Project Report

on

11.Convert MSI to MSIX using MSIX Packaging Tool. Migrate a . Traditional MSI app to MSIX Format

Submitted By

28945 - Pravin Panditrao Adde

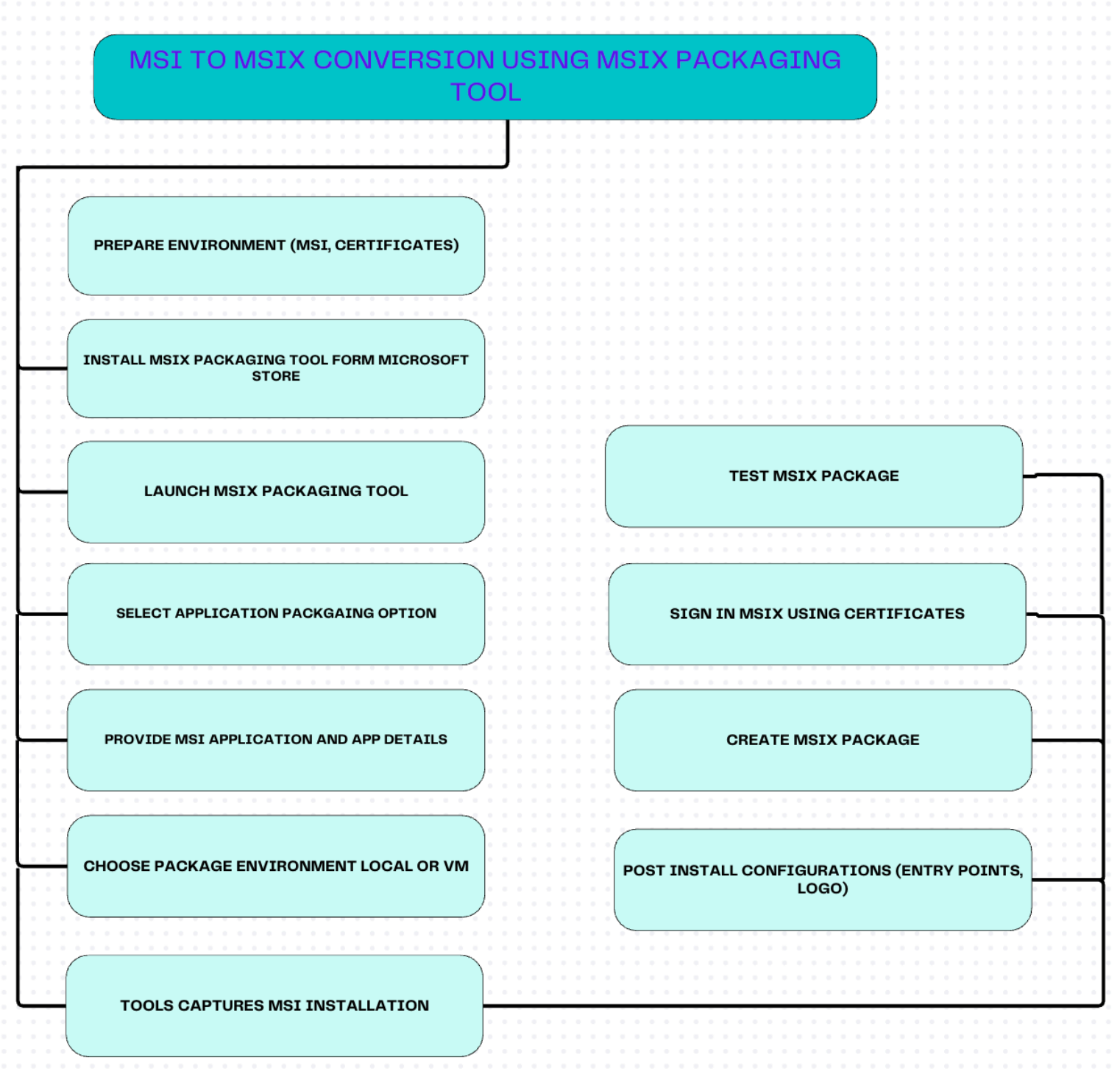
Date :- 18/06/2025

Instructor :- Harinya Mam!

Project Analysis:-

The main focus of this project is to convert .msi file to .msix that ensures

- **Application Security and Intergrety**
- **Simplifies Deployment process and Updates**
- **Go together with the Modern app Packaging Standards**



11.Convert MSI to MSIX using MSIX Packaging Tool. Migrate a . Traditional MSI app to MSIX Format

The MSIX Packaging Tool is a **Microsoft Tool** that allows users to convert existing desktop applications into the MSIX app package format.

MSIX simplifies the process of repackaging the application and makes it easier to deploy and manage them on Windows Devices

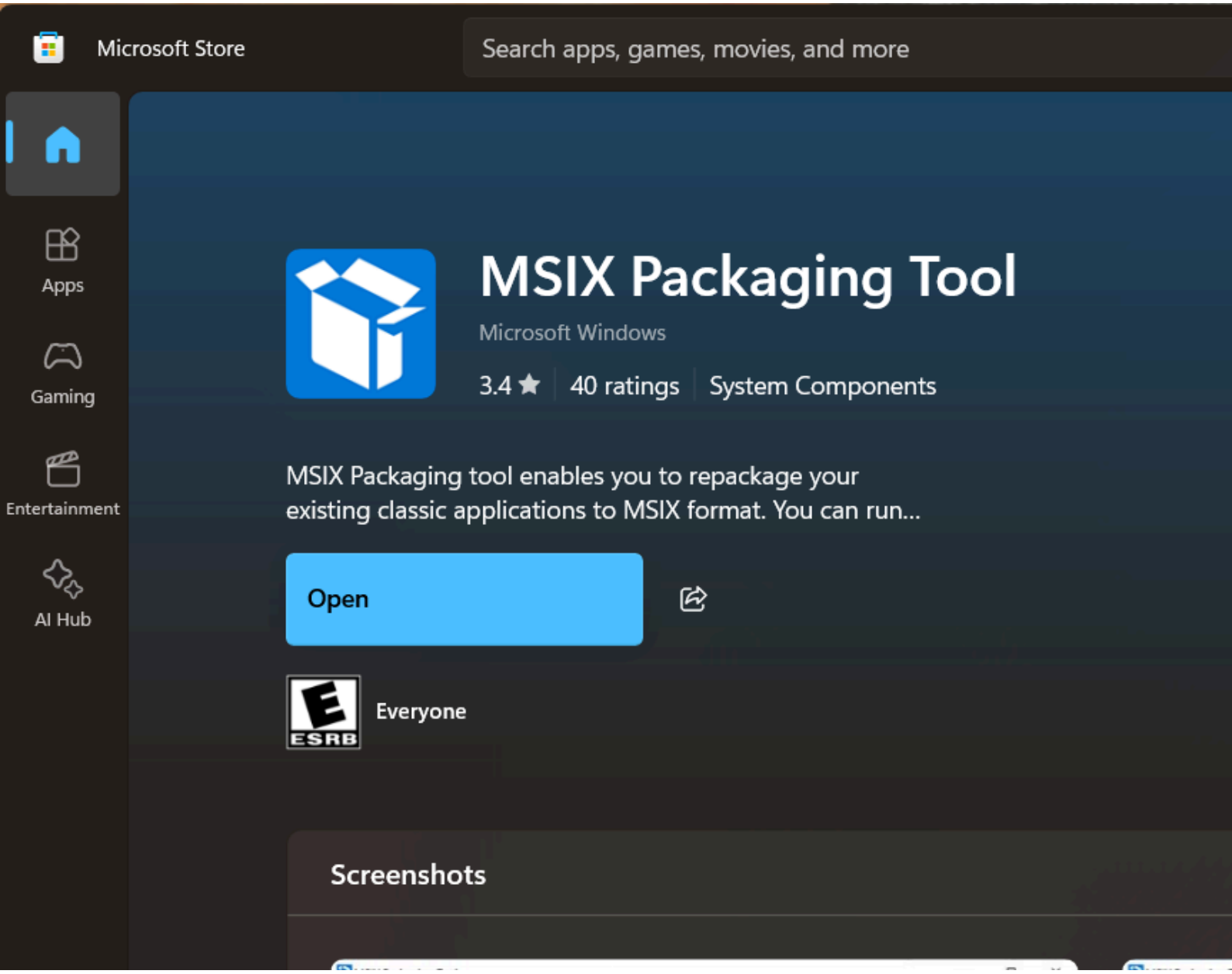
Prerequisite:-

- **Windows 10 version 1809 or later**
- **MSIX Packaging tool need to be installed on the system**
- **.msi file (**That is to be converted**)**
- **Installation Dependencies and Certificates if needed.**
- **A signing certificate (.pfx) to sign the MSIX package (can use a test certificate for internal use)**

