Introduction to MSIX

Updated April 2020

Table of Contents

Prerequisites

We recommend converting on a clean environment, such as with a new VM. If you do not have one, we provide a Quick Create MSIX Packaging Tool Environment through Hyper-V. If you are using the MSIX Packaging Tool Quick Create VM, you can skip ahead to the Lab Setup instructions.

- MSIX Packaging Tool relies on the latest Windows 10 features. Please ensure that you're on the Windows 10 1809 or later builds.
- MSIX Packaging Tool needs to be run elevated. Please ensure that you are logged in with an account that has admin privileges.
- Download the MSIX Packaging Tool from the Microsoft Store (If not already installed). https://www.microsoft.com/store/productId/9N5LW3JBCXKF
    - Ensure you are running the MSIX Packaging Tool version **1.2019.402.0 or later (To get the 1.2019.402.0 version go to this blog post**)
        - To validate the version, launch the MSIX Packaging Tool and select the gear icon in the top right corner, then select the about tab (left side).

Lab Setup

1. Download "**ITProLabs.2008.zip**" from the 'Releases' section and extract to the root of C:\ drive
2. Press the [Windows key] + [X] in combination then select "Windows PowerShell (Admin)". In the PowerShell window PowerShell window enter:

   Import-Certificate -Filepath "C:\MSIXLab\SigningCertificate\ContosoLab.cer" -CertStoreLocation cert:\LocalMachine\TrustedPeople

   NOTE: This is because the certificate with which the MSIX is going to get signed with is not a public trusted certificated and included inside your Trusted Certificate Folder. If the package is being signed with a standard trusted code signing enterprise certificate, you will not be required to do this step.

3. In the same PowerShell window enter: **Set-ExecutionPolicy RemoteSigned -force**
4. Enable sideloading of apps. From within the Start Menu launch **Settings** then navigate **Update & Security > For developers**. Select the **Sideload apps** radio button. (note: the labs will also work if developer mode is selected).
5. Launch the MSIX Packaging Tool.
    a. Select **yes** on the User Account Control pop up to run the tool in Administrative Mode.
    b. If prompted by the MSIX Packaging Tool to **Send Diagnostic data**, select the **Accept** button.
6. Go to the **Settings** by selecting the gear icon in the top right corner in the MSIX Packaging Tool window.
    a. Inside of the **Tool defaults** tab on the left, find **Default save location**, select Browse navigate to **C:\MSIXLab\Converted** then select **ok** to set your default save location.
    b. Under **Signing preference**, select **Sign with a certificate (.pfx)** from the drop down. Browse to and select **ContosoLab.pfx** from:

C:\MSIXLab\SigningCertificate folder. In the password box type in **MSIX!Lab1809**. Optionally, you can also specify a timestamp server URL.

c **Save settings** and close the MSIX Packaging Tool.

7. Look for the MSIX Packaging Tool in the Start menu, right click on it and choose **More>Pin to Taskbar**. This way you will have a way to quickly access to the tool.
8. Launch **File Explorer** and navigate to **C:**. Right Click on the **MSIXLab** folder and select **Pin to Quick Access**. This will make it easier to navigate through the lab exercises.
9. Create a checkpoint by right clicking on the VM and selecting Checkpoint so that you can easily revert to the original clean state for each exercise.

Please note

At the end of every exercise, you will be asked to copy the output packages you have created from the virtual machine to your local computer, so that you can install and try them. If the copy and paste doesn't work, you can find the output of all the exercises in the **C:\MSIXLab\Final** folder.

Exercise 1: Convert the MyEmployees using the MSIX Packaging Tool

Introduction

MSIX Packaging Tool is a tool that enables you to bring your existing Win32 desktop apps to the MSIX format. You can simply run your installer through the tool and obtain an MSIX package that you can install on your machine. MSIX Packaging tool uses a mini filter driver to listen in and capture every event on the system and isolate the actions taken by the installer to package and create the MSIX package.

Objectives

In this exercise, you will:

- Create a MSIX package using an installer (.msi).
- Install the converted MSIX.

Estimated Time to Complete This Lab

15 minutes

Scenario

In this lab, you are given the task of converting the windows installer of an application called MyEmployees into an MSIX using the MSIX Packaging Tool, then validate that it has been converted successfully.
You will use the interactive Packaging Tool UI to prepare your machine, define package properties and perform the installation of the app on the VM as the MSIX Packaging Tool monitors the system to create an MSIX package. You will then revert to the original clean state to install and test the application.

Step 1. Revert your VM to a clean state, if you haven't already.

Step 2. Launch the MSIX Packaging Tool from within your virtual machine.

Step 3. Select **Yes** on the User Account Control pop up to run the tool in Administrative Mode.

Step 4. Select the **Application package** button from the welcome screen.

Step 5. Select **Create package on this computer** and hit **Next**.

Step 6. Perform the recommended action items in the bottom table to prepare the computer for conversion by checking the checkboxes and hitting the Disable selected button. Then select **Next**. If the top table shows "pending reboot", go ahead and reboot and repeat steps 2 to 6.

Step 7. Navigate to **MyEmployees.msi** by hitting the **Browse** button and selecting **C:\MSIXLab\MyEmployees** folder in the file picker.

Step 8. Verify your signing preference is valid. If you didn't set your signing preference in the settings of the tool, select Sign with a certificate (.pfx) from the drop down then browse to and select ContosoLab.pfx from: **C:\MSIXLab\SigningCertificate** folder. In the password box type in **MSIX!Lab1809**. Optionally, you can also specify a timestamp server URL. Then hit **Next**.

Step 9. Notice that information about the package has already been populated from the MSI for you in their relative fields, then hit **Next**.

NOTE: you can leave Installation location and Description fields empty as they are optional.

Step 10. While the tool UI is in the background, perform the installation of the MyEmployees app using the setup wizard.

Step 11. Update installation completion select the **Next** button on the MSIX packaging tool page.

Step 12. Verify that MyEmployees.exe is shown as an entry in the table, then select **Next**.

NOTE: If not listed, you can manually browse to the installation location and select the MyEmployees.exe

NOTE: You can run the application to capture any first launch tasks. This app is not installing anything on first run, so it is safe to skip this step.

Step 13. Hit "Yes, move on" when prompted with the "Are you done?" pop up

Step 14. Verify **C:\MSIXLab\Converted** is your save location and select Create.

Step 15. Hit **Close** on the Package successfully created prompt to return to the home page of the MSIX Packaging Tool.

Installing the MSIX App:

Step 16. Copy **MyEmployees_1.0.0.0_x64__8h66172c634n0.msix** from **C:\MSIXLab\Converted** to your local machine or USB drive.

Step 17. Apply the latest Checkpoint on the VM to revert to a clean state.

Step 18. Copy **MyEmployees_1.0.0.0_x64__8h66172c634n0.msix** from local machine or USB drive to **C:\MSIXLab\Converted**. Double click on it then install and launch.

You now have the MyEmployees app which is installed as an MSIX.

Exercise 2: Convert the MyEmployees Export Data plugin into a Modification Package.

Introduction

A Modification packages allows you to separate your application customization from the main package. A Modification packages can be plugins, add-ins and custom configurations. Packaging your application customization separately allows you to easily manage the lifecycle of your application.

Objectives

In this exercise, you will:

- Create a MSIX modification package using an installer (.msi).
- Install the converted MSIX.
- Load and view the modification package when you launch the main application.

Estimated Time to Complete This Lab

15 minutes

Scenario

In this lab, you are given a task to convert the MyEmployees Export Plugin (.msi) into an MSIX using the MSIX Packaging Tool, install it and launch the main application to verify that the plugin works as intended at a first glance. This plugin enables a new feature inside the MyEmployees application, which allows exporting the data displayed in the application as CSV. Like creating an MSIX main package, you will use MSIX Packaging Tool to prepare your machine, define package properties and perform the installation off the plugin on the VM as the packaging tool monitors the system to create an MSIX modification package. You will then revert to the original clean state to install and test the plugin with the main application.

Step 1. Revert your VM to a clean state, if you haven't already.

Step 2. Launch the MSIX Packaging Tool from within your virtual machine.

Step 3. Click **Yes** on the User Account Control pop up to run the tool in Administrative Mode.

Step 4. Select the **Modification package** button from the home screen.

Step 5. Select Create package on this computer and select Next.

Step 6. Perform the recommended action items in the bottom table to prepare the computer for conversion by checking the checkboxes and hitting the **Disable selected** button.

Then select **Next**. If the top table shows “pending reboot”, go ahead and reboot and repeat steps 2 to 6.

Step 7. Navigate to your MSI installer by hitting Browse and selecting the MyEmployees (Export Plugin).msi installer from C:\MSIXLab\MyEmployees.

Step 8. Navigate to your parent application by hitting Browse next to the second input box and select **MyEmployees_1.0.0.0_x64__8h66172c634n0.msix** from **C:\MSIXLab\Final\Exercise 1.** This is the package we built in Exercise 1.

