 |
| Uninstall command | powershell.exe -executionpolicy bypass .\\DellPowerManager_uninstall.ps1 |

Click 'Next'

The screenshot shows the 'Program' configuration page for adding an application. The 'Install command' field contains 'powershell.exe -executionpolicy bypass .\\DellPowerManager_install.ps1' and the 'Uninstall command' field contains 'powershell.exe -executionpolicy bypass .\\DellPowerManager_uninstall.ps1'. Both fields have green checkmarks. The 'Return code' section lists four entries: 0 (Success), 1707 (Success), 3010 (Soft reboot), and 1641 (Hard reboot). The 'Code type' dropdown next to each entry has a small red box around it. At the bottom, there are 'Previous' and 'Next' buttons, with 'Next' being highlighted with a red box.

Section 'Requirements'

The screenshot shows the 'Add App' interface with the 'Requirements' tab selected. The URL in the address bar is 'Home > Apps > Windows > Add App'. Below it, it says 'Windows app (Win32)'. The tabs at the top are: App information (green checkmark), Program (green checkmark), Requirements (blue circle with a red border, indicating it's selected), Detection rules, Dependencies, Superseding (preview), Assignments, and Review + create.

| Field | Value |
|-------------------------------|--|
| Operating system architecture | 64-Bit |
| Minimum operating system | 2004 (Please note Dell support drivers and software only with the latest Win version + N -2) |

Click 'Next'

The screenshot shows the 'Add App' interface on the 'Requirements' step. The URL is 'Home > Apps | Windows > Windows | Windows apps > Add App'. The tabs at the top are: App information (green checkmark), Program (green checkmark), Requirements (blue circle with a red border, indicating it's selected), Detection rules, and Dependencies. The page title is 'Add App' with a '...' button. It says 'Windows app (Win32)'. The main content area is titled 'Specify the requirements that devices must meet before the app is installed:' and contains the following fields:
Operating system architecture * (1): 64-bit (highlighted with a red box)
Minimum operating system * (1): Windows 10 2004 (highlighted with a blue box)
Disk space required (MB): (empty)
Physical memory required (MB): (empty)
Minimum number of logical processors required (1): (empty)
Minimum CPU speed required (MHz): (empty)
At the bottom are 'Previous' and 'Next' buttons, with 'Next' being highlighted in blue.

Section 'Detection rules'

The screenshot shows the 'Add App' interface with the 'Detection rules' tab highlighted. Other tabs like 'App information', 'Program', 'Requirements', 'Dependencies', 'Supersedeance (preview)', 'Assignments', and 'Review + create' are visible but not selected.

| Field | Value |
|--------------|------------------------|
| Rules format | Use a custom detection |

Click 'Folder'

The screenshot shows the 'Add App' interface under the 'Detection rules' tab. The 'Rules format' dropdown is set to 'Use a custom detection script'. Below it, the 'Script file' input field is highlighted with a red box. A 'Select a file' button is shown next to it. Other options like 'Run script as 32-bit process on 64-bit clients' and 'Enforce script signature check and run script silently' are also visible.

Select 'DellPowerManager_DetectionRule.ps1'

The screenshot shows a file explorer window with a search bar at the top. The results list includes three files: 'DellPowerManager_DetectionRule.ps1' (selected and highlighted with a red box), 'DellPowerManager_install', and 'DellPowerManager_uninstall'. The file 'DellPowerManager_DetectionRule.ps1' is a Windows PowerShell script (2 KB).

Click 'Next'

The screenshot shows the 'Add App' interface under the 'Detection rules' tab. The 'Script file' field now contains the path 'DellPowerManager_DetectionRule.ps1'. The 'Next' button at the bottom is highlighted with a red box. The 'Previous' button is also visible.

Section 'Dependencies'



No changes

Section 'Supersedence'

A screenshot of the 'Supersedence (preview)' tab in the 'Add App' interface. The top navigation bar and tabs are identical to the previous screenshot. The main content area has a heading 'When you supersede an application, you can specify which app will be updated or replaced. To update an app, disable the uninstall previous version option. To replace an app, enable the uninstall previous version option. There is a maximum of 10 updated or replaced apps, including references to other apps. For example, your app references another app. This other app references other apps, and so on. This scenario creates a graph of apps. All apps in the graph count toward the maximum value of 10.' Below this is a note: 'If you select "Yes" please check if the detection of old app does not identify the new app.' A red oval highlights the 'Uninstall previous version' section for two entries. The first entry has 'Yes' checked and 'No' uncheckable. The second entry has 'Yes' uncheckable and 'No' checked. Both rows have a '...' button. At the bottom are 'Previous' and 'Next' buttons, with 'Next' being blue and highlighted.

No changes

Note: You can also uninstall old software via Supersede, but make sure that the detection of the old application does not also detect the new one, otherwise you will create an installation loop.

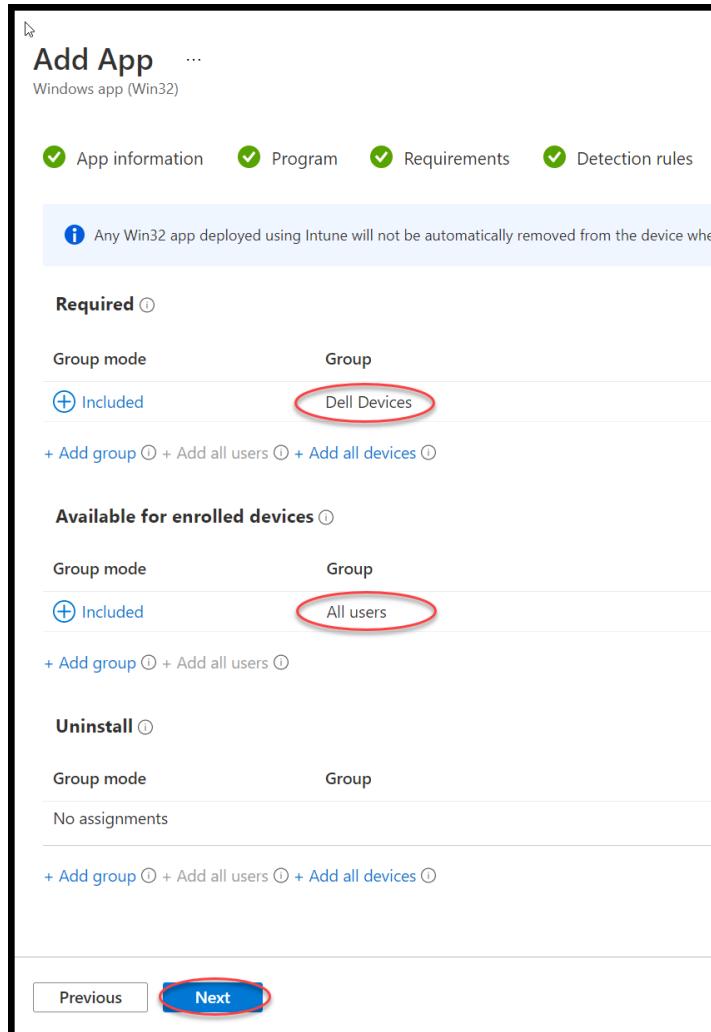
Section 'Assignments'



The Dell Power Manager supports only Dell (Latitude, Optiplex, Precision and mobile XPS) it makes sense to have da dynamic group which only incl. these systems.

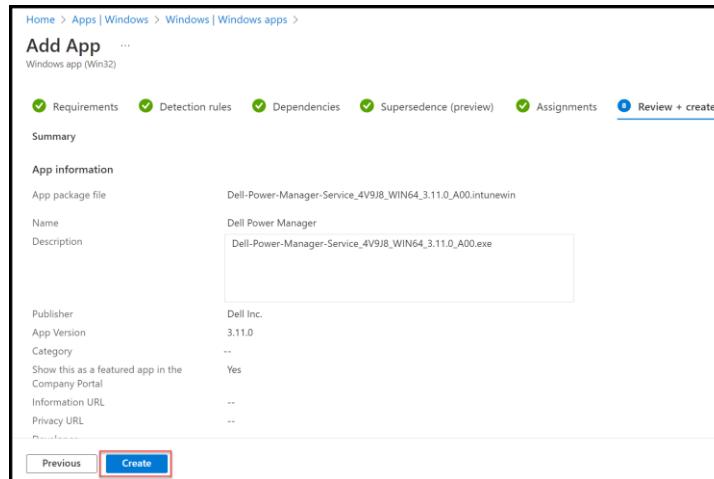
| Option | Value |
|--------------------------------|--------------------------|
| Required | Add group 'Dell Devices' |
| Available for enrolled devices | Add group 'All User' |
| Uninstall | |

Click 'Next'



The app is now finished

Click 'Create'



Ready to work.

The screenshot shows the 'Windows apps' list view. The search bar at the top contains the text 'powered'. The results table has columns: Name, Type, Status, Version, and Assigned. The first row, 'Dell Power Manager', is highlighted with a red box. The other two rows show 'Dell Power Manager Serv...' and 'Dell Power Manager Serv...', both listed as Windows app (Win32) with version 3.9.0 and 3.10.0 respectively, and assigned status 'No'.

| Name | Type | Status | Version | Assigned |
|----------------------------|---------------------|--------|---------|----------|
| Dell Power Manager | Windows app (Win32) | 3.11.0 | Yes | |
| Dell Power Manager Serv... | Windows app (Win32) | 3.9.0 | No | |
| Dell Power Manager Serv... | Windows app (Win32) | 3.10.0 | No | |

Dell Command | Update

Introduction

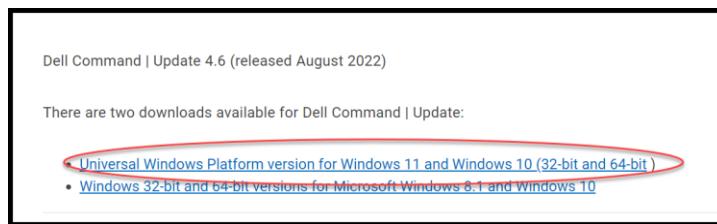
Dell Command | Update is a stand-alone application, for commercial client computers that provides updates for system software released by Dell. This application simplifies the BIOS, firmware, driver, and application update experience for Dell commercial client hardware. This application can also be used to install drivers after the operating system and network drivers are installed, based on the computer identity.

Dell Command | Update Reference Guide

<https://www.dell.com/support/home/en-us/product-support/product/command-update/docs>

Prepare Dell Command | Update for Intune

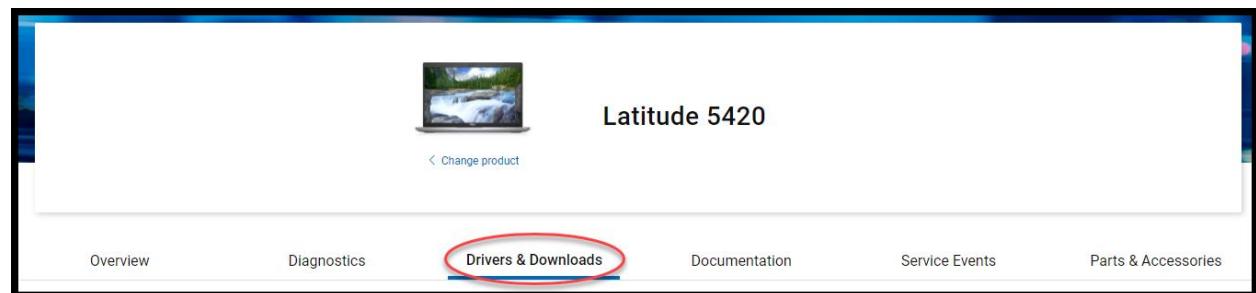
Please be aware that Dell offers two Versions of installer Dell Command | Update Application for Windows 10 (or Windows Universal Application) and Dell Command | Update. For this guide we are using the Universal Windows Application.



Download the newest Version of Dell Command | Update from our website.

<https://www.dell.com/support/home/en-us>

Note: Please select a device platform like Latitude 5420 and move to section 'Drivers & Downloads'



Choose Manually to find a specific driver

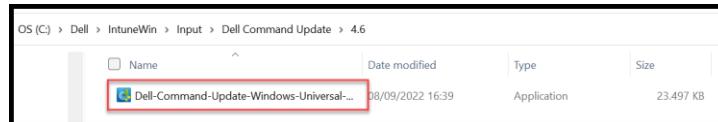
| Field | Value |
|---------|--------|
| Keyword | Update |

Find 'Dell Command | Update Windows Universal Application' and download the installer file.

The screenshot shows a search interface for drivers. The 'Keyword' field contains 'Update'. The 'Operating system' dropdown is set to 'Windows 10, 64-bit'. The search results table has columns for NAME, IMPORTANCE, CATEGORY, RELEASE DATE, and ACTION. Two items are listed: 'Dell Command | Update Application' and 'Dell Command | Update Windows Universal Application'. The second item is highlighted with a red box.

| NAME | IMPORTANCE | CATEGORY | RELEASE DATE | ACTION |
|---|------------|--------------------|--------------|---------------------------|
| Dell Command Update Application | URGENT | Systems Management | 25 Aug 2022 | <button>Download</button> |
| Dell Command Update Windows Universal Application | URGENT | Systems Management | 25 Aug 2022 | <button>Download</button> |

If you have downloaded the file from <https://www.dell.com/support>, copy file to your software repository for the next step.



Scripts for Install, Uninstall and Detection

It is possible to use the native file for installation. In this document we are using PowerShell scripts to cover different scenarios of Install new, Update, and uninstall for a later automation of uploading applications by API.

All script could be download on Github Repository:

<https://github.com/svenriebedell/Dell-Tools-Intune-Install>

Install Script

The script makes a precheck if older versions are installed and starting an uninstall first to have a clean environment.

```
#####
#### Function section #####
#####

function Get-installedcheck
{
    param
    (
        [Parameter(mandatory=$true)][string] $AppSearchString
    )

    $AppCheck = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearchString'"

    If ($null -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        return $false
    }
}

#####
#### variable section #####
#####

$InstallerName = Get-ChildItem .\*.exe | Select-Object -ExpandProperty Name
$ProgramPath = "\\" + $InstallerName
$AppSearch = "%Dell Command%Update%" #Parameter to search in registry
[Version]$ProgramVersion_target = (Get-Command $ProgramPath).FileVersionInfo.ProductVersion
$UninstallApp = Get-CimInstance -ClassName Win32_Product | Where-Object {$_.Name -like "Dell*Command*Update*"}
$SoftwareName = $UninstallApp.Name
[Version]$ProgramVersion_current = $UninstallApp.Version

#####
#### program section #####
#####

#### generate Logging Resources
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

#####
#Checking if older Version is installed and uninstall this Version #
#####

If ($null -ne $ProgramVersion_current)
{
    if ($ProgramVersion_target -gt $ProgramVersion_current)
    {
        #####
        # uninstall Software old #
        #####
        $Argument = "/x " + $UninstallApp.PackageCache + " /qn"
        Start-Process -FilePath msiexec.exe -ArgumentList "$Argument" -Wait

        #####
        # uninstall success check #
        #####
        $UninstallResult = Get-installedcheck -AppSearchString $AppSearch

        If ($UninstallResult -eq $true)
        {
            Write-Host "uninstall is unsuccessful" -BackgroundColor Red
        }
    }
}
```

```

$UninstallData = [PSCustomObject]@{
    Software = $SoftwareName
    Version = ($ProgramVersion_current).ToString()
    Uninstall = $false
} | ConvertTo-Json

Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData

}

Else
{

    Write-Host "uninstall is successful" -BackgroundColor Green

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_current).ToString()
        Uninstall = $true
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData

}

}

Else
{
    Write-Host "same version is installed"

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $false
        Reason = "same version is installed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData

}

Exit 0
}

#####
#Install new Software
#####

Start-Process -FilePath "$ProgramPath" -ArgumentList "/s" -Wait

#####
# install success check
#####

$UninstallResult = Get-installedcheck -AppSearchString $AppSearch
$UninstallApp = Get-CimInstance -ClassName Win32_Product | Where-Object {$_.Name -like "Dell*Command*Update*"}
$SoftwareName = $UninstallApp.Name

If ($UninstallResult -ne $true)
{
    Write-Host "install is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = "Dell Command | Update Classic/UWP"
        Version = ($ProgramVersion_target).ToString()
        Install = $false
        Reason = "Installation failed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Error -EventId 11 -Message $UninstallData
}

Else
{

```

```

[Version]$ProgramVersion_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearch'" | Select-Object -
ExpandProperty Version

If ($ProgramVersion_current -ge $ProgramVersion_target)
{
    Write-Host "install is successful" -BackgroundColor Green

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $true
        Reason = "Update/Install/Newer Version"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData
}

Else
{
    Write-Host "install is unsuccessful" -BackgroundColor red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $false
        Reason = "Older Version installed $ProgramVersion_current"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData
}
}

```

Uninstall Script

```
#####
#### Function section #####
#####

function Get-installedcheck
{
    param
    (
        [Parameter(mandatory=$true)][string] $AppSearchString
    )

    $AppCheck = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearchString'"

    If ($null -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        return $false
    }
}

#####
#### variable section #####
#####

$UninstallApp = Get-CimInstance -ClassName Win32_Product | Where-Object {$_.Name -like "Dell*Command*Update*"}
$AppSearch = "%Dell Command%Update%" #Parameter to search in registry
$SoftwareName = $UninstallApp.Name

#####
#### program section #####
#####

#### generate Logging Resources
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

#####
# uninstall Software #
#####

$Argument = "/x " + $UninstallApp.PackageCache + " /qn"
Start-Process -FilePath msieexec.exe -ArgumentList "$Argument" -Wait

#####
# uninstall success check #
#####

$UninstallResult = Get-installedcheck -AppSearchString $AppSearch

If ($UninstallResult -eq $true)
{
    Write-Host "uninstall is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = $UninstallApp.Version
        Uninstall = $false
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData
}

Else
{
    Write-Host "uninstall is successful" -BackgroundColor Green

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
    }
}
```

```

        Version = $UninstallApp.Version
        Uninstall = $true
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
}

```

Detection Script

The detection script showing success of installation. It could be done as well without a script but again it is helpful for later automation of upload processes in the future.

The yellow marked version must be adjusted with each new version.

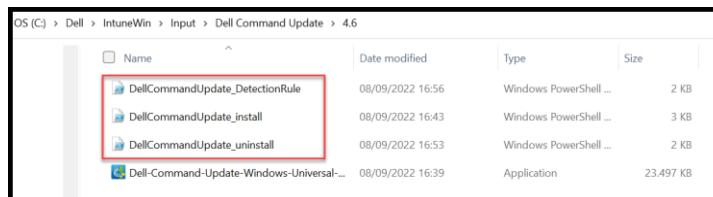
```

#####
# Program with target Version
#####
$ProgramVersion_target = '4.6.0' # need to be the same like the exe file
$ProgramVersion_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '%Dell%Command%Update%'" | Select-Object -ExpandProperty Version

if($ProgramVersion_current -eq $ProgramVersion_target)
{
    Write-Host "Found it!"
}

```

Please, copy these files in the same folder as your EXE Installer File.



Start the Microsoft Win32 Content Prep Tool (aka IntuneAppUtil.exe). In case you are not familiar with this tool, you will find documentation here: <https://docs.microsoft.com/en-us/mem/intune/apps/apps-win32-prepare>

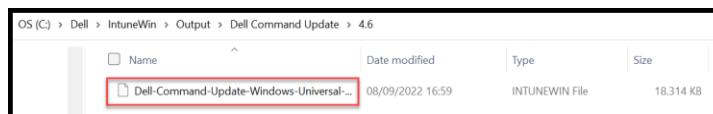
| Argument | Value |
|---------------|---|
| Source Folder | folder where you have stored the unzipped MSI |
| Setup file | Main installer file like msi/exe/ps1, etc. |
| Output Folder | where you want to store the IntuneWin |

```

Please specify the source folder: C:\Dell\IntuneWin\Input\Dell Command Update\4.6
Please specify the setup file: Dell-Command-Update-Windows-Universal-Application_DT6YC_WIN_4.6.0_A00.exe
Please specify the output folder: C:\Dell\IntuneWin\Output\Dell Command Update\4.6
Do you want to specify catalog folder (Y/N)?n

```

IntuneWin is now prepared and ready for installation by Intune.



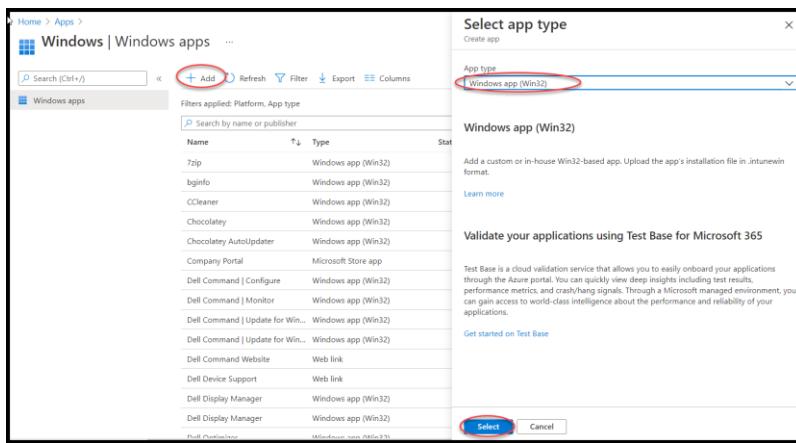
Import and Deployment settings Dell Command | Update for Intune

Click ‘Add’

| Field | Value |
|---------------|---------------------|
| Source Folder | Windows app (Win32) |

Click ‘Select’

Select Windows app (Win32) as application.

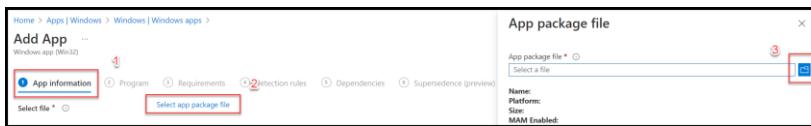


Section ‘App information’

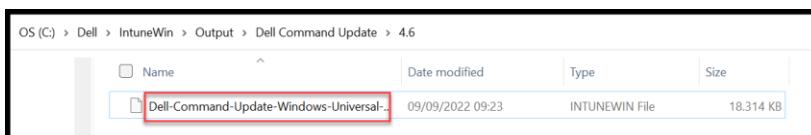


Click ‘Select app package file’

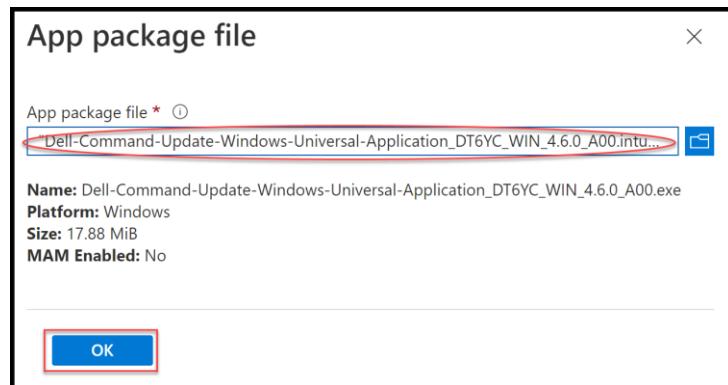
Click 'Folder'



Select 'Dell-Command-Update-Windows-Universal-Application DT6YC WIN 4.6.0 A00.intunewin'



Click 'OK'



| Field | Value |
|---|--|
| Name | Dell Command Update |
| Publisher | Dell Inc. |
| App Version | 4.6.0 Note: Use version of the Dell Command Update |
| Show this as a featured app in the Company Portal | Yes |

Click 'Next'

Home > Apps | Windows > Windows | Windows apps >

Add App ...

Windows app (Win32)

1 App information 2 Program 3 Requirements 4 Detection rules 5 Dependencies 6 Supers

Select file * ⓘ Dell-Command-Update-Windows-Universal-Application_DT6YC_WIN_4.6.0_A00.intunewin

Name * ⓘ Dell Command | Update

Description * ⓘ Dell-Command-Update-Windows-Universal-Application_DT6YC_WIN_4.6.0_A00.exe

Edit Description

Publisher * ⓘ Dell Inc.

App Version ⓘ 4.6.0

Category ⓘ 0 selected

Show this as a featured app in the Company Portal ⓘ Yes

Information URL ⓘ Enter a valid url

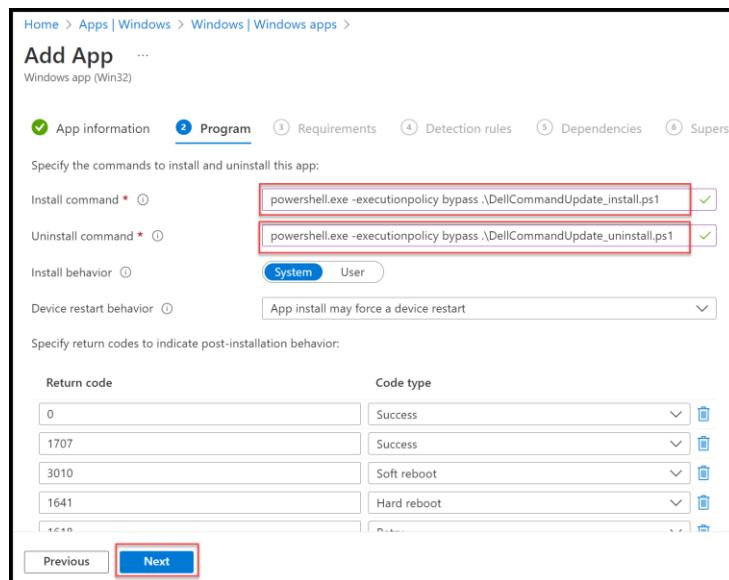
Previous Next

Section 'Program'



| Field | Value |
|-------------------|--|
| Install command | powershell.exe -executionpolicy bypass .\DellCommandUpdate_install.ps1 |
| Uninstall command | powershell.exe -executionpolicy bypass .\DellCommandUpdate_uninstall.ps1 |

Click 'Next'



Section 'Requirements'

The screenshot shows the 'Add App' interface with the 'Requirements' tab selected. The URL in the address bar is 'Home > Apps > Windows > Add App'. Below the tabs, there are several status indicators: App information (green checkmark), Program (green checkmark), Requirements (blue circle with a checkmark, highlighted with a red oval), Detection rules (grey circle), Dependencies (grey circle), Supersedence (grey circle), Assignments (grey circle), and Review + create (grey circle).

| Field | Value |
|-------------------------------|--|
| Operating system architecture | 64-Bit |
| Minimum operating system | 2004 (Please note Dell support drivers and software only with the latest Win version + N -2) |

Click 'Next'

The screenshot shows the 'Add App' interface at the 'Requirements' step. The URL in the address bar is 'Home > Apps | Windows > Windows | Windows apps > Add App'. The 'Requirements' tab is selected. The form asks to specify requirements for device installation. The 'Operating system architecture' field contains '64-bit' (highlighted with a red box). The 'Minimum operating system' field contains 'Windows 10 2004' (highlighted with a blue box). Other fields like 'Disk space required (MB)', 'Physical memory required (MB)', 'Minimum number of logical processors required', and 'Minimum CPU speed required (MHz)' are empty. At the bottom, there are 'Previous' and 'Next' buttons, with 'Next' being highlighted.

Section 'Detection rules'

The screenshot shows the 'Add App' interface with the 'Detection rules' tab highlighted. Other tabs like 'App information', 'Program', 'Requirements', 'Dependencies', 'Supersedeance (preview)', 'Assignments', and 'Review + create' are visible but not selected.

| Field | Value |
|--------------|------------------------|
| Rules format | Use a custom detection |

Click 'Folder'

The screenshot shows the 'Add App' interface under the 'Detection rules' tab. The 'Rules format' dropdown is set to 'Use a custom detection script'. Below it, there's a 'Select a file' button. The 'Script file' input field is highlighted with a red box, and the 'Select a file' button next to it is also highlighted with a red box.

Select 'DellCommandUpdate_DetectionRule.ps1'

The screenshot shows a file explorer window with the path 'OS (C) > Dell > IntuneWin > Input > Dell Command Update > 4.6'. Inside this folder, three files are listed: 'DellCommandUpdate_DetectionRule.ps1', 'DellCommandUpdate_install.ps1', and 'DellCommandUpdate_uninstall.ps1'. The first file, 'DellCommandUpdate_DetectionRule.ps1', is highlighted with a red box.

Click 'Next'

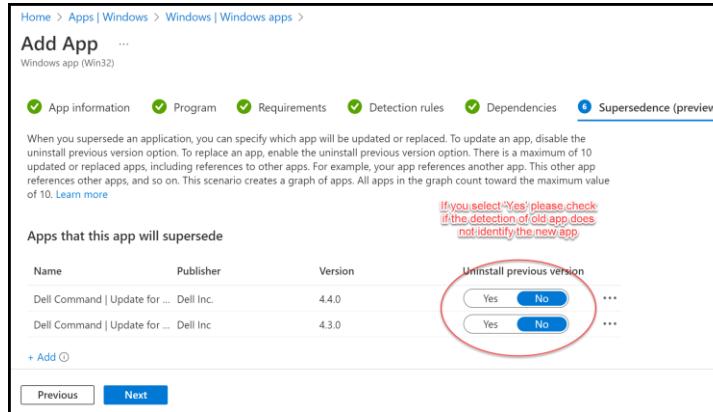
The screenshot shows the 'Add App' interface under the 'Detection rules' tab. The 'Script file' input field contains the path 'DellCommandUpdate_DetectionRule.ps1', which is highlighted with a red box. At the bottom, the 'Next' button is highlighted with a red box.

Section 'Dependencies'



No changes

Section 'Supersedence'



No changes

Note: You can also uninstall old software via Supersede, but make sure that the detection of the old application does not also detect the new one, otherwise you will create an installation loop.

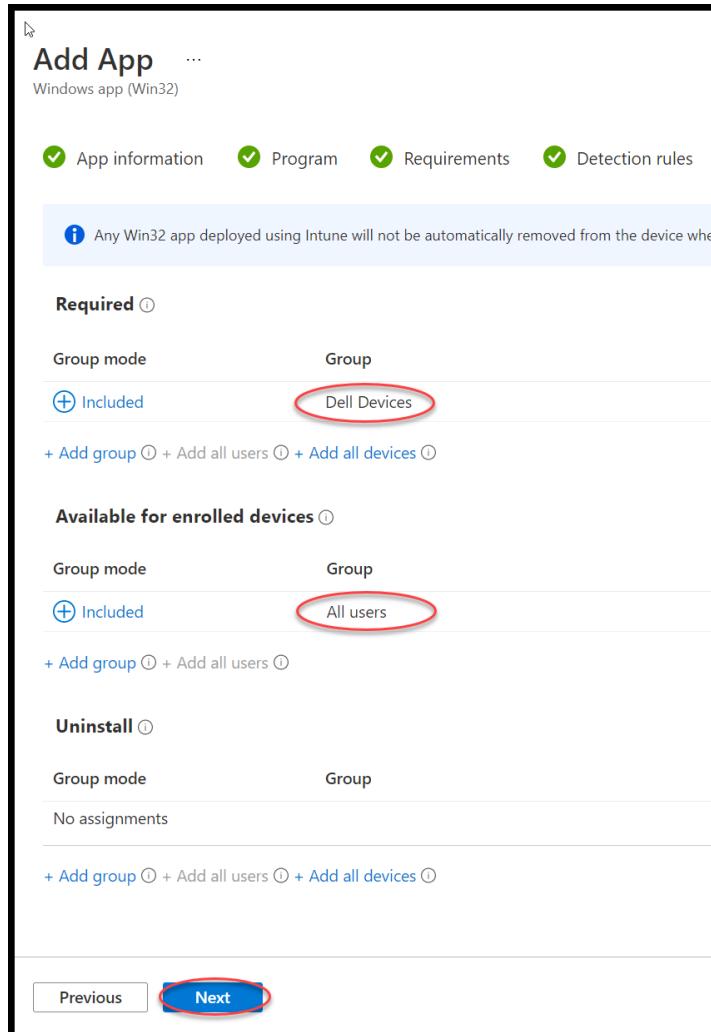
Section 'Assignments'



The Dell Command | Update supports only Dell (Latitude, Optiplex, Precision and mobile XPS) it makes sense to have da dynamic group which only incl. these systems.

| Option | Value |
|--------------------------------|--------------------------|
| Required | Add group 'Dell Devices' |
| Available for enrolled devices | Add group 'All User' |
| Uninstall | |

Click 'Next'



The app is now finished

Click 'Create'

Home > Apps | Windows > Windows | Windows apps >

Add App ...