Procedure

Step 1:-

The first and most important thing is to download the MSIX packaging tool From Microsoft Store



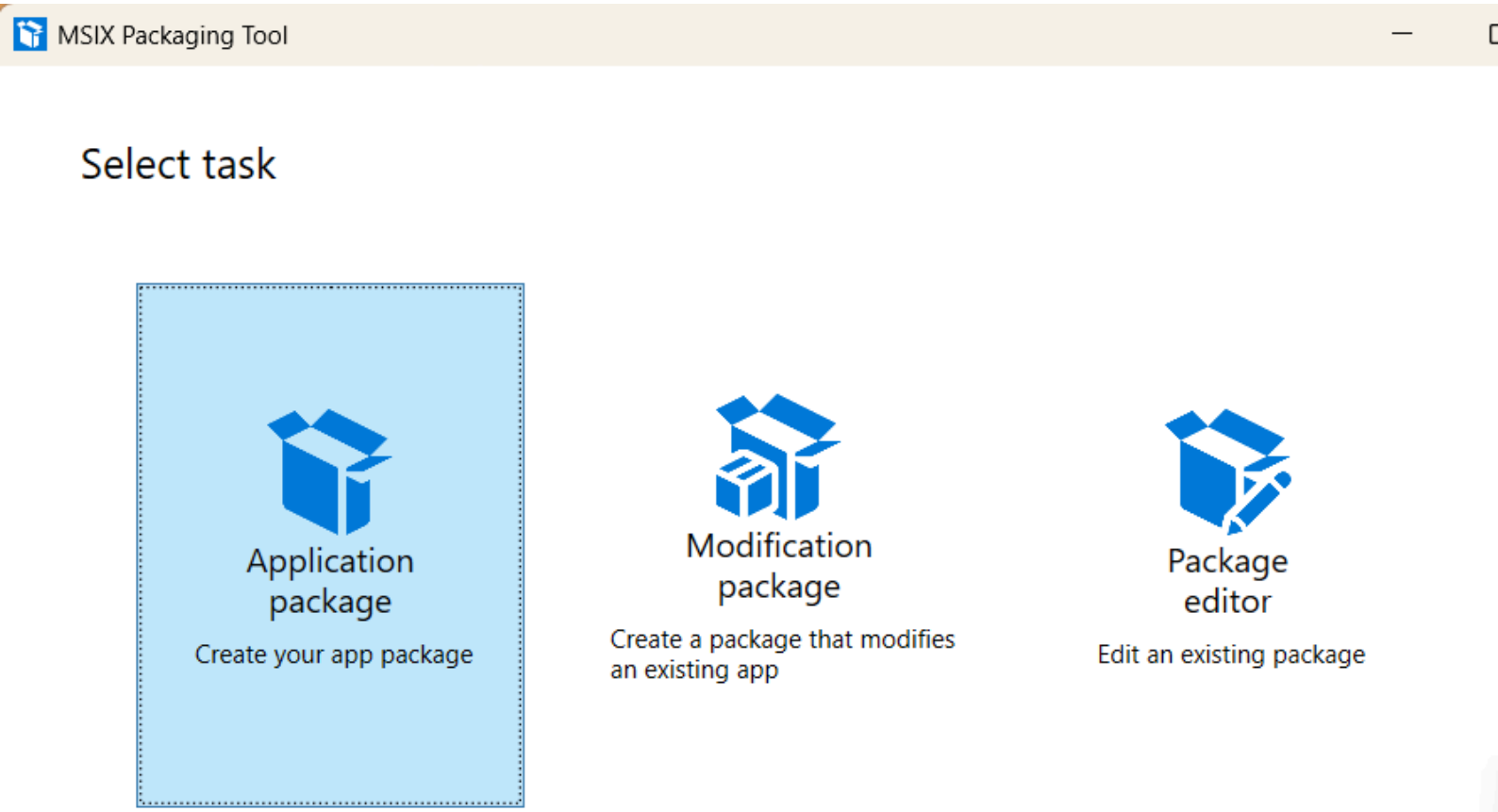
Once it is Downloaded and Installed

Click on Open to Launch MSIX Packaging tool

Step 2:-

Choose the Conversion Option

- Select **"Application package"** on the welcome screen.
- Click on **"Create package on this computer"**.



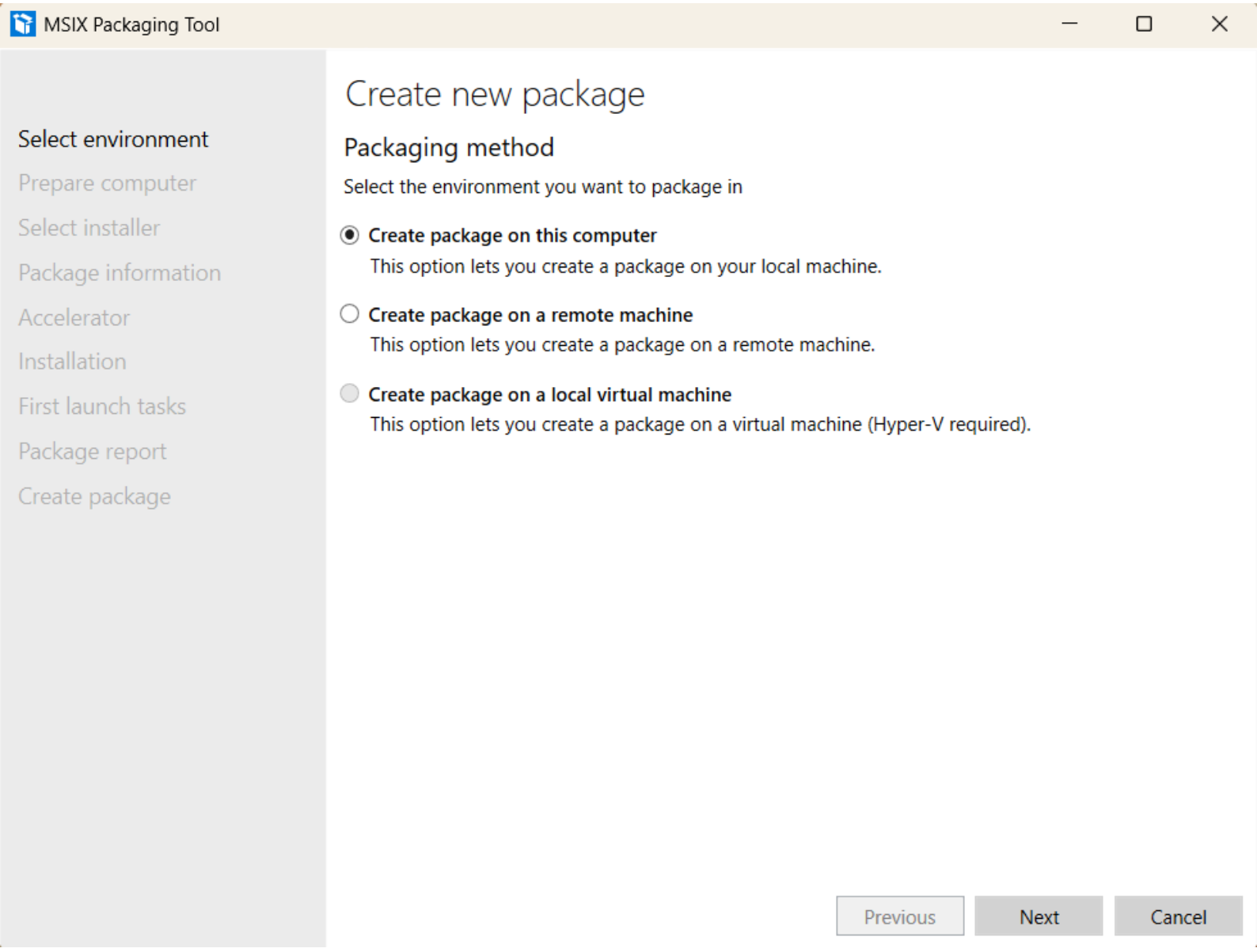
This option enables us to create a new MSIX package