NOTE: For the purposes of this lab, you do not need the win32 version of the main application installed. Outside of the lab setting, this is a requirement for modification packages.

Step 9. Verify your signing preference is valid. If you didn’t set your signing preference in the settings of the tool, select **Sign with a certificate (.pfx)** from the drop down then browse to and select **ContosoLab.pfx** from: **C:\MSIXLab\SigningCertificate** folder. In the password box type in **MSIX!Lab1809**. Optionally, you can also specify a timestamp server URL. Then hit **Next**.

Step 10. Verify the Package information fields and update them, as necessary. You will have to modify the Package Name, since the name of the plugin contains spaces and brackets, which aren’t allowed. Change it to **MyEmployees-ExportPlugin.** When you have finished, hit **Next.**

NOTE: you can leave Installation location and Description fields empty as they are optional.

Step 11. While the tool UI is in the background, perform the installation of the MyEmployees Export Data Plugin app using the setup wizard. Once installation is finished select **next** on the MSIX packaging tool page.

Step 12. Hit “**Yes, move on**” when prompted with the “Are you done?” pop up

Step 13. Verify **C:\MSIXLab\Converted** is your save location and select Create.

Step 14. Hit Close on the Package successfully created pop up to return to the home page of the MSIX Packaging Tool.

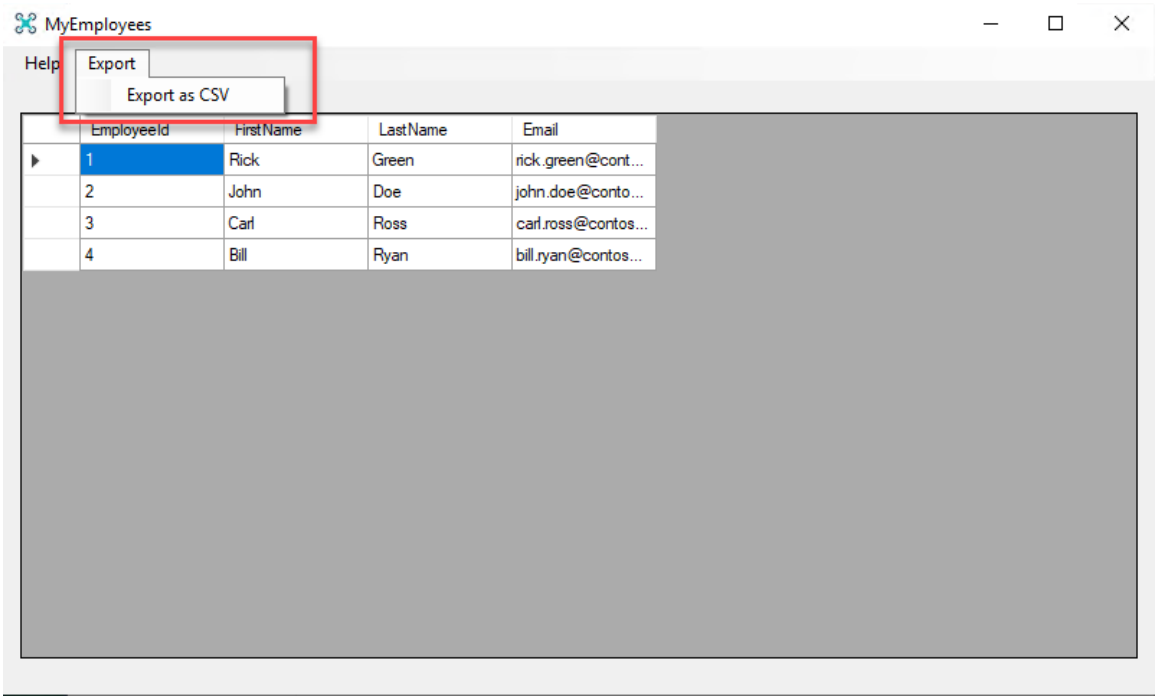
Installing the MSIX.

Step 15. Copy MyEmployees-ExportPlugin_1.0.0.0_x64__8h66172c634n0.msix from C:\MSIXLab\Converted to your local machine or USB drive.

Step 16. Apply the latest Checkpoint on the VM to revert to a clean state.

Step 17. Ensure you have installed MyEmployees (main package). You can double click on the package **MyEmployees_1.0.0.0_x64__8h66172c634n0.msix** from **C:\MSIXLab\Final\Exercise 1** folder to do it.

Step 18. Copy **MyEmployees-ExportPlugin_1.0.0.0_x64__8h66172c634n0.msix** from local machine or USB drive to **C:\MSIXLab\Converted**. Double click on it and install. Then launch MyEmployees. Notice that now there's a new option in the menu called **Export**. If you open it and you choose **Export as CSV**, you will be able to export the data you see in the grid in a CSV file on your machine.



You now have the MyEmployees Export Data Plugin which is installed as an MSIX.

Exercise 3: Convert multiple installers using MSIX Packaging Tool

Introduction

The MSIX Packaging Tool supports a command line interface which allows the user to automate the bulk conversion of application installers. The MSIX Packaging Tool uses a conversion template for its command line interface which stores the packaging information and tool settings that are going to be used for a certain packaging project.

Objectives

In this exercise, you will:

- Create a MSIX package for an installer (.msi) through a PowerShell script.
- Install the converted MSIX package.

Estimated Time to Complete This Lab

10 minutes

Scenario

In this lab, you are given the task of converting multiple installer (.msi) files into an MSIX Package using MSIX Packaging Tool and installing them. Going through the UI for each installer will be time consuming and won't scale well, so instead you are going to convert these installers silently using the command line interface of the MSIX packaging tool. You will run a simple PowerShell script that creates a conversion template file for each installer in the installers folder and kick off each conversion with the package parameters that are captured in the conversion template. After package is created the script will run Signtool command on the package.

NOTE: We have provided one sample installer file for this exercise. If you would like to convert multiple installers, add another installer to the **BulkConvert\Installers** folder.

Step 1. Revert your VM to a clean state, if you haven't already.

Step 2. Launch PowerShell in Administrative Mode by hitting Windows key + X clicking on Windows PowerShell (Admin).

Step 3. Type the following command into the **Windows PowerShell (Admin)** window.
C:\MSIXLab\BulkConvert\BulkConvert.PS1

The command line interface will start converting applications. If you see any warning, feel free to ignore them. They're expected, since the tool isn't able to automatically

translate some of the installer's actions into manifest entries, but they aren't blocking the conversion.

Installing the MSIX.

Step 4. Copy the packages from **C:\MSIXLab\BulkConvert\Converted** to your local machine or USB drive.

Step 5. Apply the latest Checkpoint on the VM to revert to a clean state.

Step 6. Copy the packages from local machine or USB drive back to **C:\MSIXLab\BulkConvert\Converted**. Double click, install and launch them.

Exercise 4: Edit the MyEmployees MSIX Package

Introduction

The MSIX Packaging Tool Package editor enables you to change the properties or an MSIX package and the content of your existing MSIX packages, without the need to repackage the installer again. Package Editor allows you to change package information that are captured in the package manifest through a UI as well as editing the manifest xml in notepad. Package editor also allows you to add or remove a file in the package or registry keys.

Objectives

In this exercise, you will:

- Open a MSIX package using MSIX Packaging Tool.
- Change the package manifest.
- Edit a registry key.
- Install the newly edited MSIX package.

Estimated Time to Complete This Lab

15 minutes

Scenario

In this lab you are given the task of updating an MSIX app to make it compliant with a specific requirement coming from the HR department. You are instructed to make the application work in a special kiosk mode, so that it can be used on special devices which will be placed around the company. The MyEmployees application supports this behavior by enabling a special key in the registry. You will need to modify using the MSIX Packaging Tool. Next you will change the display name for this package, save it and install it for further testing.

Step 1. Revert your VM to a clean state, if you haven't already.

Step 2. Launch MSIX Packaging Tool.

Step 3. Hit **yes** on the User Account Control pop up to run the tool in Administrative Mode.

Step 4. Select the **Package editor** button from the welcome screen.

Step 5. Hit Browse button and navigate to **C: \MSIXLab\Final\Exercise 1** folder and select **MyEmployees_1.0.0.0_x64__8h66172c634n0.msix** from file picker and hit open. This is the package we created in Exercise 1. Then press the **Open package** button.

Step 6. Click on the **Virtual Registry tab** and navigate to the **REGISTRY>MACHINE->SOFTWARE>WOW6432Node>Contoso>MyEmployees** node. Double click on the **KioskMode** key and change the value data from **false** to **true** then press the **Ok** button.

Step 7. Navigate to Package information tab on the left.

Step 8. Change the version number to **1.0.1.0**

Step 9. Verify your signing preference is valid. If you didn't set your signing preference in the settings of the tool, select **Sign with a certificate (.pfx)** from the drop down then browse to and select **ContosoLab.pfx** from: **C:\MSIXLab\SigningCertificate** folder. In the password box type in **MSIX!Lab1809**. Optionally, you can also specify a timestamp server URL.