Windows app (Win32)

✓ App information ✓ Program ✓ Requirements ✓ Detection rules ✓ Dependencies

Summary

App information

| | |
|---|---|
| App package file | Dell-Command-Update-Windows-Universal-Application_DT6YC_WIN_4.6.0_A00.intunewin |
| Name | Dell Command Update |
| Description | Dell-Command-Update-Windows-Universal-Application_DT6YC_WIN_4.6.0_A00.exe |
| Publisher | Dell Inc. |
| App Version | 4.6.0 |
| Category | -- |
| Show this as a featured app in the Company Portal | Yes |
| Information URL | -- |
| Privacy URL | -- |

Previous **Create**

Ready to work.

| Name | Type | Status | Version | Assigned | ... |
|----------------------------------|---------------------|--------------|------------|------------|-----|
| Chocolatey AutoUpdater | Windows app (Win32) | No | ... | | |
| Dell Command Update f... | Windows app (Win32) | 4.3.0 | No | ... | |
| Dell Command Update f... | Windows app (Win32) | 4.4.0 | No | ... | |
| Dell Command Update ... | Windows app (Win32) | 4.6.0 | Yes | ... | |

Dell Command | Monitor

Introduction

The Dell Command | Monitor software application enables IT administrators to easily manage fleet inventory, monitor system health, modify BIOS settings, and remotely collect information for deployed Dell client systems.

Active system health state monitoring can help reduce the total cost of system ownership and is part of a holistic approach to managing all networked devices.

Dell Command | Monitor is designed for Dell Enterprise client systems, Dell IoT Gateway systems, and for Dell Embedded PCs.

Dell Command | Monitor Reference Guide

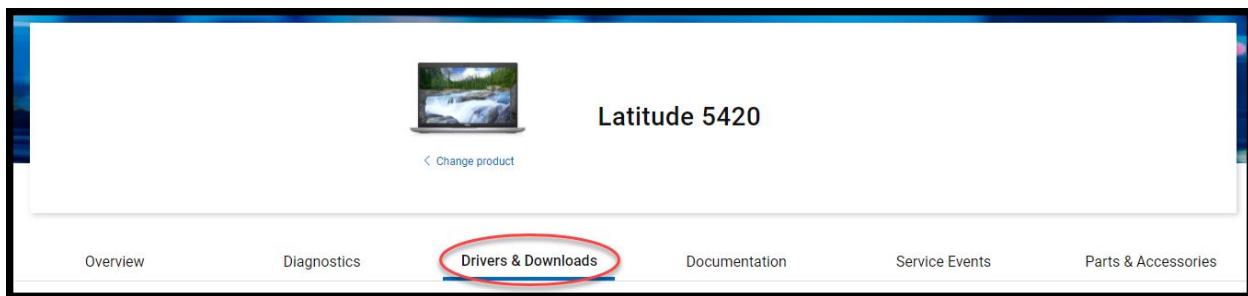
<https://www.dell.com/support/home/en-us/product-support/product/command-monitor/docs>

Prepare Dell Command | Monitor for Intune

Download the newest Version of Dell Command | Update from our website.

<https://www.dell.com/support/home/en-us>

Note: Please select a device platform like Latitude 5420 and move to section 'Drivers & Downloads'



Choose Manually to find a specific driver

| Field | Value |
|---------|--------|
| Keyword | Update |

Find 'Dell Command | Monitor' and download the installer file.

The screenshot shows a search interface for drivers. In the 'Keyword' field, 'monitor' is typed. The 'Operating system' dropdown is set to 'Windows 10, 64-bit'. Below the search bar, there are filters for 'Download Type' (All) and 'Category' (All). A note at the top states: 'This is a comprehensive list of all available downloads for your Latitude 5420. Some downloads may already exist on your device. To let Dell automatically find available updates for you, select Check for Updates. Select "This Device" to view and manually install available downloads specific to your device's unique identifier. (Show me how.)' Below the filters, there are two tabs: 'Latitude 5420 (6)' and 'This Device'. The main table lists six items, with the last item, 'Dell Command | Monitor', highlighted with a red box. The table columns include Name, Importance, Category, Release Date, and Action (Download).

| Name | Importance | Category | Release Date | Action |
|-------------------------------------|-------------|--------------------|--------------|----------|
| Dell Latitude 5420 System BIOS | URGENT | BIOS | 09 Aug 2022 | Download |
| Intel Thunderbolt Controller Driver | URGENT | Chipset | 05 Nov 2021 | Download |
| DBUtil Removal Utility | URGENT | Security | 02 Aug 2021 | Download |
| Trusted Device Agent | RECOMMENDED | Trusted Device | 31 Aug 2022 | Download |
| Dell Command Monitor | RECOMMENDED | Systems Management | 29 Jul 2022 | Download |

If you have downloaded the file from <https://www.dell.com/support/kbdoc/en-us/000177080/dell-command-monitor>, copy file to your software repository for the next step.



Scripts for Install, Uninstall and Detection

It is possible to use the native file for installation. In this document we are using PowerShell scripts to cover different scenarios of Install new, Update, and uninstall for a later automation of uploading applications by API.

All script could be download on Github Repository:
<https://github.com/svenriebedell/Dell-Tools-Intune-Install>

Install Script

The script makes a precheck if older versions are installed and starting an uninstall first to have a clean environment.

```
#####
#### Function section #####
#####

function Get-installedcheck
{
    param
    (
        [Parameter(mandatory=$true)][string] $AppSearchString
    )

    $AppCheck = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearchString'"

    If ($null -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        return $false
    }
}

#####
#### variable section #####
#####

$InstallerName = Get-ChildItem .\*.exe | Select-Object -ExpandProperty Name
$ProgramPath = "\\" + $InstallerName
[Version]$ProgramVersion_target = (Get-Command $ProgramPath).FileVersionInfo.ProductVersion
$AppSearch = "%Dell%Monitor%" #Parameter to search in registry
$UninstallApp = Get-CimInstance -ClassName Win32_Product | Where-Object {$_.Name -like "Dell*Monitor*"}
$SoftwareName = "Dell Command | Monitor"
[Version]$ProgramVersion_current = $UninstallApp.Version

#####
#### program section #####
#####

#### generate Logging Resources
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

#####
#Checking if older Version is installed and uninstall this Version #
#####

If ($null -ne $ProgramVersion_current)
{
    if ($ProgramVersion_target -gt $ProgramVersion_current)
    {
        #####
        # uninstall Software old      #
        #####
        $Argument = "x" + $UninstallApp.PackageCache + " /qn"
        Start-Process -FilePath msiexec.exe -ArgumentList "$Argument" -Wait

        #####
        # uninstall success check     #
        #####
        $UninstallResult = Get-installedcheck -AppSearchString $AppSearch

        If ($UninstallResult -eq $true)
        {
            Write-Host "uninstall is unsuccessful" -BackgroundColor Red
            $UninstallData = [PSCustomObject]@{
```

```

        Software = $SoftwareName
        Version = ($ProgramVersion_current).ToString()
        Uninstall = $false
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData

}

Else
{
    Write-Host "uninstall is successful" -BackgroundColor Green

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_current).ToString()
        Uninstall = $true
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
}

}

Else
{
    Write-Host "same version is installed"

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $false
        Reason = "same version is installed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData

}

Exit 0
}

#####
#Install new Software
#####
Start-Process -FilePath "$ProgramPath" -ArgumentList "/s" -Wait

#####
# install success check
#####
$UninstallResult = Get-installedcheck -AppSearchString $AppSearch

If ($UninstallResult -ne $true)
{
    Write-Host "install is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $false
        Reason = "Installation failed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Error -EventId 11 -Message $UninstallData
}

Else
{
    [Version]$ProgramVersion_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearch'" | Select-Object -ExpandProperty Version
}

```

```

If ($ProgramVersion_current -ge $ProgramVersion_target)
{
    Write-Host "install is successful" -BackgroundColor Green
    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $true
        Reason = "Update/Install/Newer Version"
    } | ConvertTo-Json
    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData
}
Else
{
    Write-Host "install is unsuccessful" -BackgroundColor red
    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $false
        Reason = "Older Version installed $ProgramVersion_current"
    } | ConvertTo-Json
    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData
}
}

```

Uninstall Script

The script starts the uninstalling of application.

```
#####
##### Function section #####
#####

function Get-installedcheck
{
    param
    (
        [Parameter(mandatory=$true)][string] $AppSearchString
    )

    $AppCheck = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearchString'"

    If ($null -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        return $false
    }
}

#####
##### variable section #####
#####

$UninstallApp = Get-CimInstance -ClassName Win32_Product | Where-Object {$__.Name -like "Dell*Monitor*"}
$AppSearch = "%Dell%Monitor%" #Parameter to search in registry
$Program_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearch'"
$SoftwareName = "Dell Command | Monitor"

#####
##### program section #####
#####

##### generate Logging Resources
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

#####
# uninstall Software #
#####

$Argument = "/x "+ $UninstallApp.PackageCache + " /qn"
Start-Process -FilePath msixexec.exe -ArgumentList "$Argument" -Wait

#####
# uninstall success check #
#####

$UninstallResult = Get-installedcheck -AppSearchString $AppSearch

If ($UninstallResult -eq $true)
{
    Write-Host "uninstall is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = $Program_current.Version
        Uninstall = $false
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData
}

Else
{
    Write-Host "uninstall is successful" -BackgroundColor Green
}
```

```

$UninstallData = [PSCustomObject]@{
    Software = $SoftwareName
    Version = $Program_current.Version
    Uninstall = $true
} | ConvertTo-Json

Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
}

```

Detection Script

The detection script showing success of installation. It could be done as well without a script but again it is helpful for later automation of upload processes in the future.

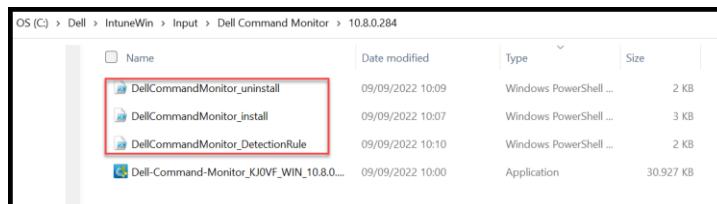
The **yellow marked** version must be adjusted with each new version.

```

#####
# Program with target Version
#####
$ProgramVersion_target = "10.8.0.284" # need to be the same like the exe file
$ProgramVersion_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '%Dell%Monitor%'" | Select-Object -ExpandProperty Version
if($ProgramVersion_current -eq $ProgramVersion_target)
{
    Write-Host "Found it!"
}

```

Please, copy these files in the same folder as your EXE Installer File.



Start the Microsoft Win32 Content Prep Tool (aka IntuneAppUtil.exe). In case you are not familiar with this tool, you will find documentation here: <https://docs.microsoft.com/en-us/mem/intune/apps/apps-win32-prepare>

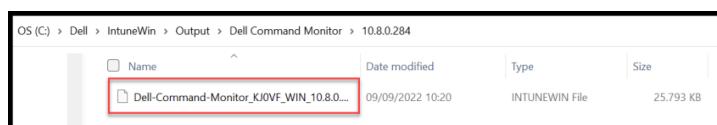
| Argument | Value |
|---------------|---|
| Source Folder | folder where you have stored the unzipped MSI |
| Setup file | Main installer file like msi/exe/ps1, etc. |
| Output Folder | where you want to store the IntuneWin |

```

Please specify the source folder: C:\Dell\IntuneWin\Input\Dell Command Monitor\10.8.0.284
Please specify the setup file: Dell-Command-Monitor_KJ0VF_WIN_10.8.0.284_A00.exe
Please specify the output folder: C:\Dell\IntuneWin\Output\Dell Command Monitor\10.8.0.284
Do you want to specify catalog folder (Y/N)?n

```

IntuneWin is now prepared and ready for installation by Intune.



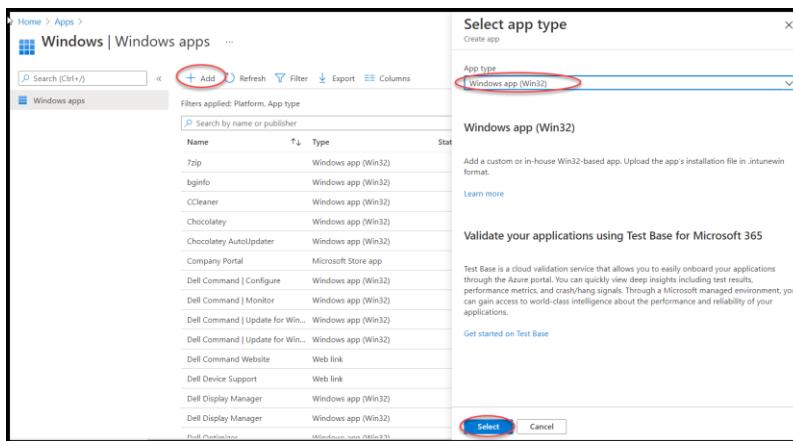
Import and Deployment settings Dell Command | Monitor for Intune

Click 'Add'

| Field | Value |
|---------------|---------------------|
| Source Folder | Windows app (Win32) |

Click 'Select'

Select Windows app (Win32) as application.

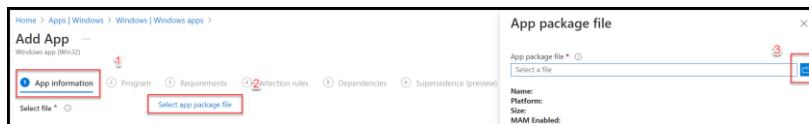


Section 'App information'

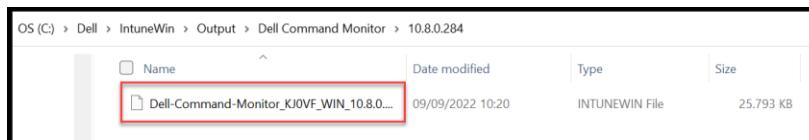


Click 'Select app package file'

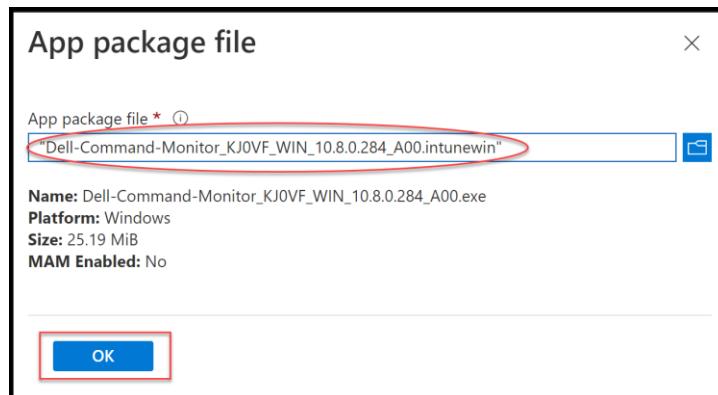
Click 'Folder'



Select 'Dell-Command-Monitor_KJ0VF_WIN_10.8.0.284_A00.intunewin'



Click 'OK'



| Field | Value |
|---|--|
| Name | Dell Command Monitor |
| Publisher | Dell Inc. |
| App Version | 10.8.0.284 Note: Use version of the Dell Command Monitor |
| Show this as a featured app in the Company Portal | No Note: This App has no User UI, and it is for Hardware Management only. |

Click 'Next'

Section 'Program'

The screenshot shows the 'Program' tab selected in the 'Add App' interface. The tab bar includes 'App information', 'Program' (highlighted in red), 'Requirements', 'Detection rules', 'Dependencies', 'Supersedence (preview)', 'Assignments', and 'Review + create'. The main area displays the command for installing the app.

| Field | Value |
|-------------------|---|
| Install command | powershell.exe -executionpolicy bypass .\\DellCommandMonitor_install.ps1 |
| Uninstall command | powershell.exe -executionpolicy bypass .\\DellCommandMonitor_uninstall.ps1 |

Click 'Next'

The screenshot shows the 'Program' configuration page. The 'Install command' field contains 'powershell.exe -executionpolicy bypass .\\DellCommandMonitor_install.ps1' and the 'Uninstall command' field contains 'powershell.exe -executionpolicy bypass .\\DellCommandMonitor_uninstall.ps1'. Both fields are highlighted with red boxes. At the bottom, the 'Next' button is also highlighted with a red box.

Section 'Requirements'

The screenshot shows the 'Add App' interface with the 'Requirements' tab selected. The URL in the address bar is 'Home > Apps > Windows > Add App'. Below it, it says 'Windows app (Win32)'. The tabs at the top are: App information (green checkmark), Program (green checkmark), Requirements (blue circle with a dot, highlighted with a red oval), Detection rules, Dependencies, Superseding (preview), Assignments, and Review + create.

| Field | Value |
|-------------------------------|--|
| Operating system architecture | 64-Bit |
| Minimum operating system | 2004 (Please note Dell support drivers and software only with the latest Win version + N -2) |

Click 'Next'

The screenshot shows the 'Add App' interface on the 'Requirements' step. The URL is 'Home > Apps | Windows > Windows | Windows apps > Add App'. The tabs at the top are: App information (green checkmark), Program (green checkmark), Requirements (blue circle with a dot, highlighted with a red oval), Detection rules, Dependencies, and Superseding (preview). The page title is 'Add App' with three dots. It says 'Windows app (Win32)'. The main content asks 'Specify the requirements that devices must meet before the app is installed:'. It includes fields for 'Operating system architecture *' (set to '64-bit') and 'Minimum operating system *' (set to 'Windows 10 2004'). Other fields like 'Disk space required (MB)', 'Physical memory required (MB)', 'Minimum number of logical processors required', and 'Minimum CPU speed required (MHz)' are present but empty. At the bottom are 'Previous' and 'Next' buttons, with 'Next' being highlighted with a blue rectangle.

Section 'Detection rules'

The screenshot shows the 'Add App' interface with the 'Detection rules' tab highlighted. Other tabs like 'App information', 'Program', 'Requirements', 'Dependencies', 'Supersede (preview)', 'Assignments', and 'Review + create' are visible but not selected.

| Field | Value |
|--------------|------------------------|
| Rules format | Use a custom detection |

Click 'Folder'

The screenshot shows the 'Add App' interface under the 'Detection rules' tab. The 'Rules format' dropdown is set to 'Use a custom detection script'. Below it, the 'Script file' input field is selected and highlighted with a red box. A 'Select a file' button is shown to its right. Other settings like 'Run script as 32-bit process on 64-bit clients' and 'Enforce script signature check and run script silently' are also visible.

Select 'DellCommandMonitor_DetectionRule.ps1'

The screenshot shows a file explorer window with the path 'OS (C) > Dell > IntuneWin > Input > Dell Command Monitor > 10.8.0.284'. The file 'DellCommandMonitor_DetectionRule.ps1' is selected and highlighted with a red box. Other files in the folder include 'DellCommandMonitor_install.ps1' and 'DellCommandMonitor_uninstall.ps1'.

Click 'Next'

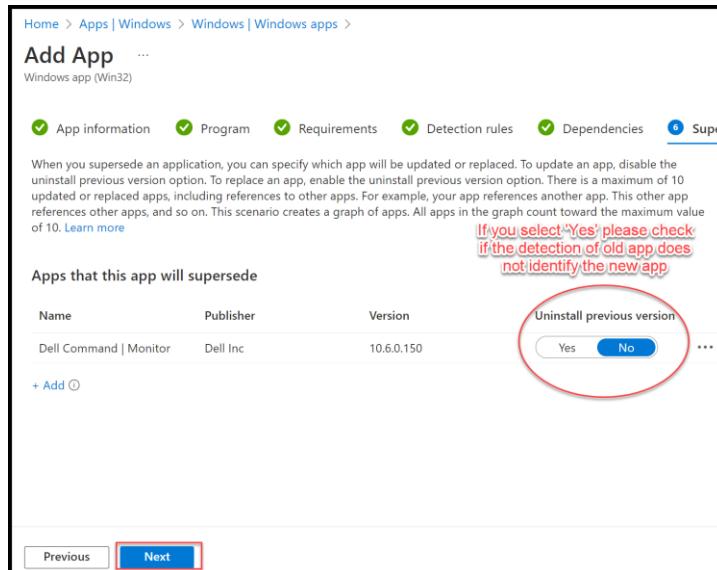
The screenshot shows the 'Add App' interface under the 'Detection rules' tab. The 'Script file' input field now contains the path 'DellCommandMonitor_DetectionRule.ps1', which is highlighted with a red oval. The 'Next' button at the bottom of the form is also highlighted with a red box. The 'Previous' button is visible to its left.

Section 'Dependencies'



No changes

Section 'Supersedence'



No changes

Note: You can also uninstall old software via Supersede, but make sure that the detection of the old application does not also detect the new one, otherwise you will create an installation loop.

Section 'Assignments'

The screenshot shows the 'Assignments' tab highlighted with a red circle. Other tabs visible include 'App information', 'Program', 'Requirements', 'Detection rules', 'Dependencies', 'Supersedence (preview)', and 'Review + create'.

The Dell Command | Monitor supports only Dell (Latitude, Optiplex, Precision and mobile XPS) it makes sense to have da dynamic group which only incl. these systems.

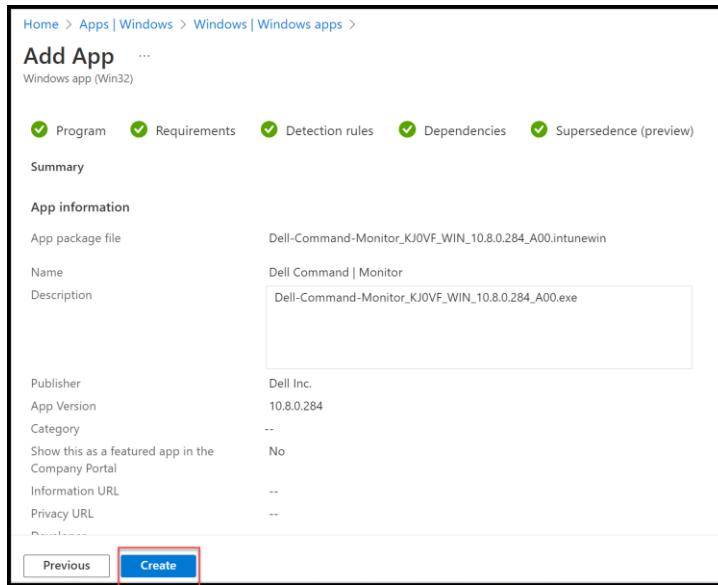
| Option | Value |
|--------------------------------|--------------------------|
| Required | Add group 'Dell Devices' |
| Available for enrolled devices | |
| Uninstall | |

Click 'Next'

The screenshot shows the 'Assignments' step of the 'Add App' wizard. It includes sections for 'Required', 'Available for enrolled devices', and 'Uninstall'. The 'Required' section has a 'Group mode' table where 'Included' is selected and 'Dell Devices' is highlighted with a red circle. The 'Available for enrolled devices' section shows 'No assignments' and a 'None' filter mode. The 'Uninstall' section is partially visible. Navigation buttons 'Previous' and 'Next' are at the bottom, with 'Next' highlighted with a red box.

The app is now finished

Click 'Create'



Ready to work.

A screenshot of the 'Windows | Windows apps' list. The search bar at the top contains 'monitor'. The table has columns: Name, Type, Status, Version, and Assigned. Two rows are visible, both for 'Dell Command | Monitor' (Windows app (Win32)). The first row has a status of 'Yes' and the second has 'No'. A red oval highlights the first row.

| Name | Type | Status | Version | Assigned |
|------------------------|---------------------|--------|------------|----------|
| Dell Command Monitor | Windows app (Win32) | Yes | 10.8.0.284 | |
| Dell Command Monitor | Windows app (Win32) | No | 10.6.0.150 | |

Dell Command | Configure

Introduction

Dell Command | Configure is a packaged software application that provides configuration capability to business client platforms.

This product consists of a Command Line Interface (CLI) and Graphical User Interface (GUI) to configure various BIOS features. Dell Command | Configure supports following Windows and Linux operating systems: Windows 10, Windows Pre-installation Environment (Windows PE), Red Hat Enterprise Linux 7, Red Hat Enterprise Linux 8, Ubuntu Desktop 16.04, Ubuntu Desktop 18.04, Ubuntu Server 18.04 and Ubuntu Desktop 20.04.

Note: Dell Command | Configure is only used by administrators and is usually not installed on all devices. We have only included them in this guide for the sake of completeness.

Dell Command | Configure Command Line Interface Reference Guide

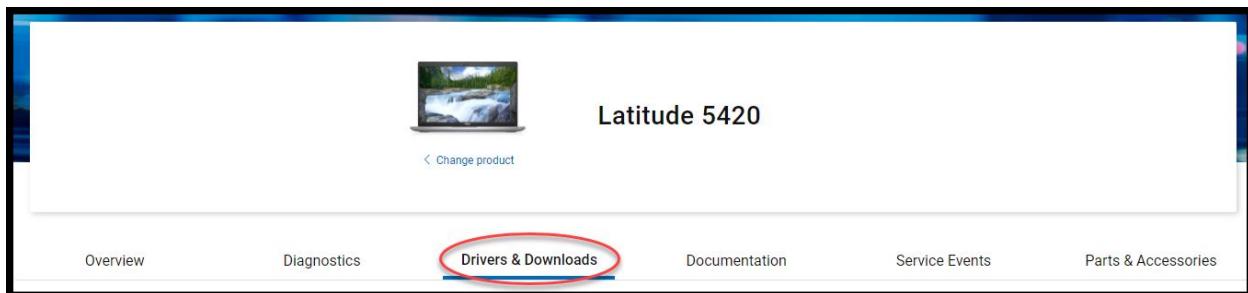
<https://www.dell.com/support/home/en-us/product-support/product/command-configure/docs>

Prepare Dell Command | Configure for Intune

Download the newest Version of Dell Command | Update from our website.

<https://www.dell.com/support/home/en-us>

Note: Please select a device platform like Latitude 5420 and move to section ‘Drivers & Downloads’



Choose Manually to find a specific driver

| Field | Value |
|---------|-----------|
| Keyword | Configure |

Find 'Dell Command | Configure' and download the installer file.

The screenshot shows a search interface for drivers. The 'Keyword' field contains 'configure'. The 'Operating system' dropdown is set to 'Windows 10, 64-bit'. The search results table has columns for Name, Importance, Category, Release Date, and Action. One result, 'Dell Command | Configure', is highlighted with a red box. The table also includes a header row with filters for Name, Importance, Category, Release Date, and Action.

If you have downloaded the file from <https://www.dell.com/support>, copy file to your software repository for the next step.



Scripts for Install, Uninstall and Detection

It is possible to use the native file for installation. In this document we are using PowerShell scripts to cover different scenarios of Install new, Update, and uninstall for a later automation of uploading applications by API.

All script could be download on Github Repository:

<https://github.com/svenriebedell/Dell-Tools-Intune-Install>

Install Script

The script makes a precheck if older versions are installed and starting an uninstall first to have a clean environment.

```
#####
#### Function section #####
#####

function Get-installedcheck
{
    param
    (
        [Parameter(mandatory=$true)][string] $AppSearchString
    )

    $AppCheck = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearchString'"

    If ($null -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        return $false
    }
}

#####
#### variable section #####
#####

$InstallerName = Get-ChildItem .\*.exe | Select-Object -ExpandProperty Name
$ProgramPath = "\\" + $InstallerName
[Version]$ProgramVersion_target = (Get-Command $ProgramPath).FileVersionInfo.ProductVersion
$AppSearch = "%Dell%Configure%" #Parameter to search in registry
$UninstallApp = Get-CimInstance -ClassName Win32_Product | Where-Object {$_.Name -like "Dell*Command*Configure"}
$SoftwareName = "Dell Command | Configure"
[Version]$ProgramVersion_current = $UninstallApp.Version

#####
#### program section #####
#####

##### generate Logging Resources
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

#####
#Checking if older Version is installed and uninstall this Version #
#####

If ($null -ne $ProgramVersion_current)
{
    if ($ProgramVersion_target -gt $ProgramVersion_current)
    {
        #####
        # uninstall Software old      #
        #####
        $Argument = "/x" + $UninstallApp.PackageCache + " /qn"
        Start-Process -FilePath msiexec.exe -ArgumentList "$Argument" -Wait

        #####
        # uninstall success check      #
        #####
        $UninstallResult = Get-installedcheck -AppSearchString $AppSearch

        If ($UninstallResult -eq $true)
        {
            Write-Host "uninstall is unsuccessful" -BackgroundColor Red
        }
    }
}
```

```

$UninstallData = [PSCustomObject]@{
    Software = $SoftwareName
    Version = ($ProgramVersion_current).ToString()
    Uninstall = $false
} | ConvertTo-Json

Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData

}

Else
{

    Write-Host "uninstall is successful" -BackgroundColor Green

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_current).ToString()
        Uninstall = $true
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData

}

}

Else
{
    Write-Host "same version is installed"

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $false
        Reason = "same version is installed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData

}

Exit 0
}

#####
#Install new Software
#####

Start-Process -FilePath "$ProgramPath" -ArgumentList "/s" -Wait

#####
# install success check
#####
$UninstallResult = Get-installedcheck -AppSearchString $AppSearch

If ($UninstallResult -ne $true)
{
    Write-Host "install is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $false
        Reason = "Installation failed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Error -EventId 11 -Message $UninstallData
}

Else
{

```

```

[Version]$ProgramVersion_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearch'" | Select-Object -ExpandProperty Version

If ($ProgramVersion_current -ge $ProgramVersion_target)
{
    Write-Host "install is successful" -BackgroundColor Green

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $true
        Reason = "Update/Install/Newer Version"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData
}

Else
{
    Write-Host "install is unsuccessful" -BackgroundColor red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $false
        Reason = "Older Version installed $ProgramVersion_current"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData
}
}

```

Uninstall Script

The script starts the uninstalling of application.

```
#####
##### Function section #####
#####

function Get-installedcheck
{
    param
    (
        [Parameter(mandatory=$true)][string] $AppSearchString
    )

    $AppCheck = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearchString'"

    If ($null -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        return $false
    }
}

#####
##### variable section #####
#####

$UninstallApp = Get-CimInstance -ClassName Win32_Product | Where-Object {$_._Name -like "Dell*Command*Configure"}
$AppSearch = "%Dell%Configure%" #Parameter to search in registry
$Program_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearch'"
$SoftwareName = "Dell Command | Configure"

#####
##### program section #####
#####

#### generate Logging Resources
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

#####
# uninstall Software #
#####

$Argument = "/x "+ $UninstallApp.PackageCache + " /qn"
Start-Process -FilePath msixexec.exe -ArgumentList "$Argument" -Wait

#####
# uninstall success check #
#####

$UninstallResult = Get-installedcheck -AppSearchString $AppSearch

If ($UninstallResult -eq $true)
{
    Write-Host "uninstall is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = $Program_current.Version
        Uninstall = $false
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData
}

Else
{
    Write-Host "uninstall is successful" -BackgroundColor Green
}
```

```

$UninstallData = [PSCustomObject]@{
    Software = $SoftwareName
    Version = $Program_current.Version
    Uninstall = $true
} | ConvertTo-Json

Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
}

```

Detection Script

The detection script showing success of installation. It could be done as well without a script but again it is helpful for later automation of upload processes in the future.

The **yellow marked** version must be adjusted with each new version.

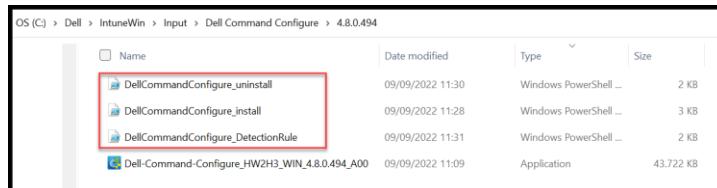
```

#####
# Program with target Version
#####
$ProgramVersion_target = '4.7.0.433' # need to be the same like the exe file
$ProgramVersion_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '%Dell%Configure%'" | Select-Object -ExpandProperty Version

if($ProgramVersion_current -eq $ProgramVersion_target)
{
    Write-Host "Found it!"
}

```

Please, copy these files in the same folder as your EXE Installer File.