Step 3:-

Select the environment we want our package to be

In this case we are going to choose to

Create the package on our local Machine



To move on to the next step

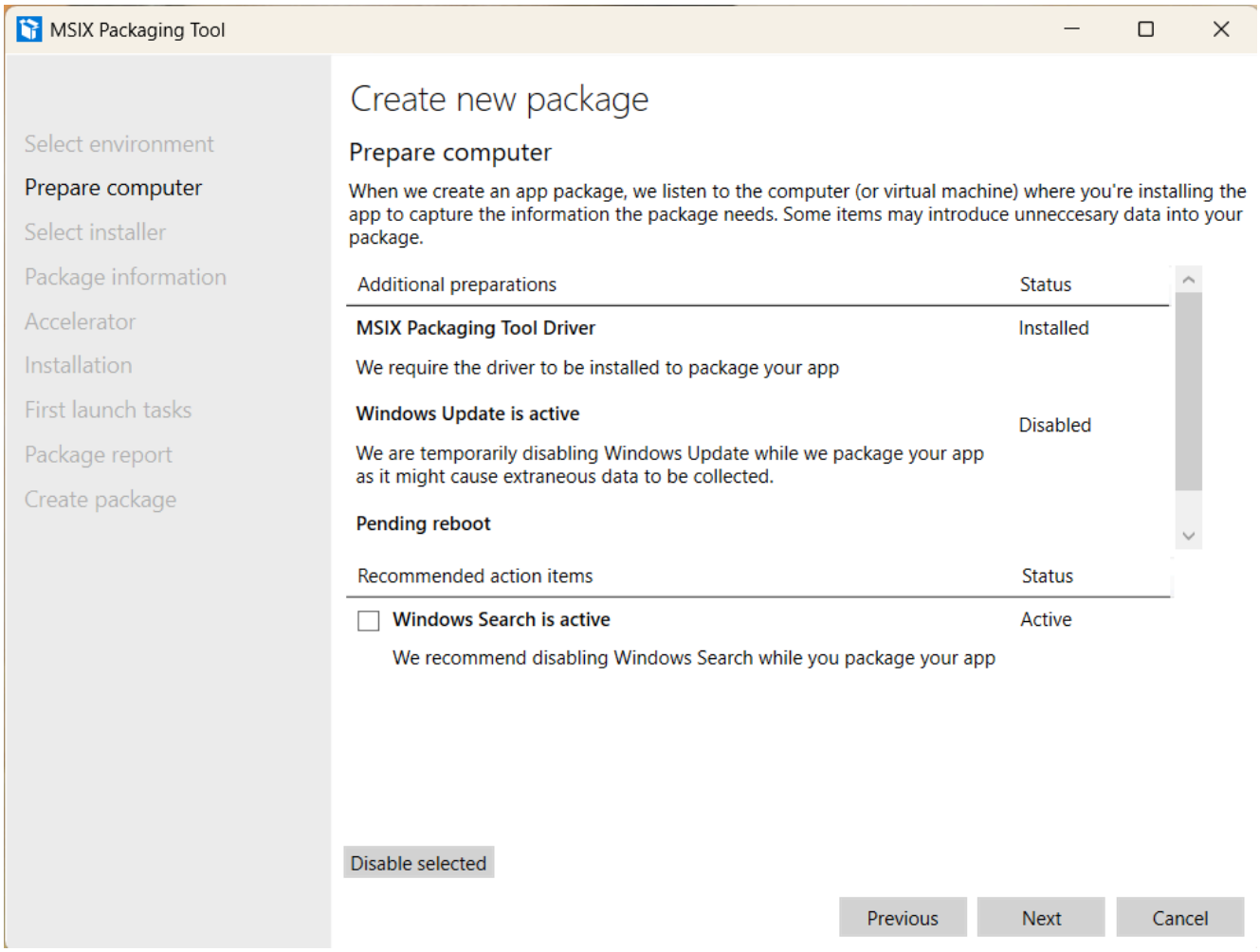
Click on the NEXT button

We then click on the NEXT button to move forward

Step 4:-

Prepare the Computer for Packaging and Check if required additional tools are required.

In this case MSIX Packaging Tool Driver is Checked and Detected as installed



Click on the NEXT button to proceed Further

Step 5:-

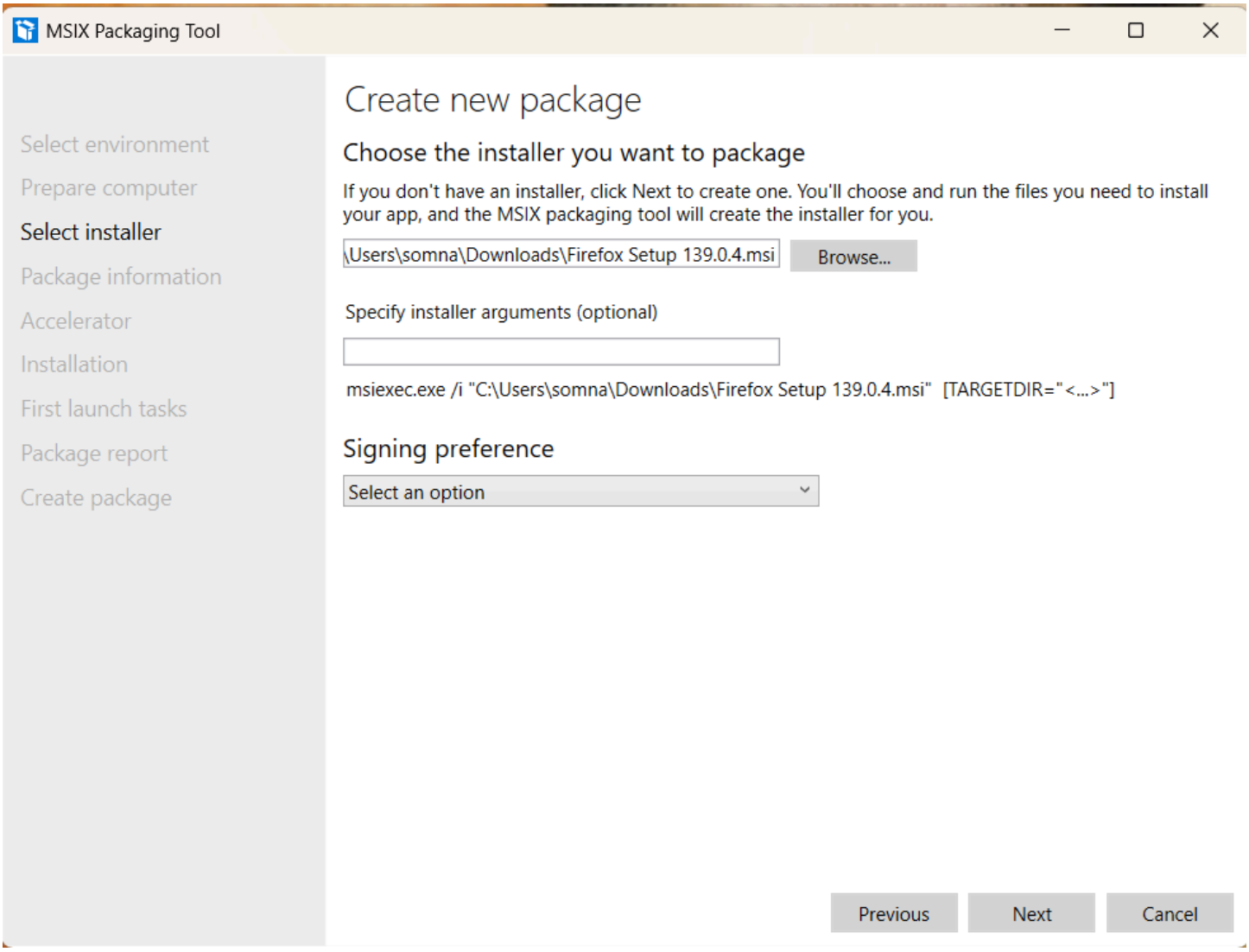
Choose the installer file.

Here we have to choose the .msi installer file that to be converted into MSIX .

Use the Browse button and select the .msi file

In this case I have used *Firefox Setup 139.0.4.msi file*

The MSIX Packaging Tool captures the installation process automatically, so it must be able to run your installer without user interaction.



