

Introduction to MSIX

Updated July 2019

Lab 1: Introduction to MSIX Packaging Tool

Contents

Prerequisites	4
Please note.....	4
Exercise 1: Convert the MyEmployees installer using MSIX Packaging Tool into a Windows app package.	5
Exercise 2: Convert the MyEmployees Export Data plugin installer using MSIX Packaging Tool into a Modification package.....	7
Exercise 3: Convert multiple installers using MSIX Packaging Tool command line interface and PowerShell.	10
Exercise 4: Edit the MyEmployees package in Packaging Tool Package Editor	12
Exercise 5: Use the Package Support Framework to solve runtime issues	14
Exercise 6: Convert the MyEmployees Customization installer using MSIX Packaging Tool into a Modification package.....	21

Prerequisites

1. MSIX Packaging Tool relies on the latest Windows 10 features. Please ensure that you're on the Windows 10 1809 or later builds.
2. MSIX Packaging tool needs to be run elevated. Please ensure that you are logged in with an account that has admin privileges.
3. Download the MSIX Packaging Tool from the Microsoft Store (If not already installed). Press Windows key + R and type *ms-windowsstore://pdp/?productid=9N5LW3JBCXKF*
4. Ensure you are running tool version **1.2019.701.0**, by launching the tool and hitting the gears icon from the top right corner and navigate to the about tab.
5. Download "[MSIXPackagingToolLabv1907.zip](#)" and unzip to C:\
6. Hit Windows key + X and 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.
7. In the same PowerShell window enter: **Set-ExecutionPolicy Bypass -force**
8. Enable sideloading of apps. Launch **Settings** and navigate **Update & Security > For developers**. **Select the Sideload Apps** radio button. (note: the labs will also work if developer mode is selected).
9. Launch the MSIX Packaging Tool.
10. Hit **yes** on the User Account Control pop up to run the tool in Administrative Mode.
11. If presented with a user consent dialog to send telemetry data to Microsoft, hit **Yes**.
12. Close the MSIX Packaging Tool by clicking the X in the upper right corner of the app.
13. 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.
14. Launch explorer and browse 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.
15. 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 back 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 installer using MSIX Packaging Tool into a Windows app package.

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 on your machines

Estimated Time to Complete This Lab

15 minutes

Scenario

In this lab, you are given the task of converting the installer of an application called MyEmployees into an MSIX using MSIX Packaging Tool and install and launch it to verify that the application works as intended at a first glance. 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 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, then launch MSIX Packaging Tool.

Step 2: Click **Yes** on the User Account Control pop up to run the tool in Administrative Mode.

Step 3: Select the **Application package** icon from the welcome screen.

Step 4: Navigate to **MyEmployees.msi** by hitting Browse button and selecting **C:\MSIXLab\MyEmployees** folder in the file picker then click **Next**.

Step 5: Check the box under sign package for testing, browse to and select **ContosoLab.pfx** from: **C:\MSIXLab\SigningCertificate** folder. In the password box type in **MSIX!Lab1809**. Then hit Next.

Step 6: Select **Create package on this computer** and hit Next.

Step 7: Notice that information about the package has already been populated from the MSIX for you in the fields, then hit Next.

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

Step 8: 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 9: While the tool UI is in the background, perform the installation of the MyEmployees app using the setup wizard. Once installation is finished hit next on the MSIX packaging tool page.

Step 10: Verify that **MyEmployees.exe** is shown as an entry point in the table, then hit Next.

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

NOTE: You can run the application by right clicking on the entry in the table and selecting Run or by double clicking on it to capture any first launch tasks. This app is not installing anything on first run, so it is safe to skip this step.

Step 11: Hit “**Yes, move on**” when prompted with the “Are you done?” pop up

Step 12: Hit Browse and select **C:\MSIXLab\Converted** as your save location and hit Create.

Step 13: Hit Close on the Package successfully created pop up to return to the home page of the MSIX Packaging Tool.

Installing the MSIX.

Step 14: Copy **MyEmployees_1.0.0.0_x64__8h66172c634n0.msix** from **C:\MSIXLab\Converted** to your local machine or USB drive.

Step 15: Apply the latest Checkpoint on the VM to revert to a clean state.

Step 16: Copy **MyEmployees_1.0.0.0_x64__8h66172c634n0.msix** from local machine or USB drive to **C:\MSIXLab\Converted**. Double click on it and install and launch.

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

Exercise 2: Convert the MyEmployees Export Data plugin installer using MSIX Packaging Tool into a Modification package.