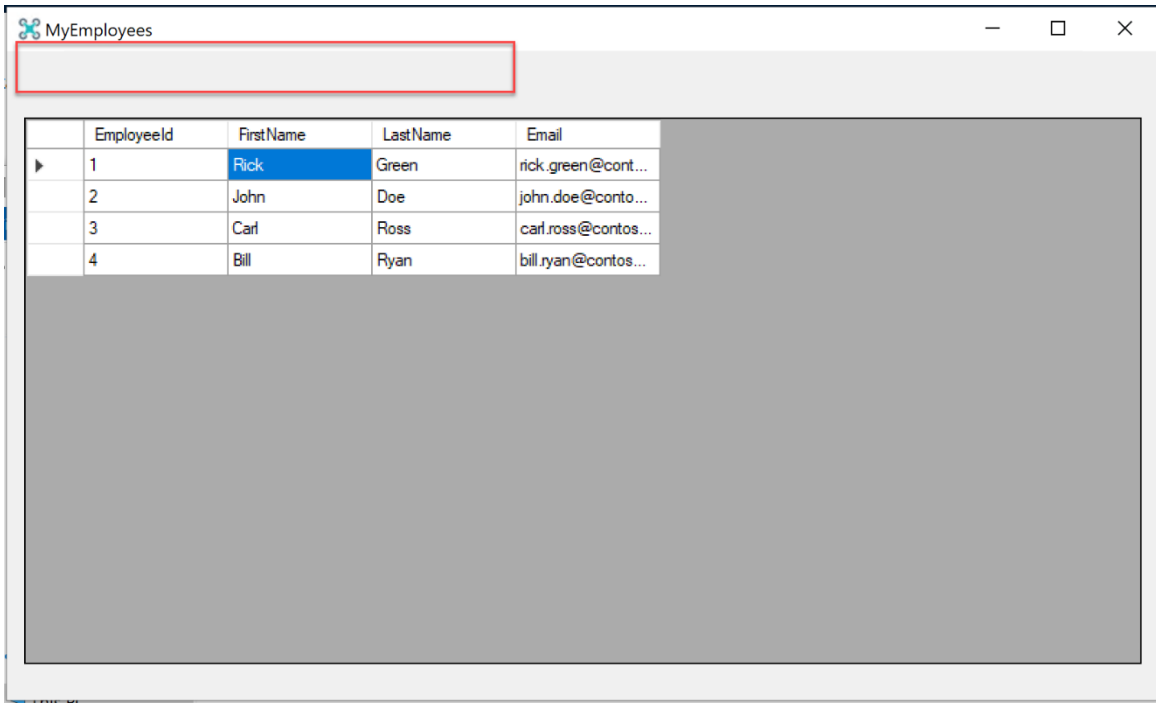
Step 10. Hit Save and select **C:\MSIXLab\Converted** folder as the path and hit save.

Step 11. Hit **Close** on the Package successfully created pop up to return to the home page of the MSIX Packaging Tool.

Installing the MSIX.

Step 12. First install the MyEmployees package you have created in Exercise 1 by double clicking on **MyEmployees_1.0.0.0_x64__8h66172c634n0.msix** in **C:\MSIXLab\Final\Exercise 1**. Install it, then launch it. Notice how the application is working normally, with all the options in the menu enabled.

Step 13. Go back to the **C:\MSIXLab\Converted** folder and double click on **MyEmployees_1.0.1.0_x64__8h66172c634n0.msix** and hit update. Launch again the application and notice how now, instead, the menu bar is missing since the application is running in kiosk mode.



You now have the updated MyEmployees app which is installed as an MSIX.

Exercise 5: Use the Package Support Framework to solve runtime issues

Introduction

When a Win32 application is packaged using MSIX, you may experience that some operations are not behaving as expected. If the application reads one or more files from the installation folder, you may notice that these files can't be found anymore. The reason is that, when an application is packaged with MSIX, you don't have any more a traditional shortcut in the Start menu and, as such, the Current Working Directory doesn't link anymore against the installation folder. Another scenario is that an application may try to write some content (for example, a log file) in the installation folder. However, since packaged apps always run at user level, they can't write or update files in the folder where packages are deployed, since it's system protected.

The Package Support Framework (PSF) is an open source framework, released by Microsoft, which can be used to change the runtime behavior of the application without modifying the source code. The PSF acts as a middle man between your application and Windows and it's able to change the behavior of low-level APIs when the application runs, for example by redirecting file operations from one folder to another.

Objectives

In this exercise, you will:

- Inject the Package Support Framework inside an existing MSIX package
- Configure the Package Support Framework in order to solve runtime issues

Estimated Time to Complete This Lab

20 minutes

Scenario

In this lab you are given the task of solving an issue that you have faced after installing the packaged version of MyEmployees. The application is running fine from a user point of view, but the logging feature isn't working as expected. To help the developer to diagnose issues, in fact, the application is writing a log file called **logfile.txt** inside the installation folder, which tracks the events that are happening during the usage of the application. When you package the application as an MSIX app, the log file isn't created. This is because the application isn't able to write the log file inside the installation folder. To solve this problem, you're going to add the Package Support Framework to the MSIX package, so that the log file is created in a different folder where the application has write access.

Step 1. Revert your VM to a clean state.

Step 2. **Step 2:** Open the folder **C:\MSIXLab\Tools\AppSdk**. Then click on **File** in File Explorer and choose **Open Windows PowerShell**.

Step 3. The first step is to unpack the content of the MSIX package which contains the MyEmployees application we have packaged in Exercise 1, so that we can inject the Package Support Framework. To do it we're going to use the **makeappx** tool, which is part of the Windows 10 SDK. Execute the following command:

```
.\makeappx unpack -p "C:\MSIXLab\Final\Exercise 1\MyEmployees_1.0.0.0_x64__8h66172c634n0.msix" -d "C:\MSIXLab\Converted\PackageFiles" -l
```

Step 4. Open the folder C:\MSIXLab\Tools\Microsoft.PackageSupportFramework\bin and copy the following files inside, to the C:\MSIXLab\Converted\PackageFiles folder:

- FileRedirectionFixup32.dll
- PsfLauncher32.exe
- PsfRunDll32.exe
- PsfRuntime32.dll

Step 5. From the PowerShell command prompt type **Notepad** and press [enter]**.**

Paste the following JSON inside Notepad:

```
{
  "applications": [
    {
      "id": "MYEMPLOYEES",

"executable":"VFS\\ProgramFilesX86\\Contoso\\MyEmployees\\MyEmployees.exe",
      "workingDirectory":"VFS\\ProgramFilesX86\\Contoso\\MyEmployees\\"
    }
  ],
  "processes": [
    {
      "executable": "MyEmployees",
      "fixups": [
        {
          "dll": "FileRedirectionFixup.dll",
          "config": {
            "redirectedPaths": {
              "knownFolders": [
                {
                  "id": "ProgramFilesX86",
                  "relativePaths": [
```

```
                    {
                      "base":"Contoso\\MyEmployees",
                      "patterns": [
                        ".*\\.txt"
                      ]
                    }
                  ]
                }
              ]
            }
          }
        }
      ]
    }
  ]
}
```

The key points of this configuration file are:

- **executable** contains the path of the main process of the application, which must be launched when the package is started
- **workingDirectory** contains the path of the folder which acts as working directory. It matches the path of the executable.
- The **fixups** section (located within the "processes" section) contains the configuration of the specific fixup for redirecting writing operations. We're configuring it so that every operation which tries to write a file with extension .txt inside the working directory will be redirected.

Step 6. In Notepad, select **File > Save As**. Navigate to the **C:\MSIXLab\Converted\PackageFiles** folder. Type **config.json** as the *File Name* and select **All Files** for the *Save as type*. Click **Save** and then close notepad.

Step 7. Now we need to change the entry point of the application. We need to start, in fact, the PSF launcher instead of the main executable. Right click on the **AppxManifext.xml** file inside **C:\MSIXLab\Converted\PackageFiles**, choose **Edit** and look for the following entry:

```
<Application Id="MYEMPLOYEES"
Executable="VFS\ProgramFilesX86\Contoso\MyEmployees\MyEmployees.exe"
EntryPoint="Windows.FullTrustApplication">
```

Change it as the following (highlighted in yellow is the changed part):

```
<Application Id="MYEMPLOYEES" Executable="PSFLauncher32.exe"
EntryPoint="Windows.FullTrustApplication">
```

Step 8. In Notepad choose **File > Save**, then close Notepad.

Step 9. Now return to the PowerShell window you had opened in step 3 and run the following command:

Step 10. ./makeappx pack -p "C:\MSIXLab\Converted\MyEmployees_1.0.0.0_x64__8h66172c634n0.msix" -d "C:\MSIXLab\Converted\PackageFiles" -l

The command will recreate a new MSIX package starting from the content of the C:\MSIXLab\Converted\PackageFiles folder.