We can also set up sign in preference in this case i choose with DO NOT SIGN IN

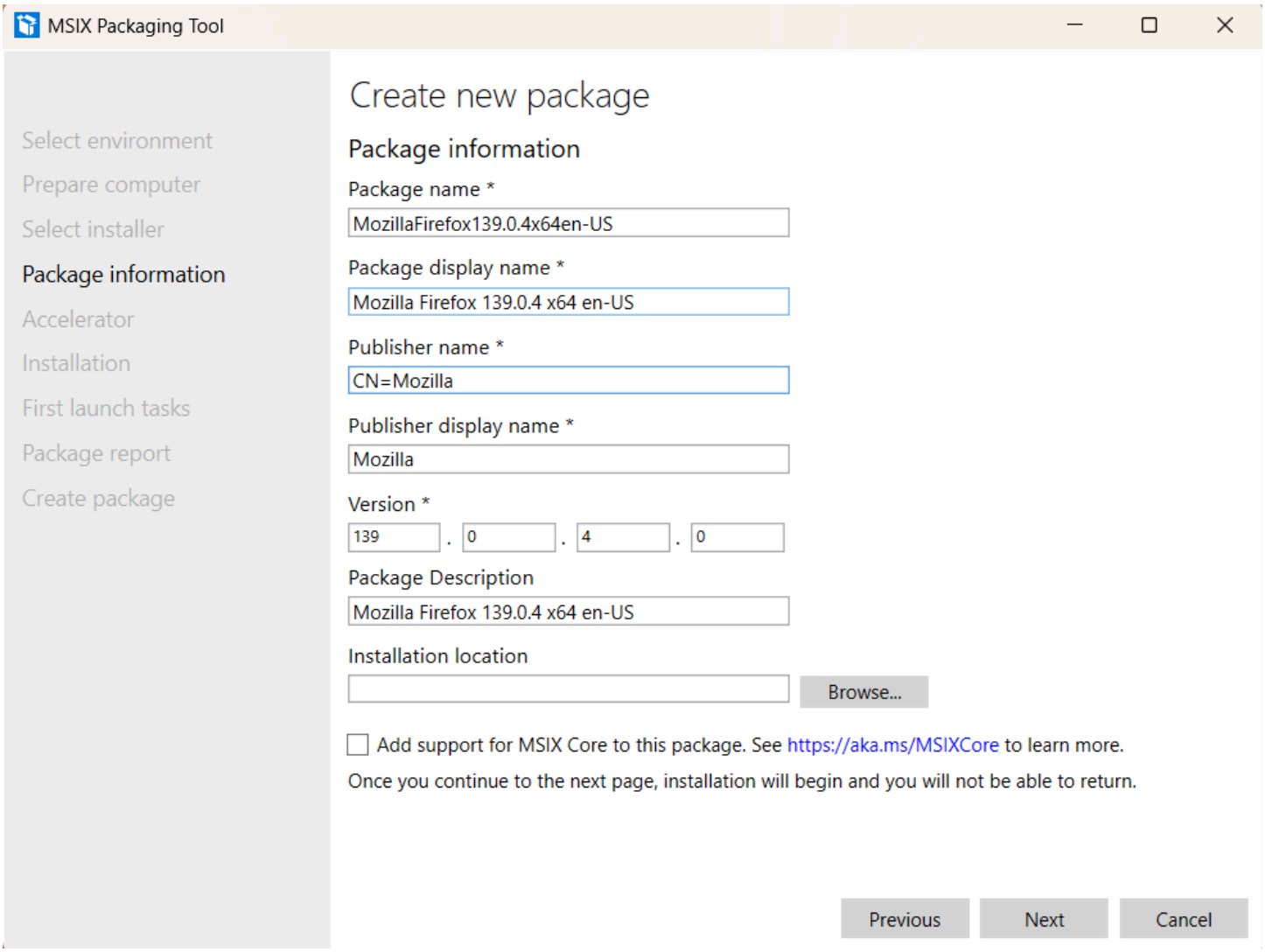
Click on the NEXT button to proceed Further

Step 6:-

Specify Package Information

The tool may ask you to provide:

- **Package Name**
- **Publisher Name (should match your certificate). (CN=Publisher Name)**
- **Version. (139.0.4.0)**
- **Display Name**
- **Logo (optional)**



We can specify Installation Location here as well

Step 6:-

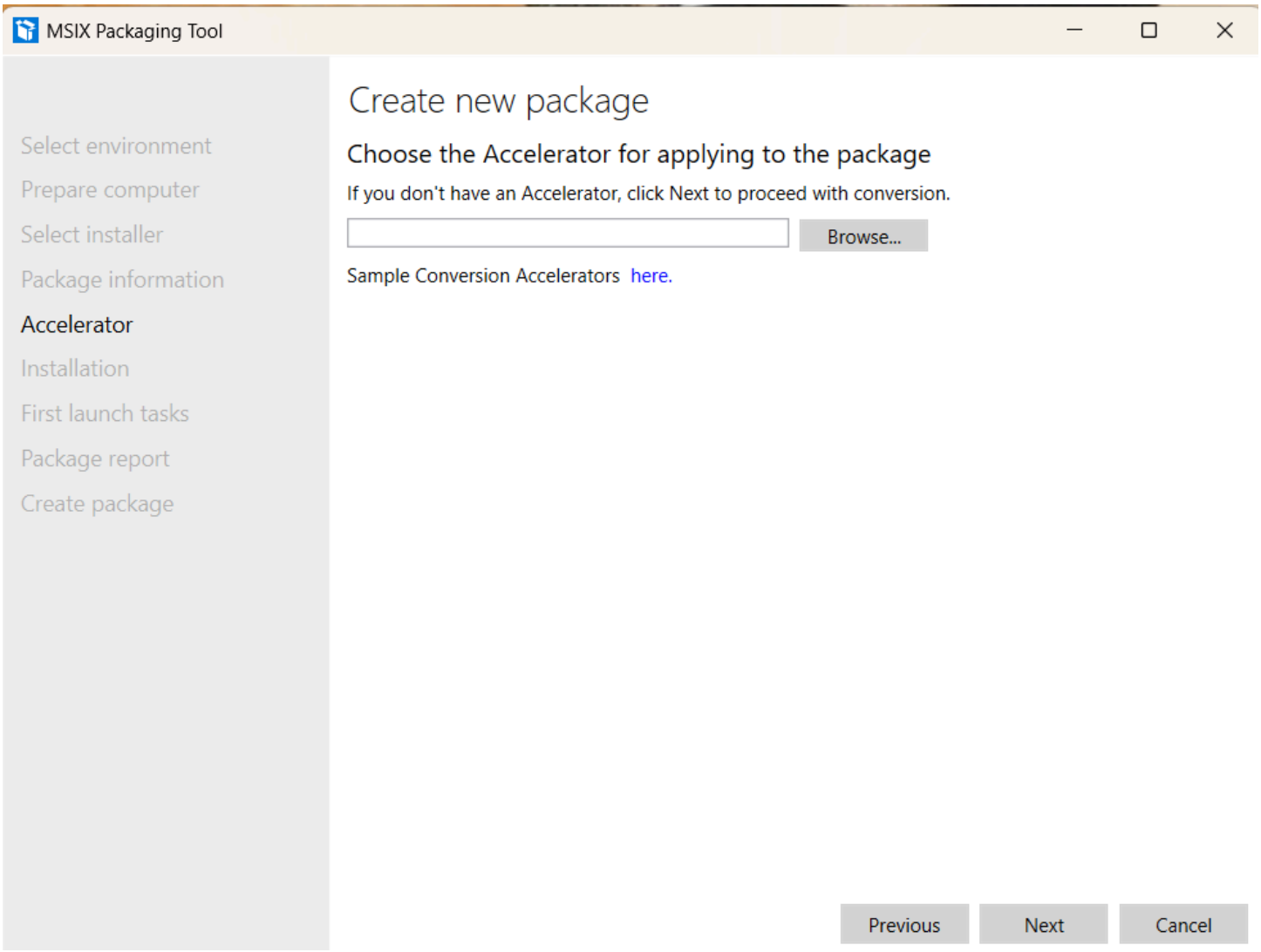
Choose an Accelator

The Accelerator is a **template or pre-configured package configuration** that helps **speed up** the repackaging process and **standardize** MSIX package creation across teams or projects.

Purpose of Accelerator

The Accelerator feature is designed to:

- **Reduce manual input** during packaging
- **Enforce consistency** (e.g., naming, capabilities, entries)
- **Save time** for repeated conversions or large-scale deployments



We haven’t Specified any Accelerator as we dont have any specific Accelerator file

