Recast

Launch Universal Windows Platform apps with Application Workspace

Last Modified on 2026-11-05

We often receive questions about how to launch provisioned Universal Windows Platform apps (UWP) from within Application Workspace.

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

In this example, we will use PowerShell to find all the Snip & Sketch details needed to create a Custom package in Application Workspace.

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

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

Then look at three properties in the output: 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** – In this example, we're looking for the 'Snip & Sketch' app.
- **InstallLocation** – Displays 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.application
  s.application.id
  ```

- **PackageFamilyName** – Identifies the app. The combination of the PackageFamility Name and ID is needed to launch Snip & Sketch successfully. The ID value can be found in the `AppxManifest.xml`.

Recast

Create an Application Workspace package

We can now bring this all together and create a package in the Application Workspace.

To create the package:

1. Navigate to **Manage** > **Workspace** > **Packages** and click **Create** in the table toolbar.
   - On the **Type** tab, select **Custom package**.
   - On the **Overview** tab
     - Enter the Name of the app, in this case ' ScreenSketch'
     - Select a desired **Icon** from the `C:\Program Files\WindowsApps\Microsoft.ScreenSketch_8wekyb3d8bbwe\Assets` path. We recommend using the highest available resolution.
   - Under **Entitlements**, add the identities to which you want to assign this package.

2. Click **Create action set** in the table toolbar.

3. In the **Create action set** window, as the **Type**, select **Launch**. Click **Confirm**.

4. Click **Create action** in the action set toolbar.

5. In the **Create action** window that opens, as the **Type**, select **Start Windows App** and configure the following:
   - Change the Name, if desired.
   - Enter the **Package Family Name**, previously discovered with the Powershell statement.
   - Enter the **Application Id**, previously discovered with the Powershell statement.