Step 11. The package is now unsigned, so we can't install it. As such, we're going to sign it using the **signtool** utility, which is part of the Windows SDK. In the same PowerShell window, run the following command:

./signtool.exe sign /a /v /fd SHA256 /f "C:\MSIXLab\SigningCertificate\ContosoLab.pfx" /p "MSIX!Lab1809" "C:\MSIXLab\Converted\MyEmployees_1.0.0.0_x64__8h66172c634n0.msix"

This command will sign the package with the certificate called **ContosoLab.pfx**, which is the same one we have used in the other exercises with the MSIX Packaging Tool. All the parameters you see are fixed, except for the **/p** one, which is used to specify the certificate's password.

Install the MSIX App:

Step 12. Double click on **MyEmployees_1.0.0.0_x64__8h66172c634n0.msix** in the **C:\MSIXLab\Converted** folder and install it. Then launch MyEmployees. Now open the **%localappdata%\Packages\MyEmployees_8h66172c634n0\LocalCache\Local\VFS\C$\Program Files (x86)\Contoso\MyEmployees** folder. Notice that it contains a file called **logfile.txt**. Open it to see the logging created by the MyEmployees application. This file was originally created in the installation folder of the application.

Exercise 6: Manage an MSIX package using PowerShell

Introduction

An MSIX package can be deployed and managed using PowerShell commands, making it easier to automate delivery and management of MSIX apps in an enterprise environment. In this exercise you're going to explore the basic commands to install, describe and remove a package.

Objectives

In this exercise, you will:

- Deploy a MSIX package with PowerShell
- Deploy a MSIX Modification package with PowerShell
- Check the status of the package on the machine
- Remove the MSIX package using PowerShell

Estimated Time to Complete This Lab

15 minutes

Scenario

In this lab, you are given a task to deploy the MyEmployees application using PowerShell. You will need to automate the deployment of MSIX packages throughout your organization.

Step 1. Revert your VM to a clean state.

Step 2. Open File Explorer on the path **C:\MSIXLab\Final\Exercise 1**

Step 3. Click on **File** and choose **Open Windows PowerShell**

Step 4. Copy and paste the following command into the PowerShell window:

Add-AppxPackage -Path ".\MyEmployees_1.0.0.0_x64__8h66172c634n0.msix"

Note: The **Add-AppxPackage** cmdlet will install the specified MSIX package for the current user. With the **-Path** parameter you can specify the full path of the MSIX package you want to deploy.

Step 5. You can check that the application has been properly installed by copying and pasting the following command into the PowerShell window:

Get-AppxPackage -Name *employees*

This command will list all the packages with **employees** in the name that are installed on your machine. Listing apps from the Store, manually sideloaded or deployed by your IT Pro department. Applications will be returned in chronological order, the last in the list should be the MyEmployees app that we just installed.

Step 6. Review the information returned by the **Get-AppxPackage** cmdlet regarding the installed MSIX app. Note the following returned values:

- The publisher
- The architecture
- The version
- The Package Family Name (PFN)
- The full location where it has been installed

Take note of the Package Full Name since we're going to reuse it later.

```
Name              : MyEmployees
Publisher         : CN=Contoso Software (FOR LAB USE ONLY), O=Contoso Corporation, C=US
Architecture      : X64
ResourceId        :
Version           : 1.0.0.0
PackageFullName   : MyEmployees_1.0.0.0_x64__8h66172c634n0
InstallLocation   : C:\Program Files\WindowsApps\MyEmployees_1.0.0.0_x64__8h66172c634n0
IsFramework       : False
PackageFamilyName : MyEmployees_8h66172c634n0
PublisherId       : 8h66172c634n0
IsResourcePackage : False
IsBundle          : False
IsDevelopmentMode : False
NonRemovable      : False
IsPartiallyStaged : False
SignatureKind     : Developer
Status            : Ok
```

Step 7. As an additional verification that the application was installed, open the Start menu. You will find the **MyEmployees** entry at the top of the **Recently added** section. Click on it to launch it and verify that the app launches.

Step 8. The same **Add-AppxPackage** cmdlet can be used to deploy modification packages. Copy and paste the following command into the PowerShell window:

Add-AppxPackage -Path "C:\MSIXLab\Final\Exercise 2\MyEmployees-ExportPlugin_1.0.0.0_x64__8h66172c634n0.msix"

Step 9. Now let's check the status of the package by copying and pasting again the following command:

Get-AppxPackage -Name *employees*

```
Name                  : MyEmployees
Publisher             : CN=Contoso Software (FOR LAB USE ONLY), O=Contoso Corporation, C=US
Architecture          : X64
ResourceId            :
Version               : 1.0.0.0
PackageFullName       : MyEmployees_1.0.0.0_x64__8h66172c634n0
InstallLocation       : C:\Program Files\WindowsApps\MyEmployees_1.0.0.0_x64__8h66172c634n0
IsFramework           : False
PackageFamilyName     : MyEmployees_8h66172c634n0
PublisherId           : 8h66172c634n0
IsResourcePackage     : False
IsBundle              : False
IsDevelopmentMode     : False
NonRemovable          : False
Dependencies          : {MyEmployees-ExportPlugin_1.0.0.0_x64__8h66172c634n0}
IsPartiallyStaged     : False
SignatureKind         : Developer
Status                : Ok
```

Notice how this time there's a new property called **Dependencies**, which identifies that this application has a new dependency. The value shown inside of the dependencies is the Package Full Name of the modification package we just installed.

Step 10. Open the Start menu and launch the MyEmployees application from the list.

Step 11. Observe how the **Export** menu is now visible. This means that the modification package has been successfully installed.

Remove/Uninstall the MSIX Application:

Step 12. Copy and paste the following command into the PowerShell window:

Remove-AppxPackage -Package MyEmployees_1.0.0.0_x64__8h66172c634n0

The cmdlet requires the **-Package** parameter with the Package Full Name of the application to Removed. We have taken note of this information in Step 7, thanks to the **Get-AppxPackage** command.

Step 13. Let's check the installation status of the MSIX app using **the Get-**AppxPackage cmdlet. copy and paste the following command in the PowerShell window:

Get-AppxPackage -Name *employees*

Notice how this time there was no results. Additionally, if you open the Start menu, the MyEmployees application won't be listed anymore in the **Recently added** section.

The **Remove-AppxPackage** command has taken care of removing all the dependencies. As such, the modification package we have previously installed is no longer present in the system.

Optional exercises

Exercise 7: Convert the MyEmployees Customization installer using MSIX Packaging Tool into a Modification package.

Introduction

Modification packages allows you to separate your application customization from the main package. MSIX supports installing multiple modification packages for the same application, so that you can easily decouple different components (plugins, customization, branding, etc.)

Objectives

In this exercise, you will:

- Create a MSIX modification package using an installer (.msi)
- Install the converted MSIX on your machines.
- Load and view the modification package when you launch the main application.

Estimated Time to Complete This Lab

15 minutes

Scenario

In this lab, you are given a task to convert the MyEmployees Customization (.msi) into an MSIX using the MSIX packaging tool, install and launch the main application to verify that the customization works as intended at a first glance. This customization adds information about the Contoso company in the About page and disables the **Check for updates** feature. Since MSIX supports multiple modification packages, you will be able to install this customization together with the Export plugin you have previously converted in Exercise 2. Like creating an MSIX main package, you will use MSIX Packaging Tool to prepare your machine, define package properties and perform the installation off the plugin on the VM as the packaging tool monitors the system to create an MSIX modification package. You will than revert to the original clean state to install and test the plugin with the main application.

Step 1. Revert your VM to a clean state then launch MSIX Packaging Tool.

Step 2. Hit yes on the User Account Control pop up to run the tool in **Administrative** Mode.

Step 3. Select **Modification package** icon from the welcome screen.

Step 4. Select Create package on this computer and hit Next.

Step 5. Perform the recommended action items in the bottom table to prepare the computer for conversion by checking the checkboxes and hitting the **Disable selected** button. Then hit **Next**. If the top table shows "pending reboot", go ahead and reboot and repeat steps 1 to 5.

Step 6. Navigate to your MSI installer by hitting Browse and selecting the MyEmployees (Customization).msi installer from C:\MSIXLab\MyEmployees then click Next.

Step 7. Navigate to your parent application by hitting Browse next to the second input box and selecting **MyEmployees_1.0.0.0_x64__8h66172c634n0.msix** from **C:\MSIXLab\Final\Exercise 1.** This is the package we built in Exercise 1.

NOTE: For the purposes of this lab, you do not need the win32 version of the main application installed. Outside of the lab setting, this is a requirement for modification packages.

Step 8. Check the box under sign package for testing, press **Browse** and select **ContosoLab.pfx** from: **C:\MSIXLab\SigningCertificate** folder. In the password box Type in **MSIX!Lab1809**. Then hit **Next**.

Step 9. Select Create package on this computer and hit Next.