Start the Microsoft Win32 Content Prep Tool (aka IntuneAppUtil.exe). In case you are not familiar with this tool, you will find documentation here: <https://docs.microsoft.com/en-us/mem/intune/apps/apps-win32-prepare>

| Argument | Value |
|---------------|---|
| Source Folder | folder where you have stored the unzipped MSI |
| Setup file | Main installer file like msi/exe/ps1, etc. |
| Output Folder | where you want to store the IntuneWin |

```

Please specify the source Folder: C:\Dell\IntuneWin\Input\Dell Command Configure\4.8.0.494
Please specify the setup file: Dell-Command-Configure_HW2H3_WIN_4.8.0.494_A00.exe
Please specify the output folder: C:\Dell\IntuneWin\Output\Dell Command Configure\4.8.0.494
Do you want to specify catalog folder (Y/N)?

```

IntuneWin is now prepared and ready for installation by Intune.



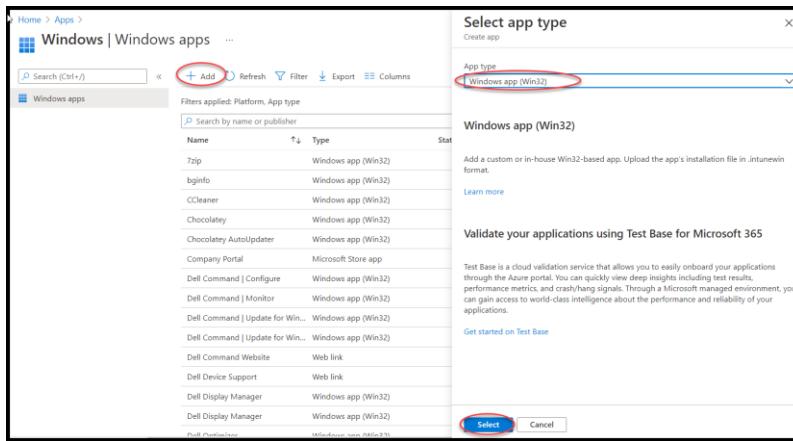
Import and Deployment settings Dell Command | Configure for Intune

Click 'Add'

| Field | Value |
|---------------|---------------------|
| Source Folder | Windows app (Win32) |

Click 'Select'

Select Windows app (Win32) as application.



Section 'App information'



Click 'Select app package file'

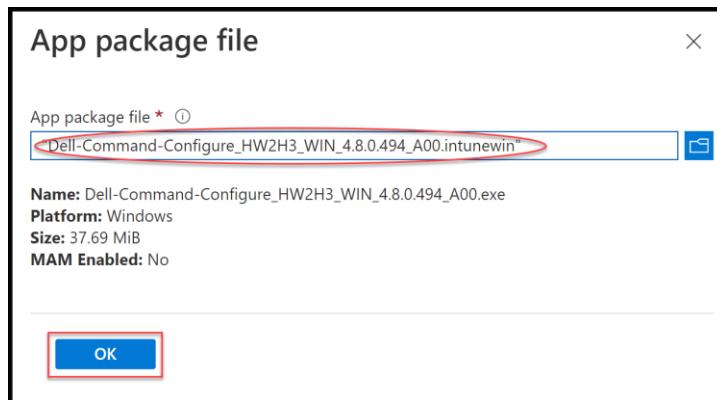
Click 'Folder'



Select 'Dell-Command-Configure_HW2H3_WIN_4.8.0.494_A00.intunewin'



Click 'OK'



| Field | Value |
|---|---|
| Name | Dell Command Configure |
| Publisher | Dell Inc. |
| App Version | 4.8.0.494 Note: Use version of the Dell Command Configure |
| Show this as a featured app in the Company Portal | Yes, but restricted to Device Admins Note: This App is for Admins only |

Click 'Next'

Section 'Program'

| Field | Value |
|-------------------|---|
| Install command | powershell.exe -executionpolicy bypass .\DellCommandConfigure_install.ps1 |
| Uninstall command | powershell.exe -executionpolicy bypass .\DellCommandConfigure_uninstall.ps1 |

Click 'Next'

Section 'Requirements'

| Field | Value |
|-------------------------------|--|
| Operating system architecture | 64-Bit |
| Minimum operating system | 2004 (Please note Dell support drivers and software only with the latest Win version + N -2) |

Click 'Next'

Home > Apps | Windows > Windows | Windows apps >
Add App ...
Windows app (Win32)
App information Program Requirements Detection rules Dependencies
Specify the requirements that devices must meet before the app is installed:
Operating system architecture * ① 64-bit
Minimum operating system * ① Windows 10 2004
Disk space required (MB) ①
Physical memory required (MB) ①
Minimum number of logical processors required ①
Minimum CPU speed required (MHz) ①
Previous Next

Section 'Detection rules'

Home > Apps > Windows >
Add App ...
Windows app (Win32)
App information Program Requirements Detection rules Dependencies Supersedeance (preview) Assignments Review + create

| Field | Value |
|--------------|------------------------|
| Rules format | Use a custom detection |

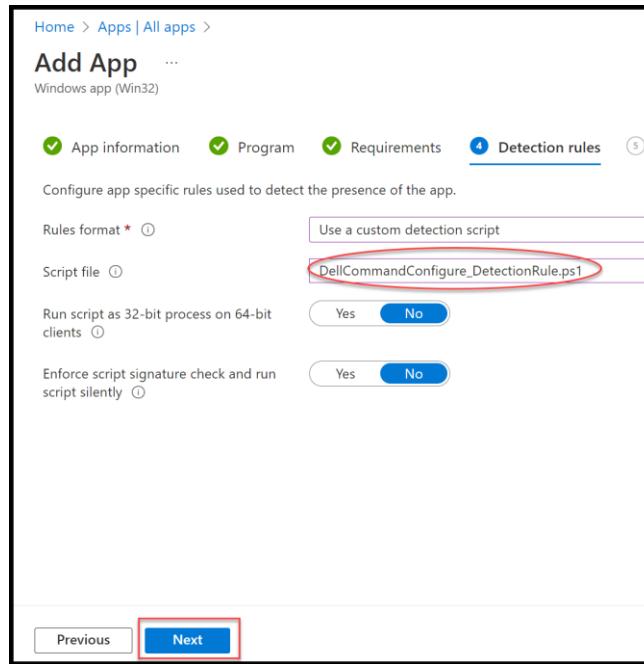
Click 'Folder'

Home > Apps | Windows > Windows | Windows apps >
Add App ...
Windows app (Win32)
App information Program Requirements Detection rules Dependencies
Configure app specific rules used to detect the presence of the app.
Rules format * ① Use a custom detection script
Script file ① Select a file
Run script as 32-bit process on 64-bit clients ① Yes No
Enforce script signature check and run script silently ① Yes No

Select 'DellCommandConfigure_DetectionRule.ps1'

OS (C) > Dell > IntuneWin > Input > Dell Command Configure > 4.8.0.494 > Search 4.8.0.494
Name Date modified Type Size
DellCommandConfigure_DetectionRule 09/09/2022 11:31 Windows PowerShell ... 2 KB
DellCommandConfigure_install 09/09/2022 11:28 Windows PowerShell ... 3 KB
DellCommandConfigure_uninstall 09/09/2022 11:30 Windows PowerShell ... 2 KB

Click 'Next'



Section 'Dependencies'



No changes

Section ‘Supersedence’

Home > Apps | All apps >

Add App ...

Windows app (Win32)

App information Program Requirements Detection rules Dependencies Supersedence (preview)

When you supersede an application, you can specify which app will be updated or replaced. To update an app, disable the uninstall previous version option. To replace an app, enable the uninstall previous version option. There is a maximum of 10 updated or replaced apps, including references to other apps. For example, your app references another app. This other app references other apps, and so on. This scenario creates a graph of apps. All apps in the graph count toward the maximum value of 10. [Learn more](#)

If you select 'Yes' please check
if the detection of old app does
not identify the new app

Apps that this app will supersede

| Name | Publisher | Version | Uninstall previous version |
|--------------------------|-----------|-----------|---|
| Dell Command Configure | Dell Inc. | 4.7.0.433 | <input type="radio"/> Yes <input checked="" type="radio"/> No |
| Dell Command Configure | Dell Inc. | 4.6.0.277 | <input type="radio"/> Yes <input checked="" type="radio"/> No |

+ Add (i)

Previous Next

No changes

Note: You can also uninstall old software via Supersede, but make sure that the detection of the old application does not also detect the new one, otherwise you will create an installation loop.

Section ‘Assignments’

Home > Apps > Windows >

Add App ...

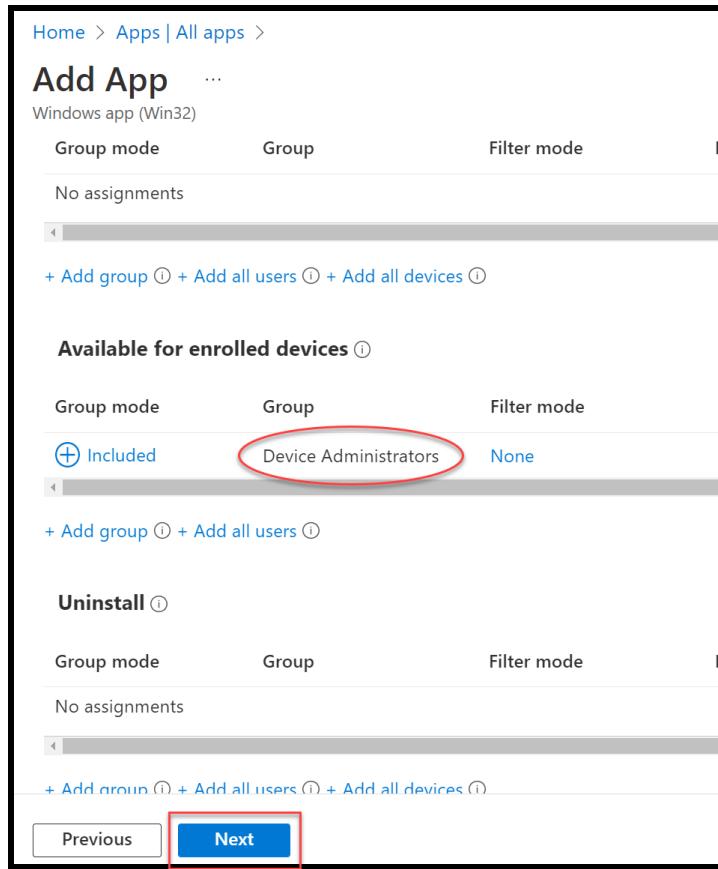
Windows app (Win32)

App information Program Requirements Detection rules Dependencies Supersidence (preview) Assignments Review + create

The Dell Command | Configure supports only Dell (Latitude, Optiplex, Precision and mobile XPS) it makes sense to have da dynamic group which only incl. these systems. Normally this Application is for Administrators only, we recommend using this Applications only on Administrator Devices if needed.

| Option | Value |
|--------------------------------|-----------------------------------|
| Required | |
| Available for enrolled devices | Add Group ‘Device Administrators’ |
| Uninstall | |

Click 'Next'



The app is now finished

Click 'Create'

Home > Apps | All apps >

Add App ...

Windows app (Win32)

Program Requirements Detection rules Dependencies Superseded

Summary

App information

| | |
|---|--|
| App package file | Dell-Command-Configure_HW2H3_WIN_4.8.0.494_A00.intunewin |
| Name | Dell Command Configure |
| Description | Dell-Command-Configure_HW2H3_WIN_4.8.0.494_A00.exe |
| Publisher | Dell Inc. |
| App Version | 4.8.0.494 |
| Category | -- |
| Show this as a featured app in the Company Portal | Yes |
| Information URL | -- |
| Privacy URL | -- |

Previous Create

Ready to work.

| Name | Type | Status | Version | Assigned |
|--------------------------|---------------------|-----------|---------|----------|
| Dell Command Configure | Windows app (Win32) | 4.7.0.433 | Yes | |
| Dell Command Configure | Windows app (Win32) | 4.6.0.277 | No | |
| Dell Command Configure | Windows app (Win32) | 4.8.0.494 | Yes | |

Dell Display Manager

Introduction

Dell Display Manager enhances everyday productivity through comprehensive management tools giving you best front of screen experience, efficient display management and easy, effortless multitasking.

With an improved Dell Display Manager, the ease of access and usability is further enhanced for the user. IT Managers will now be able to manage and control monitors remotely, improving overall productivity.

Knowledge Base Article: What is Dell Display Manager?

<https://www.dell.com/support/kbdoc/de-de/000060112/what-is-dell-display-manager?lang=en>

Prepare Dell Display Manager for Intune

Please check in advance which version or versions you need as the two Dell Display Manager versions V1.x and V2.x support different monitor models.

<https://www.dell.com/support/kbdoc/de-ch/000060112/what-is-dell-display-manager?lang=en#models>

Dell Displays (Monitor) Models supported by Dell Display Manager

Dell Displays (Monitor) Models supported by Dell Display Manager (Windows Operating System)

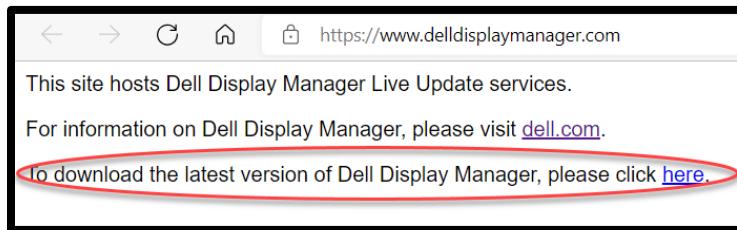
Dell Display Manager 2.0 requires Windows 11 or Windows 10 build 17763 or newer.

| MODEL | SUPPORTS DDM 2.0 | SUPPORTS DDM 1.X ONLY |
|---------------------|--|-----------------------|
| Alienware Monitors | AW2521HF AW2511FA AW2511FL AW2511FLA AW2523HF AW2720HF AW2720HFA AW2723DF AW5520QF | AW2518HF |
| Dell Gaming Monitor | G2422HS G2722HS G2723H | |

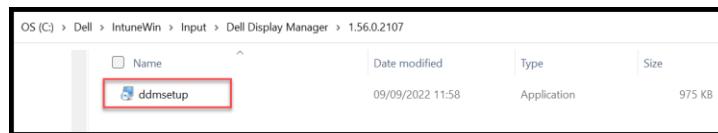
Download Dell Display Manager V1.x

Download the newest Version of Dell Display Manager from our website.

<https://www.delldisplaymanager.com/>



If you have downloaded the file from <https://www.delldisplaymanager.com/>, copy file to your software repository for the next step.



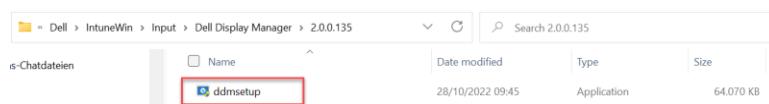
Download Dell Display Manager V2.x

Download the newest Version of Dell Display Manager from our website.

<https://www.dell.com/support/home/en-us/product-support/product/dell-display-peripheral-manager/drivers>

A screenshot of the Dell Support website's driver download section. The top navigation bar shows 'Overview', 'Drivers & Downloads' (which is active), and 'Documentation'. Below, there's a search bar for 'Find a driver for your Dell Display and Peripheral Manager' with filters for 'Operating system' (Windows 10, 64-bit), 'Download Type' (All), and 'Category' (All). A note says it's a comprehensive list of available downloads. The main table lists two items: 'Dell Peripheral Manager' and 'Dell Display Manager Application'. The 'Dell Display Manager Application' row is highlighted with a red box. The table has columns for Name, Importance, Category, Release Date, and Action (with a 'Download' button).

If you have downloaded the file from <https://www.delldisplaymanager.com/>, copy file to your software repository for the next step.



Scripts for Install, Uninstall and Detection

It is possible to use the native file for installation. In this document we are using PowerShell scripts to cover different scenarios of Install new, Update, and uninstall for a later automation of uploading applications by API.

All script could be download on Github Repository:

<https://github.com/svenriebedell/Dell-Tools-Intune-Install>

Script set for Dell Display Manager V1.x

Install Script

The script makes a precheck if older versions are installed and starting an uninstall first to have a clean environment.

```
#####
##### Function section #####
#####

function Get-installedcheck
{
    param
    (
        [Parameter(mandatory=$true)][string] $AppSearchString
    )

    $AppCheck = Get-ChildItem -Path HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall | Get-ItemProperty | Where-Object {$_.DisplayName -match "$AppSearchString" }

    If ($null -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        return $false
    }
}

#####
##### variable section #####
#####

$InstallerName = Get-ChildItem .*.exe | Select-Object -ExpandProperty Name
$ProgramPath = ".\" + $InstallerName
[Version]$ProgramVersion_target = (Get-Command $ProgramPath).FileVersionInfo.ProductVersion
$AppSearch = "Dell Display Manager" #Parameter to search in registry
$Program_current = Get-ChildItem -Path HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall | Get-ItemProperty | Where-Object {$_.DisplayName -match "$AppSearch" }
[Version]$ProgramVersion_current = $Program_current.DisplayVersion
$ApplicationID_current = $Program_current.QuietUninstallString
$SoftwareName = $Program_current.DisplayName + " 1"

#####
##### program section #####
#####

#### generate Logging Resources
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

#####
#Checking if older Version is installed and uninstall this Version #
#####

If ($null -ne $ProgramVersion_current)
{
    if ($ProgramVersion_target -gt $ProgramVersion_current)
    {
        $IDProcess = Get-Process | Where-Object {$_.ProcessName -like 'ddm'} | Select-Object -ExpandProperty ID
        Stop-Process -Id $IDProcess -Force
        Start-Process cmd.exe -ArgumentList '/c',$ApplicationID_current -Wait

        Start-Sleep -Seconds 10
    }
    # uninstall success check #
    $UninstallResult = Get-installedcheck -AppSearchString $AppSearch
```

```

If ($UninstallResult -eq $true)
{
    Write-Host "uninstall is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = $Program_current.DisplayVersion
        Uninstall = $false
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData
}

Else
{
    Write-Host "uninstall is successful" -BackgroundColor Green

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = $Program_current.DisplayVersion
        Uninstall = $true
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
}

}
Else
{
    Write-Host "same version is installed"

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = $($ProgramVersion_target).ToString()
        Install = $false
        Reason = "same version is installed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData

    Exit 0
}

#####
#Install new Software
#####

Start-Process -FilePath ".\ddmsetup.exe" -ArgumentList '/verysilent /noupdate'
Start-Sleep -Seconds 15
Start-Process -FilePath "C:\Program Files (x86)\Dell\Dell Display Manager\ddm.exe"

#####
# install success check
#####
$Program_current = Get-ChildItem -Path HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall | Get-ItemProperty |
Where-Object {$_.DisplayName -match "$AppSearch" }
$SoftwareName = $Program_current.DisplayName + " 1"
$UninstallResult = Get-installedcheck -AppSearchString $AppSearch

If ($UninstallResult -ne $true)
{
    Write-Host "install is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = $($ProgramVersion_target).ToString()
        Install = $false
    } | ConvertTo-Json
}

```

```

        Reason = "Installation failed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Error -EventId 11 -Message $UninstallData

}

Else
{
    [Version]$ProgramVersion_current = Get-ChildItem -Path HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall | Get-ItemProperty | Where-Object {$_.DisplayName -match "Dell Display Manager"} | Select-Object -ExpandProperty DisplayName

    If ($ProgramVersion_current -ge $ProgramVersion_target)
    {

        Write-Host "install is successful" -BackgroundColor Green

        $UninstallData = [PSCustomObject]@{
            Software = $SoftwareName
            Version = ($ProgramVersion_target).ToString()
            Install = $true
            Reason = "Update/Install/Newer Version"
        } | ConvertTo-Json

        Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData

    }
    Else
    {

        Write-Host "install is unsuccessful" -BackgroundColor red

        $UninstallData = [PSCustomObject]@{
            Software = $SoftwareName
            Version = ($ProgramVersion_target).ToString()
            Install = $false
            Reason = "Older Version installed $ProgramVersion_current"
        } | ConvertTo-Json

        Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData

    }
}

```

Uninstall Script

The script starts the uninstalling of application.

```
#####
##### Function section #####
#####

function Get-installedcheck
{
    param
    (
        [Parameter(mandatory=$true)][string] $AppSearchString
    )

    $AppCheck = Get-ChildItem -Path HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall | Get-ItemProperty | Where-Object {$_.DisplayName -match "$AppSearchString" }

    If ($null -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        return $false
    }
}

#####
##### variable section #####
#####

$AppSearch = "Dell Display Manager" #Parameter to search in registry
$Program_current = Get-ChildItem -Path HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall | Get-ItemProperty | Where-Object {$_.DisplayName -match "$AppSearch" }
$SoftwareName = $Program_current.DisplayName
$ApplicationID_current = $Program_current.QuietUninstallString

#####
##### program section #####
#####

##### generate Logging Resources
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

#####
#uninstall Software #
#####

$IDProcess = Get-Process | Where-Object {$__.ProcessName -like 'ddm'} | Select-Object -ExpandProperty ID
Stop-Process -Id $IDProcess -Force

Start-Process cmd.exe -ArgumentList '/c',$ApplicationID_current -Wait

#####
# uninstall success check #
#####

$UninstallResult = Get-installedcheck -AppSearchString $AppSearch

If ($UninstallResult -eq $true)
{
    Write-Host "uninstall is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = $Program_current.DisplayVersion
        Uninstall = $false
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData
}
```

```

Else
{
    Write-Host "uninstall is successful" -BackgroundColor Green

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = $Program_current.DisplayVersion
        Uninstall = $true
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
}

```

Detection Script

The detection script showing success of installation. It could be done as well without a script but again it is helpful for later automation of upload processes in the future.

The **yellow marked** version must be adjusted with each new version.

```

#####
# Program with target Version
#####
$ProgramPath = "C:\Program Files (x86)\Dell\Display Manager\ddm.exe"
$ProgramVersion_target = '1.56.0.2107' # need to be the same like the exe file
$ProgramVersion_current = [System.Diagnostics.FileVersionInfo]::GetVersionInfo($ProgramPath).FileVersion

if($ProgramVersion_current -eq $ProgramVersion_target)
{
    Write-Host "Found it!"
}

```

Script set for Dell Display Manager V2.x

Install Script

The script makes a precheck if older versions are installed and starting an uninstall first to have a clean environment.

```
#####
##### Function section #####
#####

function Get-installedcheck
{
    param
    (
    )

    $AppCheck = Test-Path -Path 'C:\Program Files\Del\lDell Display Manager 2\ddm.exe' #cover new folder
    If ($false -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        $AppCheck = Test-Path -Path 'C:\Program Files\Del\lDell Display Manager 2.0\ddm.exe' #cover old folder
        Write-Host "Old DDM Folder used by installation"
    }

    If ($false -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        return $false
    }
}

#####
##### variable section #####
#####

$InstallerName = Get-ChildItem *.exe | Select-Object -ExpandProperty Name
$ProgramPath = ".\" + $InstallerName
[Version]$ProgramVersion_target = (Get-Command $ProgramPath).FileVersionInfo.ProductVersion
$AppSearch = "ddm.exe" #Parameter to search in registry
$SoftwareName = "Dell Display Manager 2"

#####
##### program section #####
#####

##### generate Logging Resources
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

#####
##### Checking if older Version is installed and uninstall this Version #
#####

If ((Get-installedcheck) -eq $true)
{
    # get version of existing installation#
    $CheckFolder = Test-Path -Path 'C:\Program Files\Del\lDell Display Manager 2\DDM.exe'

    if ($false -eq $CheckFolder)
    {
        # get version of existing installation based old folder
```

```

[Version]$ProgramVersion_current = (Get-ItemProperty -Path 'C:\Program Files\DELL\DELL Display Manager 2.0\DDM.exe').VersionInfo |
Select-Object -ExpandProperty ProductVersion
$ApplicationID_current = "C:\Program Files\DELL\DELL Display Manager 2.0\Uninst.exe"

}
Else
{

    # get version of existing installation based new folder
    [Version]$ProgramVersion_current = (Get-ItemProperty -Path 'C:\Program Files\DELL\DELL Display Manager 2\DDM.exe').VersionInfo | Select-
Object -ExpandProperty ProductVersion
$ApplicationID_current = "C:\Program Files\DELL\DELL Display Manager 2\Uninst.exe"

}

if ($ProgramVersion_target -gt $ProgramVersion_current)
{
    $IDProcess = Get-Process | Where-Object {$_.ProcessName -ceq 'DDM'} | Select-Object -ExpandProperty ID
    if ($null -ne $IDProcess)
    {
        Stop-Process -Id $IDProcess -Force
    }

    Start-Process -FilePath $ApplicationID_current -ArgumentList "/S" -Wait
    Start-Sleep -Seconds 10

#####
# uninstall success check
#####
$UninstallResult = Get-installedcheck -AppSearchString $AppSearch

If ($UninstallResult -eq $true)
{
    Write-Host "uninstall is unsuccessful" -BackgroundColor Red
    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_current).ToString()
        Uninstall = $false
    } | ConvertTo-Json
    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData
}

Else
{
    Write-Host "uninstall is successful" -BackgroundColor Green
    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_current).ToString()
        Uninstall = $true
    } | ConvertTo-Json
    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
}

Else
{
    Write-Host "same version is installed"
    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $false
    }
}

```

```

        Reason = "same version is installed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData

    Exit 0
}
}

Else
{
    Write-Host "No Dell Display Manager 2 was installed"
}

#####
#Install new Software
#####
Start-Process -FilePath $ProgramPath -ArgumentList '/verysilent /NotifyUpdate="disable"'
Start-Sleep -Seconds 10

#####
# install success check
#####
$UninstallResult = Get-installedcheck

If ($UninstallResult -ne $true)
{
    Write-Host "install is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $false
        Reason = "Installation failed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Error -EventId 11 -Message $UninstallData
}

Else
{
    [Version]$ProgramVersion_current = (Get-ItemProperty -Path 'C:\Program Files\Del\lDell Display Manager 2\DDM.exe').VersionInfo | Select-Object -ExpandProperty ProductVersion

    If ($ProgramVersion_current -ge $ProgramVersion_target)
    {
        Write-Host "install is successful" -BackgroundColor Green

        $UninstallData = [PSCustomObject]@{
            Software = $SoftwareName
            Version = ($ProgramVersion_target).ToString()
            Install = $true
            Reason = "Update/Install/Newer Version"
        } | ConvertTo-Json

        Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData
    }

    Else
    {
        Write-Host "install is unsuccessful" -BackgroundColor red

        $UninstallData = [PSCustomObject]@{
            Software = $SoftwareName
            Version = ($ProgramVersion_target).ToString()
            Install = $false
            Reason = "Older Version installed $ProgramVersion_current"
        } | ConvertTo-Json

        Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData
    }
}
}

```

```
    }
}
```

Uninstall Script

The script starts the uninstalling of application.

```
#####
#### Function section #####
#####

function Get-installedcheck
{
    param
    (
        )
    $AppCheck = Test-Path -Path 'C:\Program Files\Dell\Display Manager 2\ddm.exe' #cover new folder
    If ($false -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        $AppCheck = Test-Path -Path 'C:\Program Files\Dell\Display Manager 2.0\ddm.exe' #cover old folder
        Write-Host "Old DDM Folder used by installation"
    }

    If ($false -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        return $false
    }
}

#####
#### variable section #####
#####

$AppSearch = "ddm.exe" #Parameter to search in registry
$SoftwareName = "Dell Display Manager 2"

#####
#### program section #####
#####

#### generate Logging Resources
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

# get version of existing installation#
$CheckFolder = Test-Path -Path 'C:\Program Files\Dell\Display Manager 2\DDM.exe'

if ($false -eq $CheckFolder)
{
    # get version of existing installation based old folder
    [Version]$ProgramVersion_current = (Get-ItemProperty -Path 'C:\Program Files\Dell\Display Manager 2.0\ddm.exe').VersionInfo | Select-Object -ExpandProperty ProductVersion
    $ApplicationID_current = "C:\Program Files\Dell\Display Manager 2.0\Uninst.exe"
}
Else
```

```

{
    # get version of existing installation based new folder
    [Version]$ProgramVersion_current = (Get-ItemProperty -Path 'C:\Program Files\Del\Del Display Manager 2\DDM.exe').VersionInfo | Select-Object -ExpandProperty ProductVersion
    $ApplicationID_current = "C:\Program Files\Del\Del Display Manager 2\Uninst.exe"

}

$IDProcess = Get-Process | Where-Object {$_.ProcessName -ceq 'DDM'} | Select-Object -ExpandProperty ID

if ($null -ne $IDProcess)
{
    Stop-Process -Id $IDProcess -Force
}

Start-Process -FilePath $ApplicationID_current -ArgumentList "/S" -Wait
Start-Sleep -Seconds 10

#####
# uninstall success check      #
#####
$UninstallResult = Get-installedcheck -AppSearchString $AppSearch

If ($UninstallResult -eq $true)
{
    Write-Host "uninstall is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_current).ToString()
        Uninstall = $false
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData
}

Else
{
    Write-Host "uninstall is successful" -BackgroundColor Green

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_current).ToString()
        Uninstall = $true
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
}

```

Detection Script

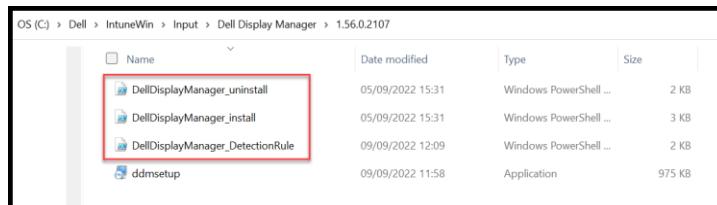
The detection script showing success of installation. It could be done as well without a script but again it is helpful for later automation of upload processes in the future.

The yellow marked version must be adjusted with each new version.

```
#####
# Program with target Version
#####
$ProgramPath = "C:\Program Files\DELL\DELL Display Manager 2\DDM.exe"
$ProgramVersion_target = [2.1.1.17]
$ProgramVersion_current = [System.Diagnostics.FileVersionInfo]::GetVersionInfo($ProgramPath).FileVersion

if($ProgramVersion_current -ge $ProgramVersion_target)
{
    Write-Host "Found it!"
}
```

Please, copy these files in the same folder as your EXE Installer File.

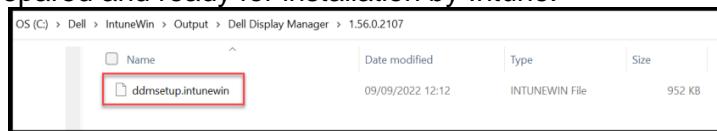


Start the Microsoft Win32 Content Prep Tool (aka IntuneAppUtil.exe). In case you are not familiar with this tool, you will find documentation here: <https://docs.microsoft.com/en-us/mem/intune/apps/apps-win32-prepare>

| Argument | Value |
|---------------|---|
| Source Folder | folder where you have stored the unzipped MSI |
| Setup file | Main installer file like msi/exe/ps1, etc. |
| Output Folder | where you want to store the IntuneWin |

```
Please specify the source folder: C:\Dell\IntuneWin\Input\DELL Display Manager\1.56.0.2107
Please specify the setup file: ddmsetup.exe
Please specify the output folder: C:\Dell\IntuneWin\Output\DELL Display Manager\1.56.0.2107
Do you want to specify catalog folder (Y/N)?n
```

IntuneWin is now prepared and ready for installation by Intune.