Introduction

Modification packages allow you to separate your application customization from the main package. 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 on your machine.
- 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 of 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 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: Navigate to your MSI installer by hitting Browse and selecting the **MyEmployees (Export Plugin).msi** installer from **C:\MSIXLab\MyEmployees** then click Next.

Step 5: 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 6: Check the box **Specify your own certificate to sign with**, press Browse and select **ContosoLab.pfx** from: **C:\MSIXLab\SigningCertificate** folder. In the password box type in **MSIX!Lab1809**. Then hit Next.

Step 7: Select **Create package on this computer** and hit Next.

Step 8: 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 empty as it is an optional field.

Step 9: 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 10: 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 hit next on the MSIX packaging tool page.

Step 11: Hit **"Yes, move on"** when prompted with the "Are you done?" pop up

Step 12: Select **C:\MSIXLab\Converted** as your save location and then hit Create.

Step 13: Hit Close on the Package successfully created pop up to return to the home page of the MSIX Packaging Tool.

Installing the MSIX.

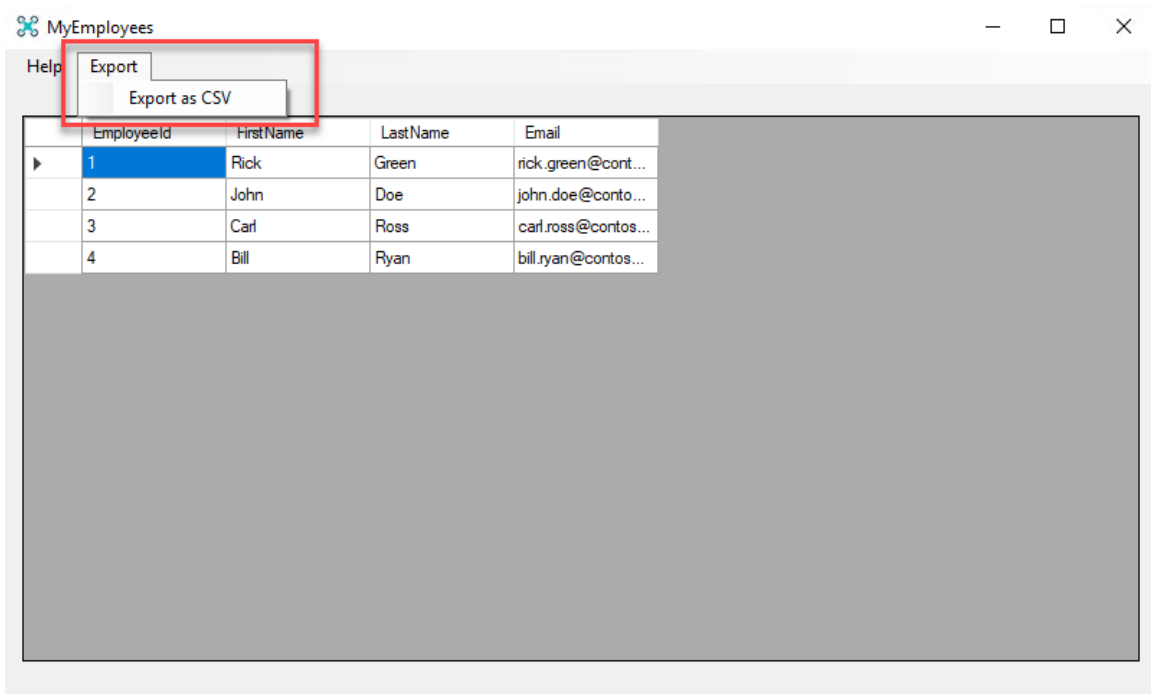
Step 14: Copy **MyEmployees-ExportPlugin_1.0.0.0_x64__8h66172c634n0.msix** from **C:\MSIXLab\Converted** to your local machine or USB drive.

Step 15: Apply the latest Checkpoint on the VM to revert to a clean state.

Step 16: 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 17: 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 a Windows App.

Exercise 3: Convert multiple installers using MSIX Packaging Tool command line interface and PowerShell.

Introduction

MSIX Packaging Tool supports a command line interface which allows the user to automate the conversion of application installers in bulk. 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

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

Step 3: Type in in this command in the command line.

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. When the script has completed, move onto step 4.

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 package in Packaging Tool Package Editor

Introduction

MSIX Packaging Tool Package editor enables you to change package properties and content of your existing MSIX packages, without the need to repack 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 as well as registry keys.

Objectives

In this exercise, you will:

- Open a MSIX package using MSIX Packaging Tool package editor
- 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 a MSIX to make it compliant with a specific requirement coming from the HR department. You are instructed to make the application working 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, which you're going to modify thanks to the Package Editor. 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 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 the **Package editor** icon from the welcome screen.