Step 10. Verify the Package information fields and update them, as necessary. You will have to modify the Package Name, since the name of the plugin contains spaces and brackets, which aren't allowed. Change it to **MyEmployees-Customization.** When you have finished, hit Next.

NOTE: you can leave Installation location empty as it is an optional field.

Step 11. Perform the recommended action items in the bottom table to prepare the computer for conversion by checking the checkboxes and hitting the Disable selected button. Then hit Next. If the top table shows "pending reboot", go ahead and reboot and repeat steps 1 to 8.

Step 12. While the tool UI is in the background, perform the installation of the MyEmployees Customization app using the setup wizard. Once installation is finished hit next on the MSIX packaging tool page.

Step 13. Hit "**Yes, move on**" when prompted with the "Are you done?" pop-up

Step 14. Select **C:\MSIXLab\Converted** as your save location and then hit Create.

Step 15. Hit Close on the Package successfully created pop up to return to the home page of the MSIX Packaging Tool.

Installing the MSIX

Step 16. Copy **MyEmployees-Customization_1.0.0.0_x64__8h66172c634n0.msix** from **C:\MSIXLab\Converted** to your local machine or USB drive.

Step 17. Apply the latest Checkpoint on the VM to revert to a clean state.

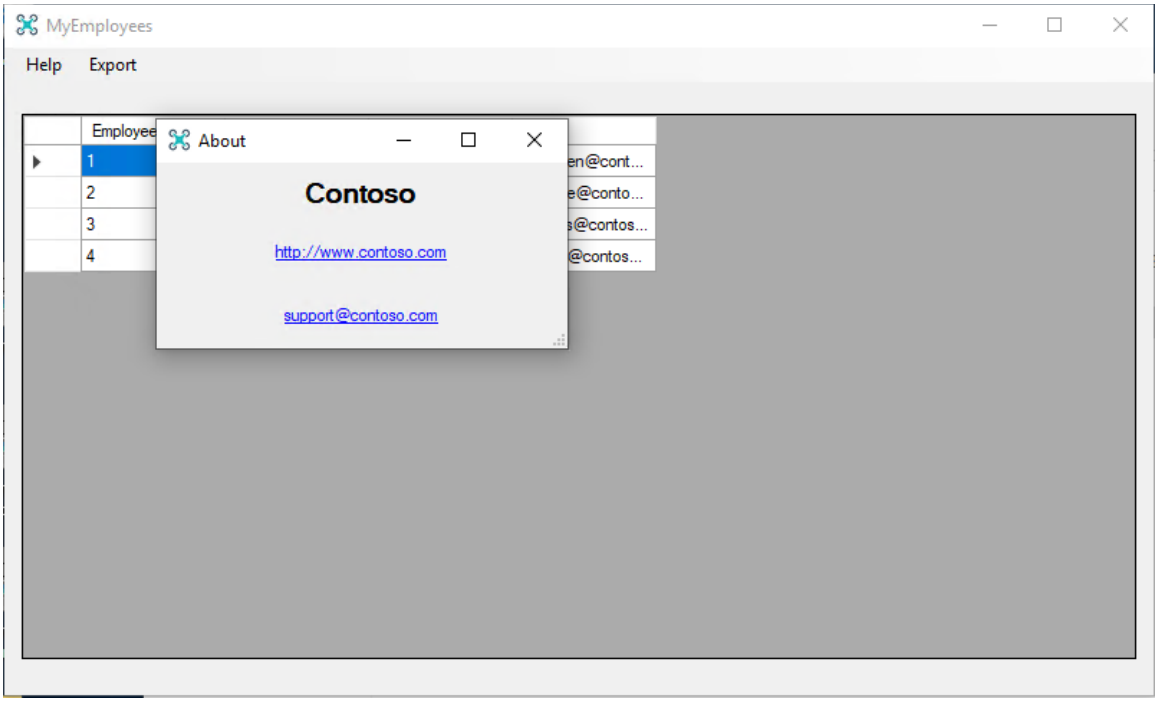
Step 18. Ensure you have installed MyEmployees (main package) and the Export plugin. You can double click on MyEmployees_1.0.0.0_x64__8h66172c634n0.msix from C:\MSIXLab\Final\Exercise 1 and on MyEmployees-ExportPlugin_1.0.0.0_x64__8h66172c634n0.msix from C:\MSIXLab\Final\Exercise 2 folders to do it.

Step 19. Copy **MyEmployees-Customization_1.0.0.0_x64__8h66172c634n0.msix** from local machine or USB drive to **C:\MSIXLab\Converted**. Double click on it and install. Then launch MyEmployees. Click on Help and notice:

Step 20. The **Check for updates** option is now missing.

Step 21. If you click on **About**, you will see a pop-up with different information about the Contoso company.

Notice also how the Export feature is still present since the Export Plugin modification package can live side by side with the Customization one you have just created.



You now have the MyEmployees Customization which is installed as a Windows App.

Exercise 8: Manage a MSIX package using PowerShell for all the users on the machine

Introduction

By default, when you deploy an MSIX package it's installed for the currently logged on user of the machine, or under the account which triggered the install. A multi-user scenarios may exist for some machine. Thanks to PowerShell, we can deploy the application to a device allowing all the users access to it.

Objectives

In this exercise, you will:

- Deploy a MSIX package with PowerShell for all the Windows users
- Create a new user on the machine and use the application
- Remove the MSIX package for all the existing users with PowerShell

Estimated Time to Complete This Lab

20 minutes

Scenario

In this lab, you are given a task to deploy the MyEmployees application using PowerShell, so that you can explore the opportunity to automate the management of MSIX packages inside your enterprise environment. However, many machines inside the company leverage a multi-user environment, since they aren't owned by a specific employee, but they can be used by any employee who is authorized to perform the task.

Step 1. Revert your VM to a clean state.

Step 2. Open File Explorer and navigate to **C:\MSIXLab\Final\Exercise 1**.

Step 3. Click on File, move your mouse over Open **Windows PowerShell** for a few seconds and then choose **Open Windows PowerShell as administrator**.

Step 4. Press **Yes** when you're asked for a confirmation.

Step 5. Copy and paste the following command:

```
Add-AppxProvisionedPackage -Online -PackagePath
".\MyEmployees_1.0.0.0_x64__8h66172c634n0.msix" -SkipLicense
```

We use the **Add-AppxProvisionedPackage** command with the following set of parameters:

- **Online**, which means that we want to apply this configuration to the current Windows installation
- **PackagePath**, which is the path of the MSIX package you want to deploy
- **SkipLicense**: since we are deploying a package which isn't coming from the Microsoft Store, we don't need a license so we can skip the verification

Step 6. You can check that the application has been properly installed by copying and pasting the following command into the PowerShell window:

Get-AppxPackage -Name *employees*

This command will list all the packages with **employees** in the name that are installed on your machine. Listing apps from the Store, manually sideloaded or deployed by your IT Pro department. Applications will be returned in chronological order, the last in the list should be the MyEmployees app that we just installed.

Example:

```
Name              : MyEmployees
Publisher         : CN=Contoso Software (FOR LAB USE ONLY), O=Contoso Corporation, C=US
Architecture      : X64
ResourceId        :
Version           : 1.0.0.0
PackageFullName   : MyEmployees_1.0.0.0_x64__8h66172c634n0
InstallLocation   : C:\Program Files\WindowsApps\MyEmployees_1.0.0.0_x64__8h66172c634n0
IsFramework       : False
PackageFamilyName : MyEmployees_8h66172c634n0
PublisherId       : 8h66172c634n0
IsResourcePackage : False
IsBundle          : False
IsDevelopmentMode : False
NonRemovable      : False
IsPartiallyStaged : False
SignatureKind     : Developer
Status            : Ok
```

Step 7. As an additional verification that the application was installed, open the Start menu. You will find the **MyEmployees** entry at the top of the **Recently added** section. Click on it to launch it and verify that the app launches.

Step 8. Now let's verify that the application has been indeed deployed for all the current and future users on the machine.

Step 9. Right click on the Start menu and choose **Computer management**.

Step 10. In the left tree, click on **Local Users and Groups** under **System Tools**.

Step 11. Click on **Users**.

Step 12. Right click in empty space in the user's list and choose **New user**.

Step 13. Type **MsixLab** in the **User name** field.

Step 14. Type **msix** in the **Password** and **Confirm password** fields.

Step 15. Uncheck the User must change password at next logon option.

Step 16. Click the **Create** button, then click on the **close** button.

Step 17. Now open the Start menu, click on the avatar icon and choose **Sign out**

Step 18. Once you're back to the Windows login screen, login with the user you have just created:

- Username: MsixLab
- Password: msix

Step 19. Select **Accept** leaving the default values for Windows privacy settings.

Step 20. Once you're on the desktop of the new user, right click on the Start menu and choose **Windows PowerShell**.

Step 21. Copy and paste the following command:

Get-AppxPackage -Name *employees*

Notice how the package will be found for the current user.

