

Launch Universal Windows Platform apps with PowerShell

Last Modified on 04.16.26

Overview

We get many questions about how to launch provisioned [Universal Windows Platform apps](#) from within Application Workspace. In this article, we will use a common app: Snip & Sketch.

UWP apps are designed to work across platforms and can be installed on multiple ones including Windows client, Windows Phone, and Xbox. All UWP apps are also Windows apps, but not all Windows apps are UWP apps.

[Appx](#) is Microsoft's file format used to distribute and install these apps on the Universal Windows Platform. Appx is a .ZIP-based file containing the app's payload files plus info needed to validate, deploy, manage, and update the app.

Some Windows apps are already provisioned in Windows by default. Read [Overview of apps on Windows client devices](#) to find out how you can use Windows PowerShell to find out the name of all these provisioned apps.

Retrieve PackageFamilyName and App ID using PowerShell

Next, we will use PowerShell to find about Snip & Sketch all the details needed to create a Launch type package in the Application Workspace.

We use the `Get-AppxPackage` cmdlet to list information about the Appx package only:

```
Get-AppxPackage | Where {$_.Name -match 'ScreenSketch'}
```

We're going to have a look at three properties here, Name, InstallLocation and PackageFamilyName:

```
Name : Microsoft.ScreenSketch
Publisher : CN=Microsoft Corporation, O=Microsoft Corporation, L=Redmond, S=Washington, C=US
Architecture : X64
ResourceId :
Version : 11.2407.3.0
PackageFullName : Microsoft.ScreenSketch_11.2407.3.0_x64_8wekyb3d8bbwe
InstallLocation : C:\Program Files\WindowsApps\Microsoft.ScreenSketch_11.2407.3.0_x64_8wekyb3d8bbwe
IsFramework : False
PackageFamilyName : Microsoft.ScreenSketch_8wekyb3d8bbwe
PublisherId : 8wekyb3d8bbwe
IsResourcePackage : False
IsBundle : False
IsDevelopmentMode : False
NonRemovable : False
Dependencies : {Microsoft.UI.Xaml.2.8_8.2310.30001.0_x64_8wekyb3d8bbwe, Microsoft.WindowsAppRuntime.1.5_5001.214.1843.0_x64_8wekyb3d8bbwe,
Microsoft.VCLibs.140.00_14.0.33519.0_x64_8wekyb3d8bbwe, Microsoft.VCLibs.140.00.UWPDesktop.14.0.33728.0_x64_8wekyb3d8bbwe...}
IsPartiallyStaged : False
SignatureKind : Store
Status : OK
```

Name – is something we need to make certain, that we are looking at the details of the correct app; the 'Snip & Sketch' app.

InstallLocation – is where we can find the `AppxManifest.xml` file which contains the info the system needs to deploy, display, or update this Windows app. Within it, we can find the ID of the app which in this case is 'App'. But we can also retrieve it by running:

```
(Get-AppxPackage | Where {$_.Name -match 'ScreenSketch'} | Get-AppxPackageManifest).package.applications.application.id
```

Recast

PackageFamilyName – is what we need to tell `explorer.exe` to launch the app. To launch Snip & Sketch using `explorer.exe` successfully, we need to add the "!App" string which is the ID value found in the `AppxManifest.xml`.

