Windows Store Guide

With Windows 10, the good old win32 executable got a new sibling: The Universal Windows Platform. The new `.appx` format does not only enable a number of new powerful APIs like Cortana or Push Notifications, but through the Windows Store, also simplifies installation and updating.

Microsoft developed a tool that compiles Electron apps as `.appx` packages, enabling developers to use some of the goodies found in the new application model. This guide explains how to use it - and what the capabilities and limitations of an Electron AppX package are.

Background and Requirements

Windows 10 "Anniversary Update" is able to run win32 `.exe` binaries by launching them together with a virtualized filesystem and registry. Both are created during compilation by running app and installer inside a Windows Container, allowing Windows to identify exactly which modifications to the operating system are done during installation. Pairing the executable with a virtual filesystem and a virtual registry allows Windows to enable one-click installation and uninstallation.

In addition, the exe is launched inside the appx model - meaning that it can use many of the APIs available to the Universal Windows Platform. To gain even more capabilities, an Electron app can pair up with an invisible UWP background task launched together with the `exe` - sort of launched as a sidekick to run tasks in the background, receive push notifications, or to communicate with other UWP applications.

To compile any existing Electron app, ensure that you have the following requirements:

- Windows 10 with Anniversary Update (released August 2nd, 2016)
- The Windows 10 SDK, downloadable here
- At least Node 4 (to check, run `node -v`)

Then, go and install the `electron-windows-store` CLI:

```
npm install -g electron-windows-store
```

Step 1: Package Your Electron Application

Package the application using electron-packager (or a similar tool). Make sure to remove `node_modules` that you don't need in your final application, since any module you don't actually need will increase your application's size.

The output should look roughly like this:

```
├── Ghost.exe
├── LICENSE
├── content_resources_200_percent.pak
├── content_shell.pak
├── d3dcompiler_47.dll
├── ffmpeg.dll
├── icudtl.dat
├── libEGL.dll
├── libGLESv2.dll
├── locales
│   ├── am.pak
```

```
│   ├── ar.pak
│   ├── [...]
├── node.dll
├── resources
│   └── app.asar
├── v8_context_snapshot.bin
├── squirrel.exe
└── ui_resources_200_percent.pak
```

Step 2: Running electron-windows-store

From an elevated PowerShell (run it "as Administrator"), run `electron-windows-store` with the required parameters, passing both the input and output directories, the app's name and version, and confirmation that `node_modules` should be flattened.

```
electron-windows-store `
    --input-directory C:\myelectronapp `
    --output-directory C:\output\myelectronapp `
    --package-version 1.0.0.0 `
    --package-name myelectronapp
```

Once executed, the tool goes to work: It accepts your Electron app as an input, flattening the `node_modules`. Then, it archives your application as `app.zip`. Using an installer and a Windows Container, the tool creates an "expanded" AppX package - including the Windows Application Manifest (`AppXManifest.xml`) as well as the virtual file system and the virtual registry inside your output folder.

Once the expanded AppX files are created, the tool uses the Windows App Packager (`MakeAppx.exe`) to create a single-file AppX package from those files on disk. Finally, the tool can be used to create a trusted certificate on your computer to sign the new AppX package. With the signed AppX package, the CLI can also automatically install the package on your machine.

Step 3: Using the AppX Package

In order to run your package, your users will need Windows 10 with the so-called "Anniversary Update" - details on how to update Windows can be found here.

In opposition to traditional UWP apps, packaged apps currently need to undergo a manual verification process, for which you can apply here. In the meantime, all users will be able to install your package by double-clicking it, so a submission to the store might not be necessary if you're looking for an easier installation method. In managed environments (usually enterprises), the `Add-AppxPackage` PowerShell Cmdlet can be used to install it in an automated fashion.

Another important limitation is that the compiled AppX package still contains a win32 executable - and will therefore not run on Xbox, HoloLens, or Phones.

Optional: Add UWP Features using a BackgroundTask

You can pair your Electron app up with an invisible UWP background task that gets to make full use of Windows 10 features - like push notifications, Cortana integration, or live tiles.

To check out how an Electron app that uses a background task to send toast notifications and live tiles, check out the Microsoft-provided sample.

Optional: Convert using Container Virtualization

To generate the AppX package, the `electron-windows-store` CLI uses a template that should work for most Electron apps. However, if you are using a custom installer, or should you experience any trouble with the generated package, you can attempt to create a package using compilation with a Windows Container - in that mode, the CLI will install and run your application in blank Windows Container to determine what modifications your application is exactly doing to the operating system.

Before running the CLI for the first time, you will have to setup the "Windows Desktop App Converter". This will take a few minutes, but don't worry - you only have to do this once. Download and Desktop App Converter from here. You will receive two files: `DesktopAppConverter.zip` and `BaseImage-14316.wim`.

1. Unzip `DesktopAppConverter.zip`. From an elevated PowerShell (opened with "run as Administrator", ensure that your systems execution policy allows us to run everything we intend to run by calling `Set-ExecutionPolicy bypass`.
2. Then, run the installation of the Desktop App Converter, passing in the location of the Windows base Image (downloaded as `BaseImage-14316.wim`), by calling `.\DesktopAppConverter.ps1 -Setup -BaseImage .\BaseImage-14316.wim`.
3. If running the above command prompts you for a reboot, please restart your machine and run the above command again after a successful restart.

Once installation succeeded, you can move on to compiling your Electron app.