Step 22. Open the Start menu and scroll down the list of applications. Notice how, under the **M** section, you will see the MyEmployees application. Click on it to launch and observe it running as usual.

Step 23. Now let's open again the Start menu, click on the avatar icon and choose **Sign out**.

Step 24. Once you're back to the Windows login screen, login back with the original user that you have created when you have setup the machine.

Step 25. Right click on the Start menu and choose **Windows PowerShell (Admin)**.

Step 26. Press **Yes** when you're asked for a confirmation.

Step 27. Copy and paste the following command:

Get-AppxPackage -Name *employee* -AllUsers

The output will be similar to the below one:

```
Name                   : MyEmployees
Publisher              : CN=Contoso Software (FOR LAB USE ONLY), O=Contoso Corporation, C=US
Architecture           : X64
ResourceId             :
Version                : 1.0.0.0
PackageFullName        : MyEmployees_1.0.0.0_x64__8h66172c634n0
InstallLocation        : C:\Program Files\WindowsApps\MyEmployees_1.0.0.0_x64__8h66172c634n0
IsFramework            : False
PackageFamilyName      : MyEmployees_8h66172c634n0
PublisherId            : 8h66172c634n0
PackageUserInformation : {S-1-5-21-2586070391-1003120513-3189238791-1000 [msix]: Installed,
                         S-1-5-21-2586070391-1003120513-3189238791-1002 [MsixLab]: Installed}
IsResourcePackage      : False
IsBundle               : False
IsDevelopmentMode      : False
NonRemovable           : False
IsPartiallyStaged      : False
SignatureKind          : Developer
Status                 : Ok
```

Notice how, thanks to the **-AllUsers** parameter, we can see a new property called **PackageUserInformation**, which lists all the users on the machine who have installed the package, identified by their SID and user name.

Step 28. Now let's remove the application for all the users on the machine. Copy and paste the following command:

Remove-AppxPackage -Package MyEmployees_1.0.0.0_x64__8h66172c634n0 -AllUsers

Step 29. Let's check the status of the package with the usual **Get-AppxPackage** cmdlet. Copy and paste the following command:

Get-AppxPackage -Name *employee* -AllUsers

Notice how the search will return no results. The package has been removed for all the users on the machine.

Step 30. If you want to double check the outcome of the operation, open the Start menu and click on the avatar icon. Choose **Sign out**.

Step 31. Once you're back to the Windows login screen, login with the user that we had previously created:

- Username: MsixLab
- Password: msix

Step 32. Once you're on the desktop, right click on the Start menu and choose **Windows PowerShell**.

Step 33. Copy and paste the following command:

Get-AppxPackage -Name *employee*

Notice how no results will be found.

Step 34. Open the Start menu and scroll down the list of applications. Notice how, under the **M** section, the MyEmployees application is not listed anymore.

Exercise 9: Run pre-launch script with the Package Support Framework

Introduction

Scripts enable IT Pros to customize an application dynamically to the user's environment after it is packaged using MSIX and installed on the computer. For example, you can use scripts to configure your database, set up a VPN, mount a shared drive, or perform a license check dynamically. Scripts provide a lot of flexibility. They may change registry keys or perform file modifications based on the machine or server configuration.

You can use the Package Support Framework (PSF) to run one PowerShell script before a packaged application executable runs and one PowerShell script after the application executable ran in order to perform some clean up. Each application executable defined in the application manifest can have its own scripts. You can configure the script to run once only on the first app launch and without showing the PowerShell window, so users won't end the script prematurely by mistake.

Objectives

In this exercise, you will:

- Add a pre-launch script to the package
- Inject the Package Support Framework

Estimated Time to Complete This Lab

20 minutes

Scenario

MyEmployees app can operate in 2 modes – Regular and Manager view. The MyEmployees application supports this behavior by enabling a special key in the registry. The IT Pro would like to on unique installer and be able to choose the mode with a startup script after the installation. We can imagine that the script is checking for permissions on a third part system in order to activate or not the Manager view.

In this lab, you are given the task of updating the MSIX to prompt the users with their choice of mode every time the application launches and accordingly the changes should reflect in the UI (In the real script, they will be no prompt but the permission check).

To ensure the script appears first and later the application is launched, you will use Package Support Framework and a startup script.

Step 1. Revert your VM to a clean state.

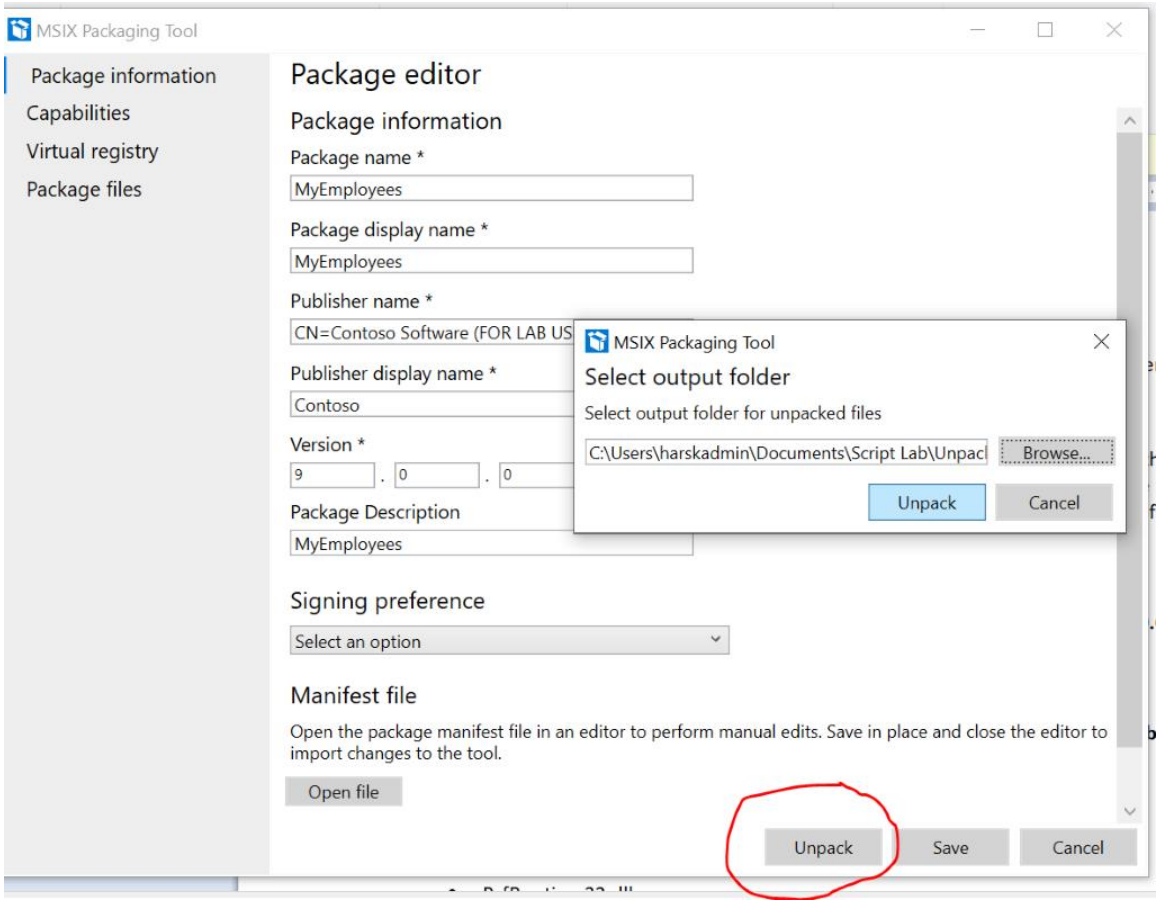
Step 2. Under folder "**C:\MSIXLab\Converted**" create a new folder named PackageFiles2

Step 3. Go to "**C:\MSIXLab\Final\Exercise 9**". Right click on the **MyEmployees.Package_9.0.0.0_x86_Initial.msix** file and select "Edit with MSIX Packaging Tool"

Step 4. In the **MSIX Packaging Tool**, click on the **"Unpack"** button at the right bottom.

Step 5. On the prompt to **Browse**, select the folder **"C:\MSIXLab\Converted\PackageFiles2".**

Step 6. Click on **Unpack** button of the dialog box.



Note: If you are using an older version of MSIX Packaging Tool which does not support unpacking of an msix, you could use the makeappx tool to unpack the app, using the following instructions:

- Open the folder **C:\MSIXLab\Tools\AppSdk**. Then click on **File** in File Explorer and choose **Open Windows PowerShell**.

- Run the following command:

  .\makeappx unpack -p "C:\MSIXLab\Final\Exercise 9\MyEmployees.Package_9.0.0.0_x86_Initial.msix" -d "C:\MSIXLab\Converted\PackageFiles2" -l

Step 7. Open the folder **C:\MSIXLab\Final\Exercise 9** and copy the following files inside, to the **C:\MSIXLab\Converted\PackageFiles2** folder:

- PsfLauncher32.exe
- PsfRunDll32.exe
- PsfRuntime32.dll

Step 8. Click on Start > **Open Notepad**.