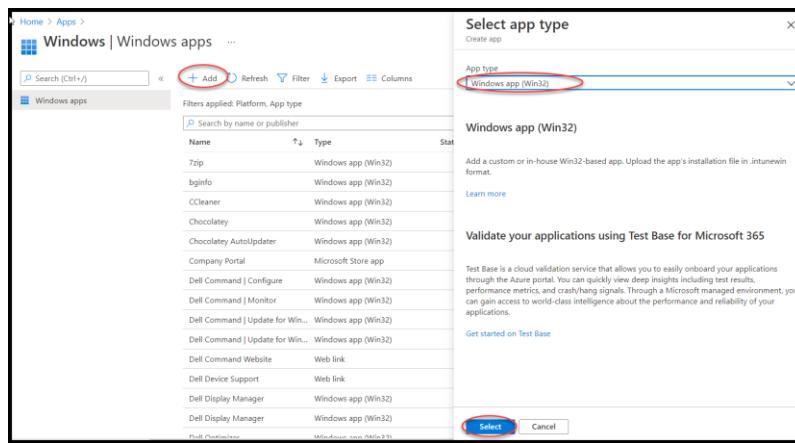
Import and Deployment settings Dell Display Manager for Intune

Click 'Add'

| Field | Value |
|---------------|---------------------|
| Source Folder | Windows app (Win32) |

Click 'Select'

Select Windows app (Win32) as application.



Section 'App information'

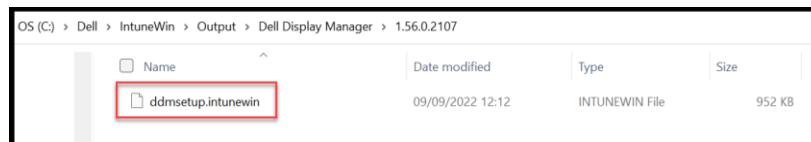


Click 'Select app package file'

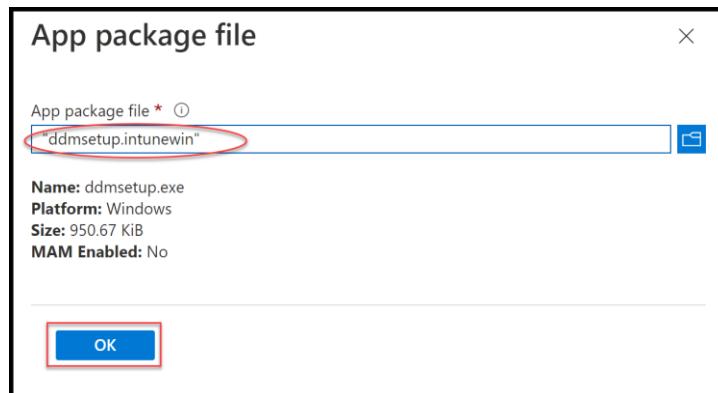
Click 'Folder'



Select 'ddmsetup.intunewin'



Click 'OK'



| Field | Value |
|---|--|
| Name | Dell Display Manager |
| Publisher | Dell Inc. |
| App Version | 1.56.0.2107 |
| | Note: Use version of the Dell Display Manager |
| Show this as a featured app in the Company Portal | Yes |

Click 'Next'

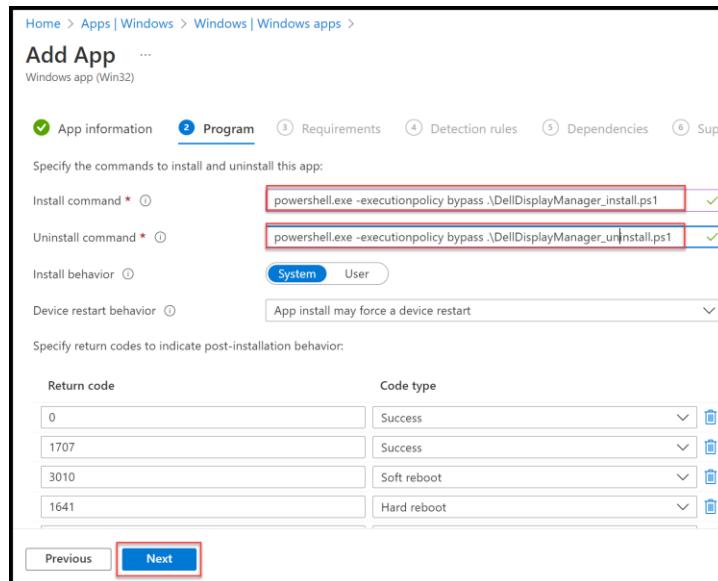
The screenshot shows an 'Add App' form. At the top, it says 'Home > Apps | Windows > Windows | Windows apps > Add App'. The form has several input fields: 'Name *' (Dell Display Manager), 'Description *' (ddmsetup.exe), 'Publisher *' (Dell Inc.), 'App Version' (1.56.0.2107), and a 'Category' section with '0 selected'. Below these, there is a checkbox for 'Show this as a featured app in the Company Portal' with options 'Yes' and 'No' (Yes is selected). At the bottom, there is an 'Information URL' field with the placeholder 'Enter a valid url' and a 'Next' button, which is highlighted with a red box.

Section 'Program'



| Field | Value |
|-------------------|---|
| Install command | powershell.exe -executionpolicy bypass .\\DellDisplayManager_install.ps1 |
| Uninstall command | powershell.exe -executionpolicy bypass .\\DellDisplayManager_uninstall.ps1 |

Click 'Next'



Section 'Requirements'

The screenshot shows a software interface for adding an application. At the top, there's a breadcrumb navigation: Home > Apps > Windows >. Below it, the title 'Add App' and a subtitle 'Windows app (Win32)'. A horizontal navigation bar at the top right includes tabs: App information (green checkmark), Program (green checkmark), Requirements (blue circle with a white dot, highlighted with a red oval), Detection rules (grey), Dependencies (grey), Superseding (grey), Assignments (grey), and Review + create (grey). The 'Requirements' tab is currently active.

| Field | Value |
|-------------------------------|--|
| Operating system architecture | 32/64-Bit |
| Minimum operating system | 2004 (Please note Dell support drivers and software only with the latest Win version + N -2) |

Click 'Next'

The screenshot shows the 'Add App' interface on the 'Requirements' tab. The URL in the address bar is 'Home > Apps | Windows > Windows | Windows apps >'. The title 'Add App' and subtitle 'Windows app (Win32)' are present. The 'Requirements' tab is active, indicated by a blue circle with a white dot and a red oval around it. Below the tabs, a message says 'Specify the requirements that devices must meet before the app is installed:'. There are several input fields:

- 'Operating system architecture *' with a note '(1)' and a dropdown menu showing '2 selected'.
- 'Minimum operating system *' with a note '(1)' and a dropdown menu showing 'Windows 10 2004'.
- 'Disk space required (MB)' with a note '(1)' and an empty input field.
- 'Physical memory required (MB)' with a note '(1)' and an empty input field.
- 'Minimum number of logical processors required' with a note '(1)' and an empty input field.
- 'Minimum CPU speed required (MHz)' with a note '(1)' and an empty input field.

Below these fields is a section titled 'Configure additional requirement rules' with a table:

| Type | Path/Script |
|--------------------------------|-------------|
| No requirements are specified. | |

At the bottom, there are 'Previous' and 'Next' buttons. The 'Next' button is highlighted with a red rectangle.

Section 'Detection rules'

The screenshot shows the 'Add App' interface with the 'Detection rules' tab highlighted. Other tabs like 'App information', 'Program', 'Requirements', 'Dependencies', 'Supersedeance (preview)', 'Assignments', and 'Review + create' are also visible.

| Field | Value |
|--------------|------------------------|
| Rules format | Use a custom detection |

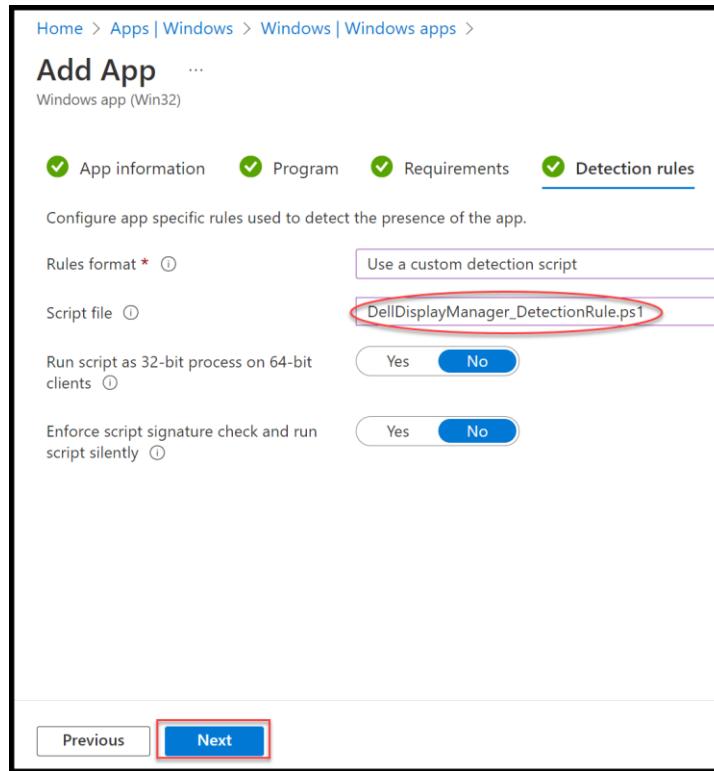
Click 'Folder'

The screenshot shows the 'Add App' configuration page under the 'Detection rules' tab. It includes fields for 'Rules format' (set to 'Use a custom detection script'), 'Script file' (with a 'Select a file' button), and options for running the script as 32-bit or 64-bit.

Select 'DellDisplayManager_DetectionRule.ps1'

The screenshot shows a file explorer window displaying files in the path 'OS (C:) > Dell > IntuneWin > Input > Dell Display Manager > 1.56.0.2107'. The file 'DellDisplayManager_DetectionRule.ps1' is highlighted with a red box.

Click 'Next'



Section 'Dependencies'



No changes

Section 'Supersedence'

The screenshot shows the 'Add App' interface with the 'Sup' tab selected. A note at the top right says: "If you select 'Yes', please check if the detection of old app does not identify the new app". Below is a table of existing apps:

| Name | Publisher | Version | Uninstall previous version |
|----------------------|-----------|-------------|---|
| Dell Display Manager | Dell Inc. | 1.54.0.2068 | <input type="radio"/> Yes <input checked="" type="radio"/> No |
| Dell Display Manager | Dell Inc. | 1.53.0.2065 | <input type="radio"/> Yes <input checked="" type="radio"/> No |

Buttons at the bottom include 'Previous' and 'Next' (highlighted with a red box).

No changes

Note: You can also uninstall old software via Supersede, but make sure that the detection of the old application does not also detect the new one, otherwise you will create an installation loop.

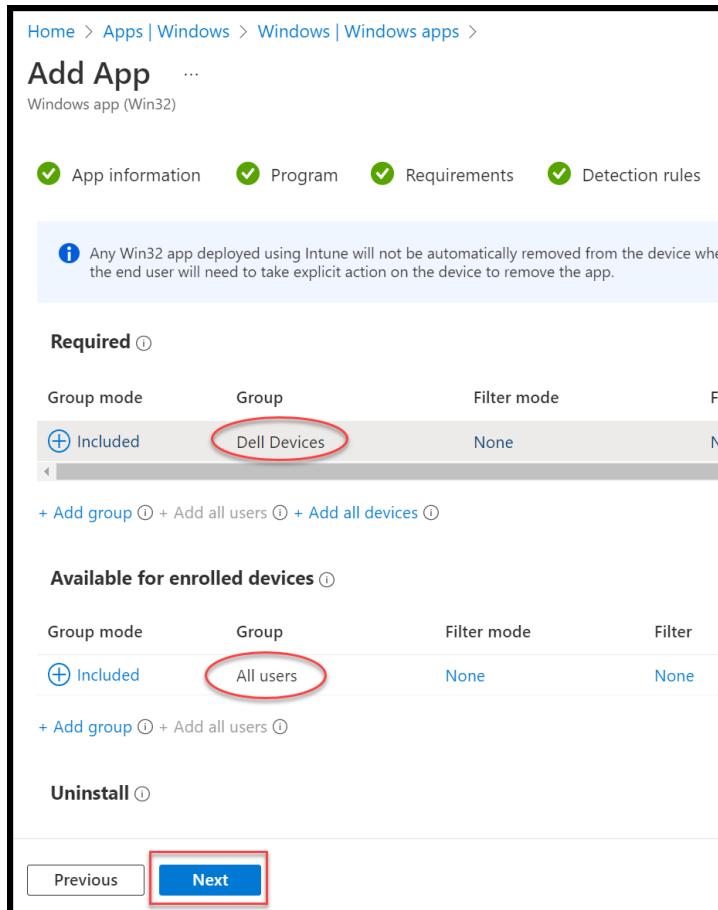
Section 'Assignments'



The Dell Display Manager supports only Dell Displays (Latitude, Optiplex, Precision and mobile XPS) it makes sense to have a dynamic group which only incl. these systems who have connect to a Dell Display.

| Option | Value |
|--------------------------------|-------------------------|
| Required | Add Group 'Dell Device' |
| Available for enrolled devices | Add Group 'All User' |
| Uninstall | |

Click 'Next'



The app is now finished

Click 'Create'

Home > Apps | Windows > Windows | Windows apps >

Add App

Windows app (Win32)

Program Requirements Detection rules

Summary

App information

| | |
|---|----------------------|
| App package file | ddmsetup.intunewin |
| Name | Dell Display Manager |
| Description | ddmsetup.exe |
| Publisher | Dell Inc. |
| App Version | 1.56.0.2107 |
| Category | -- |
| Show this as a featured app in the Company Portal | Yes |
| Information URL | -- |
| Privacy URL | -- |

Previous Create

Ready to work.

| Name | Type | Status | Version | Assigned |
|----------------------|---------------------|--------|-------------|----------|
| Dell Display Manager | Windows app (Win32) | | 1.53.0.2065 | No |
| Dell Display Manager | Windows app (Win32) | | 1.56.0.2107 | Yes |
| Dell Display Manager | Windows app (Win32) | | 1.54.0.2068 | No |

Dell SupportAssist for Business PCs

Introduction

SupportAssist is a proactive and predictive technology that provides automated technical support for your Dell PCs. It enables IT administrators to manage their PC fleet from TechDirect anytime, anywhere.

When deployed, SupportAssist monitors each PC and proactively detects both hardware and software issues. Depending on your service plan, when an issue is detected, SupportAssist automatically opens a support case with technical support and sends you an email notification.

SupportAssist collects and sends the required information securely to Dell technical support. The collected information enables Dell to provide you with an enhanced, efficient, and accelerated support experience.

SupportAssist enables you to optimize your PC by removing unwanted files, optimizing network settings, tuning-up system performance, and removing viruses and malware. It also identifies driver updates available for your PC.

SupportAssist also collects telemetry, application experience, health, and security data proactively from your PCs and provides various performance insights about your PCs, based on your service plan.

After you have deployed SupportAssist on your PCs, you can manage the PC fleet using the Connect and manage service in TechDirect.

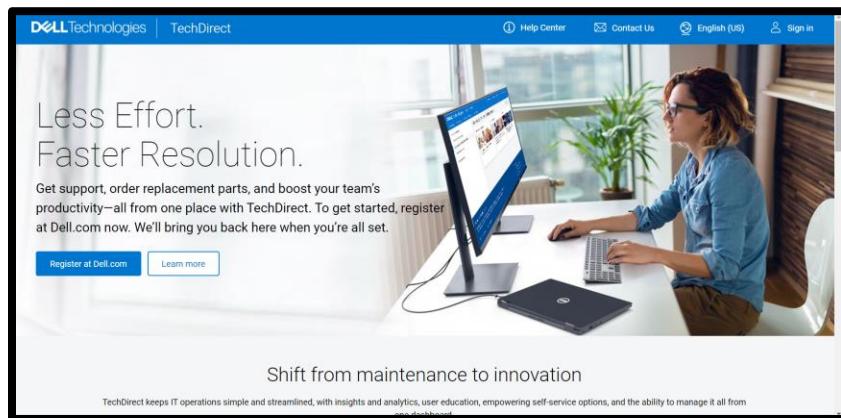
SupportAssist for Business PCs with Windows OS Administrator Guide

<https://www.dell.com/support/home/en-us/product-support/product/supportassist-business-pcs/docs>

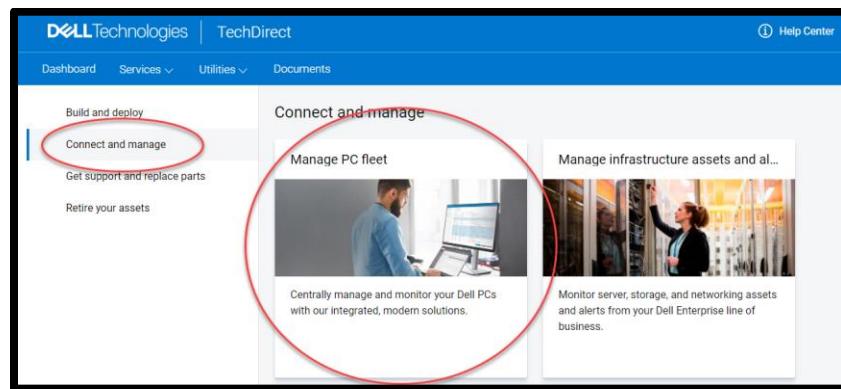
Prepare Dell SupportAssist for Business PCs for Intune

Download the newest Version of Dell SupportAssist for Business PCs from our website.

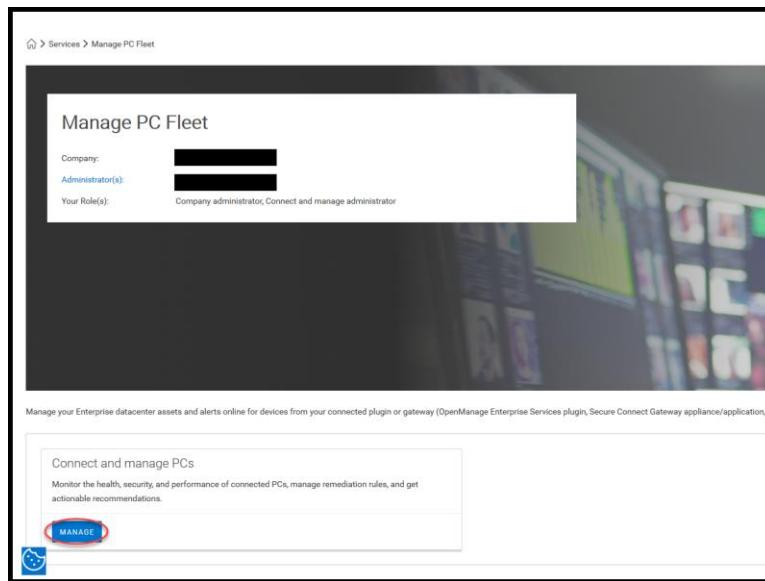
This requires a Dell TechDirect Account <https://techdirect.dell.com/>



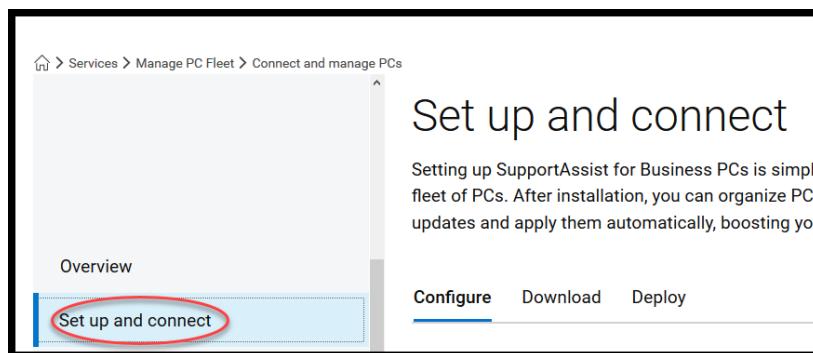
After Login move to Connect and Manage section and open Manage PC fleet.



Click 'Manage'

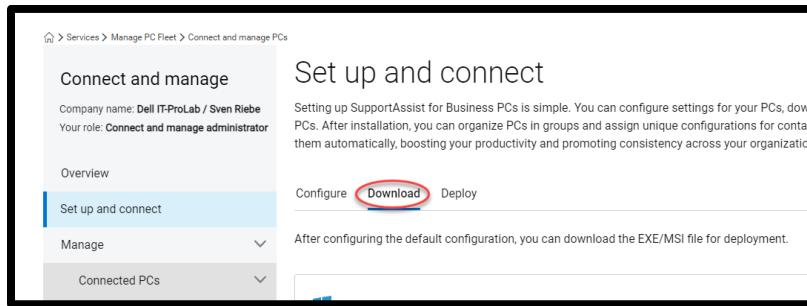


Click on Set up and connect



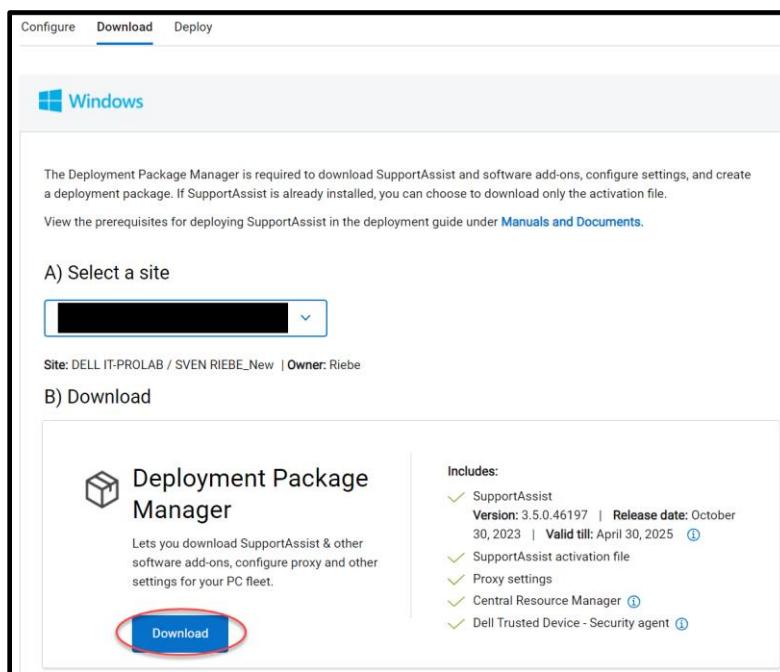
Configure Dell SupportAssist for Business first - please refer to the support documentation.
<https://www.dell.com/support/home/product-support/product/supportassist-business-pcs/docs>

After you have fulfilled the configure section, you can download the software.



Download the Deployment Package Manager now.

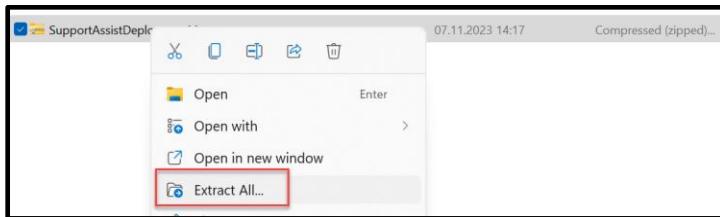
Click 'Download'



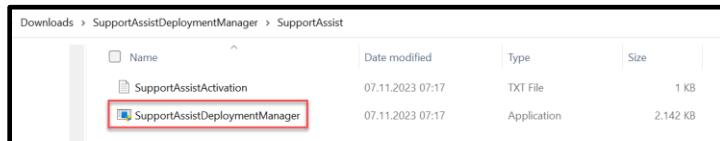
Go to your download folder and check for this file: SupportAssistDeploymentManager.zip



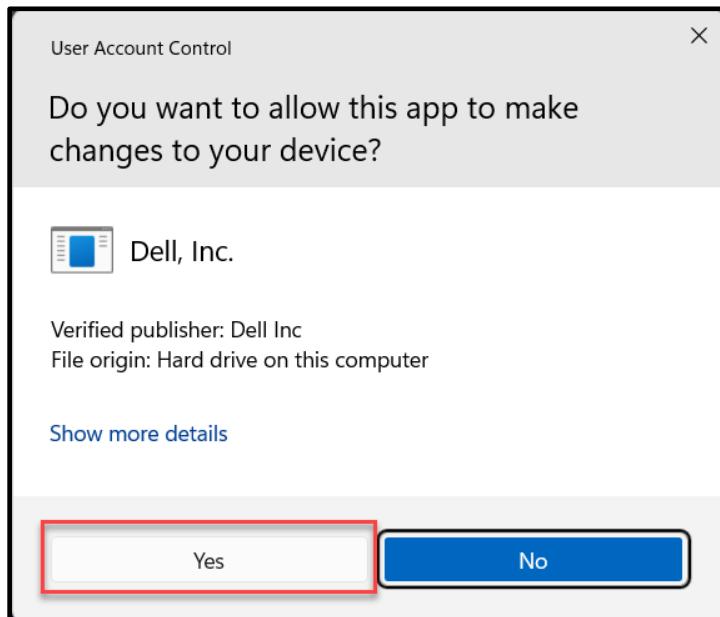
Extracting the file SupportAssistDeploymentManager.zip



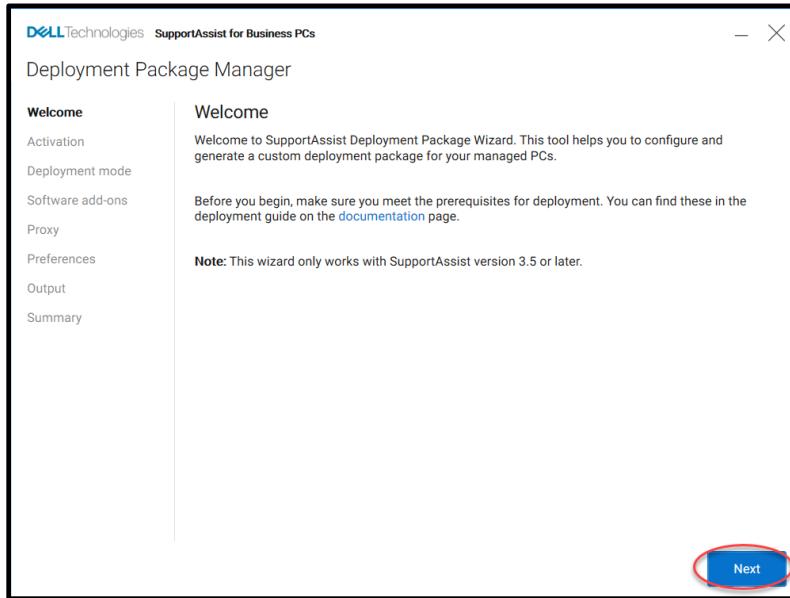
Extract the file and click this file SupportAssistDeploymentManager.exe (need administrator rights)



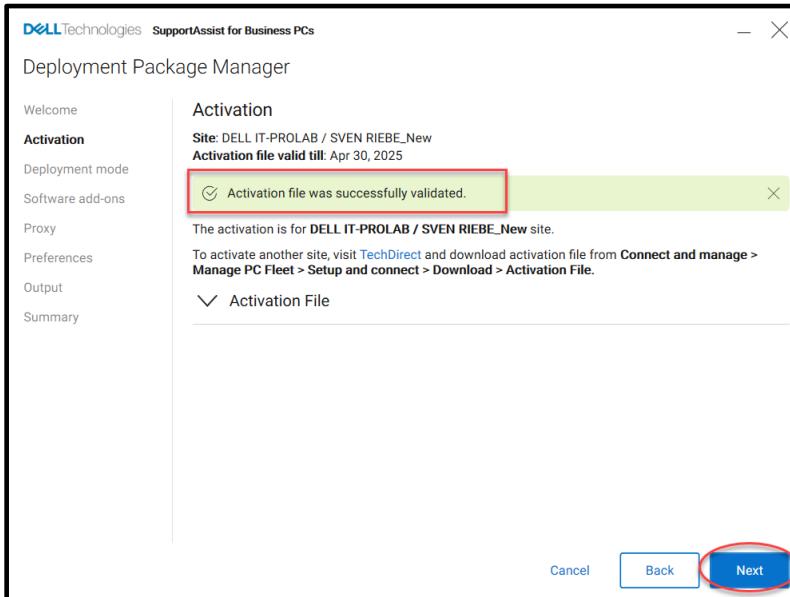
Click 'Yes'



Prepare the MSI and MST for the deployment.
Click 'Next'



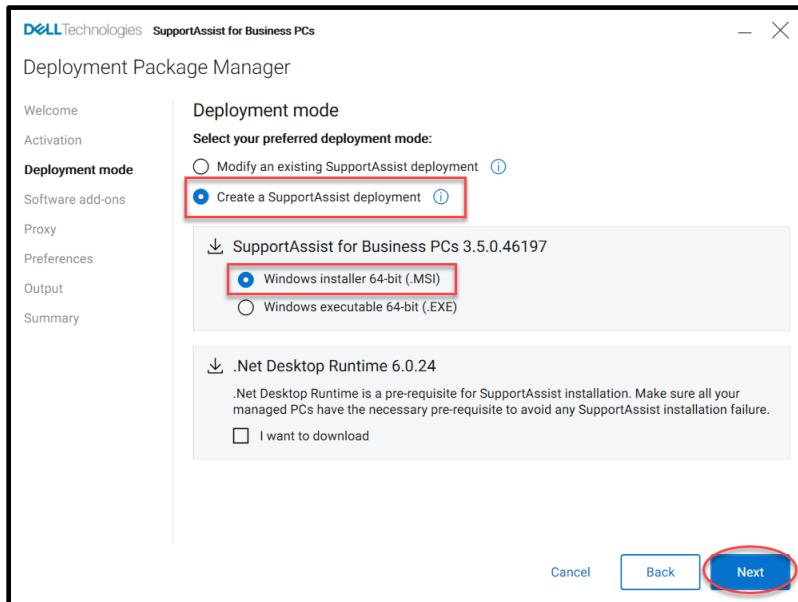
Check if Activation file is valid.
Click 'Next'



Select 'Create a SupportAssist deployment'

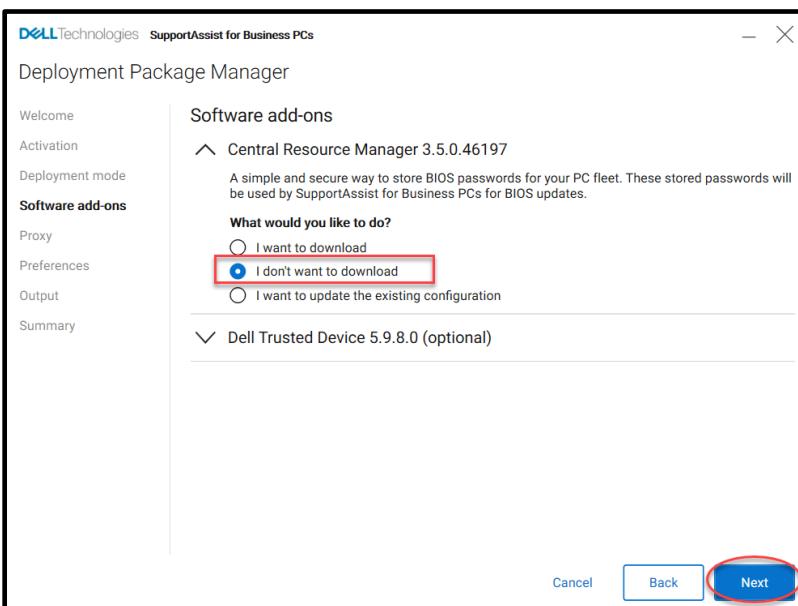
Select 'Windows installer 64-Bit (.MSI)'

Click 'Next'

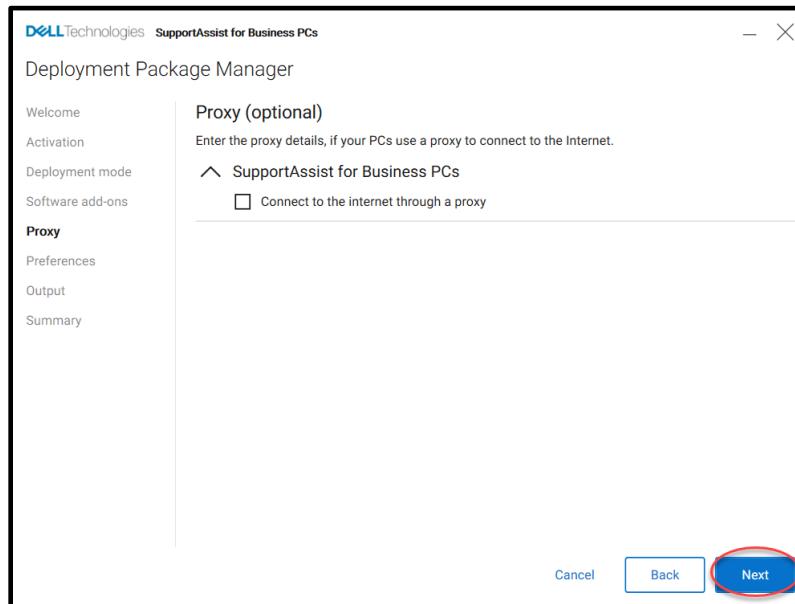


Select 'I don't want to download'

Click 'Next'

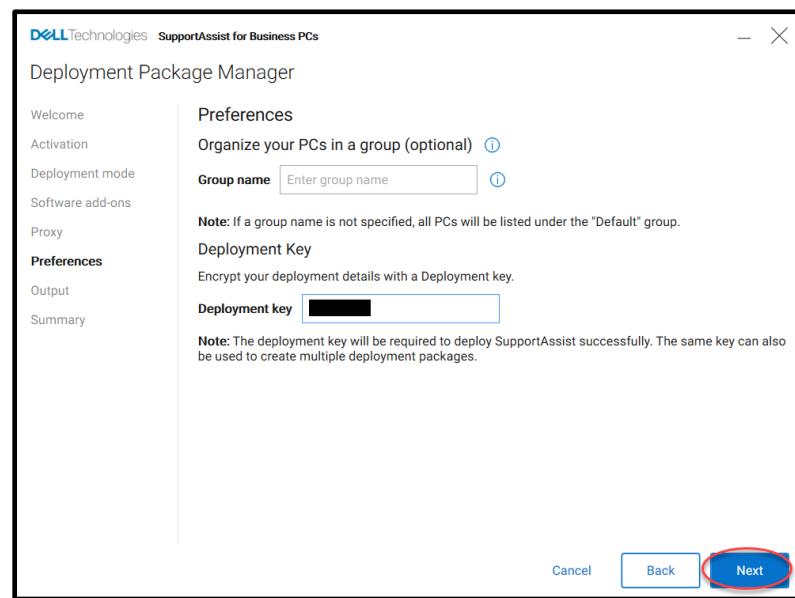


Click 'Next' (or if needed configure your Proxy settings here.)