Step 4: 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 5: 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**.

Step 6: Navigate to Package information tab on the left.

Step 7: Change the version number to **1.0.1.0**

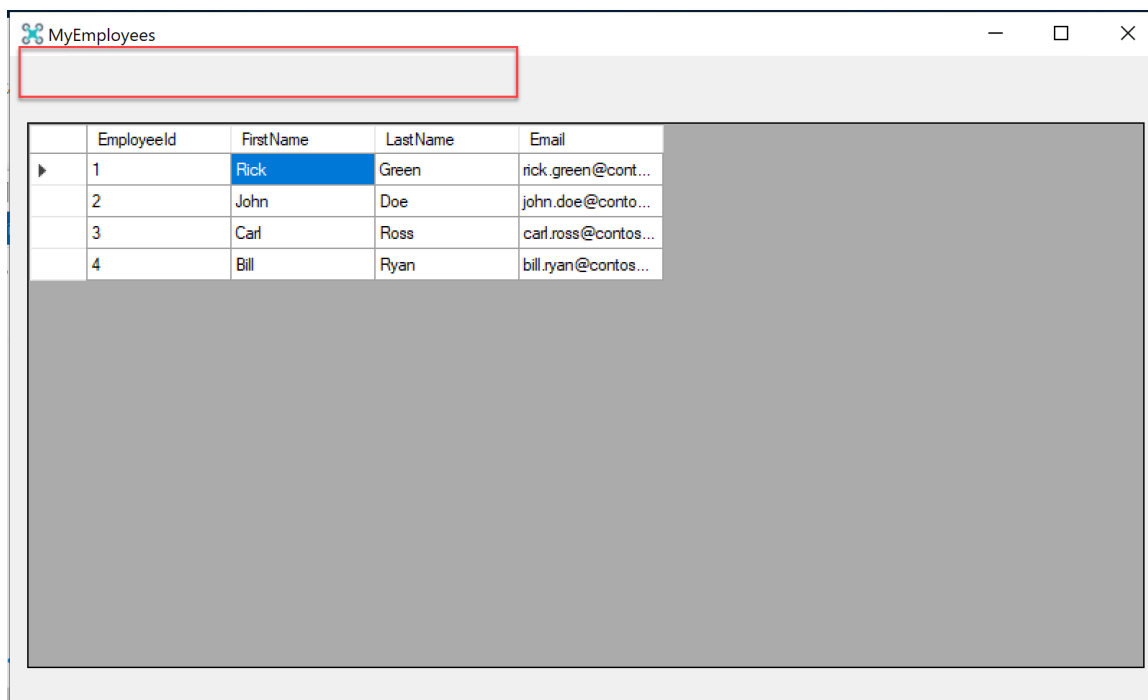
Step 8: Check the box under sign package for testing, press **Browse** and select **ContosoLab.pfx** from: **C:\MSIXLab\SigningCertificate** folder. In the password box type in **MSIX!Lab1809**. Then hit Next.

Step 9: Hit Save and select **C:\MSIXLab\Converted** folder as the path and hit save.

Step 10: Hit Close on the Package successfully created pop up to return to the home page of the MSIX Packaging Tool.

Installing the MSIX.

Step 11: 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. Then 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 a Windows App.

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 file 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 trying 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. However, when we package the application as MSIX, this file isn't created, since 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: 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].

Step 6: 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": [  

```

```
./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 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_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.

Installing the MSIX.

Step 15: 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 a MSIX package using PowerShell

Introduction

MSIX packages can be deployed and managed also with PowerShell commands, making easier to automate some operation 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, so that you automate the management of MSIX packages inside your enterprise environment.

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:

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

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 6: You can check that the application has been properly installed by copying and pasting the following command:

Get-AppxPackage

The command will list all the packages installed on your system, either from the Store, manually sideloaded or deployed by your IT Pro department. Applications will be listed in chronological order, so the last displayed one should be the MyEmployees one we have just installed.

If you can't see it, you can search for it using the **-Name** parameter, which also accepts wildcards. This makes easier to search for applications even if you don't know the exact name.

Get-AppxPackage -Name *employees*

Step 7: Observe how the **Get-AppxPackage** return many useful information about the installed package, like:

- 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 8: As a final verification that the application has been indeed installed, open the Start menu. You will find the **MyEmployees** entry in the at the top of the **Recently added** section. Click on it to launch it and observe it running as expected.

Step 9: The **Add-AppxPackage** command can be used also to deploy modification packages. Let's test it. Copy and paste the following command in the same command prompt:

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 shows that this application has a new dependency. The displayed value is the Package Full Name of the modification package we have just installed.