Step 9. Paste the following JSON inside Notepad:

```
{
  "applications": [
    {
      "id": "MYEMPLOYEES",
      "executable": "VFS/ProgramFilesX86/Contoso/MyEmployees/MyEmployees.exe",
      "workingDirectory": "VFS/ProgramFilesX86/Contoso/MyEmployees/",
      "stopOnScriptError": true,
      "startScript": {
        "scriptPath": "ModeChangePreLaunch.ps1",
        "runInVirtualEnvironment": true,
        "showWindow": true,
        "waitForScriptToFinish": true,
        "runOnce": false
      }
    }
  ]
}
```

Step 10. The key points of this configuration file are:

- **executable** contains the path of the main process of the application, which will be launched when the package is started (.exe to be launched post the pre-launch script).
- **startScript** contains details of the pre-launch script.
- **scriptPath** is the path to the script including the name and extension.
- **workingDirectory** contains the path of the folder which acts as working directory. It matches the path of the executable.
- **showWindow** mentions whether the PowerShell window is shown.
- **runOnce** Specifies whether the script should run once per user, per version.

Step 11. In Notepad, select **File > Save As**. Navigate to the **C:\MSIXLab\Converted\PackageFiles2** folder. Type **config.json** as the File Name and select **All Files** for the Save as type. Click **Save** and then close Notepad.

Step 12. Now we need to change the entry point of the application. We need to start, in fact, the PSF launcher instead of the main executable. Right click on the **AppxManifext.xml** file inside **C:\MSIXLab\Converted\PackageFiles2**, choose **Edit** and look for the following entry:

<Application Id=" MYEMPLOYEES"
Executable="VFS\ProgramFilesX86\Contoso\MyEmployees\MyEmployees.exe"
EntryPoint="Windows.FullTrustApplication">

Change it as the following (highlighted in yellow is the changed part):

<Application Id="MYEMPLOYEES" Executable="PSFLauncher32.exe"
EntryPoint="Windows.FullTrustApplication">

Step 13. In Notepad choose **File > Save**, then close Notepad.

Step 14. Copy the **StartingScriptWrapper.ps1** and **ModeChangePreLaunch.ps1** from the **"C:\MSIXLab\Final\Exercise 9"** folder to **"C:\MSIXLab\Converted\PackageFiles2\VFS\ProgramFilesX86\Contoso\MyEmployees"**

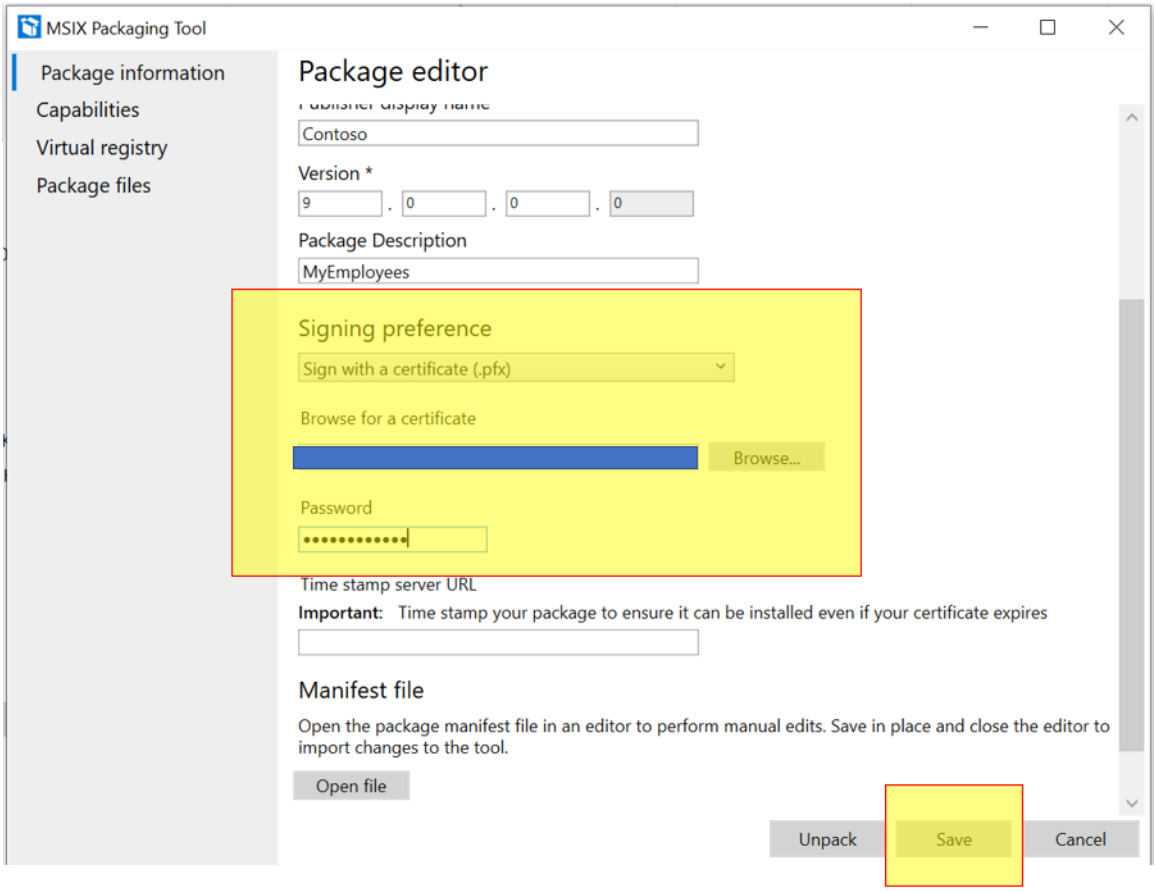
Step 15. Open the folder **C:\MSIXLab\Tools\AppSdk** in Explorer. Then click on **File** in File Explorer and choose **Open Windows PowerShell**. Run the following command:

./makeappx pack -p "C:\MSIXLab\Converted\MyEmployees.Package_9.0.0.0_x86_Final.msix" -d "C:\MSIXLab\Converted\PackageFiles2" -l

Step 16. The command will recreate a new MSIX package starting from the content of the **C:\MSIXLab\Converted\PackageFiles2** folder.

Step 17. The package is now unsigned, so we can't install it. Use the MSIX Packaging Tool to sign the MSIX using the .pfx certificate provided in the lab.

- Right click on the **MyEmployees.Package_9.0.0.0_x86_Final.msix** file and select "Edit with MSIX Packaging Tool"
- In the MSIX Packaging Tool, Select **"Sign with certificate(.pfx)"** option.
- Browse the certificate .pfx file : **C:\MSIXLab\SigningCertificate\ContosoLab.pfx** and add the corresponding password: **MSIX!Lab1809**.
- Click on Save to save the signed MSIX.



Note: You can sign the MSIX package using the **signtool** utility as well, which is part of the Windows SDK. In the same command prompt, run the following command:

./signtool.exe sign /a /v /fd SHA256 /f "C:\MSIXLab\SigningCertificate\ContosoLab.pfx" /p "MSIX!Lab1809" "C:\MSIXLab\Converted\MyEmployees.Package_9.0.0.0_x86_Final.msix"

This command will sign the package with the certificate called **ContosoLab.pfx**, which is the same one we have used in the other exercises with the MSIX Packaging Tool. All the parameters you see are fixed, except for the **/p** one, which is used to specify the certificate's password.

Verifying requirements for scripts with PSF

To be able to run Powershell scripts with Package Support Framework, we have to modify the Powershell script execution policy on the client computer.