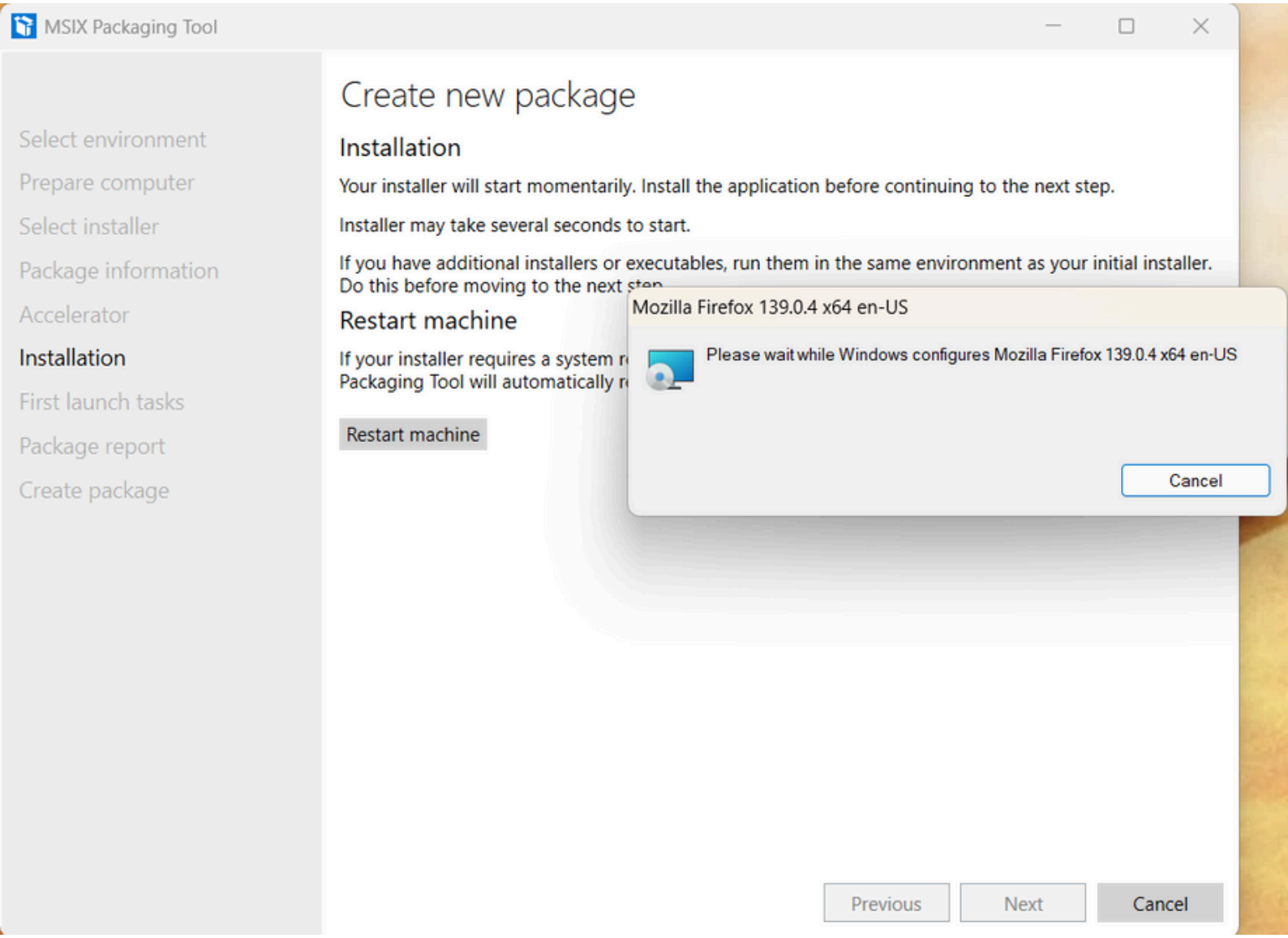
Click on the Next Button to move on to Next Step

Step 6:-

Installation

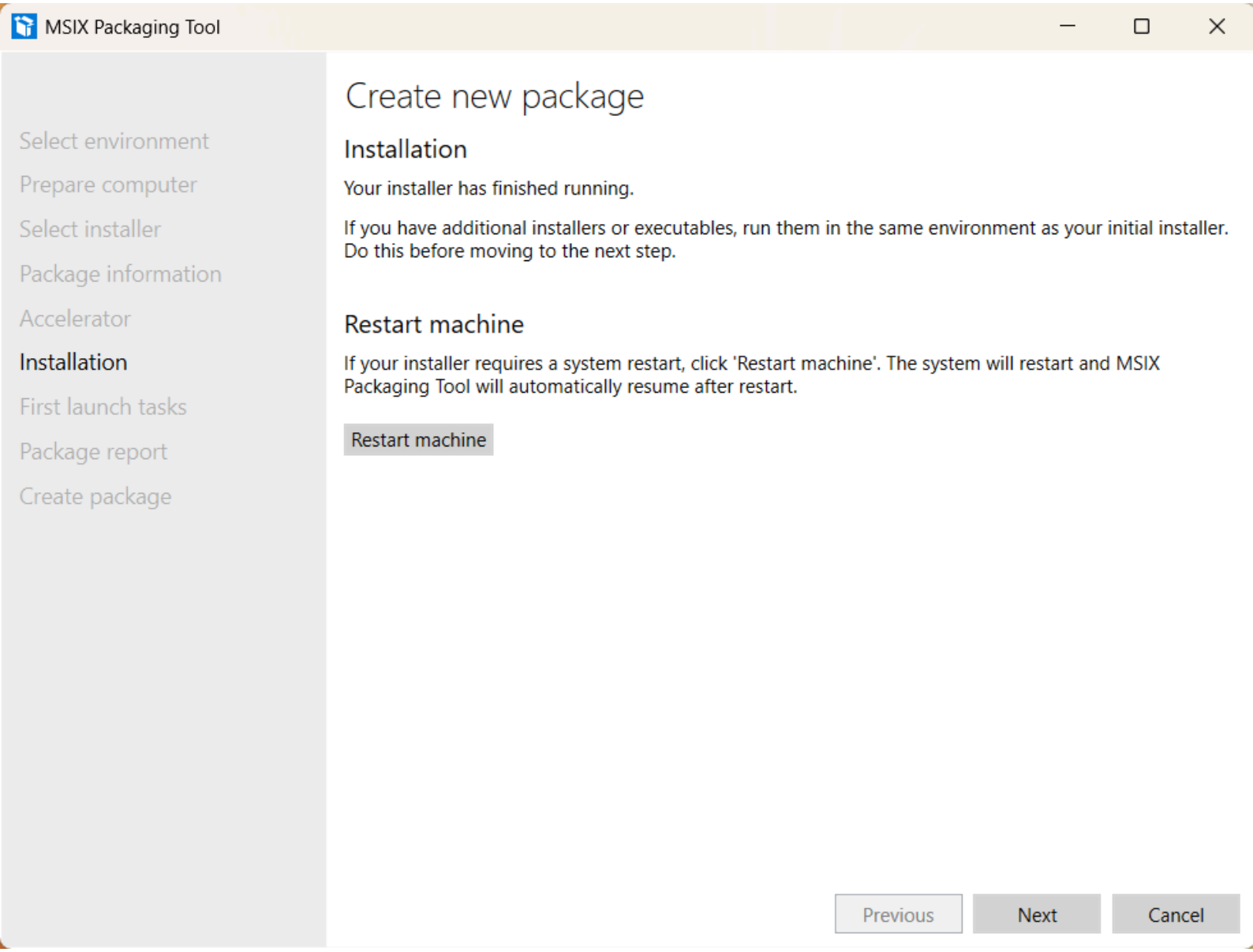
This step monitors and records the changes made to the system while your MSI or EXE installer is running, including:

- **Files created or modified**
- **Registry entries added or changed**
- **Shortcuts created**
- **Services installed**
- **Application binaries and dependencies**



How it works:

1. **The MSIX Packaging Tool launches your installer silently (if you specified installer arguments).**
2. **It captures all changes made to the system during the installation.**
3. **You can manually install other dependencies at this point (e.g., additional tools or runtimes).**
4. **Once installation is complete, MSIX Packaging Tool prepares to wrap everything into the .msix package.**



We can Restart the Machine if Needed

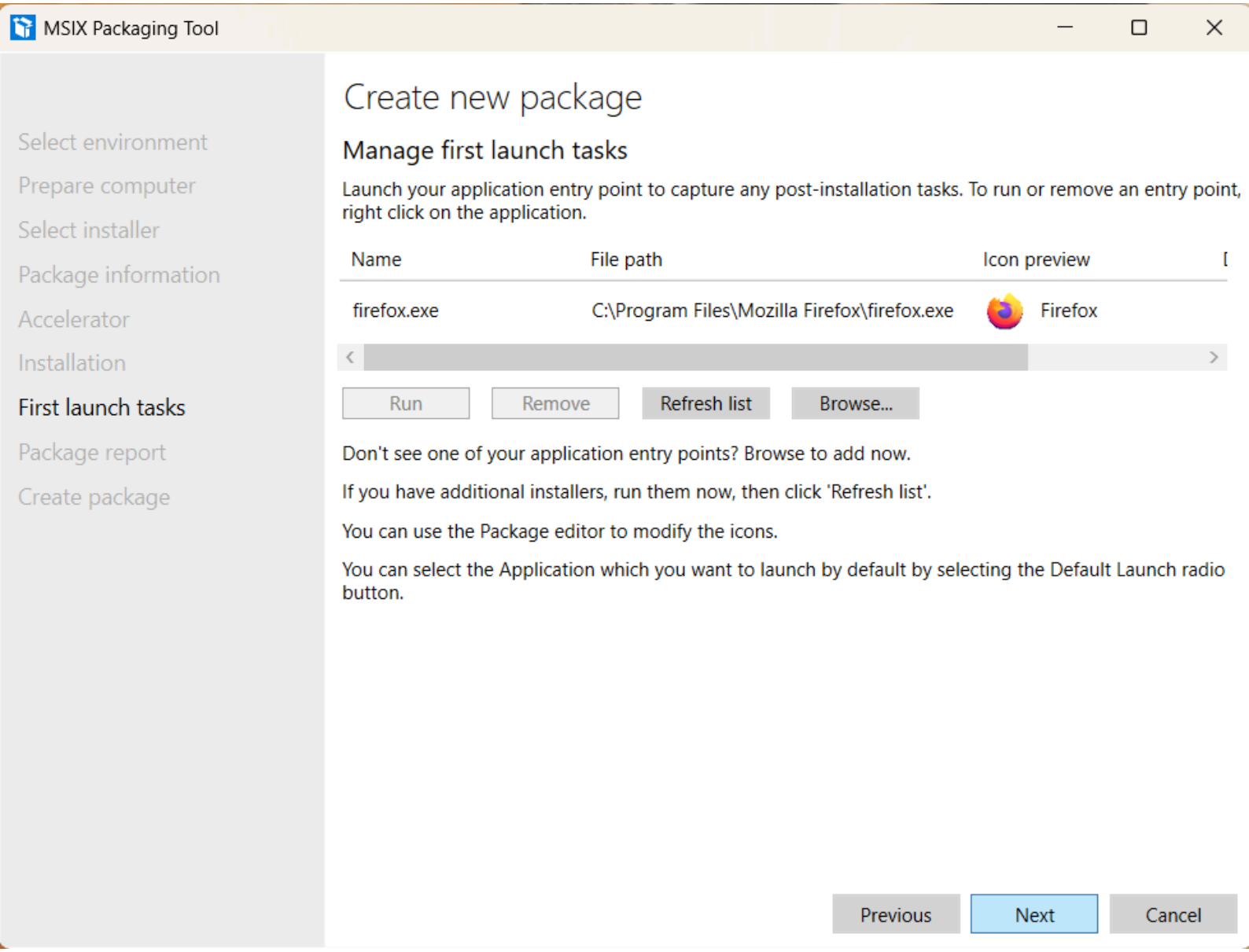
Click on the NEXT Button to Move Further