Step 10: Open again 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.

Step 12: Now we need to remove the application. We can use the **Remove-AppxPackage** PowerShell command. Copy and paste the following command in the prompt:

Remove-AppxPackage -Package MyEmployees_1.0.0.0_x64__8h66172c634n0

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

Step 13: Let's check the status of the package for the last time. Copy and paste the following command:

Get-AppxPackage -Name *employees*

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

Step 14: The **Remove-AppxPackage** command has taken care of removing also 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 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 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: Navigate to your MSI installer by hitting Browse and selecting the **MyEmployees (Customization).msi** installer from **C:\MSIXLab\MyEmployees** then click Next.

Step 5: 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 6: 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 7: Select **Create package on this computer** and hit Next.

Step 8: 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 9: 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 10: 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 11: Hit "Yes, move on" when prompted with the "Are you done?" pop up

Step 12: Select **C:\MSIXLab\Converted** as your save location and then hit Create.

Step 13: Hit Close on the Package successfully created pop up to return to the home page of the MSIX Packaging Tool.

Installing the MSIX

Step 14: Copy **MyEmployees-Customization_1.0.0.0_x64__8h66172c634n0.msix** from **C:\MSIXLab\Converted** to your local machine or USB drive.

Step 15: Apply the latest Checkpoint on the VM to revert to a clean state.

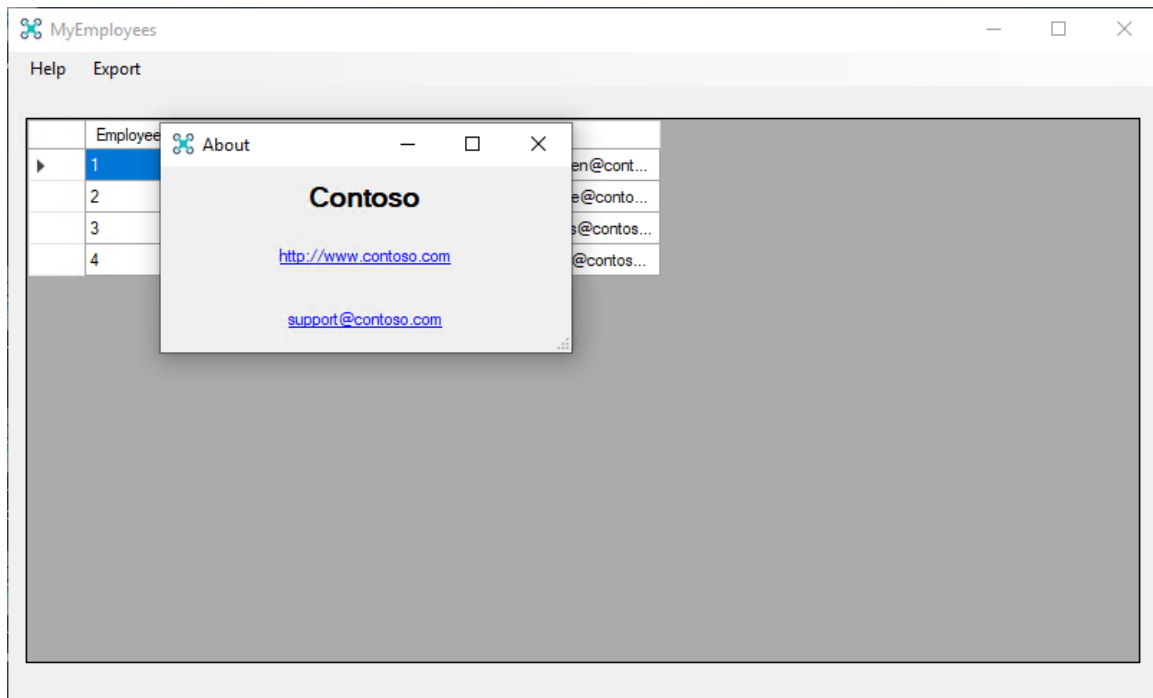
Step 16: Ensure you have installed MyEmployees (main package) and the Export plugin. You can double click on **MyEmployees_1.0.0.0_x64__8h66172c634n0.msix** from **C:\MSIXLab\Final\Exercise**

1 and on **MyEmployees-ExportPlugin_1.0.0.0_x64__8h66172c634n0.msix** from **C:\MSIXLab\Final\Exercise 2** folders to do it.

