* **Difference Between User, Admin, and System Context:**

1. **User**:

This is the person who used the system (like a website, app, or software)

They can only access their information and do basic tasks (like reading content, creating an account, etc)

Ex. You are logging into your email.

1. **Admin: (Administrator)**

This person has higher access than the normal user.

They can manage users, change settings, and oversee how the system works.

Ex. The person who manages users and settings on the company’s website.

1. **System**:

This refers to the backend (the technology or software itself) that keeps everything running.

It includes automated processes, servers, and core configurations that users and admins don’t usually see.

Ex. The software and databases that power your email system

* **In short:**

**User**: uses the system

**Admin**: manages the system and users

**System**: the software and technology itself

* **Logon Scripts to Populate User Profile Data:**

A logon script is a small file (usually a batch file or PowerShell script) that runs automatically when a user logs into Windows

It tells the computer to do certain tasks to set up the user’s environment

Ex. Set printers: connect to the default printers

Set environment variables, like paths

Copy files, like custom templates or documents

Set registry value: for special configurations

* **Why use logon scripts:**

To automate the setup for every user so they get the same experience

To enforce policies without relying on manual setups

* **How does it work?**
* You write a script (for example: logon.bat or logon.ps1) with instructions.
* You configure group policy or Active Directory to run this script at login
* When the user logs in, the system runs the script automatically
* **Simple Analogy:**

Imagine you’re setting up a hotel room for a guest:

The logon script is like the checklist for housekeeping (set towels, fill mini-bar, turn on AC)

When the guest (user) enters (logs in), everything is ready based on the checklist

* **Active setup versioning to ensure it runs each time:**

Active setup is a feature in Windows that lets you run a command once per user when they log into the system.

It's especially useful for things like installing software or setting up user-specific settings after a fresh login

* **What does it do?**

When a user logs in, Windows checks a special area in the registry (active setup entries)

If the user has never run a particular setup before (or if the version number has changed), Windows runs the command

This command can do things like:

* Configure app settings for the user
* Set up shortcuts or user-specific registry keys
* Install small per-user components
* **Why use active setup?**

It's great for deploying software to multiple users on a computer

Ensures every user (including new ones) gets the same setup

Usually, when an admin installs an app (like Chrome or Office), it needs some configuration when each user logs in

* **How does it work?**
* The admin sets up an entry in the registry under:
* **HKLM\SOFTWARE\Microsoft\ActiveSetup\Installed Components\{YourComponentGUID}**

StubPath: the command or script to run

Version: the version number (increase it to force a re-run)

* It includes a command (in StubPath) to run and a version number
* When a user logs in, Windows checks if that user has run this active setup before
* If no, it runs this active setup and marks it has done
* If yes (and version matches), it does nothing
* If the version has changed, it runs again
* **Simple analogy:**

Active setup is like a welcome kit for every new employee at a company:

The system checks, “Has this person already received the kit?”

It no, it gives them the kit (runs the command)

If yes, it skips it. If the kit has been updated (version change), it gives them the new kit

* **Why versioning?**

Action setup keeps track of a version number (in the registry) for each component

If you increase the version number, it forces Active Setup to run again for every user (including fresh installs or after updates)

Versioning lets you control when the active setup runs again (by changing the version in the registry)

* **In short:**

**Logon script**: runs every time a user logs in to set up their environment

**Active setup**: runs once per user (or when version changes) to configure app/user settings during login

* **Windows 10 vs Windows 11 – key differences relevant to app packaging:**

1. **MSIX App Packaging:**

Both Windows 10 and Windows 11 support MSIX (the new packaging format).

Windows 11 has some extra improvements to make MSIX apps run faster and smoother.

1. **Windows Package Manager (Winget):**

Windows 11 comes with Winget built in (makes it easier to install packaged apps).

In Windows 10, you have to install Winget manually.

1. **Faster Logon & Setup in Win11:**

Windows 11 has a faster startup and login process, so older scripts (like logon scripts or Active Setup) might not run well if they take too long.

You should use modern packaging methods (like MSIX, Intune, or winget) for better compatibility.

1. **Better Security in Windows 11:**

Windows 11 requires things like TPM 2.0 and Secure Boot, which can affect apps that depend on older drivers or system tweaks.

Packaged apps should be tested on Windows 11’s stricter security.

1. **Compatibility:**

Most Win32 apps and MSIX packages will work the same on Windows 10 and 11.

However, Windows 11 has newer APIs and features that might only work on it.