Note: For the deployment you need to setup a password (4-10 characters using numbers, letters, and special characters).

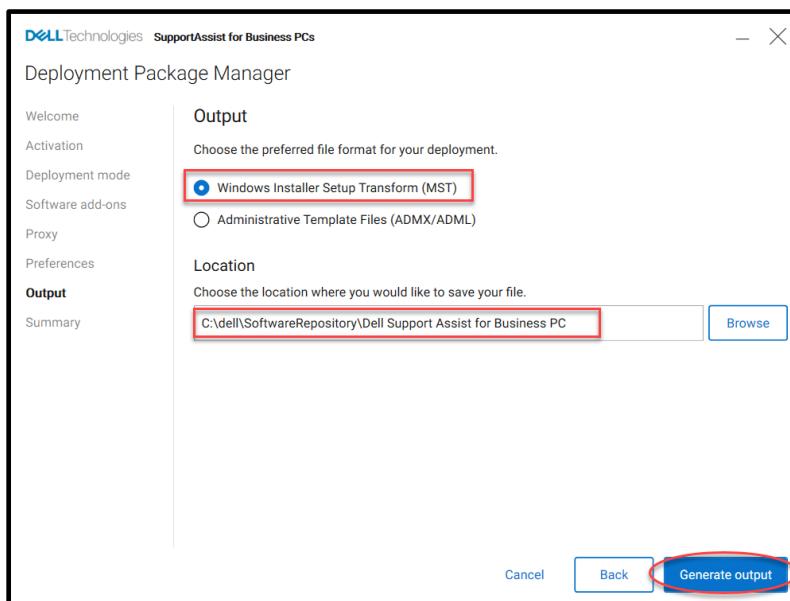
Click 'Next'



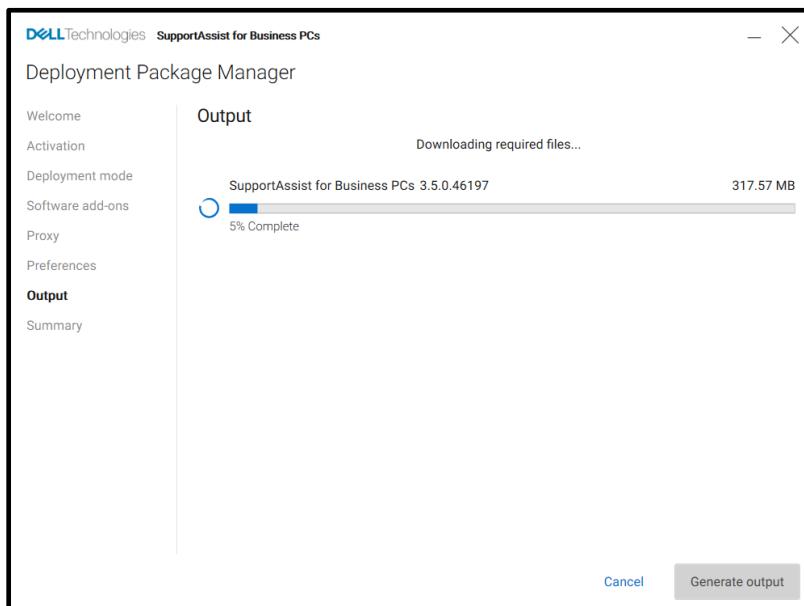
Select Windows Installer Setup Transform (MST)

Choose the path where the installer should be saved, e.g., C:\Dell\DSA

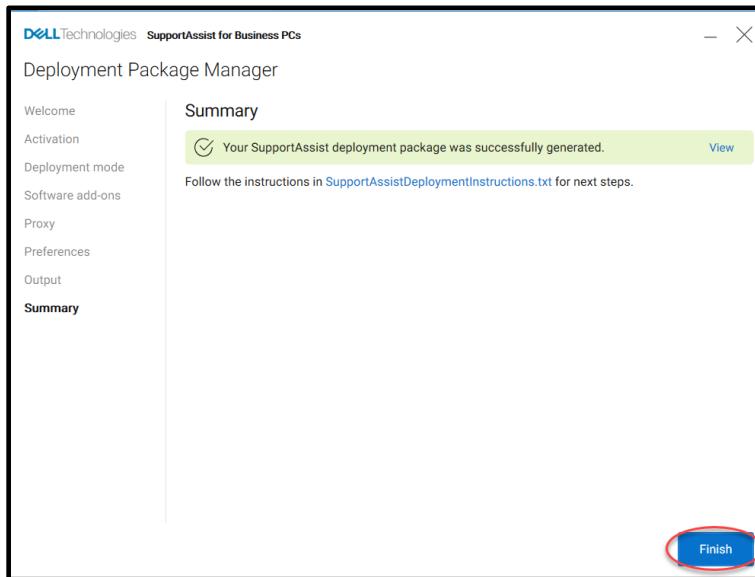
Click 'Generate output'



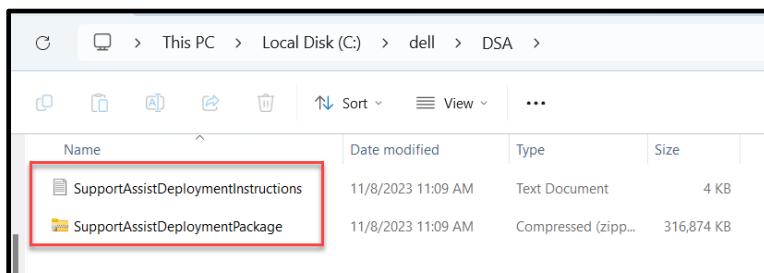
Download is starting.



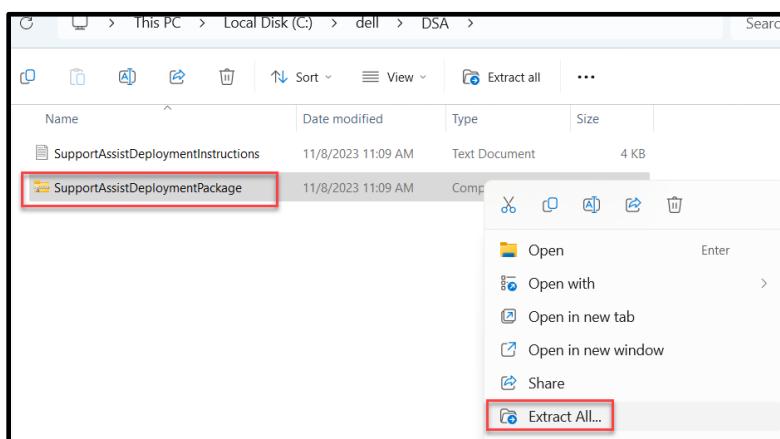
Final Summary.
Click 'Close'



Files are now available.



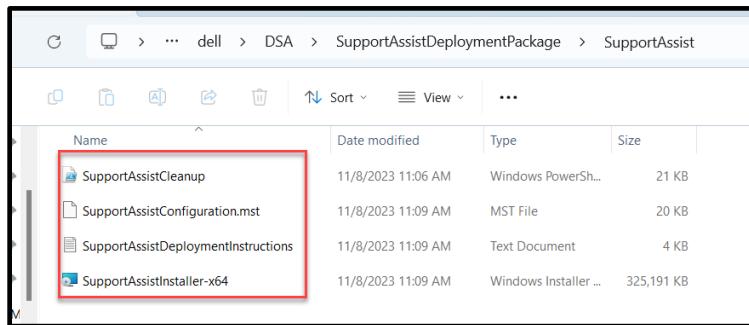
Extract the file 'SupportAssistDeploymentPackage.zip'



If you have finished this step, copy file to your software repository for the next step. We planning to build a dependency app too. We will have now two folders one for the Application we want to install and the second one for the dependency Application we need to run first.

Folder for SupportAssist for Business PC

This folder includes all files who are generated in the steps before.

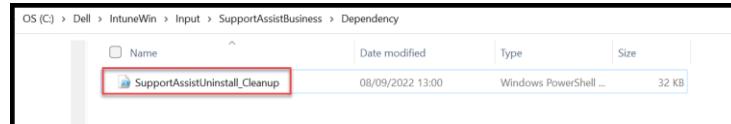


| Name | Date modified | Type | Size |
|-------------------------------------|--------------------|-----------------------|------------|
| SupportAssistCleanup | 11/8/2023 11:06 AM | Windows PowerShell... | 21 KB |
| SupportAssistConfiguration.mst | 11/8/2023 11:09 AM | MST File | 20 KB |
| SupportAssistDeploymentInstructions | 11/8/2023 11:09 AM | Text Document | 4 KB |
| SupportAssistInstaller-x64 | 11/8/2023 11:09 AM | Windows Installer ... | 325,191 KB |

Folder for SupportAssist Dependency Application

Copy file ‘SupportAssistUninstall_Cleanup.ps1’ to a new folder as base for the dependency application.

Note: It is recommended to run the ‘SupportAssistUninstall_Cleanup.ps1’ first to uninstall all SupportAssist installations, otherwise it could be the new installation does not work correctly.



| Name | Date modified | Type | Size |
|--------------------------------|------------------|------------------------|-------|
| SupportAssistUninstall_Cleanup | 08/09/2022 13:00 | Windows PowerShell ... | 32 KB |

Scripts for Install, Uninstall and Detection

It is possible to use the native file for installation. In this document we are using PowerShell scripts to cover different scenarios of Install new, Update, and uninstall for a later automation of uploading applications by API.

All script could be download on Github Repository:

<https://github.com/svenriebedell/Dell-Tools-Intune-Install>

Scripts Folder for SupportAssist for Business PC

Install Script for SupportAssist (main install)

The script makes a precheck if older versions are installed and starting an uninstall first to have a clean environment.

You need to fill in your Deployment Key, you have generated before on Page 113.

```
#####
#### Function section #####
#####

function Get-installedcheck
{
    param
    (
        [Parameter(mandatory=$true)][string] $AppSearchString
    )

    $AppCheck = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearchString'"

    If ($null -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        return $false
    }
}

#### function based on https://stackoverflow.com/questions/8743122/how-do-i-find-the-msi-product-version-number-using-powershell

function Get-MsiFileVersion {
    param
    (
        [Parameter(mandatory=$true)][string] $msiName
    )

    try {
        $FullPath = (Resolve-Path $msiName).Path
        $windowsInstaller = New-Object -com WindowsInstaller.Installer

        $database = $windowsInstaller.GetType().InvokeMember(
            "OpenDatabase", "InvokeMethod", $Null,
            $windowsInstaller, @($FullPath, 0)
        )

        $q = "SELECT Value FROM Property WHERE Property = 'ProductVersion'"
        $View = $database.GetType().InvokeMember(
            "OpenView", "InvokeMethod", $Null, $database, ($q)
        )

        $View.GetType().InvokeMember("Execute", "InvokeMethod", $Null, $View, $Null)

        $record = $View.GetType().InvokeMember(
            "Fetch", "InvokeMethod", $Null, $View, $Null
        )

        [Version]$productVersion = $record.GetType().InvokeMember(
            "GetStringData", "GetProperty", $Null, $record, 1
        )

        $View.GetType().InvokeMember("Close", "InvokeMethod", $Null, $View, $Null)
    }

    return $productVersion
} catch {
    throw "Failed to get MSI file version the error was: {0}." -f $_
}
```

```

}

#####
#### variable section #####
#####

$InstallerName = Get-ChildItem .\*.msi | Select-Object -ExpandProperty Name
$MSTName = Get-ChildItem .\*.mst | Select-Object -ExpandProperty Name
$ProgramPath = $InstallerName
$MSTPath = $MSTName
$ProgramVersion_target = Get-MsiFileVersion -msiName $InstallerName
$AppSearch = "%Dell%Support%Business%" #Parameter to search in registry
$Program_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearch'"
[Version]$ProgramVersion_current = $Program_current.Version
$ApplicationID_current = $Program_current.IdentifyingNumber
$Argumentstring = "/i " + "" + $ProgramPath + " TRANSFORMS=" + $MSTPath + " DEPLOYMENTKEY="UseYourKey# /norestart /qn /l+
"c:\SupportAssistMsi.log" #Your Deployment Key you generated before with the installer

$SoftwareName = $Program_current.Name

#####
#### Program section #####
#####

#### generate Logging Resources
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

#####
#Checking if older Version is installed and uninstall this Version #
#####

If ($null -ne $ProgramVersion_current)
{
    if ($ProgramVersion_target -gt $ProgramVersion_current)
    {
        #####
        # uninstall Software old #
        #####
        Start-Process -FilePath msixexec.exe -ArgumentList "/x $ApplicationID_current /qn" -Wait

        #####
        # uninstall success check #
        #####
        $UninstallResult = Get-installedcheck -AppSearchString $AppSearch

        If ($UninstallResult -eq $true)
        {
            Write-Host "uninstall is unsuccessful" -BackgroundColor Red

            $UninstallData = [PSCustomObject]@{
                Software = $SoftwareName
                Version = ($ProgramVersion_current).ToString()
                Uninstall = $false
            } | ConvertTo-Json

            Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData
        }
        Else
        {
            Write-Host "uninstall is successful" -BackgroundColor Green

            $UninstallData = [PSCustomObject]@{
                Software = $SoftwareName
                Version = ($ProgramVersion_current).ToString()
                Uninstall = $true
            } | ConvertTo-Json

            Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
        }
    }
}

```

```

Else
{
    Write-Host "same version is installed"
    [String]$VersionString = $ProgramVersion_target

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = $VersionString
        Install = $false
        Reason = "same version is installed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData

    Exit 0
}

#####
#Install new Software
#####

Start-Process -FilePath msiexec.exe -ArgumentList $Argumentstring -Wait

#####
# install success check #
#####

$Program_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearch'"
[Version]$ProgramVersion_current = $Program_current.Version
$SoftwareName = "Dell Support Assist for Business PCs"
$UninstallResult = Get-installedcheck -AppSearchString $AppSearch
[String]$VersionString = $ProgramVersion_target

If ($UninstallResult -ne $true)
{
    Write-Host "install is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = $VersionString
        Install = $false
        Reason = "Installation failed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Error -EventId 11 -Message $UninstallData
}

Else
{
    If ($ProgramVersion_current -ge $VersionString)
    {
        Write-Host "install is successful" -BackgroundColor Green

        $UninstallData = [PSCustomObject]@{
            Software = $SoftwareName
            Version = $VersionString
            Install = $true
            Reason = "Update/Install/Newer Version"
        } | ConvertTo-Json

        Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData
    }
}

Else
{
    Write-Host "install is unsuccessful" -BackgroundColor red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
}

```

```
    Version = $VersionString
    Install = $false
    Reason = "Older Version installed $ProgramVersion_current"
} | ConvertTo-Json

Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData

}

}
```

Uninstall Script

'SupportAssistUninstall_Cleanup.ps1' from the installer package

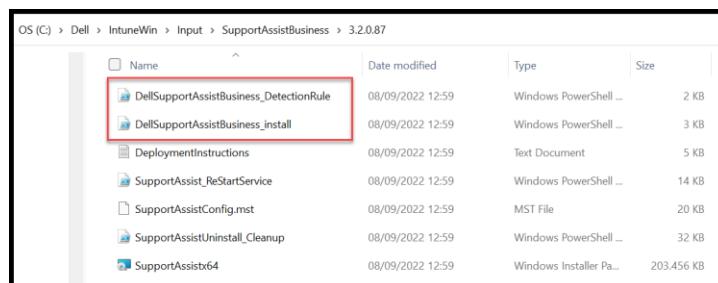
Detection Script (main install)

The detection script showing success of installation. It could be done as well without a script but again it is helpful for later automation of upload processes in the future.

The yellow marked version must be adjusted with each new version.

```
#####
# Program with target Version
#####
$ProgramVersion_target = '3.2.0.87' # need to be the same like the msi file
$ProgramVersion_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '%Dell%SupportAssist%Business%' | select-object -ExpandProperty Version
if($ProgramVersion_current -eq $ProgramVersion_target)
{
    Write-Host "Found it!"
}
```

Please, copy these files in the same folder as your MSI Installer File.

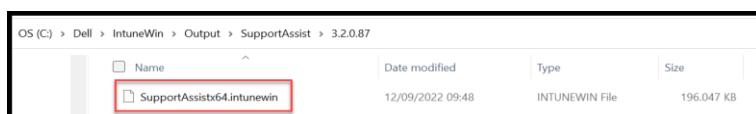


Start the Microsoft Win32 Content Prep Tool (aka IntuneAppUtil.exe). In case you are not familiar with this tool, you will find documentation here: <https://docs.microsoft.com/en-us/mem/intune/apps/apps-win32-prepare>

| Argument | Value |
|---------------|---|
| Source Folder | folder where you have stored the unzipped MSI |
| Setup file | Main installer file like msi/exe/ps1, etc. |
| Output Folder | where you want to store the IntuneWin |

```
Please specify the source folder: C:\Dell\IntuneWin\Input\DSA\3.2.0.87
Please specify the setup file: SupportAssistx64.msi
Please specify the output folder: C:\Dell\IntuneWin\Output\SupportAssist\3.2.0.87
Do you want to specify catalog folder (Y/N)?n_
```

IntuneWin is now prepared and ready for installation by Intune.



Scripts Folder for Dependency

Install Script (dependency install)

'SupportAssistUninstall_Cleanup.ps1' from the installer package

Detection Script (dependency install)

The detection script showing success of installation. It could be done as well without a script but again it is helpful for later automation of upload processes in the future.

The yellow marked version must be adjusted with each new version.

```
##### Variables
$ProgramVersion_target = '3.2.0.87' # need to be the same like the msi file
$ProgramVersion_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '%Dell%SupportAssist%Business%" | Select-Object -
ExpandProperty Version
$Program_current = Get-CimInstance -ClassName Win32_Product | Where-Object {($_.Name -like "*Dell SupportAssist*" -and $_.Name -notlike "*OS
Recovery*" -and $_.Name -notlike "Dell*SupportAssist*Remediation")} | Select-Object -ExpandProperty Name

#####
# Checking if a SupportAssist existing after Cleanup script
#####

if ($null -eq $Program_current)
{
    Write-Host "Found it!"

}
Else
{
    #####
    # Cover newest version is installed on the machine
    #####
    if ($ProgramVersion_target -eq $ProgramVersion_current)
    {
        Write-Host "Found it!"
    }
}
```

Please, copy these files in the folder Dependency together with the Uninstaller PowerShell Script.



Start the Microsoft Win32 Content Prep Tool (aka IntuneAppUtil.exe). In case you are not familiar with this tool, you will find documentation here: <https://docs.microsoft.com/en-us/mem/intune/apps/apps-win32-prepare>

| Argument | Value |
|---------------|---|
| Source Folder | folder where you have stored the unzipped MSI |
| Setup file | Main installer file like msi/exe/ps1, etc. |
| Output Folder | where you want to store the IntuneWin |

```
Please specify the source folder: C:\Dell\IntuneWin\Input\SupportAssistBusiness\Dependency  
Please specify the setup file: SupportAssistUninstall_Cleanup.ps1  
Please specify the output folder: C:\Dell\IntuneWin\Output\SupportAssist\Dependency  
Do you want to specify catalog folder (Y/N)?
```

IntuneWin is now prepared and ready for installation by Intune.



Import and Deployment settings Dell SupportAssist for Business PCs for Intune

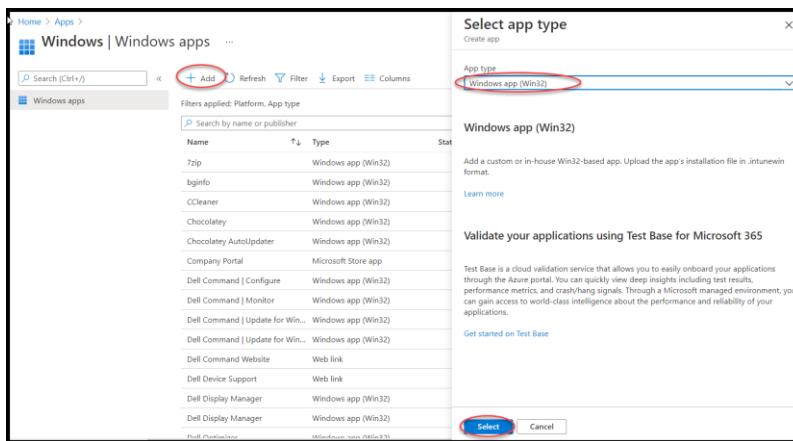
Part Dependency Application

Click 'Add'

| Field | Value |
|---------------|---------------------|
| Source Folder | Windows app (Win32) |

Click 'Select'

Select Windows app (Win32) as application.



Section 'App information'

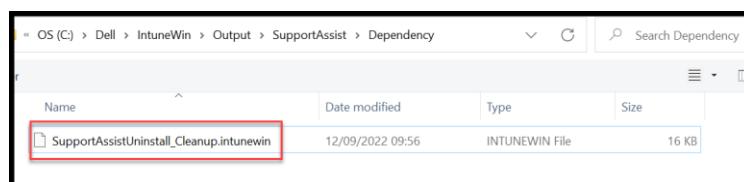


Click 'Select app package file'

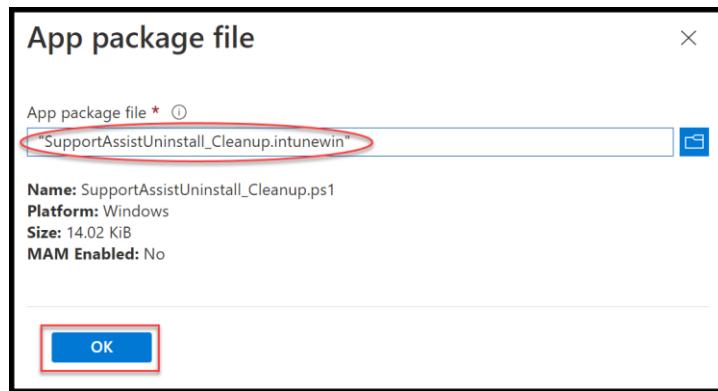
Click 'Folder'



Select 'SupportAssistUninstall_Cleanup.intunewin'



Click 'OK'



| Field | Value |
|---|---|
| Name | SupportAssist Uninstall Cleanup |
| Publisher | Dell Inc. |
| App Version | 3.2.0.87 Note: Use same Version as your SupportAssist for easier find the right combination of Install and Dependency |
| Show this as a featured app in the Company Portal | No |

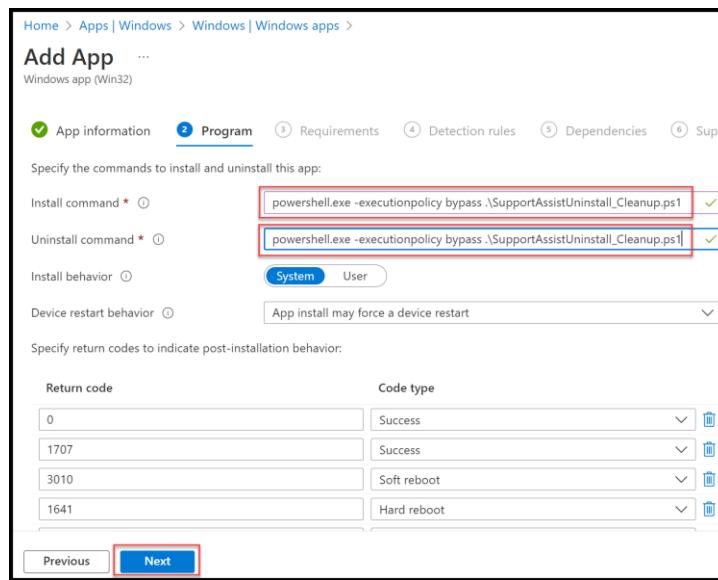
Click 'Next'

Section 'Program'



| Field | Value |
|-------------------|---|
| Install command | powershell.exe -executionpolicy bypass .\\SupportAssistUninstall_Cleanup.ps1 |
| Uninstall command | powershell.exe -executionpolicy bypass .\\SupportAssistUninstall_Cleanup.ps1 |

Click 'Next'



Section 'Requirements'

The screenshot shows a software interface for adding an application. At the top, there's a breadcrumb navigation: Home > Apps > Windows > Add App. Below it, the title 'Add App' and a subtitle 'Windows app (Win32)'. A horizontal navigation bar at the top right includes tabs: App information (green checkmark), Program (green checkmark), Requirements (blue circle with a red border, indicating it's selected), Detection rules (grey), Dependencies (grey), Superseding (grey), Assignments (grey), and Review + create (grey). The main content area is titled 'Requirements' and contains a table with two rows:

| Field | Value |
|-------------------------------|--|
| Operating system architecture | 64-Bit |
| Minimum operating system | 2004 (Please note Dell support drivers and software only with the latest Win version + N -2) |

Click 'Next'

The screenshot shows the 'Add App' interface again, this time on the 'Requirements' step. The breadcrumb navigation is identical. The horizontal navigation bar now has the 'Requirements' tab highlighted with a blue underline. The main content area is titled 'Specify the requirements that devices must meet before the app is installed:' and contains several input fields:

- Operating system architecture * (dropdown menu showing '64-bit')
- Minimum operating system * (dropdown menu showing 'Windows 10 2004')
- Disk space required (MB) (empty input field)
- Physical memory required (MB) (empty input field)
- Minimum number of logical processors required (empty input field)
- Minimum CPU speed required (MHz) (empty input field)

At the bottom, there are 'Previous' and 'Next' buttons. The 'Next' button is highlighted with a red border.

Section 'Detection rules'

The screenshot shows the 'Add App' interface with the 'Detection rules' tab highlighted. Other tabs like 'App information', 'Program', 'Requirements', 'Dependencies', 'Supersedeance (preview)', 'Assignments', and 'Review + create' are also visible.

| Field | Value |
|--------------|------------------------|
| Rules format | Use a custom detection |

Click 'Folder'

The screenshot shows the 'Add App' interface under the 'Detection rules' tab. The 'Rules format' dropdown is set to 'Use a custom detection script'. The 'Script file' input field is highlighted with a red box. Below it, there are options for running the script as 32-bit or 64-bit and for silent execution.

Select 'DellSupportAssistBusinessDependency_DetectionRule.ps1'

The screenshot shows a file explorer window with the path 'OS (C) > Dell > IntuneWin > Input > SupportAssistBusiness > Dependency'. It lists two files: 'DellSupportAssistBusinessDependency_DetectionRule.ps1' and 'SupportAssistUninstall_Cleanup.ps1'. The first file is highlighted with a red box.

Click 'Next'

The screenshot shows the 'Add App' interface under the 'Detection rules' tab. The 'Script file' input field contains the path 'D:\IntuneWin\Input\SupportAssistBusiness\Dependency\DellSupportAssistBusinessDependency_DetectionRule.ps1'. The 'Next' button at the bottom is highlighted with a red box.

Section 'Dependencies'



No changes

Section 'Supersedence'

No changes

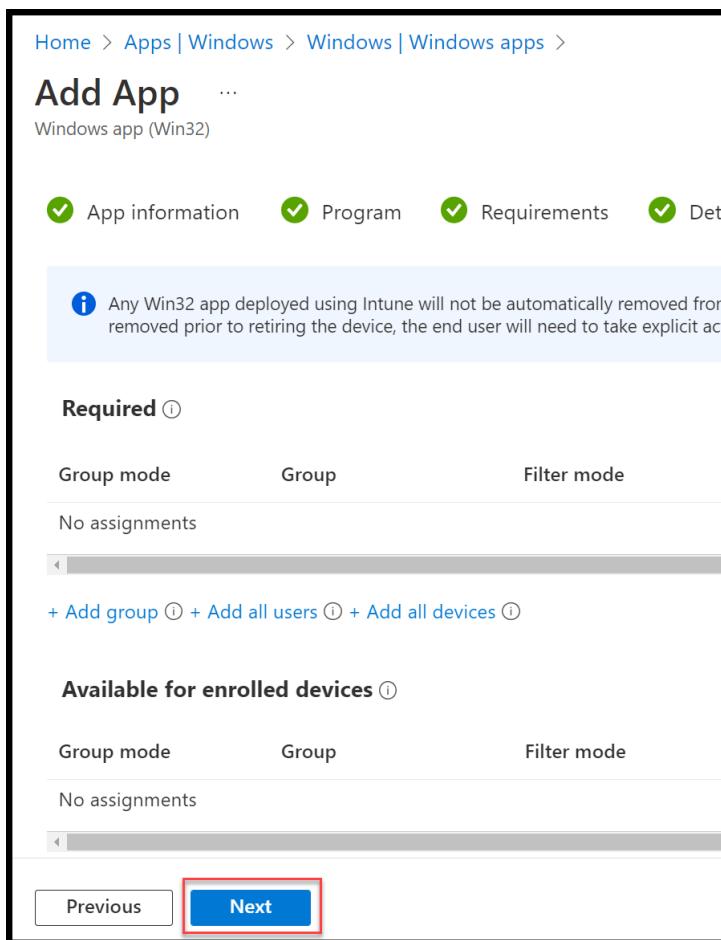
Section ‘Assignments’



No changes

| Option | Value |
|--------------------------------|-------|
| Required | |
| Available for enrolled devices | |
| Uninstall | |

Click ‘Next’



The app is now finished

Click 'Create'

Home > Apps | Windows > Windows | Windows apps >

Add App ...

Windows app (Win32)

✓ Program ✓ Requirements ✓ Detection rules ✓ Dependencies

Summary

App information

| | |
|---|--|
| App package file | SupportAssistUninstall_Cleanup.intunewin |
| Name | SupportAssist Uninstall Cleanup |
| Description | SupportAssistUninstall_Cleanup.ps1 |
| Publisher | Dell Inc. |
| App Version | 3.2.0.87 |
| Category | -- |
| Show this as a featured app in the Company Portal | No |
| Information URL | -- |
| Privacy URL | -- |

Previous **Create**

Ready to work.

Windows apps ...