Step 7:

Many applications perform important tasks after installation but before first use, such as:

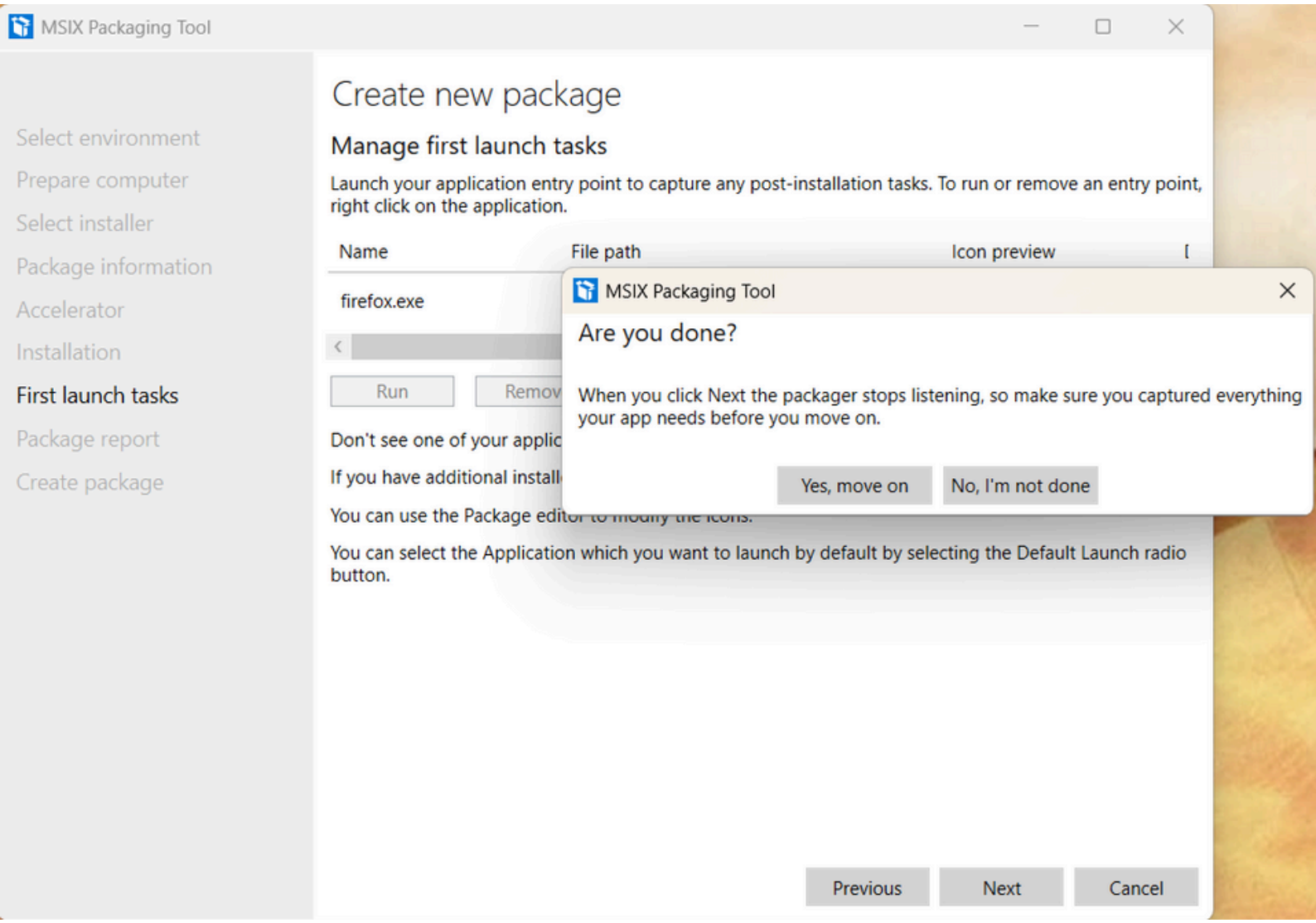
- **Creating configuration files**
- **Registering user-specific settings**
- **Downloading components**
- **Initializing databases**
- **Setting up plugins or extensions**

The MSIX Packaging Tool can capture these changes and include them in the MSIX package to ensure the app is ready to use immediately after deployment.



Click on NEXT

Step 8:



A Dilog box Appears that says

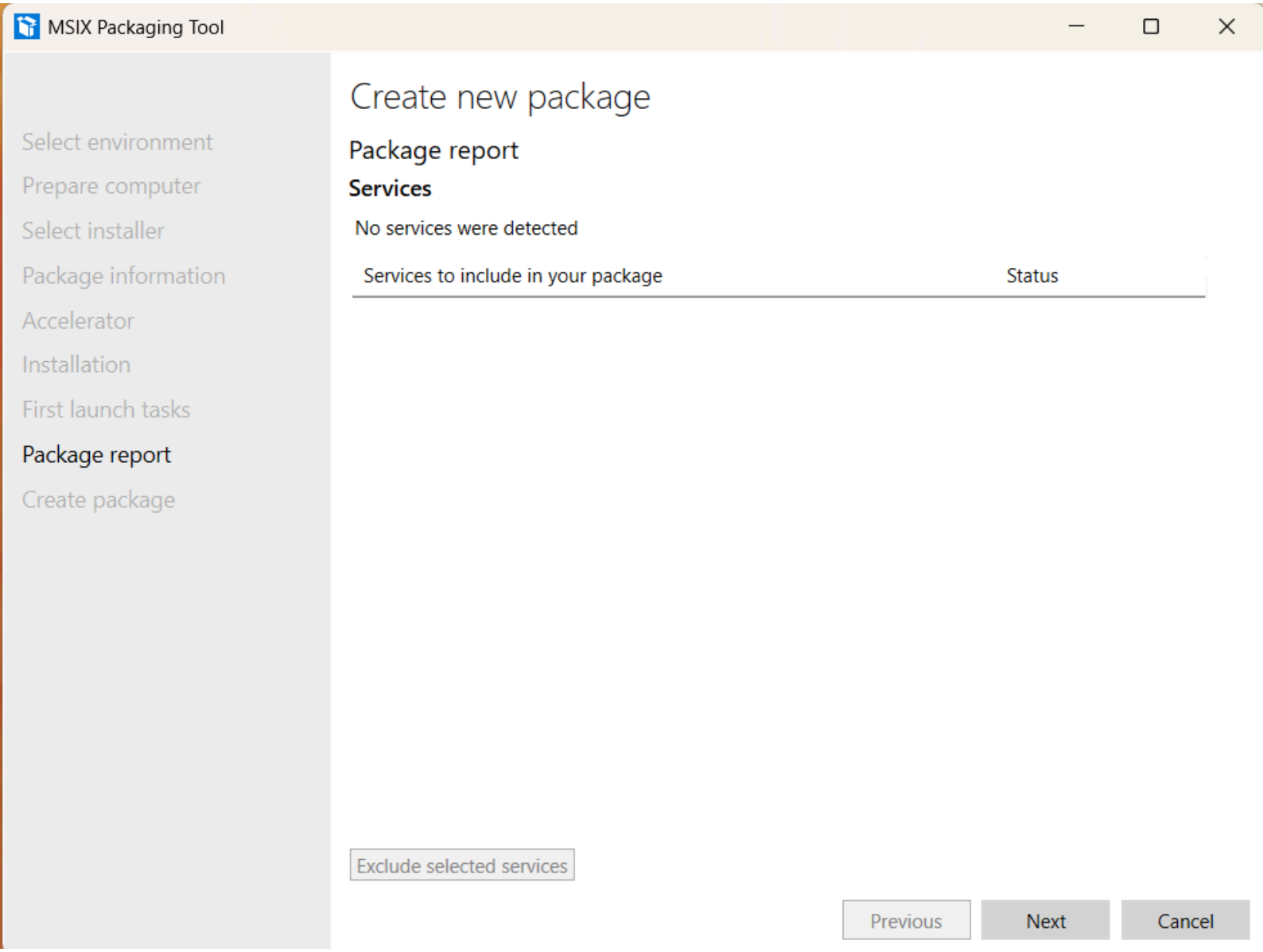
When you click Next the packager Stops Listening and we have to make sure we have captures everything that the app needs Before we Move Further