* **Simple Table:**

| **Feature** | **Windows 10** | **Windows 11** |
| --- | --- | --- |
| **MSIX Support** | Yes | Yes (with improvements) |
| **Winget Built-In** | No (manual install) | Yes |
| **Login Speed** | Slower | Faster (could skip scripts) |
| **Security** | Standard | Stricter (TPM, Secure Boot) |
| **Compatibility** | Excellent | Excellent, but test for stricter rules |

* **In Simple Words:**

Windows 11 is faster, more secure, and more modern.

It handles new app packaging formats (like MSIX and winget) a bit better than Windows 10.

Older scripts (logon scripts, Active Setup) may need updates for Win11’s faster logins.

* **Handling Scheduled Tasks and their use cases. In a simple way:**
* **What are Scheduled Tasks?**

A Scheduled Task is a built-in feature in Windows that lets you automatically run a program, script, or command at a specific time or under certain conditions (like startup, login, or when idle).

It’s like setting a reminder for your computer to do something automatically, without needing you to start it.

* **Why Use Scheduled Tasks?**

They are used to:

Automate repetitive tasks (like backups, updates, or file cleanup).

Run programs/scripts at specific times (like every night at 2 AM).

Trigger actions when events happen (like when a user logs in or when the system starts).

* **Common Use Cases**

| **Use Case** | **What It Does** |
| --- | --- |
| **Backup Files** | Automatically back up important files daily at 1 AM. |
| **Run Maintenance Scripts** | Clean up temporary files or optimize system on startup. |
| **Send Notifications/Emails** | Send a report to IT if disk space is low. |
| **Start a Program at Login** | Launch a monitoring app when a user logs in. |
| **Re-run a Script on Schedule** | Ensure a data sync runs every 15 minutes. |
| **Software Updates** | Run a script to update apps or fetch patches daily. |

* **How Does it Work?**

1. You use **Task Scheduler** in Windows (a control panel app) to:
   * Set the **trigger** (like “at startup” or “every day at 5 PM”).
   * Choose the **action** (run a script, launch a program).
   * Optionally set **conditions** (like “only if the computer is idle”).
2. The system **watches for the trigger** and runs the action when conditions are met.

* **Simple Analogy:**

Think of a **Scheduled Task** like setting an **alarm clock**:

**When** it rings (trigger): at 7 AM every day.

**What** happens (action): it plays music (runs a script).

* **In Summary:**

**Scheduled Tasks automate things** you’d normally do manually.

They help keep the system running smoothly, reduce human error, and improve efficiency.

* **User Prompts for Various Use Cases. (Reboot Prompts, Close Open Apps, etc.):**
* **What Are User Prompts?**

User prompts are messages or dialog boxes that pop up to ask the user to take an action (like rebooting, closing an app, saving work, etc.).

They help the system communicate with users before taking actions that might affect their work.

* **Use Cases for User Prompts**

**1. Reboot Prompt**

**When?** After installing software or updates that require a system restart.

**What?** A prompt says:

* + “Your computer needs to restart to finish installing updates. Do you want to restart now or later?”

**Why?** To ensure users are aware that a restart is required, and can save their work before restarting.

**2. Close Open Apps Prompt**

**When?** When installing updates or patches, or when running a script that needs exclusive access to files.

**What?** A prompt says:

* + “Please close the following applications to continue: [List of Apps]”
  + Sometimes gives an option to close them automatically or cancel.

**Why?** To avoid errors or data loss by ensuring no apps are using the files.

1. **Save Work Prompt**

**When?** When an app is closing unexpectedly (e.g., forced reboot, system shutdown).

**What?**

* + - “Do you want to save your changes to [Document] before closing?”

**Why?** To prevent data loss.

**4. Confirmation Prompt Before Action**

**When?** Before running scripts, applying configurations, or making system changes.

**What?**

* + - “Are you sure you want to proceed with the system cleanup?”
    - Options: Yes / No

**Why?** To confirm the user’s intent before proceeding.

**5. Informational Prompt**

**When?** After completing an action or update.

**What?**

* + - “Update complete! Please restart your computer to apply changes.”

**Why?** To notify the user and guide the next steps.

**Simple Summary Table**

| **Use Case** | **Example Message** |
| --- | --- |
| **Reboot Prompt** | “Restart required. Restart now or later?” |
| **Close Open Apps** | “Please close [App] to continue installation.” |
| **Save Work** | “Save changes to [Document] before closing?” |
| **Confirm Action** | “Are you sure you want to delete these files?” |
| **Inform User** | “Update complete. Restart your PC to finish.” |

* **Analogy**

Think of user prompts like a **polite assistant**:

They **ask for permission** before making big changes.

They **remind you** to save your work.

They **explain what’s happening**.