+ Add ⏪ Refresh ⏴ Filter ⏴ Export ⏴ Columns

Filters applied: Platform, App type

| Name | Type | Status | Version | Assigned |
|--|---------------------|-----------------|---------|----------|
| Dell SupportAssist Cleanup | Windows app (Win32) | 1.0.0.2 | No | ... |
| Dell SupportAssist for Business PCs | Windows app (Win32) | 3.1.1.18 | No | ... |
| Dell SupportAssist for Business PCs | Windows app (Win32) | 3.1.0.64 | No | ... |
| Dell SupportAssist for Business PCs | Windows app (Win32) | 3.2.0.87 | Yes | ... |
| Dell SupportAssist for Business PCs | Windows app (Win32) | 3.0.0.34 | No | ... |
| SupportAssist Uninstall Cleanup | Windows app (Win32) | 3.2.0.87 | No | ... |

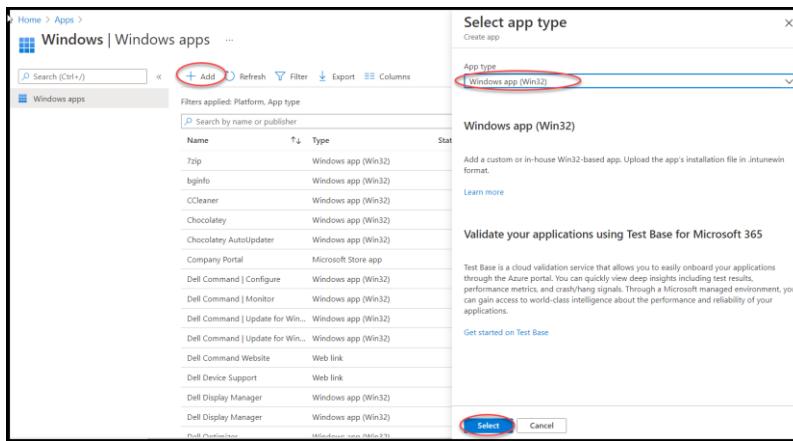
Part SupportAssist for Business Application

Click 'Add'

| Field | Value |
|---------------|---------------------|
| Source Folder | Windows app (Win32) |

Click 'Select'

Select Windows app (Win32) as application.



Section 'App information'

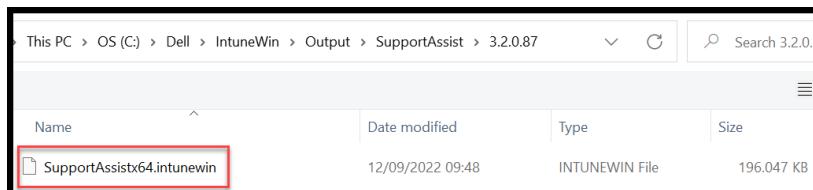


Click 'Select app package file'

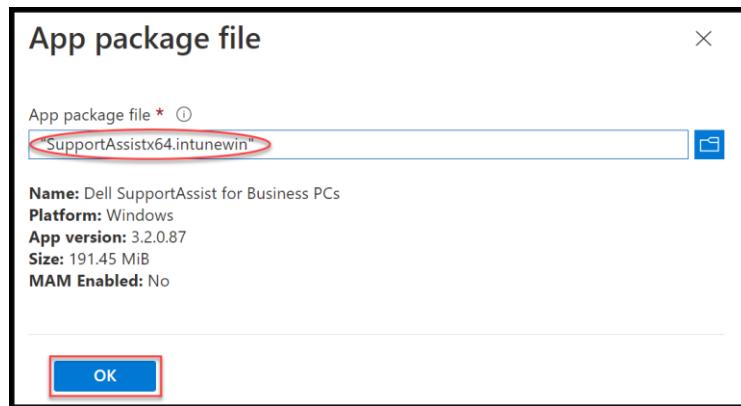
Click 'Folder'



Select 'SupportAssistx64.intunewin'



Click 'OK'



| Field | Value |
|---|-----------|
| Publisher | Dell Inc. |
| Show this as a featured app in the Company Portal | Yes |
| | |

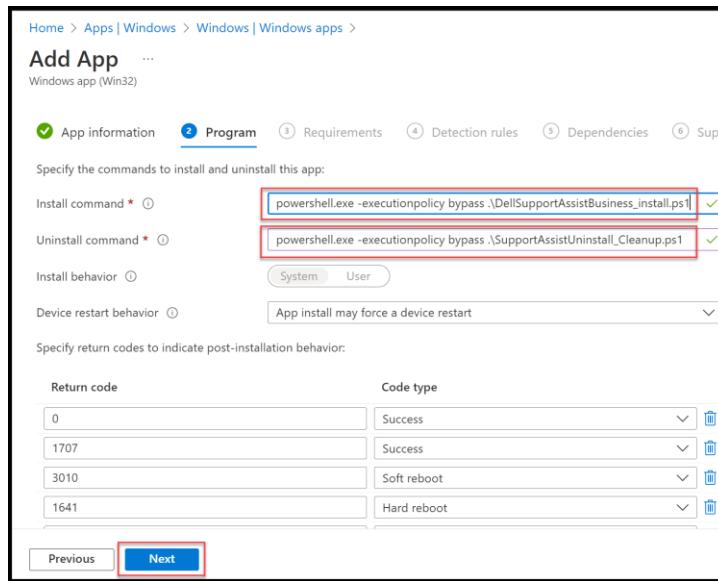
Click 'Next'

Section 'Program'



| Field | Value |
|-------------------|--|
| Install command | powershell.exe -executionpolicy bypass .\\DellSupportAssistBusiness_install.ps1 |
| Uninstall command | powershell.exe -executionpolicy bypass .\\SupportAssistUninstall_Cleanup.ps1 |

Click 'Next'



Home > Apps | Windows > Windows | Windows apps >

Add App ...

Windows app (Win32)

Specify the commands to install and uninstall this app:

Install command * ⓘ powershell.exe -executionpolicy bypass .\\DellSupportAssistBusiness_install.ps1 ✓

Uninstall command * ⓘ powershell.exe -executionpolicy bypass .\\SupportAssistUninstall_Cleanup.ps1 ✓

Install behavior ⓘ System User

Device restart behavior ⓘ App install may force a device restart

Specify return codes to indicate post-installation behavior:

| Return code | Code type |
|-------------|-------------|
| 0 | Success |
| 1707 | Success |
| 3010 | Soft reboot |
| 1641 | Hard reboot |

Previous Next

Section 'Requirements'

The screenshot shows a software interface for adding an application. At the top, there's a breadcrumb navigation: Home > Apps > Windows > Add App. Below it, the title 'Add App' and a sub-label 'Windows app (Win32)'. A horizontal navigation bar at the top right includes tabs: App information (green checkmark), Program (green checkmark), Requirements (blue circle with a question mark, highlighted with a red oval), Detection rules (grey), Dependencies (grey), Superseding (grey), Assignments (grey), and Review + create (grey). The 'Requirements' tab is currently active.

| Field | Value |
|-------------------------------|--|
| Operating system architecture | 64-Bit |
| Minimum operating system | 2004 (Please note Dell support drivers and software only with the latest Win version + N -2) |

Click 'Next'

The screenshot shows the 'Add App' interface on the 'Requirements' page. The top navigation bar is identical to the previous screenshot. The main area is titled 'Specify the requirements that devices must meet before the app is installed:' and contains several input fields:

- Operating system architecture * (radio button selected): 64-bit (highlighted with a red rectangle)
- Minimum operating system * (radio button selected): Windows 10 2004 (highlighted with a blue rectangle)
- Disk space required (MB): (empty input field)
- Physical memory required (MB): (empty input field)
- Minimum number of logical processors required: (empty input field)
- Minimum CPU speed required (MHz): (empty input field)

At the bottom, there are 'Previous' and 'Next' buttons. The 'Next' button is highlighted with a red rectangle.

Section 'Detection rules'

The screenshot shows the 'Add App' interface with the 'Detection rules' tab selected. The tabs at the top are: App information, Program, Requirements, Detection rules (selected), Dependencies, Supersedeance (preview), Assignments, and Review + create.

| Field | Value |
|--------------|------------------------|
| Rules format | Use a custom detection |

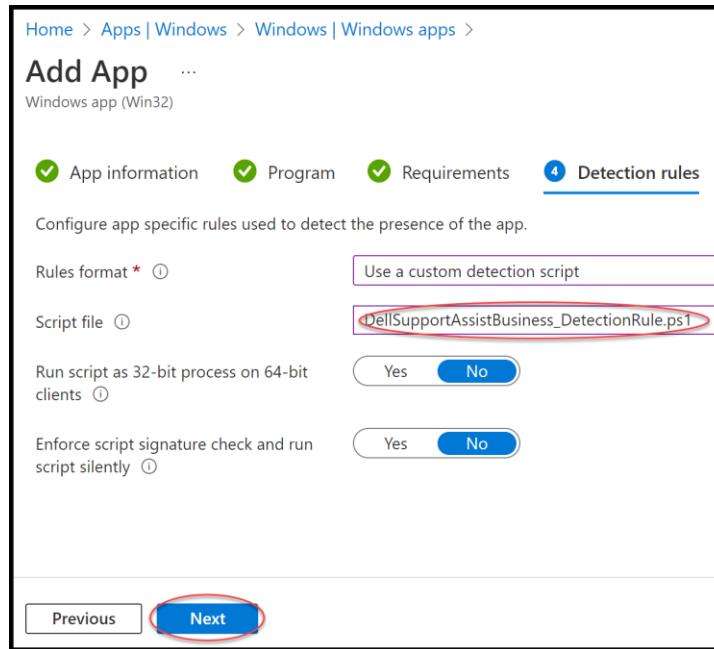
Click 'Folder'

The screenshot shows the 'Add App' interface under the 'Detection rules' tab. It includes fields for 'Rules format' (set to 'Use a custom detection script'), 'Script file' (with a 'Select a file' button highlighted by a red box), and options for running the script as 32-bit or 64-bit and enforcing silent execution.

Select 'DellSupportAssistBusiness_DetectionRule.ps1'

The screenshot shows a file explorer window displaying the contents of the 'SupportAssistBusiness' folder. The files listed are: DellSupportAssistBusiness_DetectionRule.ps1 (highlighted by a red box), DellSupportAssistBusiness_install.ps1, SupportAssist_ReStartService.ps1, and SupportAssistUninstall_Cleanup.ps1.

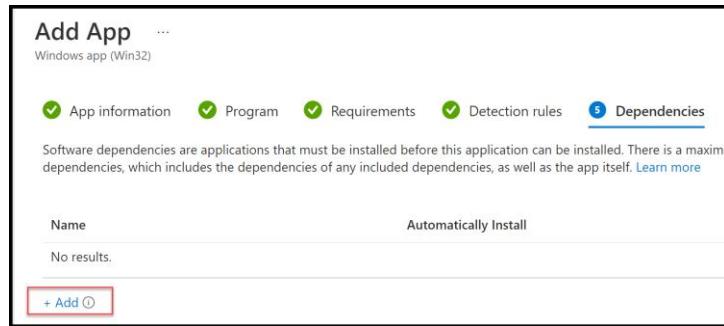
Click 'Next'



Section 'Dependencies'

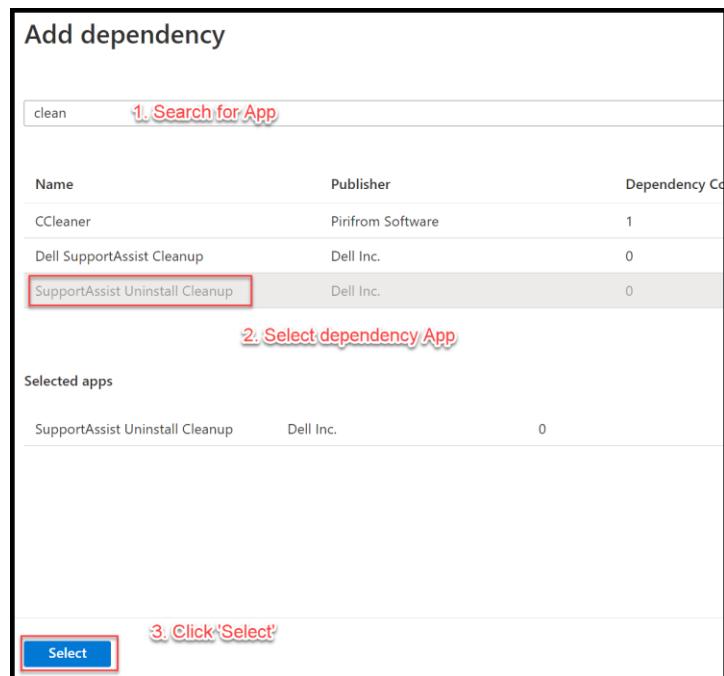


Click 'Add'



Search for the new created dependency Application 'SupportAssist Uninstall Cleanup' and select this Application.

Click 'Select'



| Field | Value |
|-----------------------|-------|
| Automatically Install | Yes |

Click 'Next'

Home > Apps | Windows > Windows | Windows apps >
Add App ...
 Windows app (Win32)

App information Program Requirements Detection rules Dependencies

Software dependencies are applications that must be installed before this application can be installed. There is a maximum of 100 dependencies, which includes the dependencies of any included dependencies, as well as the app itself. [Learn more](#)

| Name | Automatically Install |
|---------------------------------|---|
| SupportAssist Uninstall Cleanup | <input checked="" type="button"/> Yes <input type="button"/> No |

+ Add ⓘ

Previous **Next**

Section 'Supersedence'

No changes

Section 'Assignments'

Home > Apps > Windows >
Add App ...
 Windows app (Win32)

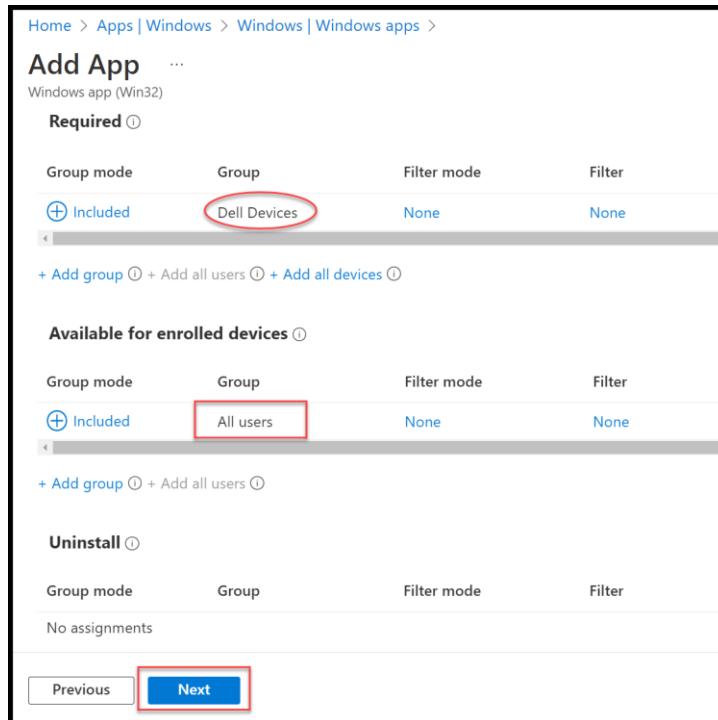
App information Program Requirements Detection rules Dependencies Supersedence (preview) **Assignments** ⓘ Review + create

No changes

The Dell SupportAssist for Business PC supports only Dell (Latitude, Optiplex, Precision and mobile XPS) it makes sense to have da dynamic group which only incl. these systems.

| Option | Value |
|--------------------------------|-------------------------|
| Required | Add Group 'Dell Device' |
| Available for enrolled devices | Add Group 'All User' |
| Uninstall | |

Click 'Next'



The app is now finished

Click 'Create'

Home > Apps | Windows > Windows | Windows apps >

Add App

Windows app (Win32)

Program Requirements Detection rules Dependencies

Summary

App information

| | |
|---|-------------------------------------|
| App package file | SupportAssistx64.intunewin |
| Name | Dell SupportAssist for Business PCs |
| Description | Dell SupportAssist for Business PCs |
| Publisher | Dell Inc. |
| App Version | 3.2.0.87 |
| Category | -- |
| Show this as a featured app in the Company Portal | Yes |
| Information URL | -- |
| Privacy URL | -- |

Previous Create

Ready to work.

Windows apps

+ Add Refresh Filter Export Columns

Filters applied: Platform, App type

| Name | Type | Status | Version | Assigned |
|--------------------------------|---------------------|----------|---------|----------|
| Dell SupportAssist for Busi... | Windows app (Win32) | 3.1.1.18 | No | |
| Dell SupportAssist for Busi... | Windows app (Win32) | 3.1.0.64 | No | |
| Dell SupportAssist for Busi... | Windows app (Win32) | 3.2.0.87 | Yes | |
| Dell SupportAssist for Busi... | Windows app (Win32) | 3.0.0.34 | No | |

Dell Optimizer

Introduction

Dell Optimizer is a software application that intelligently optimizes the performance of your system by using artificial intelligence and machine learning. Dell Optimizer dynamically configures your system settings to optimize the performance of your applications. It improves productivity, performance, and user experience through system usage analysis and learning.

On Dell Precision workstations, Dell Optimizer for Precision includes analytics feature that collects extensive data about your system and helps identify potential issues.

Dell Optimizer Version 4.x User's Guide

<https://www.dell.com/support/home/en-us/product-support/product/dell-optimizer/docs>

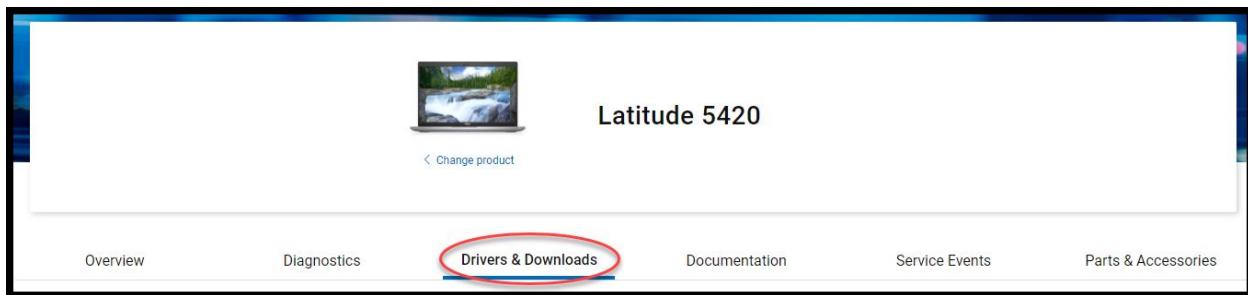
Prepare Dell Optimizer for Intune

Download Dell Optimizer

Download the newest Version of Dell Optimizer from our website.

<https://www.dell.com/support/home/en-us>

Note: Please select a device platform like Latitude 5420 and move to section ‘Drivers & Downloads’



Choose Manually to find a specific driver

| Field | Value |
|---------|-----------|
| Keyword | Optimizer |

Find 'Dell Optimizer Application' and download the installer file.

Manually find a specific driver for your Latitude 5320

Keyword: **boptimizer**

Operating system: Windows 10, 64-bit

Download Type: All

Category: All

This is a comprehensive list of all available downloads for your Latitude 5320. Some downloads may already exist on your device. To let Dell automatically find available updates for you, select [Check for Updates](#).

| NAME | IMPORTANCE | CATEGORY | RELEASE DATE | ACTION |
|-----------------------------------|-------------|--------------------|--------------|--------------------------|
| Dell Power Manager Service | CRITICAL | Systems Management | 03 Apr 2023 | Download |
| Dell Optimizer Application | RECOMMENDED | Application | 16 Mar 2023 | Download |

If you have downloaded the file from dell.com/support, copy file to your software repository for the next step.

This PC > OS (C) > Dell > IntuneWin > Input > Dell Optimizer > 4.0.310.0

| Name | Date modified | Type | Size |
|---|------------------|-------------|------------|
| Dell-Optimizer_92TP7_WIN_4.0.310.0_A00 | 03.04.2023 16:49 | Application | 614.662 KB |

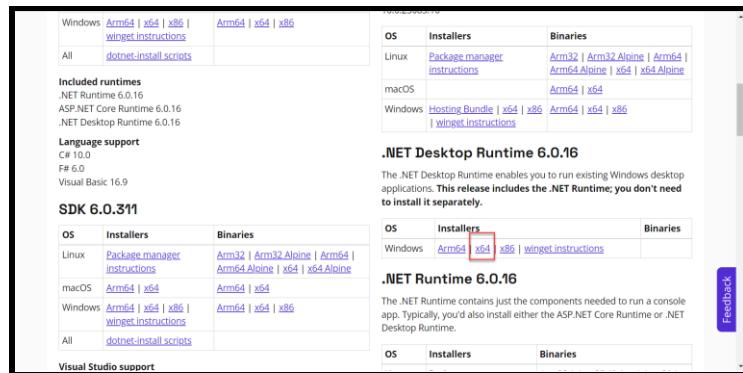
Download Microsoft .net Runtime 6.x

Dell Optimizer since Version 4.0.201.0 is Microsoft .net Runtime 6.0.15 or higher required. If you cover this by standard, you can ignore this part otherwise we will show you how you can build a dependency application in Intune which allows to check if Microsoft .net Runtime is installed or not and install the application if needed.

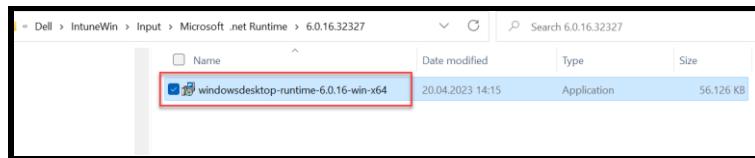
Download the latest version of .NET Desktop Runtime 6.x

<https://dotnet.microsoft.com/en-us/download/dotnet/6.0>

Find .NET Desktop Runtime 6.x for Windows x64' and download the installer file.



If you have downloaded the file from dell.com/support, copy file to your software repository for the next step.



Scripts for Install, Uninstall and Detection

It is possible to use the native file for installation. In this document we are using PowerShell scripts to cover different scenarios of Install new, Update, and uninstall for a later automation of uploading applications by API.

All script could be download on Github Repository:

<https://github.com/svenriebedell/Dell-Tools-Intune-Install>

Script for Dell Optimizer

Install Script

The script makes a precheck if older versions are installed and starting an uninstall first to have a clean environment.

```
#####
##### Function section #####
#####

function Get-installedcheck
{
    param
    (
        [Parameter(mandatory=$true)][string] $AppSearchString
    )

    $AppCheck = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearchString'"

    If ($null -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        return $false
    }
}

function get-uninstallstring
{
    param
    (
        [Parameter(mandatory=$true)][ValidateSet($true,$false)][string]$PowerShell64,
        [Parameter(mandatory=$true)][ValidateSet("Optimizer", "WebView2", "Runtime6", "DDM", "DPM")][string]$App,
        [Parameter(mandatory=$true)][ValidateSet("UninstallString", "InstallLocation", "Quietuninstallstring")][string]$Value
    )

    $AppSearch = switch ($App)
    {
        Optimizer {"Dell*Optimizer*Core"}
        WebView2 {"Microsoft*WebView2**"}
        Runtime6 {"Microsoft Windows Desktop Runtime - 6*(x64)*"}
        DDM {"Dell*Display*Manager**"}
        DPM {"Dell*Power*Manager**"}
    }

    if ($PowerShell64 -eq $true)
    {
        If (($AppSearch -like "*Display*") -or ($AppSearch -like "Dell*Power*"))
        {
            # get 64 Bit uninstall
            Get-ChildItem -Path HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall | Get-ItemProperty | Where-Object {$_.DisplayName -like "$AppSearch"} | Select-Object -ExpandProperty $Value
        }
        else
        {
            # get 32 Bit uninstall
            Get-ChildItem -Path HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall | Get-ItemProperty | Where-Object {$_.DisplayName -like "$AppSearch"} | Select-Object -ExpandProperty $Value
        }
    }
}
```

```

else
{
    # get 32 Bit uninstall
    Get-ChildItem -Path HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall | Get-ItemProperty | Where-Object
{$_._DisplayName -like "$AppSearch" } | Select-Object -ExpandProperty $Value
}

#####

##### variable section #####
#####
$InstallerName = Get-ChildItem .\*.exe | Select-Object -ExpandProperty Name
$ProgramPath = "." + $InstallerName
[Version]$ProgramVersion_target = (Get-Command $ProgramPath).FileVersionInfo.ProductVersion
$AppSearch = "%Dell %Optimizer%" #Parameter to search in registry
$Program_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearch'"
[Version]$ProgramVersion_current = $Program_current.Version
$PS64 = [Environment]::Is64BitProcess # Check 32 or 64 bit PowerShell
$SoftwareName = $Program_current.Name

#####

##### Program section #####
#####

##### generate Logging Resources
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

#####

#Checking if older Version is installed and uninstall this Version #
#####

If ($Null -ne $ProgramVersion_current)
{
    if ($ProgramVersion_target -gt $ProgramVersion_current)
    {
        #####
        # Uninstall older Versions #
        #####
        ### Get UninstallString from registry
        $ApplicationUninstallString = get-uninstallstring -PowerShell64 $PS64 -App Optimizer -Value UninstallString
        $ApplicationUninstallString = $ApplicationUninstallString + " /Silent"

        Start-Process cmd.exe -ArgumentList '/c',"$ApplicationUninstallString" -Wait -NoNewWindow
        Start-Sleep -Seconds 15

        #####
        # uninstall success check #
        #####
        $UninstallResult = Get-installedcheck -AppSearchString $AppSearch

        If ($UninstallResult -eq $true)
        {
            Write-Host "uninstall is unsuccessful" -BackgroundColor Red

            $UninstallData = [PSCustomObject]@{
                Software = $SoftwareName
                Version = $($ProgramVersion_current).ToString()
                Uninstall = $false
            } | ConvertTo-Json

            Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData
        }
        Else
        {
            Write-Host "uninstall is successful" -BackgroundColor Green

            $UninstallData = [PSCustomObject]@{
                Software = $SoftwareName
                Version = $($ProgramVersion_current).ToString()
            }
    }
}

```

```

        Uninstall = $true
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
}

}

Else
{
    Write-Host "same version is installed"

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $false
        Reason = "same version is installed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData

}

Exit 0
}

#####
#Install new Software
#####
Start-Process -FilePath "$ProgramPath" -ArgumentList "/s" -Wait

#####
# install success check
#####
$Program_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearch'"
[Version]$ProgramVersion_current = $Program_current.Version
$SoftwareName = $Program_current.Name
$UninstallResult = Get-installedcheck -AppSearchString $AppSearch

If ($UninstallResult -ne $true)
{
    Write-Host "install is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $false
        Reason = "Installation failed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Error -EventId 11 -Message $UninstallData
}

Else
{
    If ($ProgramVersion_current -ge $ProgramVersion_target)
    {
        Write-Host "install is successful" -BackgroundColor Green

        $UninstallData = [PSCustomObject]@{
            Software = $SoftwareName
            Version = ($ProgramVersion_target).ToString()
            Install = $true
            Reason = "Update/Install/Newer Version"
        } | ConvertTo-Json

        Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData
    }
}

```

```
{  
  
    Write-Host "install is unsuccessful" -BackgroundColor red  
  
    $UninstallData = [PSCustomObject]@{  
        Software = $SoftwareName  
        Version = ($ProgramVersion_target).ToString()  
        Install = $false  
        Reason = "Older Version installed $ProgramVersion_current"  
    } | ConvertTo-Json  
  
    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData  
  
}  
}
```

Uninstall Script

The script starts the uninstalling of application.

```
#####
##### Variables section #####
#####

$WebViewUninstall = $false # $true/$false to enable/disable uninstall of Microsoft Edge WebView2, be careful and check if other applications need this app before you uninstall this software
$RuntimeUninstall = $false # $true/$false to enable/disable uninstall of Microsoft Windows Desktop Runtime 6.x, be careful and check if other applications need this app before you uninstall this software
$DDMUninstall = $false # $true/$false to enable/disable uninstall of Dell Display Manager 2.x
$DPMUUninstall = $false # $true/$false to enable/disable uninstall of Dell Power Manager Service
$PS64 = [Environment]::Is64BitProcess # Check 32 or 64 bit PowerShell
$appSearch = "%Dell %Optimizer%" #Parameter to search in registry
$Program_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$appSearch'"
[Version]$ProgramVersion_current = $Program_current.Version
$SoftwareName = $Program_current.Name

#####
# function section #
#####

function get-uninstallstring
{
    param
    (
        [Parameter(mandatory=$true)][ValidateSet($true,$false)][string]$PowerShell64,
        [Parameter(mandatory=$true)][ValidateSet("Optimizer","WebView2","Runtime6","DDM","DPM")][string]$App,
        [Parameter(mandatory=$true)][ValidateSet("UninstallString","InstallLocation","QuietUninstallString")][string]$Value
    )

    $AppSearch = switch ($App)
    {
        Optimizer {"Dell*Optimizer*Core"}
        WebView2 {"Microsoft*WebView2*"}
        Runtime6 {"Microsoft Windows Desktop Runtime - 6*(x64)*"}
        DDM {"Dell*Display*Manager*"}
        DPM {"Dell*Power*Manager*"}
    }

    if ($PowerShell64 -eq $true)
    {
        If (($AppSearch -like "*Display*") -or ($AppSearch -like "Dell*Power*"))
        {
            # get 64 Bit uninstall
            Get-ChildItem -Path HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall | Get-ItemProperty | Where-Object {$__.DisplayName -like "$AppSearch" } | Select-Object -ExpandProperty $Value
        }

        else
        {
            # get 32 Bit uninstall
            Get-ChildItem -Path HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall | Get-ItemProperty | Where-Object {$__.DisplayName -like "$AppSearch" } | Select-Object -ExpandProperty $Value
        }
    }
    else
    {
        # get 32 Bit uninstall
        Get-ChildItem -Path HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall | Get-ItemProperty | Where-Object {$__.DisplayName -like "$AppSearch" } | Select-Object -ExpandProperty $Value
    }
}
}
```

```

function Get-installedcheck
{
    param
    (
        [Parameter(mandatory=$true)][string] $AppSearchString
    )

    $AppCheck = Get-CimInstance -ClassName Win32_Product -Filter "Name like '$AppSearchString'"

    If ($null -ne $AppCheck)
    {
        return $true
    }
    Else
    {
        return $false
    }
}

#####
#### program section
#####

##### generate Logging Resources
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

##### Get UninstallString from registry
$ApplicationUninstallString = get-uninstallstring -PowerShell64 $PS64 -App Optimizer -Value UninstallString
$ApplicationUninstallString = $ApplicationUninstallString + " /Silent"

##### Uninstall Dell Optimizer
Start-Process cmd.exe -ArgumentList'/c', "$ApplicationUninstallString" -Wait -NoNewWindow
Start-Sleep -Seconds 20

#####
# uninstall success check #
#####
$UninstallResult = Get-installedcheck -AppSearchString $AppSearch

If ($UninstallResult -eq $true)
{
    Write-Host "uninstall is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = $Program_current.Version
        Uninstall = $false
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData
}

Else
{
    Write-Host "uninstall is successful" -BackgroundColor Green

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = $Program_current.Version
        Uninstall = $true
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
}

#####
# Uninstall additional Software #
#####

