# **Wipro training**

# **Assignment Day -2**

Batch no.-1 Date:22-05-2025

25VID1334-DWS

### **User Context, System Context, and Admin Context:**

In Windows operating systems, understanding the distinction between **User Context**, **System Context**, and **Admin Context** is critical for effective application packaging, deployment, and troubleshooting. **User Context** refers to processes and settings that are specific to the currently logged-in user. Actions in this context use the user’s permissions and are stored in the hkcu (HKEY\_CURRENT\_USER) part of the Windows Registry. This context is essential for applications that require user-specific configurations, like interface themes, recently opened files, or user-specific paths. In contrast, **System Context** refers to processes that run independent of the user and apply globally to the operating system, affecting all users. Configurations in this context are typically stored in (HKEY\_LOCAL\_MACHINE) and require elevated or administrator privileges to modify. System services, startup programs, and machine-wide software installations operate in this context. Lastly, **Admin Context** refers to tasks that require elevated privilegeeither under the built-in Administrator account or using User Account Control (UAC) elevation. This context is essential for making changes to protected areas of the system, installing applications, configuring security settings, and modifying system-wide registry keys. Understanding these three contexts helps administrators and packagers ensure correct behavior and security for software installations and system configurations.

### **Logon Scripts:**

**Logon scripts** are a staple of enterprise IT environments, commonly used to automate tasks that should occur when a user logs into their machine. These scripts can be assigned via **Group Policy** or placed in the user’s profile directory. They may execute tasks like mapping network drives, configuring printers, setting environment variables, or launching applications that must run at user login. Logon scripts can be written in batch files, VBScript or modern PowerShell scripts Among these, PowerShell is generally preferred for its advanced capabilities, integration with Windows Management Instrumentation (WMI), and better error handling. While logon scripts execute in **user context**, they can inherit elevated privileges if configured correctly in policy. A best practice is to keep scripts as short and efficient as possible, logging execution details to help with debugging and using condition checks to avoid repeated executions when unnecessary. Additionally, IT administrators should ensure the scripts are compatible with both Windows 10 and Windows 11, especially when dealing with path changes or modern authentication methods.

### **Active Setup in MSI Packages:**

**Active Setup** is a Windows feature used during software deployment to perform **per-user configuration** for applications that are installed in **system context** (machine-wide). This is particularly useful in enterprise environments where software is deployed using tools like SCCM,

Best practices

* Error handling
* Security
* Testing
* Documentation

### **Windows 10 and 11 Benefits:**

Both **Windows 10** and **Windows 11** offer a solid foundation for enterprise application deployment and packaging, but Windows 11 brings several enhancements that are particularly relevant to modern IT practices. Windows 10 introduced broad compatibility, stability, and support for legacy applications, which made it a preferred platform for many years. It supports App-V, PowerShell, and a wide range of device management and deployment tools. However, Windows 11 builds on this by improving support for **MSIX**, a modern packaging format that enhances security, reduces application conflicts, and simplifies updates. It also features a more consistent and user-friendly UI, better integration with virtualization platforms like **Windows 365 and Azure Virtual Desktop**, and optimizations for the latest hardware architectures. Windows 11 includes improvements in **application isolation, containerization**, and **memory management**, which collectively reduce compatibility issues and increase overall performance. Additionally, security features such as **Windows Hello, Virtualization-Based Security (VBS)**, and **Secure Boot** are more tightly integrated in Windows 11, making it a better choice for secure enterprise environments. When planning application packaging, organizations should evaluate these benefits and consider transitioning to Windows 11 where possible.

### **Considerations for Application Packaging:**

Successful **application packaging** involves multiple technical and operational considerations. One of the most important decisions is whether the application should be installed **per-user** or **per-machine**. Per-user installs typically store configuration settings in HKCU and don’t require administrative privileges, making them ideal for apps like plugins or user-customized tools. In contrast, per-machine installs store settings in HKLM and apply to all users, but require admin rights. Packaging should also address dependencies, digital certificate signing (especially important for MSIX packages), and compatibility with **User Account Control (UAC)**. IT admins must ensure that installers don’t hard-code file paths or registry entries and instead rely on environment variables for portability. Modern packaging tools, such as **Advanced Installer, Flexera AdminStudio**, or even PowerShell scripts, can streamline the process. Testing is crucial—packages should be validated across different OS versions (e.g., Windows 10 vs. 11), user roles, and hardware profiles. Also, consider uninstall and rollback processes, logging for troubleshooting, and integration with deployment systems like **Intune, SCCM, or Endpoint Manager**.

### **How to Handle Scheduled Tasks:**

**Scheduled Tasks** allow you to automate scripts or applications to run at defined times, system events, or user logins. Managed through **Task Scheduler**, PowerShell or they can be configured to run in **system or user context** depending on their purpose. Tasks running under **system context** (e.g., SYSTEM, LOCAL SERVICE) have full privileges and are ideal for background operations or maintenance scripts. Tasks can be triggered by logon, startup, idle time, or on a schedule. In scripting, you can define the task’s triggers, actions, and security context. For example, a PowerShell command can register a task that runs a script at user logon without prompting for credentials. When creating scheduled tasks, ensure you manage permissions carefully—avoid saving plain-text credentials and use service accounts where needed. Use proper logging mechanisms within the task to track execution status. For visibility, always test the task under different user environments (admin, standard user, domain user) to ensure reliability and security.

### **HKLM vs HKCU:**

In the Windows Registry, the choice between **HKLM (HKEY\_LOCAL\_MACHINE)** and **HKCU (HKEY\_CURRENT\_USER)** directly affects the behavior and accessibility of configurations. HKLM is used for system-wide settings that apply to all users of the machine and requires administrative rights to modify. It is typically used by services, drivers, and applications that operate in the system context. Settings stored here are persistent across all user sessions. On the other hand, HKCU stores user-specific preferences and configurations, and does not require admin rights to access or modify. It is dynamically loaded from the user’s profile during login and only applies to the currently logged-in user. When packaging applications, the general rule is: use HKLM for installations, licensing, and configuration that should apply machine-wide; use HKCU for UI preferences, recently used file lists, and settings that vary per user. A clear understanding of these registry hives ensures correct software behavior and improves security by adhering to the principle of least privilege.