Step 17: 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:

1. The **Check for updates** option is now missing
2. If you click on **About**, you will see a popup 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 a MSIX package it's installed for the current user of the machine. However, in the enterprise world, multi-user scenarios on the same machine are common and it would be painful for the IT Pro department to have to install the same package for each current and future user. Thanks to PowerShell, we can deploy the application directly in the current Windows installation, making sure that all the users on a machine will have 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 on the path **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:

Get-AppxPackage

The command will list all the packages installed on your system, either from the Store, manually sideloaded or deployed by your IT Pro department. Applications will be listed in chronological order, so the last displayed one should be the MyEmployees one we have just installed.

If you can't see it, you can search for it using the **-Name** parameter, which also accepts wildcards. This makes easier to search for applications even if you don't know the exact name.

Get-AppxPackage -Name *employees*

You should see the following output, which means the package has been deployed for the current user:

```
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 a final verification that the application has been indeed installed, open the Start menu. You will find the **MyEmployees** entry in the at the top of the **Recently added** section. Click on it to launch it and see it running as expected.

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 **Create**

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

Accept all the default Windows privacy settings that will be proposed.

Step 19: Once you're on the desktop of the new user, right click on the Start menu and choose **Windows PowerShell**.

Step 20: Copy and paste the following command:

Get-AppxPackage -Name *employees*

Notice how the package will be found for the current user, despite you haven't installed it directly.

Step 21: 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 22: Now let's open again the Start menu, click on the avatar icon and choose **Sign out**.

Step 23: 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.

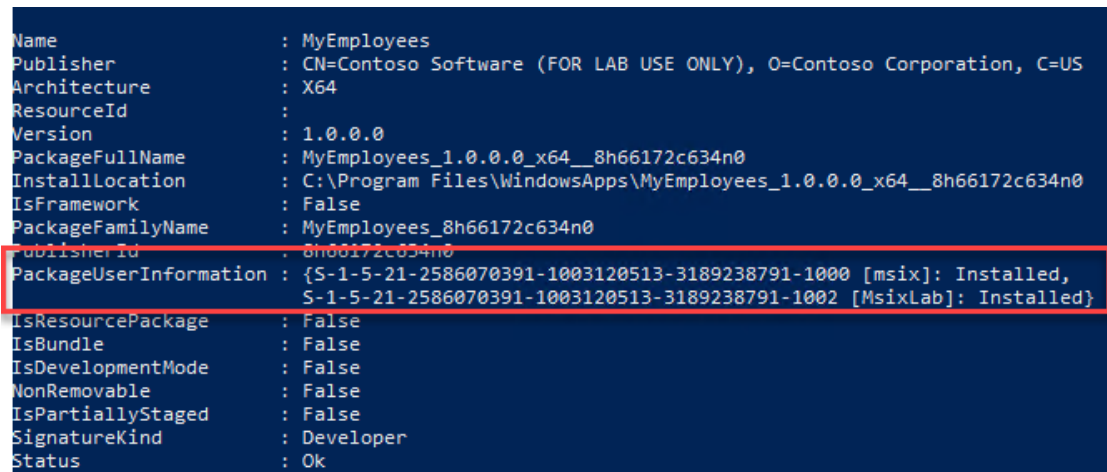
Step 24: Right click on the Start menu and choose **Windows PowerShell (Admin)**.

Step 25: Press **Yes** when you're asked for a confirmation.

Step 26: Copy and paste the following command:

Get-AppxPackage -Name *employee* -AllUsers

The output will be similar to the below one:

A screenshot of a PowerShell terminal window with a dark blue background and white text. The output of the command 'Get-AppxPackage -Name *employee* -AllUsers' is displayed. The output shows various properties of the 'MyEmployees' appx package. A red rectangular box highlights the 'PackageUserInformation' property, which lists two users: 'S-1-5-21-2586070391-1003120513-3189238791-1000 [msix]: Installed,' and 'S-1-5-21-2586070391-1003120513-3189238791-1002 [MsixLab]: Installed'. Other properties shown include Name, Publisher, Architecture, Version, PackageFullName, InstallLocation, IsFramework, PackageFamilyName, PublisherId, IsResourcePackage, IsBundle, IsDevelopmentMode, NonRemovable, IsPartiallyStaged, SignatureKind, and Status.

```
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 27: 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 28: 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 29: 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 30: Once you're back to the Windows login screen, login with the user that we have previously created:

- Username: MsixLab
- Password: msix

Step 31: Once you're on the desktop, right click on the Start menu and choose **Windows PowerShell**.

Step 32: Copy and paste the following command:

Get-AppxPackage -Name *employee*

Notice how no results will be found.

Step 21: Open the Start menu and scroll down the list of applications. Notice how, under the **M** section, the MyEmployees application is not listed anymore.