```

```

##### uninstall Microsoft Edge WebView2 and/or Microsoft Windows Desktop Runtime 6.x if enabled
If($WebViewUninstall -eq $true)
{
    $WebViewUninstallString = get-uninstallstring -PowerShell64 $PS64 -App WebView2 -Value UninstallString

    If($null -ne $WebViewUninstallString)
    {
        Write-Host "WebView2 Runtime found"
        $WebViewUninstallString = $WebViewUninstallString + " --force-uninstall"
        Start-Process cmd.exe -ArgumentList '/c',"$WebViewUninstallString" -Wait -NoNewWindow

        $UninstallResultWebView = Get-installedcheck -AppSearchString $AppSearch

        If ($UninstallResultWebView -eq $true)
        {
            Write-Host "uninstall is unsuccessful" -BackgroundColor Red

            $UninstallData = [PSCustomObject]@{
                Software = "WebView2 Runtime"
                Version = ""
                Uninstall = $false
            } | ConvertTo-Json

            Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData
        }
        Else
        {
            Write-Host "uninstall is successful" -BackgroundColor Green

            $UninstallData = [PSCustomObject]@{
                Software = "WebView2 Runtime"
                Version = ""
                Uninstall = $true
            } | ConvertTo-Json

            Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
        }
    }
}

If($RuntimeUninstall -eq $true)
{
    $Runtime6UninstallString = get-uninstallstring -PowerShell64 $PS64 -App Runtime6 -Value Quietuninstallstring

    If($null -ne $Runtime6UninstallString)
    {
        Write-Host "Desktop Runtime 6 found"
        Start-Process cmd.exe -ArgumentList '/c',"$Runtime6UninstallString" -Wait -NoNewWindow

        $UninstallResultDTRuntime = Get-installedcheck -AppSearchString $AppSearch

        If ($UninstallResultDTRuntime -eq $true)
        {
            Write-Host "uninstall is unsuccessful" -BackgroundColor Red

            $UninstallData = [PSCustomObject]@{
                Software = "Microsoft Desktop Runtime 6"
                Version = ""
                Uninstall = $false
            } | ConvertTo-Json

            Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData
        }
        Else
        {
            Write-Host "uninstall is successful" -BackgroundColor Green
        }
    }
}

```

```

$UninstallData = [PSCustomObject]@{
    Software = "Microsoft Desktop Runtime 6"
    Version = ""
    Uninstall = $true
} | ConvertTo-Json

Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData

}

}

If($DDMUninstall -eq $true)
{
$DDMUninstallString = get-uninstallstring -PowerShell64 $PS64 -App DDM -Value UninstallString

If($null -ne $DDMUninstallString)
{
    Write-Host "Display Manager 2"
    Start-Process -FilePath $DDMUninstallString -ArgumentList "/S" -Wait -NoNewWindow

    UninstallResultDDM2 = Get-installedcheck -AppSearchString $AppSearch

    If ($UninstallResultDDM2 -eq $true)
    {
        Write-Host "uninstall is unsuccessful" -BackgroundColor Red

        $UninstallData = [PSCustomObject]@{
            Software = "Display Manager 2"
            Version = ""
            Uninstall = $false
        } | ConvertTo-Json

        Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData
    }
}
Else
{
    Write-Host "uninstall is successful" -BackgroundColor Green

    $UninstallData = [PSCustomObject]@{
        Software = "Display Manager 2"
        Version = ""
        Uninstall = $true
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
}
}
else
{
# Cover problem of PS32 can not access 64 Bit Registry
$CheckPath = Test-Path "C:\Program Files\Display Manager 2\uninst.exe"

If ($CheckPath -eq $true)
{
    Write-Host "Display Manager 2"
    $DDMUninstallString = "C:\Program Files\Display Manager 2\uninst.exe /S"
    Start-Process -FilePath "C:\Program Files\Display Manager 2\uninst.exe" -ArgumentList "/S" -Wait -NoNewWindow

    $UninstallResultDDM2 = Get-installedcheck -AppSearchString $AppSearch

    If ($UninstallResultDDM2 -eq $true)
    {
        Write-Host "uninstall is unsuccessful" -BackgroundColor Red

        $UninstallData = [PSCustomObject]@{
            Software = "Display Manager 2"
            Version = ""
            Uninstall = $false
        } | ConvertTo-Json
    }
}
}

```

```

        Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData

    }

Else
{
    Write-Host "uninstall is successful" -BackgroundColor Green

    $UninstallData = [PSCustomObject]@{
        Software = "Display Manager 2"
        Version = ""
        Uninstall = $true
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
}

}
}

If($DPMUninstall -eq $true)
{
    $DPMUninstallString = Get-CimInstance -ClassName Win32_Product -Filter "Name like '%Dell%Power%Manager%'" | Select-Object -ExpandProperty IdentifyNumber

    If($null -ne $DPMUninstallString)
    {
        Write-Host "Dell Power Manager Service"
        Start-Process -FilePath msiexec.exe -ArgumentList "/x $DPMUninstallString /qn" -Wait -NoNewWindow

        $UninstallResultDPM = Get-installedcheck -AppSearchString $AppSearch

        If ($UninstallResultDPM -eq $true)
        {
            Write-Host "uninstall is unsuccessful" -BackgroundColor Red

            $UninstallData = [PSCustomObject]@{
                Software = "Dell Power Manager Service"
                Version = ""
                Uninstall = $false
            } | ConvertTo-Json

            Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData
        }

Else
{
    Write-Host "uninstall is successful" -BackgroundColor Green

    $UninstallData = [PSCustomObject]@{
        Software = "Dell Power Manager Service"
        Version = ""
        Uninstall = $true
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
}
}
}

```

Detection Script

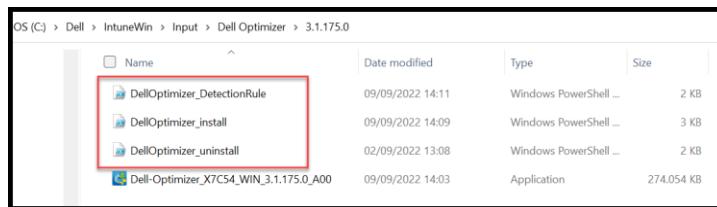
The detection script showing success of installation. It could be done as well without a script but again it is helpful for later automation of upload processes in the future.

The yellow marked version must be adjusted with each new version.

```
#####
# Program EXE with target Version
#####
$ProgramVersion_target = '4.0.310.0' # need to be the same like the exe file
$ProgramVersion_current = Get-CimInstance -ClassName Win32_Product -Filter "Name like '%Dell %Optimizer%'" | select -ExpandProperty Version

if($ProgramVersion_current -eq $ProgramVersion_target)
{
    Write-Host "Found it!"
}
```

Please, copy these files in the same folder as your EXE Installer File.



Start the Microsoft Win32 Content Prep Tool (aka IntuneAppUtil.exe). In case you are not familiar with this tool, you will find documentation here: <https://docs.microsoft.com/en-us/mem/intune/apps/apps-win32-prepare>

| Argument | Value |
|---------------|---|
| Source Folder | folder where you have stored the unzipped MSI |
| Setup file | Main installer file like msi/exe/ps1, etc. |
| Output Folder | where you want to store the IntuneWin |

```
Please specify the source folder: C:\Dell\IntuneWin\Input\Dell Optimizer\3.1.175.0
Please specify the setup file: Dell-Optimizer_X7C54_WIN_3.1.175.0_A00.exe
Please specify the output folder: C:\Dell\IntuneWin\Output\Dell Optimizer\3.1.175.0
Do you want to specify catalog folder (Y/N)?n
```

IntuneWin is now prepared and ready for installation by Intune.



Script for Microsoft .net Runtime

Install Script

The script makes a precheck if older versions are installed and starting an uninstall first to have a clean environment.

```
##### Variables
$InstallerName = Get-ChildItem .\*.exe | Select-Object -ExpandProperty Name
$ProgramPath = ".\" + $InstallerName
[Version]$ProgramVersion_target = (Get-Command $ProgramPath).FileVersionInfo.ProductVersion
[Version]$ProgramVersion_current = Get-ChildItem -Path HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall | Get-ItemProperty | Where-Object {$_._DisplayName -like "Microsoft .NET Runtime - 6.*(x64)" } | Select-Object -ExpandProperty DisplayName

#####
#Checking if older Version is installed and uninstall this Version
#####

If ($ProgramVersion_current -ne $null)
{
    if ($ProgramVersion_target -gt $ProgramVersion_current)
    {
        Start-Process cmd.exe -ArgumentList '/c'$ApplicationID_current -Wait -NoNewWindow
    }
    Else
    {
        Write-Host "same version is installed"
        Exit 0
    }
}

#####
#Install new Software
#####

Start-Process -FilePath "$ProgramPath" -ArgumentList "/install /quiet /norestart" -Wait
```

Uninstall Script

The script starts the uninstalling of application.

```
##### Variables
$ApplicationID_current = Get-ChildItem -Path HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall | Get-ItemProperty | Where-Object {$_._DisplayName -like "Microsoft .NET Runtime - 6.*(x64)" } | Select-Object -ExpandProperty QuietUninstallString

#####
#uninstall Software
#####

Start-Process cmd.exe -ArgumentList '/c'$ApplicationID_current -Wait -NoNewWindow
```

Detection Script

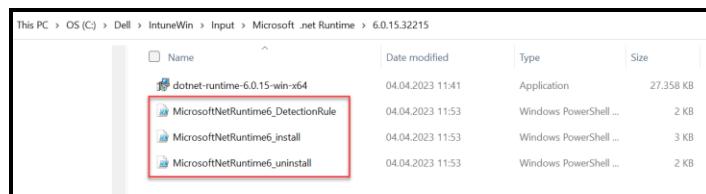
The detection script showing success of installation. It could be done as well without a script but again it is helpful for later automation of upload processes in the future.

The yellow marked version must be adjusted with each new version.

```
#####
# Program with target Version
#####
$ProgramVersion_target = [6.0.16.32327] # need to be the same like the exe file
$ProgramVersion_current = Get-ChildItem -Path HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall | Get-ItemProperty
| Where-Object {$_.DisplayName -like "Microsoft .NET Runtime - 6.*(x64)"} | Select-Object -ExpandProperty DisplayName

if($ProgramVersion_current -eq $ProgramVersion_target)
{
    Write-Host "Found it!"
}
```

Please, copy these files in the same folder as your EXE Installer File.

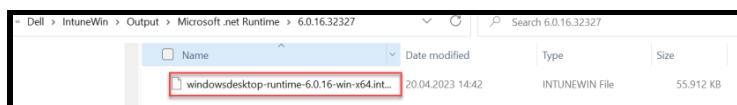


Start the Microsoft Win32 Content Prep Tool (aka IntuneAppUtil.exe). In case you are not familiar with this tool, you will find documentation here: <https://docs.microsoft.com/en-us/mem/intune/apps/apps-win32-prepare>

| Argument | Value |
|---------------|---|
| Source Folder | folder where you have stored the unzipped MSI |
| Setup file | Main installer file like msi/exe/ps1, etc. |
| Output Folder | where you want to store the IntuneWin |

```
Please specify the source folder: C:\Dell\IntuneWin\Input\Microsoft .NET Runtime\6.0.16.32327
Please specify the setup file: windowsdesktop-runtime-6.0.16-win-x64.exe
Please specify the output folder: C:\Dell\IntuneWin\Output\Microsoft .NET Runtime\6.0.16.32327
Do you want to specify catalog folder (Y/N)?
```

IntuneWin is now prepared and ready for installation by Intune.



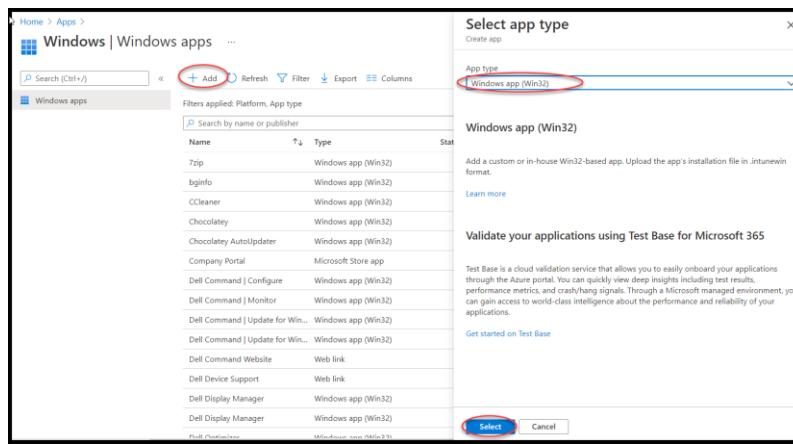
Import and Deployment settings Dell Optimizer for Intune Part for Microsoft .net Runtime (Dependency)

Click 'Add'

| Field | Value |
|---------------|---------------------|
| Source Folder | Windows app (Win32) |

Click 'Select'

Select Windows app (Win32) as application.



Section 'App information'

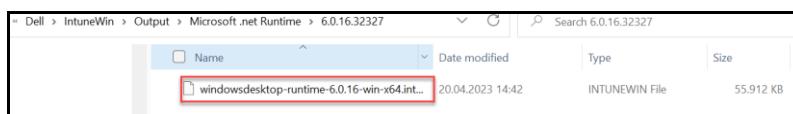


Click 'Select app package file'

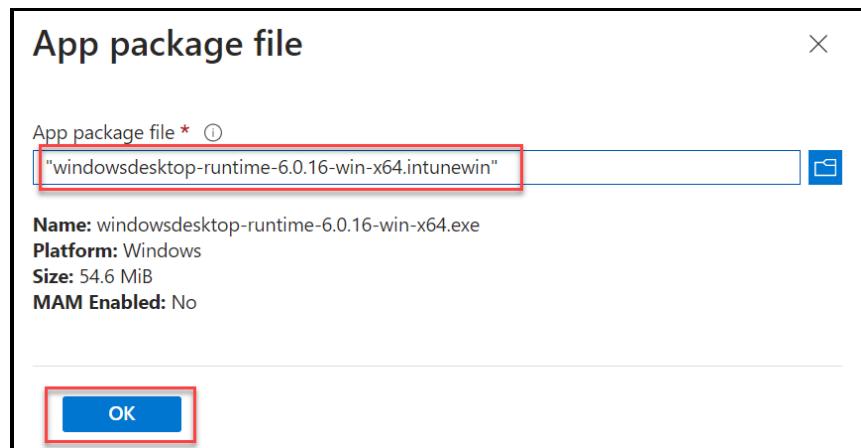
Click 'Folder'



Select 'windowsdesktop-runtime-6.0.16-win-x64.intunewin'



Click 'OK'



| Field | Value |
|---|--|
| Name | Microsoft Windows Desktop Runtime 6 |
| Publisher | Microsoft |
| App Version | 6.0.16.32327 |
| | Note: Use version of the Microsoft .NET |
| Show this as a featured app in the Company Portal | No |

Click 'Next'

Home > Apps | Windows > Windows | Windows apps >

Add App ...

Windows app (Win32)

Select file * ⓘ windowsdesktop-runtime-6.0.16-win-x64.intunewin

Name * ⓘ Microsoft Windows Desktop Runtime 6

Description * ⓘ windowsdesktop-runtime-6.0.16-win-x64.exe

Edit Description

Publisher * ⓘ Microsoft

App Version ⓘ 6.0.16.32327

Category ⓘ 0 selected

Show this as a featured app in the Company Portal ⓘ Yes **No**

Information URL ⓘ Enter a valid url

Privacy URL ⓘ Enter a valid url

Developer ⓘ

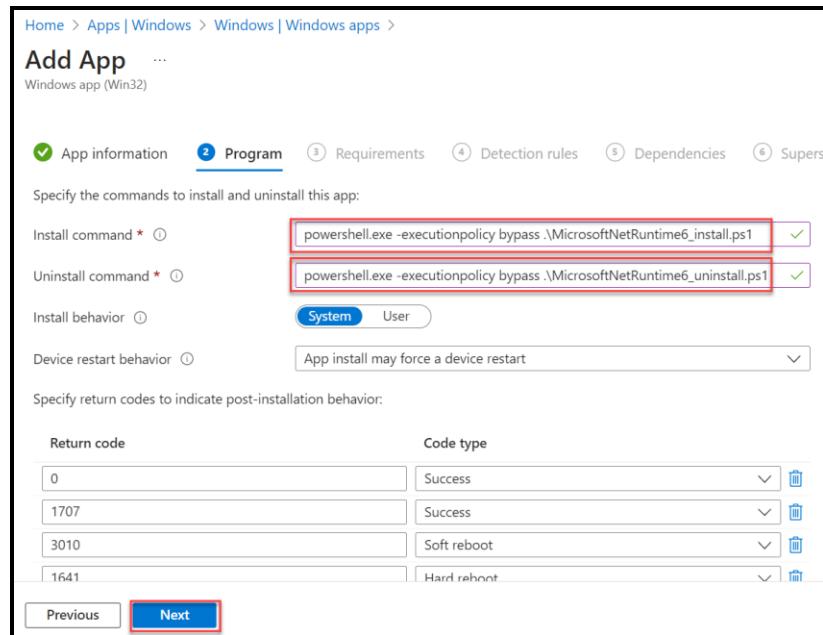
Previous Next

Section 'Program'



| Field | Value |
|-------------------|---|
| Install command | powershell.exe -executionpolicy bypass .\\MicrosoftNetRuntime6_install.ps1 |
| Uninstall command | powershell.exe -executionpolicy bypass .\\MicrosoftNetRuntime6_uninstall.ps1 |

Click 'Next'



Section 'Requirements'

The screenshot shows the 'Add App' interface with the 'Requirements' tab selected. The URL in the address bar is 'Home > Apps > Windows > Add App'. Below it, it says 'Windows app (Win32)'. The tabs at the top are: App information (green checkmark), Program (green checkmark), Requirements (blue circle with a dot, highlighted with a red oval), Detection rules (grey), Dependencies (grey), Supersedence (preview) (grey), Assignments (grey), and Review + create (grey). The 'Requirements' tab is active.

| Field | Value |
|-------------------------------|--|
| Operating system architecture | 64-Bit |
| Minimum operating system | 2004 (Please note Dell support drivers and software only with the latest Win version + N -2) |

Click 'Next'

The screenshot shows the 'Add App' interface on the 'Requirements' step. The URL is 'Home > Apps | Windows > Windows | Windows apps > Add App'. The tabs at the top are: App information (green checkmark), Program (green checkmark), Requirements (blue circle with a dot, highlighted with a red oval), Detection rules (grey), Dependencies (grey). The sub-section title is 'Specify the requirements that devices must meet before the app is installed:'. The fields filled are: 'Operating system architecture *' (64-bit) and 'Minimum operating system *' (Windows 10 2004). Other fields like 'Disk space required (MB)', 'Physical memory required (MB)', 'Minimum number of logical processors required', and 'Minimum CPU speed required (MHz)' are empty. At the bottom are 'Previous' and 'Next' buttons, with 'Next' being highlighted in blue.

Section 'Detection rules'

The screenshot shows the 'Add App' interface with the 'Detection rules' tab highlighted. Other tabs like 'App information', 'Program', 'Requirements', 'Dependencies', 'Supersedeance (preview)', 'Assignments', and 'Review + create' are also visible.

| Field | Value |
|--------------|------------------------|
| Rules format | Use a custom detection |

Click 'Folder'

The screenshot shows the 'Add App' interface under the 'Detection rules' tab. A red box highlights the 'Select a file' button next to the 'Script file' input field. Other settings shown include 'Rules format' set to 'Use a custom detection script' and 'Run script as 32-bit process on 64-bit clients' set to 'No'.

Select 'MicrosoftNetRuntime6_DetectionRule.ps1'

The screenshot shows a file explorer window with the path 'Dell > IntuneWin > Input > Microsoft .net Runtime > 6.0.15.32215'. A red box highlights the file 'MicrosoftNetRuntime6_DetectionRule.ps1' in the list, which is the PowerShell script for detection.

Click 'Next'

Home > Apps | Windows > Windows | Windows apps >

Add App

Windows app (Win32)

App information Program Requirements Detection rules (5)

Configure app specific rules used to detect the presence of the app.

Rules format * ⓘ Use a custom detection script

Script file ⓘ MicrosoftNetRuntime6_DetectionRule.ps1

Script content

```
ÿ»¿<#  
_author_ = Sven Riebe <sven_riebe@Dell.com>  
_twitter_ = @SvenRiebe  
_version_ = 1.0  
_Dev_Status_ = Test  
Copyright © 2022 Dell Inc. or its subsidiaries. All Rights Reserved.
```

Run script as 32-bit process on 64-bit clients ⓘ Yes No

Enforce script signature check and run script silently ⓘ Yes No

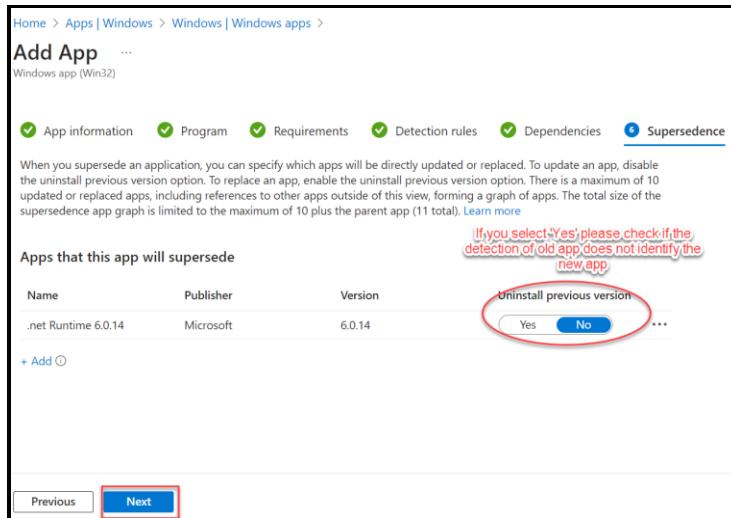
Previous Next

Section 'Dependencies'



No changes

Section 'Supersedence'



No changes

Note: You can also uninstall old software via Supersede, but make sure that the detection of the old application does not also detect the new one, otherwise you will create an installation loop.

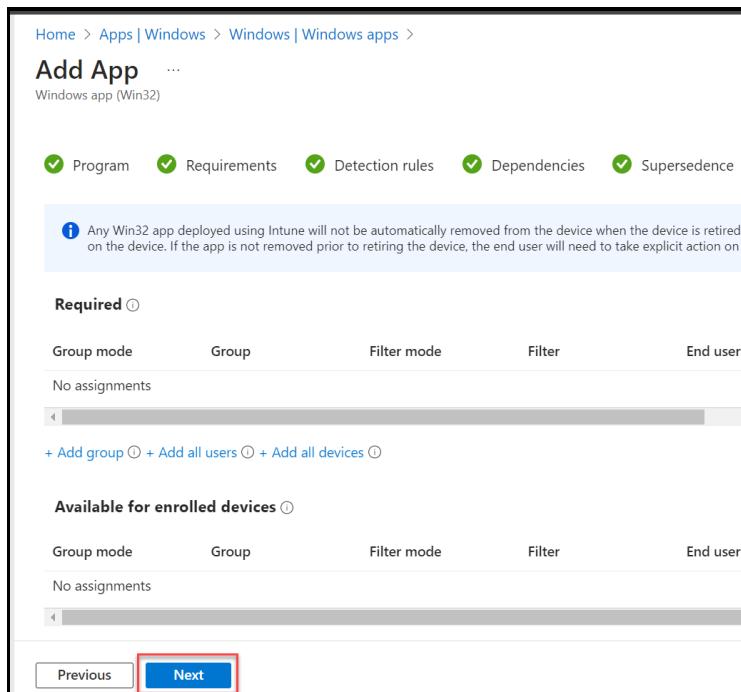
Section ‘Assignments’



No changes

| Option | Value |
|--------------------------------|-------|
| Required | |
| Available for enrolled devices | |
| Uninstall | |

Click ‘Next’



The app is now finished

Click 'Create'

This screenshot shows the 'Review + create' step of the 'Add App' wizard. At the top, there are several green checkmarks indicating successful validation: Requirements, Detection rules, Dependencies, Supersedence, and Assignments. The 'Review + create' button is highlighted with a blue border.

Summary

App information

| | |
|---|---|
| App package file | windowsdesktop-runtime-6.0.16-win-x64.intunewin |
| Name | Microsoft Windows Desktop Runtime 6 |
| Description | windowsdesktop-runtime-6.0.16-win-x64.exe |
| Publisher | Microsoft |
| App Version | 6.0.16.32327 |
| Category | -- |
| Show this as a featured app in the Company Portal | No |
| Information URL | -- |
| Privacy URL | -- |
| Developer | -- |

Buttons: Previous, Create (highlighted), Next

Ready to work.

This screenshot shows the 'Windows | Windows apps' list page. The search bar contains 'Microsoft Windows Desktop Runtime 6'. A red oval highlights the 'Assigned' column for the first item, which is listed as 'No'.

| Name | Type | Status | Version | Assigned | ... |
|--------------------------|---------------------|--------|--------------|----------|-----|
| Microsoft Windows Des... | Windows app (Win32) | Normal | 6.0.16.32327 | No | ... |

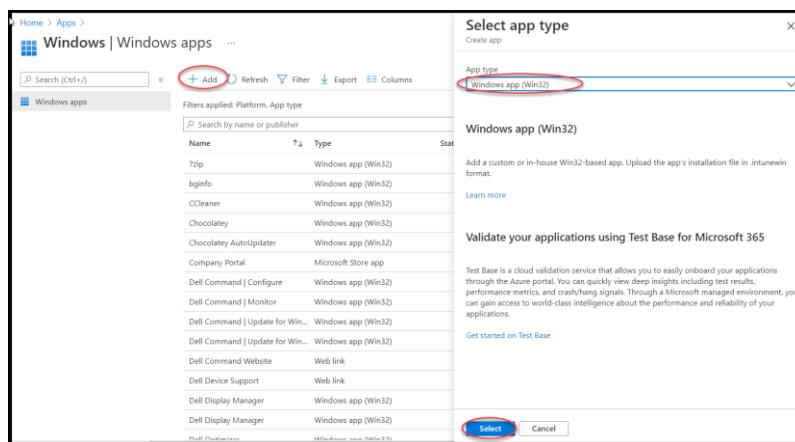
Part for Dell Optimizer

Click 'Add'

| Field | Value |
|---------------|---------------------|
| Source Folder | Windows app (Win32) |

Click 'Select'

Select Windows app (Win32) as application.



Section 'App information'



Click 'Select app package file'

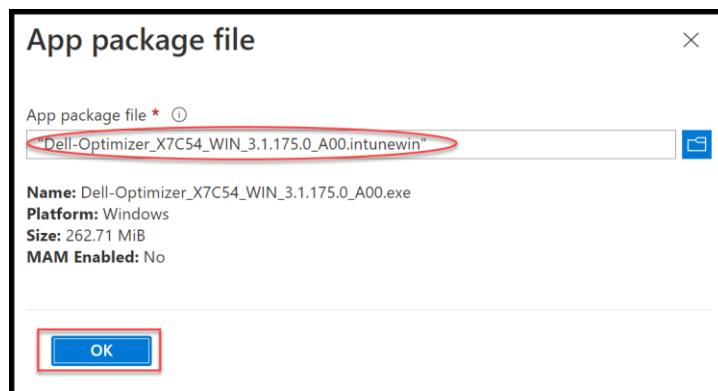
Click 'Folder'



Select 'Dell-Optimizer_X7C54_WIN_3.1.175.0_A00.intunewin'



Click 'OK'



| Field | Value |
|---|---|
| Name | Dell Optimizer |
| Publisher | Dell Inc. |
| App Version | 3.1.175.0 Note: Use version of the Dell Optimizer |
| Show this as a featured app in the Company Portal | Yes |

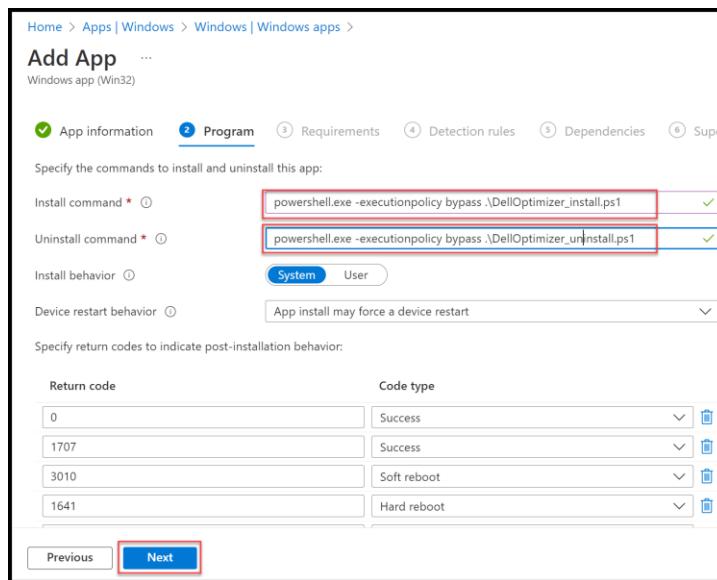
Click 'Next'

Section 'Program'



| Field | Value |
|-------------------|--|
| Install command | powershell.exe -executionpolicy bypass .\\DellOptimizer_install.ps1 |
| Uninstall command | powershell.exe -executionpolicy bypass .\\DellOptimizer_uninstall.ps1 |

Click 'Next'



Section 'Requirements'

The screenshot shows the 'Add App' interface with the 'Requirements' tab selected. The URL in the address bar is 'Home > Apps > Windows > Add App'. Below the address bar, it says 'Windows app (Win32)'. The tabs at the top are: App information (green checkmark), Program (green checkmark), Requirements (blue circle with a red border, indicating it's selected), Detection rules, Dependencies, Superseding (preview), Assignments, and Review + create.

| Field | Value |
|-------------------------------|--|
| Operating system architecture | 64-Bit |
| Minimum operating system | 2004 (Please note Dell support drivers and software only with the latest Win version + N -2) |

Click 'Next'

The screenshot shows the 'Add App' interface on the 'Requirements' step. The URL in the address bar is 'Home > Apps | Windows > Windows | Windows apps > Add App'. Below the address bar, it says 'Windows app (Win32)'. The tabs at the top are: App information (green checkmark), Program (green checkmark), Requirements (blue circle with a red border, indicating it's selected), Detection rules, Dependencies, and Superseding (preview). The main area asks 'Specify the requirements that devices must meet before the app is installed:'. It includes fields for Operating system architecture (set to '64-bit') and Minimum operating system (set to 'Windows 10 2004'). Other fields like Disk space required, Physical memory required, Minimum number of logical processors required, and Minimum CPU speed required are present but empty. At the bottom are 'Previous' and 'Next' buttons, with 'Next' being highlighted in blue.

Section 'Detection rules'

The screenshot shows the 'Add App' interface with the 'Detection rules' tab highlighted. Other tabs like 'App information', 'Program', 'Requirements', 'Dependencies', 'Supersedeance (preview)', 'Assignments', and 'Review + create' are also visible.

| Field | Value |
|--------------|------------------------|
| Rules format | Use a custom detection |

Click 'Folder'

The screenshot shows the 'Add App' interface under the 'Detection rules' tab. The 'Rules format' dropdown is set to 'Use a custom detection script'. Below it, the 'Script file' input field is highlighted with a red box. Other options like 'Run script as 32-bit process on 64-bit clients' and 'Enforce script signature check and run script silently' are also visible.

Select 'DellOptimizer_DetectionRule.ps1'

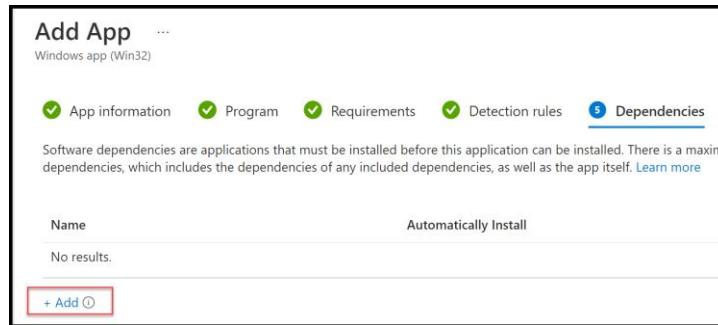
The screenshot shows a file explorer window displaying PowerShell scripts in a folder path: 'This PC > OS (C) > Dell > IntuneWin > Input > Dell Optimizer > 3.1.175.0'. The file 'DellOptimizer_DetectionRule.ps1' is selected and highlighted with a red box.

Click 'Next'

The screenshot shows the 'Add App' interface under the 'Detection rules' tab. The 'Script file' input field now contains the path 'DellOptimizer_DetectionRule.ps1', which is also highlighted with a red box. The 'Next' button at the bottom is also highlighted with a red box.

