**Today Assignment –**

**Date: 5/08/2025**

Name: Prasang Kumar Singh

Gmail: [thakurprasang23@gmail.com](mailto:thakurprasang23@gmail.com)

Topics:-

**1.Device Drivers - Handling Within MSI, Related Files and Locations, Examining Logs**

Device drivers are software pieces that let Windows talk to hardware like printers, graphics cards, or keyboards. In MSI packages, adding drivers requires extra care because MSI doesn’t handle complex driver installs by itself. You can include driver files such as .inf, .sys, and .cat and call tools like pnputil or DPInst to install them during setup. These tools copy drivers to system folders like C:\Windows\System32\drivers or the DriverStore. To confirm the driver installed properly, you can check the Windows Event Viewer or create MSI logs by running msiexec /i setup.msi /l\*v log.txt. The logs will show any errors or success messages, which helps troubleshoot problems. Always make sure you use signed drivers so that Windows allows installation without warnings or blocking.

**Summary:**  
MSIs use extra tools to install drivers. Always check logs to ensure drivers are installed correctly.

**2.Identifying Kernel Drivers. Handling Kernel Drivers within MSI**

Kernel drivers operate deep inside Windows and control critical hardware functions. These drivers usually end with .sys and must be signed to meet Windows security standards. MSI packages alone can’t register or start kernel drivers automatically, so scripts or commands like sc.exe are often used in combination to register them as services. When packaging, you need to verify compatibility with the operating system version and architecture (32-bit or 64-bit). To check installed drivers, you can use Device Manager or the driverquery command in Command Prompt. If the driver is not signed or installed correctly, Windows may block it or show errors at startup.

**Summary:**  
Kernel drivers are low-level components needing signed files and special commands. MSI alone doesn’t handle them fully.

**3.Handling Services and Printers in MSI**

MSI makes it possible to install and control Windows services using its built-in ServiceInstall and ServiceControl tables. This lets you create, start, or stop services during installation. Services often run in the background to support the main app. For installing printers, MSI doesn’t provide direct support, so you typically call a script or utility like rundll32 printui.dll to add a printer driver or configure a printer. Both services and printers usually require administrator rights to install successfully. When packaging, you should test on clean systems to confirm the service starts correctly and the printer is available to all users.

**Summary:**  
Services can be set up within MSI. Printers need extra scripts or tools for installation.

**4.Excel, PowerPoint, and Word Add-ins**

Office add-ins are small programs or files that extend what Excel, Word, or PowerPoint can do. They might be .xlam files for Excel or .dll and .vsto files for other Office apps. To install add-ins, you copy them to specific folders and create registry entries under HKCU\Software\Microsoft\Office\ to tell Office where the add-in is and when to load it. MSI can do this automatically using file and registry tables. One important detail is making sure Office is installed before running the MSI. You should also test in a user profile to confirm the add-in shows up in the Office app’s add-ins list.

Summary:  
Office add-ins are installed by copying files and updating registry keys. MSI makes this easier by automating those steps.

**5.COM Add-ins and Load Behaviour Value**

COM Add-ins are special components that connect with Office programs or other apps. They are usually .dll files registered in the Windows registry. A key part of their setup is the Load Behavior registry value, which controls when the add-in loads. A Load Behavior of 3 means the add-in loads automatically and stays enabled, while 2 means it won’t load by default. MSI can set these values when installing the add-in by adding the right registry entries. Sometimes problems come up if the add-in has missing files or wrong permissions, so it’s important to test that it loads correctly in the target Office app.

**Summary:**  
COM Add-ins use the Load Behavior value to control loading. MSI can set this up with registry entries.

**6.Difference Between User, Admin, and System Context**

In MSI installations, the context decides who gets access to the software and what permissions it has. User context means the app installs only for the currently logged-in user and doesn’t need admin rights. This setup stores files and settings in the user’s profile. Admin context installs the software for all users and requires administrator permission to change system areas. System context is used in automated deployments with tools like SCCM, where the installer runs in the background as the Local System account without user prompts. Each context affects how the app is installed and who can use it.

**Summary:**  
User context is for one user without admin rights. Admin and system contexts install for everyone with higher permissions.

**7.Windows 10 vs Windows 11 – Basics Relevant to App Packaging**

Windows 10 and Windows 11 are similar under the hood, but Windows 11 introduces stricter security and some visual changes. Windows 11 enforces driver signing more strictly, which means unsigned drivers may fail to install. Smart App Control and virtualization-based security (VBS) can block older apps. The Start menu and taskbar behave differently, so shortcuts and UI integrations may need adjustments. System folders and registry paths largely remain the same, so most MSI packages made for Windows 10 work on Windows 11 with no changes. However, it’s always important to test your package to catch compatibility problems early.

**Summary:**  
Windows 11 adds stricter security and UI changes. Most MSIs still work but testing is needed to be sure.

**8.Handling Scheduled Tasks and Their Use Cases**

Scheduled tasks let you run commands or scripts automatically on a schedule or when events happen, such as when a user logs on or the system starts. MSI can’t create scheduled tasks directly using built-in tables, but you can include custom actions or scripts to call schtasks.exe, which sets them up. Scheduled tasks are useful for running maintenance scripts, applying updates, or setting up user preferences after login. They can be created for all users or just specific accounts. When adding them, always make sure your MSI runs with enough permissions to create the task successfully.

**Summary:**  
Scheduled tasks are made using schtasks.exe in MSI setups. They help automate actions after install or at login.

**9.Logon Scripts to Populate User Profile Data in MSI Application Packaging**

When an MSI installs an application for all users, it usually can’t set up personalized settings for each user because it installs at the system level. This becomes a problem when the application needs user-specific files, folders, or registry entries, especially under HKEY\_CURRENT\_USER. To solve this, logon scripts are used. These are small batch or PowerShell scripts that automatically run when a user logs into Windows. They help complete tasks that MSI can’t do directly, like copying config files to a user’s profile, creating folders under C:\Users, or adding registry values under the current user’s key. These scripts can be deployed along with the MSI or triggered by group policies or scheduled tasks. For example, if an app needs a settings file inside each user’s Documents folder, the logon script will create that file during login. This ensures every user has their setup ready without manual work. It’s especially useful in enterprise environments where many users share a single device.

**Summary:**  
Logon scripts add user-specific settings after installation. They help finish tasks MSI can’t do on its own.