CLICK ON YES, MOVE ON

Step 9:

This Step allows Review Windows services that were installed or modified by the application

But in this case there is no services Detected so we m0ve on to the next part



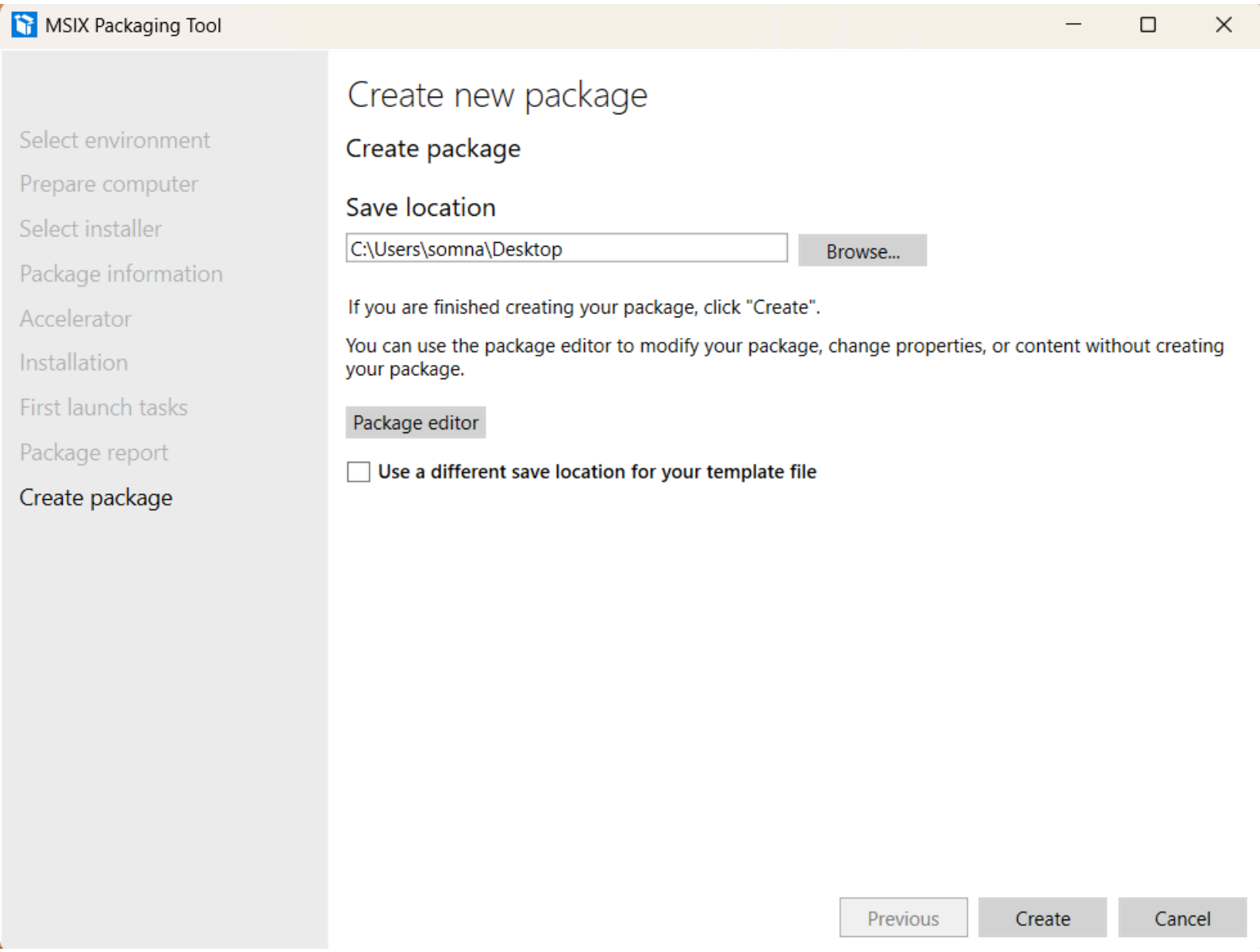
Click on the NEXT Button to move further

Step 10:

Create New Package

Now we have moved to the final step that says create a Package

Specify the Path where the Package is to be Stored



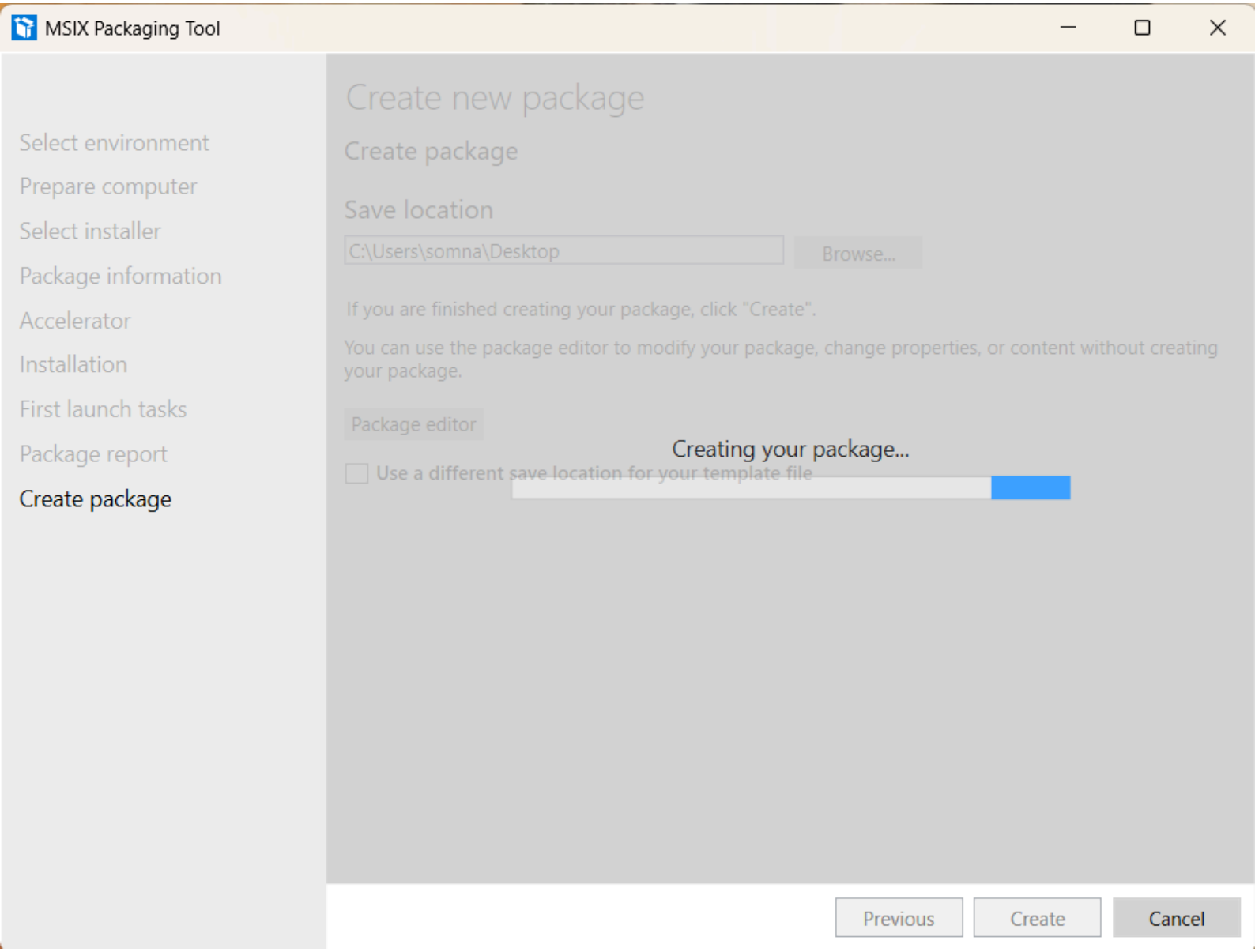
Also we can use Package Editor Option to Modify, Change the Properties of the Package without Creating it