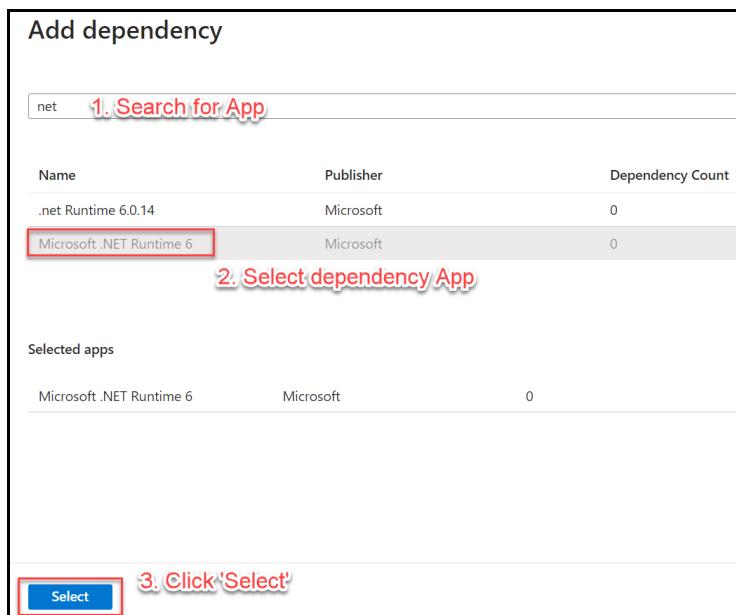
Section 'Dependencies'

Click 'Add'



Search for the new created dependency Application 'Microsoft Windows Desktop Runtime 6' and select this Application.

Click 'Select'



| Field | Value |
|-----------------------|-------|
| Automatically Install | Yes |

Click 'Next'

Home > Apps | All apps >
Add App ...
Windows app (Win32)

App information Program Requirements Detection rules Dependencies

Software dependencies are applications that must be installed before this application can be installed. To automatically child dependency app before installing the current parent app, enable the automatically install option. To only install the parent app if the child dependency app is already detected on the device, disable the automatically install option. The maximum of 100 child dependency apps, including references to other apps outside of this view, forming a graph of a total size of the dependency app graph is limited to the maximum of 100 plus the parent app (101 total). [Learn more](#)

| Name | Automatically Install |
|--------------------------|---|
| Microsoft .NET Runtime 6 | <input checked="" type="button"/> Yes <input type="button"/> No |

+ Add ⓘ

Previous Next

Section 'Supersedence'

Home > Apps | Windows > Windows | Windows apps >
Add App ...
Windows app (Win32)

Program Requirements Detection rules Dependencies Supersedence (preview)

When you supersede an application, you can specify which app will be updated or replaced. To update an app, disable the uninstall previous version option. To replace an app, enable the uninstall previous version option. There is a maximum of 10 updated or replaced apps, including references to other apps. For example, your app references another app. This other app references other apps, and so on. This scenario creates a graph of apps. All apps in the graph count toward the maximum value of 10. [Learn more](#)

If you select 'Yes' please check if the detection of old app does not identify the new app.

Apps that this app will supersede

| Name | Publisher | Version | Uninstall previous version |
|----------------|-----------|-----------|---|
| Dell Optimizer | Dell Inc. | 2.0.753.0 | <input checked="" type="button"/> Yes <input type="button"/> No |

+ Add ⓘ

Previous Next

No changes

Note: You can also uninstall old software via Supersede, but make sure that the detection of the old application does not also detect the new one, otherwise you will create an installation loop.

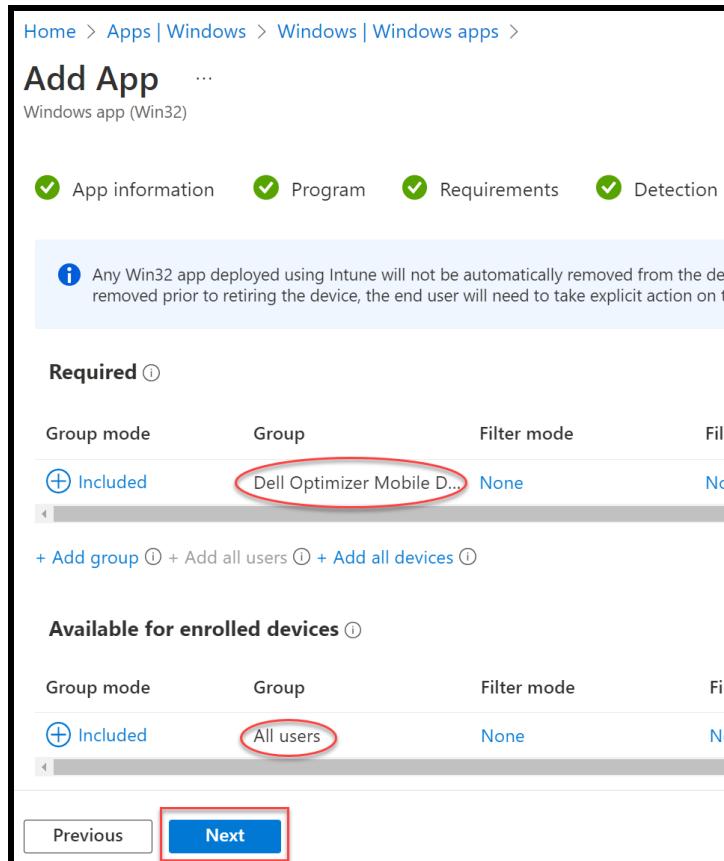
Section 'Assignments'



The Dell Optimizer supports only Dell (Latitude, Optiplex, Precision and mobile XPS with Intel Core CPU 11G and newer) it makes sense to have a dynamic group which only incl. these systems.

| Option | Value |
|--------------------------------|------------------------------------|
| Required | Add Group 'Dell Optimizer Devices' |
| Available for enrolled devices | Add Group 'All User' |
| Uninstall | |

Click 'Next'



The app is now finished

Click 'Create'

Home > Apps | Windows > Windows | Windows apps >

Add App

Windows app (Win32)

Program Requirements Detection rules Dependencies Supersedence (preview)

Summary

App information

| | |
|---|--|
| App package file | Dell-Optimizer_X7C54_WIN_3.1.175.0_A00.intunewin |
| Name | Dell Optimizer |
| Description | Dell-Optimizer_X7C54_WIN_3.1.175.0_A00.exe |
| Publisher | Dell Inc. |
| App Version | 3.1.175.0 |
| Category | -- |
| Show this as a featured app in the Company Portal | Yes |
| Information URL | -- |
| Privacy URL | -- |

Previous Create

Ready to work.

Windows apps

Filters applied: Platform, App type

| Name | Type | Status | Version | Assigned |
|----------------|---------------------|-----------|---------|----------|
| Dell Optimizer | Windows app (Win32) | 3.1.175.0 | | Yes |
| Dell Optimizer | Windows app (Win32) | 2.0.753.0 | | No |

Dell Peripheral Manager

Introduction

The Dell Peripheral Manager software enables you to easily customize and manage your Dell peripherals such as keyboards, mice, speakerphones, stylus/pens and webcams. The software allows you to customize your settings and get the latest firmware updates.

Note: Dell Peripheral Manager supports RF pairing for the following devices: KM714, KM636, WM514, WM326, WM126, KM717, WM527, WK717

Dell Peripheral Manager automatically installs to your system when you add any supported Dell accessory.

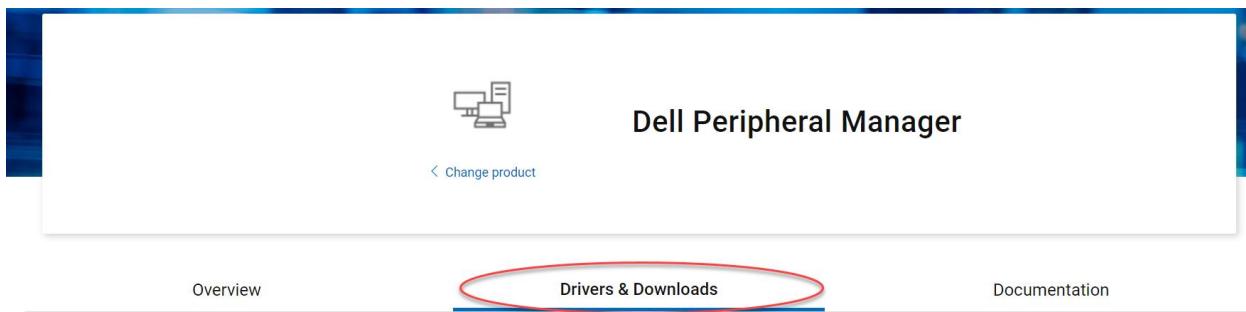
Dell Peripheral Manager User Guide

<https://dl.dell.com/content/manual14690880-dell-peripheral-manager-user-guide.pdf?language=en-us>

Prepare Dell Peripheral Manager for Intune

Download the newest Version of Dell Command | Update from our website.

<https://www.dell.com/support/home/en-us/product-support/product/dell-peripheral-manager/drivers>



'Click' Download

| Dell Peripheral Manager (1) | | | | |
|-----------------------------|-------------------------|-------------|-------------|--------------|
| | NAME | IMPORTANCE | CATEGORY | RELEASE DATE |
| | Dell Peripheral Manager | RECOMMENDED | Application | 31 Mar 2023 |

If you have downloaded the file from <https://www.dell.com/support>, copy file to your software repository for the next step.

| This PC > OS (C) > Dell > SoftwareRepository > Dell Peripheral Manager > 1.6.6 | | | | |
|--|--------------------------|------------------|-------------|------------|
| | Name | Date modified | Type | Size |
| | DPM_KYFGD_1.6.6.WN64_A00 | 16.05.2023 14:13 | Application | 148.131 KB |

Scripts for Install, Uninstall and Detection

It is possible to use the native file for installation. In this document we are using PowerShell scripts to cover different scenarios of Install new, Update, and uninstall for a later automation of uploading applications by API.

All script could be download on Github Repository:

<https://github.com/svenriebedell/Dell-Tools-Intune-Install>

Install Script

The script makes a precheck if older versions are installed and starting an uninstall first to have a clean environment.

```
#####
##### variable section #####
#####
$InstallerName = Get-ChildItem .\*.exe | Select-Object -ExpandProperty Name
$ProgramPath = "." + $InstallerName
[Version]$ProgramVersion_target = (Get-Command $ProgramPath).FileVersionInfo.ProductVersion
$SoftwareName = "Dell Peripheral Manager"

#####
##### program section #####
#####

##### generate Logging Resources
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

##### Check first if Dell Peripheral Manager is installed by looking file path
$CheckInstall = Test-path -path 'C:\Program Files\Dell\Dell Peripheral Manager\DP.M.exe'

if($CheckInstall -eq $true)
{
    [Version]$ProgramVersion_current = (Get-ItemProperty 'C:\Program Files\Dell\Dell Peripheral Manager\DP.M.exe').VersionInfo | Select-Object -ExpandProperty ProductVersion
    $ApplicationPath = "C:\Program Files\Dell\Dell Peripheral Manager"
    $NameUninstallFile = "Uninstall.exe"
}
else
{
    [Version]$ProgramVersion_current = $null
}

#####
#Checking if older Version is installed and uninstall this Version #
#####

If ($ProgramVersion_current -ne $null)
{
    if ($ProgramVersion_target -gt $ProgramVersion_current)
    {
        Start-Process -FilePath $ApplicationPath\$NameUninstallFile -ArgumentList "/S" -Wait

        #####
        # uninstall success check #
        #####
        $UninstallResult = Test-path -path 'C:\Program Files\Dell\Dell Peripheral Manager\DP.M.exe'

        If ($UninstallResult -eq $true)
        {
            Write-Host "uninstall is unsuccessful" -BackgroundColor Red

            $UninstallData = [PSCustomObject]@{
                Software = $SoftwareName
                Version = ($ProgramVersion_current).ToString()
                Uninstall = $false
            } | ConvertTo-Json

            Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData
        }
    }
    Else
    {
        Write-Host "uninstall is successful" -BackgroundColor Green

        $UninstallData = [PSCustomObject]@{
            Software = $SoftwareName
            Version = ($ProgramVersion_current).ToString()
            Uninstall = $true
        }
    }
}
```

```

        } | ConvertTo-Json

        Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
    }

}

Else
{
    Write-Host "same version is installed"

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $false
        Reason = "same version is installed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData

    Exit 0
}

#####
#Install new Software
#####

Start-Process -FilePath "$ProgramPath" -ArgumentList "/S" -Wait

#####
# install success check
#####
$UninstallResult = Test-path -path 'C:\Program Files\DELL\DELL Peripheral Manager\DPM.exe'

If ($UninstallResult -ne $true)
{
    Write-Host "install is unsuccessful" -BackgroundColor Red

    $UninstallData = [PSCustomObject]@{
        Software = $SoftwareName
        Version = ($ProgramVersion_target).ToString()
        Install = $false
        Reason = "Installation failed"
    } | ConvertTo-Json

    Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Error -EventId 11 -Message $UninstallData
}

Else
{
    [Version]$ProgramVersion_current = (Get-ItemProperty 'C:\Program Files\DELL\DELL Peripheral Manager\DPM.exe').VersionInfo | Select-Object -ExpandProperty ProductVersion

    If ($ProgramVersion_current -ge $ProgramVersion_target)
    {
        Write-Host "install is successful" -BackgroundColor Green

        $UninstallData = [PSCustomObject]@{
            Software = $SoftwareName
            Version = ($ProgramVersion_target).ToString()
            Install = $true
            Reason = "Update/Install/Newer Version"
        } | ConvertTo-Json

        Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData
    }
}

```

```
Write-Host "install is unsuccessful" -BackgroundColor red

$UninstallData = [PSCustomObject]@{
    Software = $SoftwareName
    Version = ($ProgramVersion_target).ToString()
    Install = $false
    Reason = "Older Version installed $ProgramVersion_current"
} | ConvertTo-Json

Write-EventLog -LogName Dell -Source "Dell Software Install" -EntryType Information -EventId 10 -Message $UninstallData

}

}
```

Uninstall Script

The script starts the uninstalling of application.

```
#####
#### program section #####
#####

##### generate Logging Resources
New-EventLog -LogName "Dell" -Source "Dell Software Install" -ErrorAction Ignore
New-EventLog -LogName "Dell" -Source "Dell Software Uninstall" -ErrorAction Ignore

#####
# uninstall Software #
#####

##### Check first if Dell Peripheral Manager is installed by looking registry path
$DDMPATH = $ApplicationPath + "DPM.exe"
$CheckInstall = Test-path -path $DDMPATH

if($CheckInstall -eq $true)
{
    ###### Variables
    $ProgramVersion_current = (Get-ItemProperty $DDMPATH).VersionInfo | Select-Object -ExpandProperty ProductVersion

    #####
    #uninstall Software #
    #####
    Start-Process -FilePath $ApplicationPath\$NameUninstallFile -ArgumentList "/S" -Wait -NoNewWindow

    #####
    # uninstall success check #
    #####
    $UninstallResult = Test-path -path 'C:\Program Files\Del\Peripheral Manager\DPM.exe'

    If ($UninstallResult -eq $true)
    {
        Write-Host "uninstall is unsuccessful" -BackgroundColor Red

        $UninstallData = [PSCustomObject]@{
            Software = $SoftwareName
            Version = $ProgramVersion_current
            Uninstall = $false
        } | ConvertTo-Json

        Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Error -EventId 11 -Message $UninstallData
    }
    Else
    {
        Write-Host "uninstall is successful" -BackgroundColor Green

        $UninstallData = [PSCustomObject]@{
            Software = $SoftwareName
            Version = $ProgramVersion_current
            Uninstall = $true
        } | ConvertTo-Json

        Write-EventLog -LogName Dell -Source "Dell Software Uninstall" -EntryType Information -EventId 10 -Message $UninstallData
    }
}
```

Detection Script

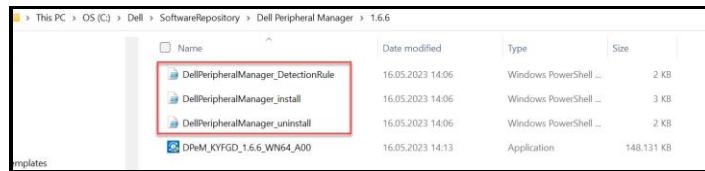
The detection script showing success of installation. It could be done as well without a script but again it is helpful for later automation of upload processes in the future.

The **yellow marked** version must be adjusted with each new version.

```
#####
# Program with target Version
#####
$ProgramVersion_target = '1.6.6' # need to be the same like the exe file
[Version]$ProgramVersion_current = (Get-ItemProperty 'C:\Program Files\DELL\DELL Peripheral Manager\DPM.exe').VersionInfo | Select-Object -ExpandProperty ProductVersion

if($ProgramVersion_current -eq $ProgramVersion_target)
{
    Write-Host "Found it!"
}
```

Please, copy these files in the same folder as your EXE Installer File.



Start the Microsoft Win32 Content Prep Tool (aka IntuneAppUtil.exe). In case you are not familiar with this tool, you will find documentation here: <https://docs.microsoft.com/en-us/mem/intune/apps/apps-win32-prepare>

| Argument | Value |
|---------------|---|
| Source Folder | folder where you have stored the unzipped MSI |
| Setup file | Main installer file like msi/exe/ps1, etc. |
| Output Folder | where you want to store the IntuneWin |

```
Please specify the source folder: C:\Dell\SoftwareRepository\DELL Peripheral Manager\1.6.6
Please specify the setup file: DPeM_KYFGD_1.6.6_WN64_A00.exe
Please specify the output folder: C:\Dell\IntuneWin\Output\DELL Peripheral Manager\1.6.6
Do you want to specify catalog folder (Y/N)?n
```

IntuneWin is now prepared and ready for installation by Intune.



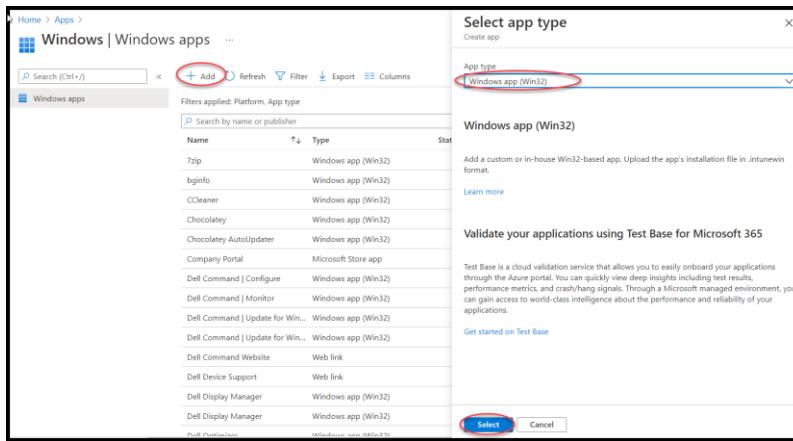
Import and Deployment settings Dell Peripheral Manager for Intune

Click 'Add'

| Field | Value |
|---------------|---------------------|
| Source Folder | Windows app (Win32) |

Click 'Select'

Select Windows app (Win32) as application.



Section 'App information'

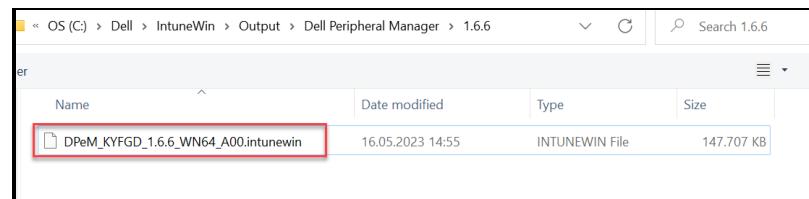


Click 'Select app package file'

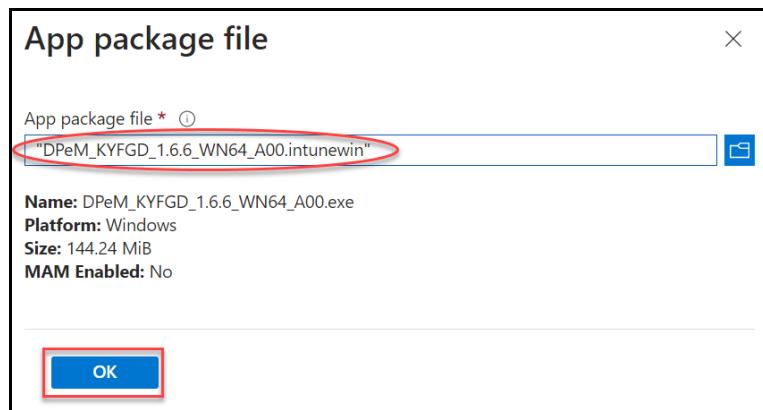
Click 'Folder'



Select 'DPeM_KYFGD_1.6.6_WN64_A00.intunewin'



Click 'OK'



| Field | Value |
|---|--|
| Name | Dell Peripheral Manager |
| Publisher | Dell Inc. |
| App Version | 1.6.6 Note: Use version of the Dell Peripheral Manager |
| Show this as a featured app in the Company Portal | Yes |

Click 'Next'

Home > Apps | Windows > Windows | Windows apps >

Add App ...

Windows app (Win32)

① App information ② Program ③ Requirements ④ Detection rules ⑤ Depen...

Select file * ⓘ DPeM_KYFGD_1.6.6_WN64_A00.intunewin

Name * ⓘ Dell Peripheral Manager

Description * ⓘ DPeM_KYFGD_1.6.6_WN64_A00.exe

Edit Description

Publisher * ⓘ Dell Inc.

App Version ⓘ 1.6.6

Category ⓘ 2 selected

Show this as a featured app in the Company Portal ⓘ Yes

Information URL ⓘ Enter a valid url

Previous Next

Section 'Program'

| Field | Value |
|-------------------|--|
| Install command | powershell.exe -executionpolicy bypass .\DellPeripheralManager_install.ps1 |
| Uninstall command | powershell.exe -executionpolicy bypass .\DellPeripheralManager_uninstall.ps1 |

Click 'Next'

Section 'Requirements'

| Field | Value |
|-------------------------------|--|
| Operating system architecture | 64-Bit |
| Minimum operating system | 2004 (Please note Dell support drivers and software only with the latest Win version + N -2) |

Click 'Next'

Home > Apps | Windows > Windows | Windows apps >
Add App ...
Windows app (Win32)

App information Program Requirements Detection rules Dependencies

Specify the requirements that devices must meet before the app is installed:

Operating system architecture * ① **64-bit**

Minimum operating system * ① **Windows 10 2004**

Disk space required (MB) ①

Physical memory required (MB) ①

Minimum number of logical processors required ①

Minimum CPU speed required (MHz) ①

Previous Next

Section 'Detection rules'

Home > Apps > Windows >
Add App ...
Windows app (Win32)

App information Program Requirements **Detection rules** Dependencies Superseding (preview) Assignments Review + create

| Field | Value |
|--------------|------------------------|
| Rules format | Use a custom detection |

Click 'Folder'

Home > Apps | Windows > Windows | Windows apps >
Add App ...
Windows app (Win32)

App information Program Requirements Detection rules Dependencies

Configure app specific rules used to detect the presence of the app.

Rules format * ① **Use a custom detection script**

Script file ① Select a file

Run script as 32-bit process on 64-bit clients ① Yes No

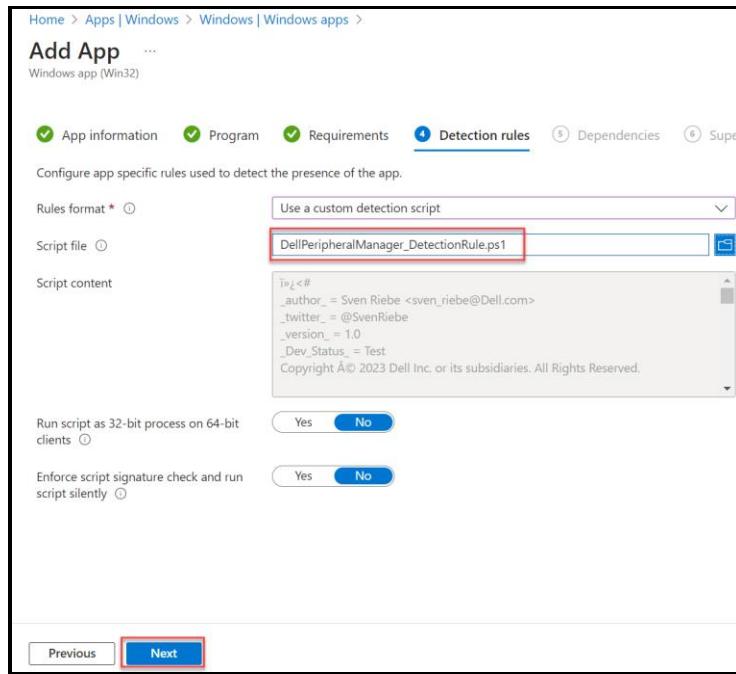
Enforce script signature check and run script silently ① Yes No

Select 'DellPeripheralManager_DetectionRule.ps1'

OS (C) > Dell > SoftwareRepository > Dell Peripheral Manager > 1.6.6

| Name | Date modified | Type | Size |
|-------------------------------------|------------------|------------------------|------|
| DellPeripheralManager_DetectionRule | 16.05.2023 14:06 | Windows PowerShell ... | 2 KB |
| DellPeripheralManager_install | 16.05.2023 14:06 | Windows PowerShell ... | 3 KB |
| DellPeripheralManager_uninstall | 16.05.2023 14:06 | Windows PowerShell ... | 2 KB |

Click 'Next'

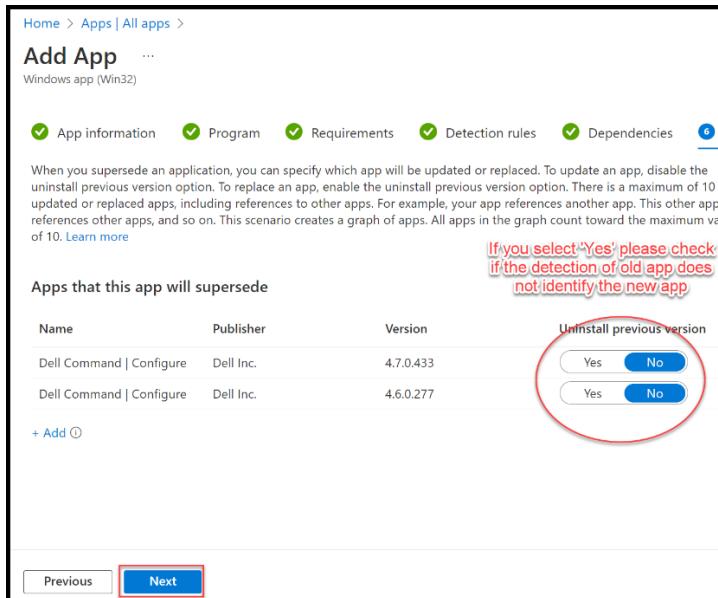


Section 'Dependencies'



No changes

Section ‘Supersedence’



No changes

Note: You can also uninstall old software via Supersede, but make sure that the detection of the old application does not also detect the new one, otherwise you will create an installation loop.

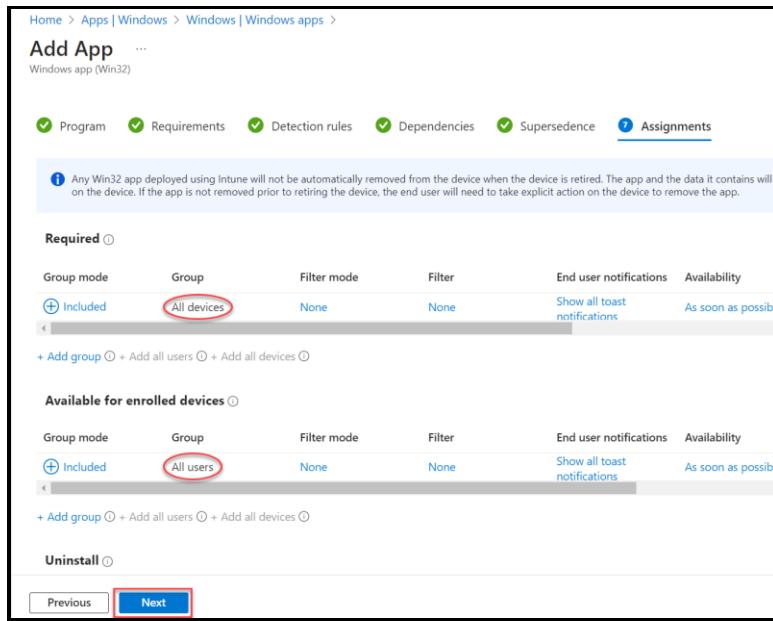
Section ‘Assignments’



The Dell Peripheral Manager supports only Dell (Dell Peripherals like, Webcam, Wireless Keyboards, etc.) it makes sense to have da dynamic group which only incl. these systems. Normally this Application is for Administrators only, we recommend using this Applications only on Administrator Devices if needed.

| Option | Value |
|--------------------------------|-------------------------|
| Required | Add Group ‘Dell Device’ |
| Available for enrolled devices | Add Group ‘All User’ |
| Uninstall | |

Click 'Next'



The app is now finished

Click 'Create'

Home > Apps | Windows > Windows | Windows apps >

Add App

Windows app (Win32)

Detection rules Dependencies Supersedence Assignments Review + create

Summary

App information

| | |
|---|-------------------------------------|
| App package file | DPEM_KYFGD_1.6.6_WN64_A00.intunewin |
| Name | Dell Peripheral Manager |
| Description | DPEM_KYFGD_1.6.6_WN64_A00.exe |
| Publisher | Dell Inc. |
| App Version | 1.6.6 |
| Category | Dell Tools Computer management |
| Show this as a featured app in the Company Portal | Yes |
| Information URL | -- |
| Privacy URL | -- |

Buttons: Previous **Create**

Ready to work.

Home > Apps | Windows > Windows | Windows apps

Application deleted
Application deleted 1

Search Add Refresh Filter Export Columns

Filters applied: Platform, App type

per

| Name | Type | Status | Version | Assigned |
|-------------------------|---------------------|--------|---------|----------|
| Dell Peripheral Manager | Windows app (Win32) | 1.6.6 | Yes | |

References

Customer Ready Scripts

<https://github.com/svenriebedell/Dell-Tools-Intune-Install>

Microsoft IntuneWin converter

<https://docs.microsoft.com/en-us/mem/intune/apps/apps-win32-prepare>

Dell Trusted Device Agent

<https://www.dell.com/support/home/en-us/product-support/product/trusted-device/docs>

<https://www.dell.com/support/home/en-us/product-support/product/trusted-device/drivers>

Dell | Command Power Manager

<https://www.dell.com/support/home/en-us/product-support/product/dell-command-power-manager/docs>

<https://www.dell.com/support/home/en-us/product-support/product/power-manager/drivers>

Dell Command | Update

<https://www.dell.com/support/home/en-us/product-support/product/command-update/docs>

<https://www.dell.com/support/home/en-us/product-support/product/command-update/drivers>

Dell Command | Monitor

<https://www.dell.com/support/home/en-us/product-support/product/command-monitor/docs>

<https://www.dell.com/support/home/en-us/product-support/product/command-monitor/drivers>

Dell Command | Configure

<https://www.dell.com/support/home/en-us/product-support/product/command-configure/docs>

<https://www.dell.com/support/home/en-us/product-support/product/command-configure/drivers>

Dell Display Manager 1

<https://www.dell.com/support/kbdoc/de-de/000060112/what-is-dell-display-manager?lang=en>

<https://www.delldisplaymanager.com/>

Dell Display Manager 2

<https://www.dell.com/support/kbdoc/de-de/000060112/what-is-dell-display-manager?lang=en>
<https://www.dell.com/support/home/en-us/product-support/product/dell-display-peripheral-manager/drivers>

SupportAssist for Business PCs

<https://www.dell.com/support/home/en-us/product-support/product/supportassist-business-pcs/docs>
<https://techdirect.dell.com/>

Dell Optimizer

<https://www.dell.com/support/home/en-us/product-support/product/dell-optimizer/docs>
<https://www.dell.com/support/home/en-us/product-support/product/dell-optimizer/drivers>

Dell Peripheral Manager

<https://www.dell.com/support/home/en-us/product-support/product/dell-peripheral-manager/docs>
<https://www.dell.com/support/home/en-us/product-support/product/dell-peripheral-manager/drivers>