Script

```
# Import the Liquit Workspace module from its default location
$moduleName = "Liquit.Server.PowerShell"
$moduleInstalled = Get-Module -ListAvailable | Where-Object { $_.Name -eq $moduleName }

if ($moduleInstalled) {
    Write-Host "The '$moduleName' module is already installed." -NoNewLine -ForegroundColor green
    Import-Module -Name "Liquit.Server.PowerShell"
    Write-Host ""
} else {
    Write-Host "The '$moduleName' module is not installed. Please install the Application Workspace PowerShell Module" -NoNewLine -ForegroundColor yellow
}

Add-Type -AssemblyName System.Windows.Forms
Add-Type -AssemblyName System.Drawing

# Define all Liquit-related variables
$url = $(Write-Host "Type Liquit Zone Url: " -NoNewLine -ForegroundColor green; Read-Host)
$usr = $(Write-Host "Please enter the Username (For example: local\admin): " -NoNewLine -ForegroundColor green; Read-Host)
$password = $(Write-Host "Please enter the Password: " -NoNewLine -ForegroundColor green; Read-Host -AsSecureString)
$credentials = New-Object System.Management.Automation.PSCredential $usr,$password

Connect-LiquitWorkspace -URI "$url" -Credential $credentials

# Define the form parameters
$form = New-Object System.Windows.Forms.Form
$form.Text = 'Provisioned Windows Apps'
$form.Size = New-Object System.Drawing.Size(300,675)
$form.StartPosition = 'CenterScreen'
$form.MinimumSize = New-Object System.Drawing.Size(300,675)

$OKButton = New-Object System.Windows.Forms.Button
$OKButton.Location = New-Object System.Drawing.Point(120,550)
$OKButton.Size = New-Object System.Drawing.Size(75,25)
$OKButton.Text = 'OK'
$OKButton.DialogResult = [System.Windows.Forms.DialogResult]::OK
$OKButton.Anchor = [System.Windows.Forms.AnchorStyles]::Bottom -bor
[System.Windows.Forms.AnchorStyles]::Right
$form.AcceptButton = $OKButton
$form.Controls.Add($OKButton)

$CancelButton = New-Object System.Windows.Forms.Button
$CancelButton.Location = New-Object System.Drawing.Point(195,550)
$CancelButton.Size = New-Object System.Drawing.Size(75,25)
$CancelButton.Text = 'Cancel'
$CancelButton.DialogResult = [System.Windows.Forms.DialogResult]::Cancel
$CancelButton.Anchor = [System.Windows.Forms.AnchorStyles]::Bottom -bor
[System.Windows.Forms.AnchorStyles]::Right
$form.CancelButton = $CancelButton
$form.Controls.Add($CancelButton)

$label = New-Object System.Windows.Forms.Label
$label.Location = New-Object System.Drawing.Point(10,20)
```

Recast

```
$label.Size = New-Object System.Drawing.Size(280,20)
$label.Text = 'Select a Provisioned Windows app:'
$form.Controls.Add($label)

$listBox = New-Object System.Windows.Forms.ListBox
$listBox.Location = New-Object System.Drawing.Point(10,40)
$listBox.Size = New-Object System.Drawing.Size(260,940)
$listBox.Height = 500
$listBox.Anchor = [System.Windows.Forms.AnchorStyles]::Bottom -bor
[System.Windows.Forms.AnchorStyles]::Top -bor
[System.Windows.Forms.AnchorStyles]::Left -bor
[System.Windows.Forms.AnchorStyles]::Right

[void] $listBox.Items.Add('HEIF Image Extensions')
[void] $listBox.Items.Add('Clipchamp')
[void] $listBox.Items.Add('Microsoft Paint')
[void] $listBox.Items.Add('Snip & Sketch')

$form.Controls.Add($listBox)
$form.Topmost = $true

# Show the form
$result = $form.ShowDialog()

if ($result -eq [System.Windows.Forms.DialogResult]::OK)
{
    $packageName = $listBox.SelectedItem

    if ($packageName -eq "Microsoft Paint")
    {
        $selectedAppxPackagename = "Microsoft.Paint"
    }

    elseif ($packageName -eq "Snip & Sketch")
    {
        $selectedAppxPackagename = "Microsoft.ScreenSketch"
    }

    elseif ($packageName -eq "Clipchamp")
    {
        $selectedAppxPackagename = "Clipchamp.Clipchamp"
    }

    elseif ($packageName -eq "HEIF Image Extensions")
    {
        $selectedAppxPackagename = "Microsoft.HEIFImageExtension"
    }

    # Fetch all Appx packages
    $AppxPackages = Get-AppxPackage | Where-Object {$_.Name -match $selectedAppxPackagename}

    foreach ($AppxPackage in $AppxPackages)
    {
        $packageArguments = "$($AppxPackage.PackageFamilyName + "!App")"
        $packageDisplayName = ""
        $packageDescription = ""

        $package = New-LiquidPackage -Name $packageName `
            -Type "Launch" `
            -DisplayName $packageDisplayName `
            -Description $packageDescription `
            -Priority 100 -Enabled $true `
            -Offline $True `
```

Recast

```
        -Web $false `
        -Icon $content
    $snapshot = New-LiquitPackageSnapshot -Package $package
    New-LiquitFilterSet -Snapshot $snapshot

# Define the Launch action set
$actionset = New-LiquitActionSet -snapshot $snapshot `
    -Type Launch `
    -name 'Launch' `
    -Enabled $true `
    -Frequency Always `
    -Process Sequential `

# Define actions for the Launch action set
New-LiquitAction -ActionSet $actionset `
    -Name 'Start Windows app' `
    -Type 'processstart' `
    -Enabled $true `
    -IgnoreErrors $false `
    -Settings @{name = '${SystemRoot}explorer.exe'; parameters = 'shell:Appsfolder' + $packageArgu
ments;} `
    -Context User

# Move the snapshot into the production stage
    Publish-LiquitPackageSnapshot -Snapshot $snapshot -stage Production
}
}
```