The command is:

Set-ExecutionPolicy -ExecutionPolicy RemoteSigned

On a 64-bit computer, you will have to launch 2 Admin Powershell instances and run the "Set-ExecutionPolicy" command above in each Powershell instance:

- 64-bit executable: %SystemRoot%\system32\WindowsPowerShell\v1.0\powershell.exe

- 32-bit executable:
  %SystemRoot%\SysWOW64\WindowsPowerShell\v1.0\powershell.exe

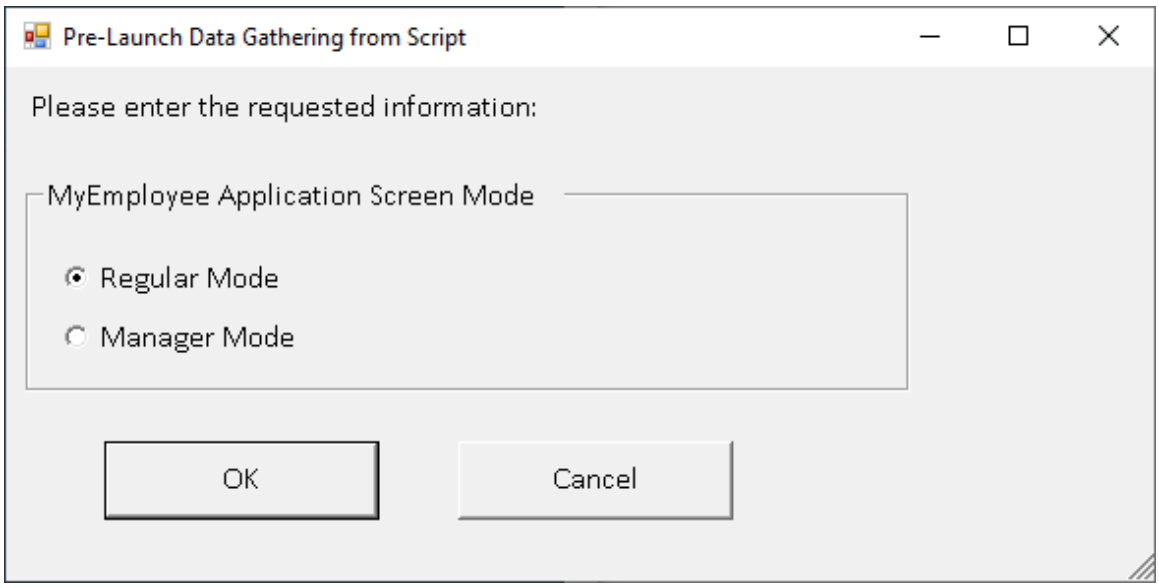
On a 32-bit computer, there is only the 32-bit Admin Powershell instance to launch:

- 32-bit executable:
  %SystemRoot%\system32\WindowsPowerShell\v1.0\powershell.exe

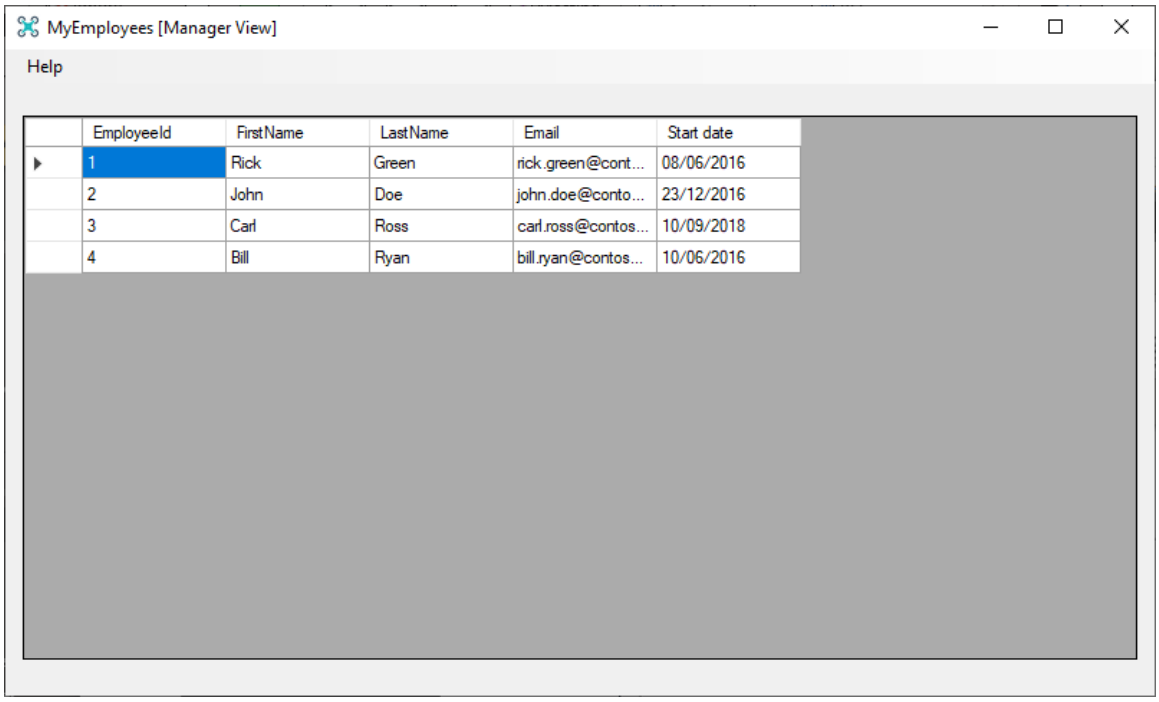
Installing the MSIX

Step 18. Double click on **MyEmployees.Package_9.0.0.0_x86_Final.msix** in the **C:\MSIXLab\Converted** folder and install it.

Step 19. Launch MyEmployees. It would prompt you with a similar dialog:



Based on the selection of this, the application will either show the show normal UI or the Manager View which displays an additional column in the datagrid:


MyEmployees [Manager View]
Help
EmployeeId
FirstName
LastName
Email
Start date
1
Rick
Green
rick.green@cont...
08/06/2016
2
John
Doe
john.doe@conto...
23/12/2016
3
Carl
Ross
carl.ross@contos...
10/09/2018
4
Bill
Ryan
bill.ryan@contos...
10/06/2016

Exercise 10: Using MSIX Core to install an MSIX down level

Introduction

MSIX Core enables the installation of MSIX apps on previous versions of Windows, provided that the apps are already built to work on those versions of Windows. MSIX Core is built for the following Windows versions that don't currently natively support MSIX:

- Windows 7 SP1
- Windows 8.1
- Currently supported Windows Server (with Desktop Experience)
- Windows 10 versions prior to 1709

MSIX Core is designed for both developers and IT pros. Developers can use the MSIX Core Library to enable their existing installers to install their MSIX packaged apps on previous Windows versions, so they can produce just one MSIX package to target all Windows users. IT pros can download the MSIX Core Installer. MSIX Core installer enables command line installation of MSIX along with the ability for users to install MSIX packages simply by double clicking them.

Objectives

In this exercise, you will:

- Add MSIX Core support to an MSIX Package
- Install an MSIX on a down level Windows OS where MSIX is not natively supported

Estimated Time to Complete This Lab

10 minutes

Scenario

You have been given the task of supporting an app on devices ranging from Windows 7 to Windows 10. You create an MSIX that works on Windows 10, but now you want to be able to leverage that same package on your Windows 7 machines, so that you only have one package to manage across your environments.

In this lab, you will add MSIX Core Support to an existing MSIX application that you have. Then you will deploy and install it on a Windows 7 machine.

Additional setup

This exercise requires the additional setup of a virtual machine with an OS previous to Windows 10 1709. For this lab, we are going to walk through the steps using a Windows 7 VM as well as using our recommended Quick Create Windows 10 virtual machine in order to add MSIX Core support.

It also requires the MSIX Core msi that enlightens your previous OS's with information about what an MSIX package is, so that you can install and use them.

Be sure to install the Lab Certificate on your Windows 7 VM so that you can install the MSIX packages.

Lab Instructions

Step 1. Revert your Windows 10 VM to a clean state.

Step 2. Open File Explorer and navigate to **C:\MSIXLab\Final\Exercise 1** and copy the **MyEmployees_1.0.0.0_x64__8h66172c634n0.msix**.

Step 3. Launch your Windows 7 VM. Paste in the MyEmployees MSIX and try to install it.

Notice how it doesn't work, and the file doesn't even get recognized by the machine. This is because the Windows 7 VM has no awareness of what an MSIX package is.

Step 4. Now, navigate to **C:\MSIXLab\MSIXCore** and copy the **msixmgr-setup msi** that matches your Windows 7 VM architecture to your Windows 7 VM and install it.

You should now be able to observe that your MSIX package has a new icon next to it, meaning that the OS now understands that this is an MSIX pacakge.

If you try to install the MSIX package now, you will notice that it still doesn't work. This is because that version doesn't have the MSIX Core support in its manifest.

Step 5. Go back to your Windows 10 VM and launch **File Explorer** and navigate to **C:\MSIXLab\Final\Exercise 1** then right click on the **MyEmployees_1.0.0.0_x64__8h66172c634n0.msix** and select '**Edit with Package Editor**'

Step 6. In the Package Information page, scroll down to the checkbox to **Add Support for MSIX Core** and select it to turn it on, then select **Windows 7, SP1 (6.1.7601.0)** as your version number.

Alternatively, if you prefer to manually add it, you can open your manifest, and under **<Dependencies>** you can add this line:

```
<TargetDeviceFamily Name="MSIXCore.Desktop" MinVersion="6.1.7601.0"
MaxVersionTested="10.0.10240.0" />
```

Note: This is **in addition to** the existing Target Device Family node that currently targets Windows 10.

Save and close your manifest file.

Step 7. Update your version number, and sign the package if you didn't have defaults set already. **Save** the updated MSIX package.

Step 8. Copy the updated MSIX with MSIX Core Support over to your Window 7 machine.

Step 9. Install the updated MSIX package. Notice that you are now able to launch the same MSIX package that works on your Windows 10 machines, down level all the way to Windows 7.