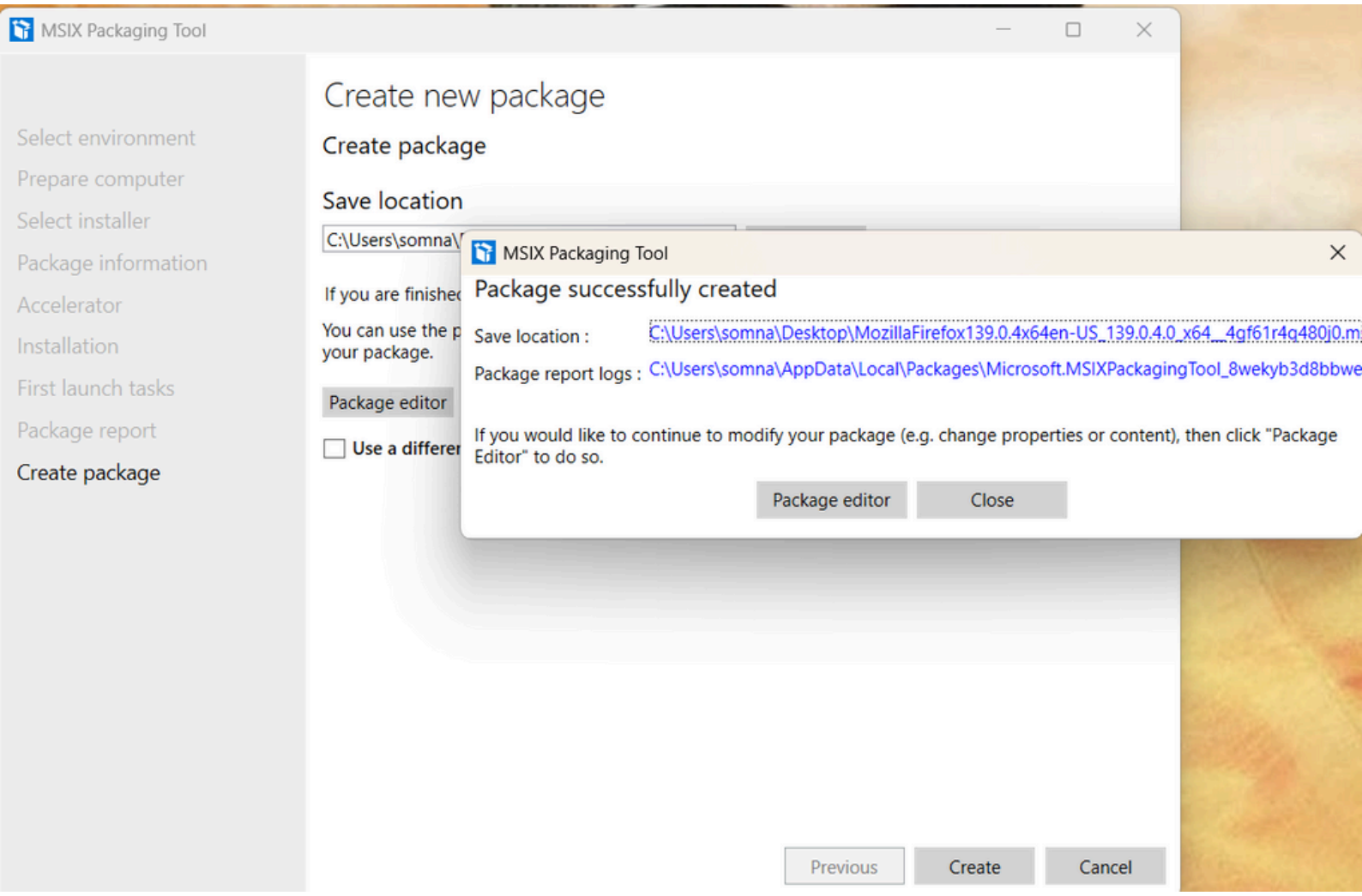
Step 10:

Finally there is a Button to Create Package

Clicking on it a pop up Window is Shown

- The Pop up states that The Package has been Successfully Created
- It Specifies the Save Location for the PAckage
- Also the Location for the Report Log Files

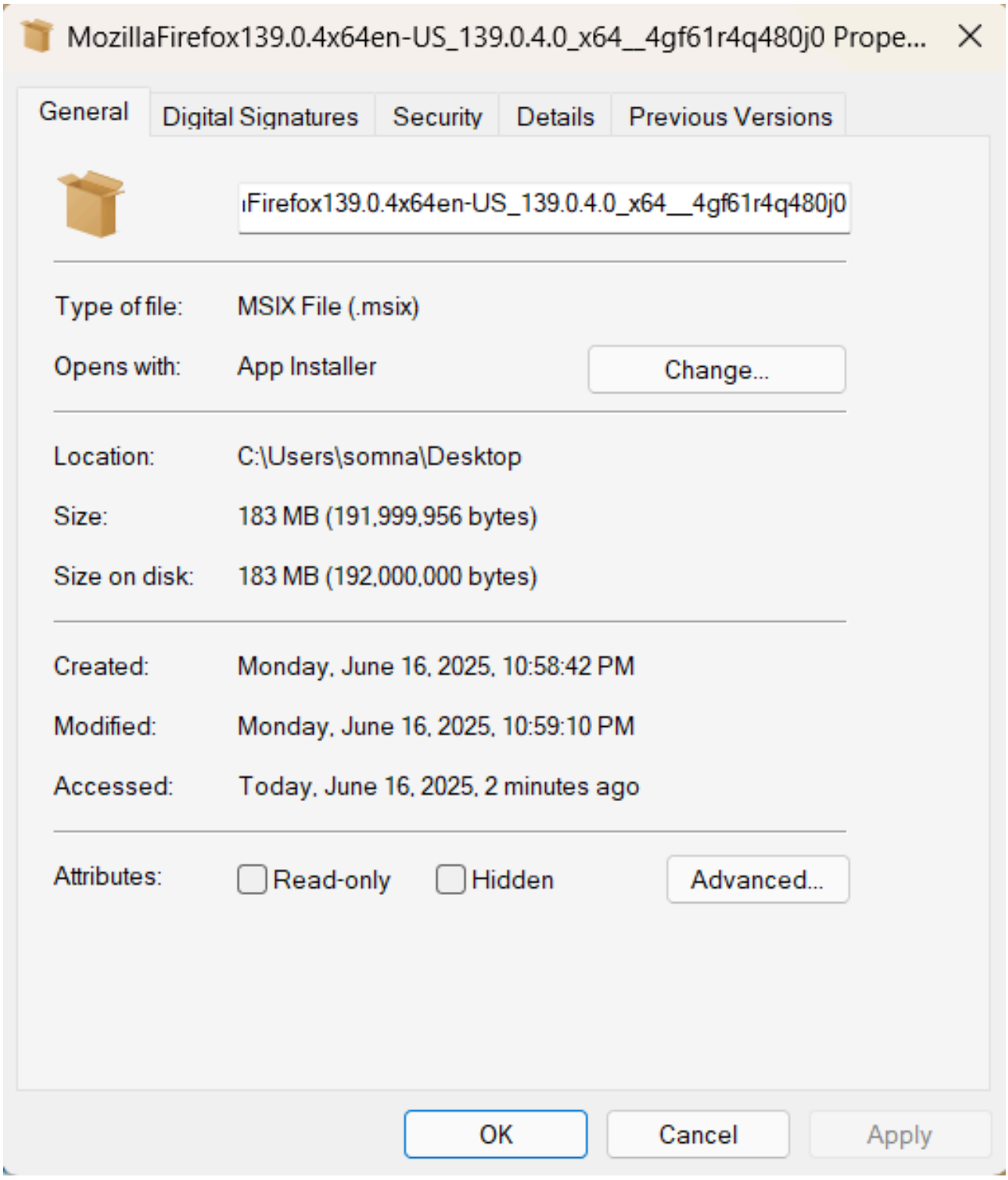




Click on the Close Button if we need not to Edit the Package At this stage our package is Successfully Created

We can now Close the setup Tool

Output



Now We have our .MSIX File Ready as shown above

Conclusion

MSIX Packaging Tool allows developers and IT professionals to modernize their application deployment strategy.

MSIX brings enhanced security, better management, and cleaner installs/uninstalls. T

his project outlines a structured approach to migrate legacy MSI setups into the MSIX format,

enabling future-proof app deployment and improved end-user experience.