**Assignment 3**

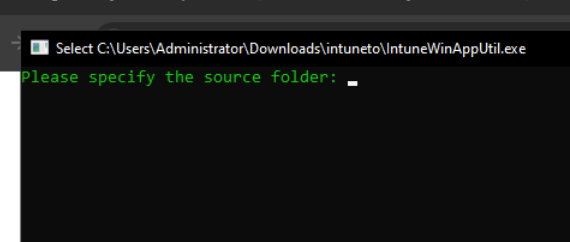
Name: Rahul kumar

Batch Start Date: 2025-08-04

Batch Name: WiproNGA\_DWS\_B5\_25VID2550

User ID: 34722

* **IntuneWin conversion - Compatible version to upload to Intune**
* We will gonna need Win32 content prep tool for converting files to .intunewin file

**

This is how the tool will look like

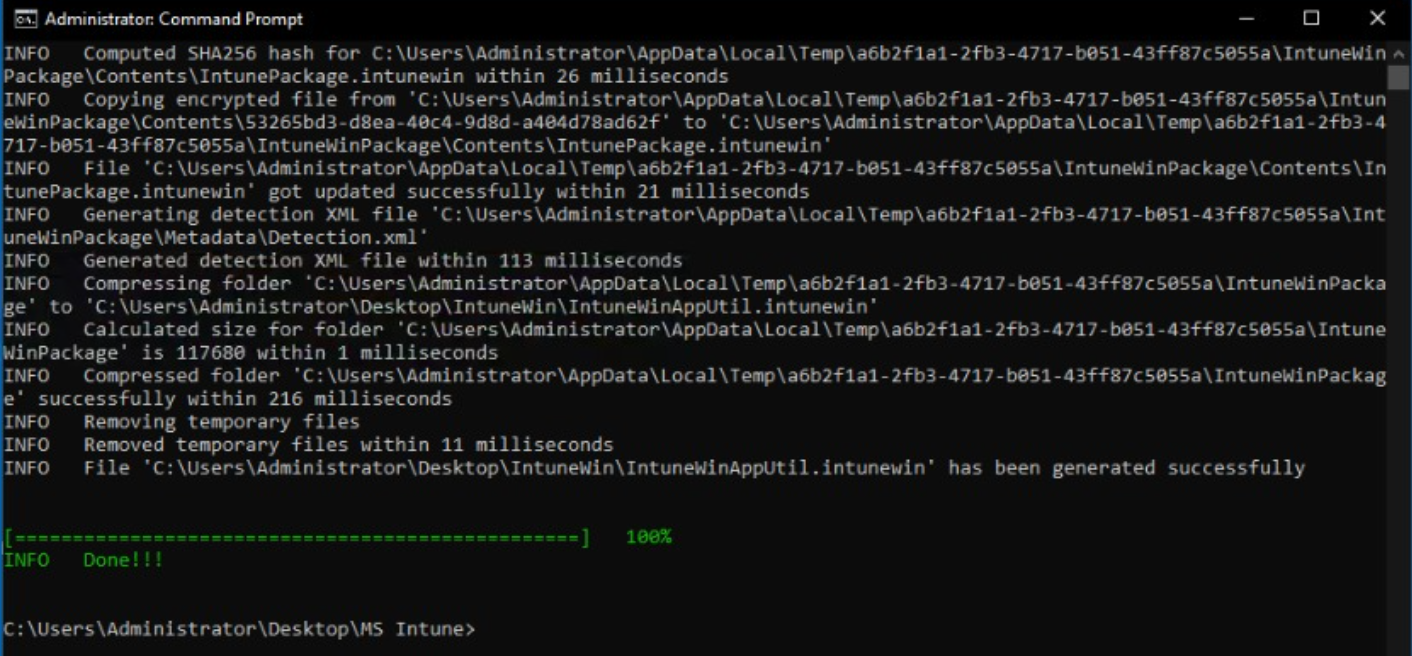
**Steps for conversion of file to intunewin**

1. **Prepare your application**

***A computer with a cd and a computer

AI-generated content may be incorrect.***

1. **Use the win32contentprep tool to convert it to .intunewin file**

******

1. **Create the .intunewin file**

*A white paper with a corner curled up

AI-generated content may be incorrect.*

1. **Upload to intune**

These steps are below on how to upload

**Walkthrough over Intune Applications (Windows)**

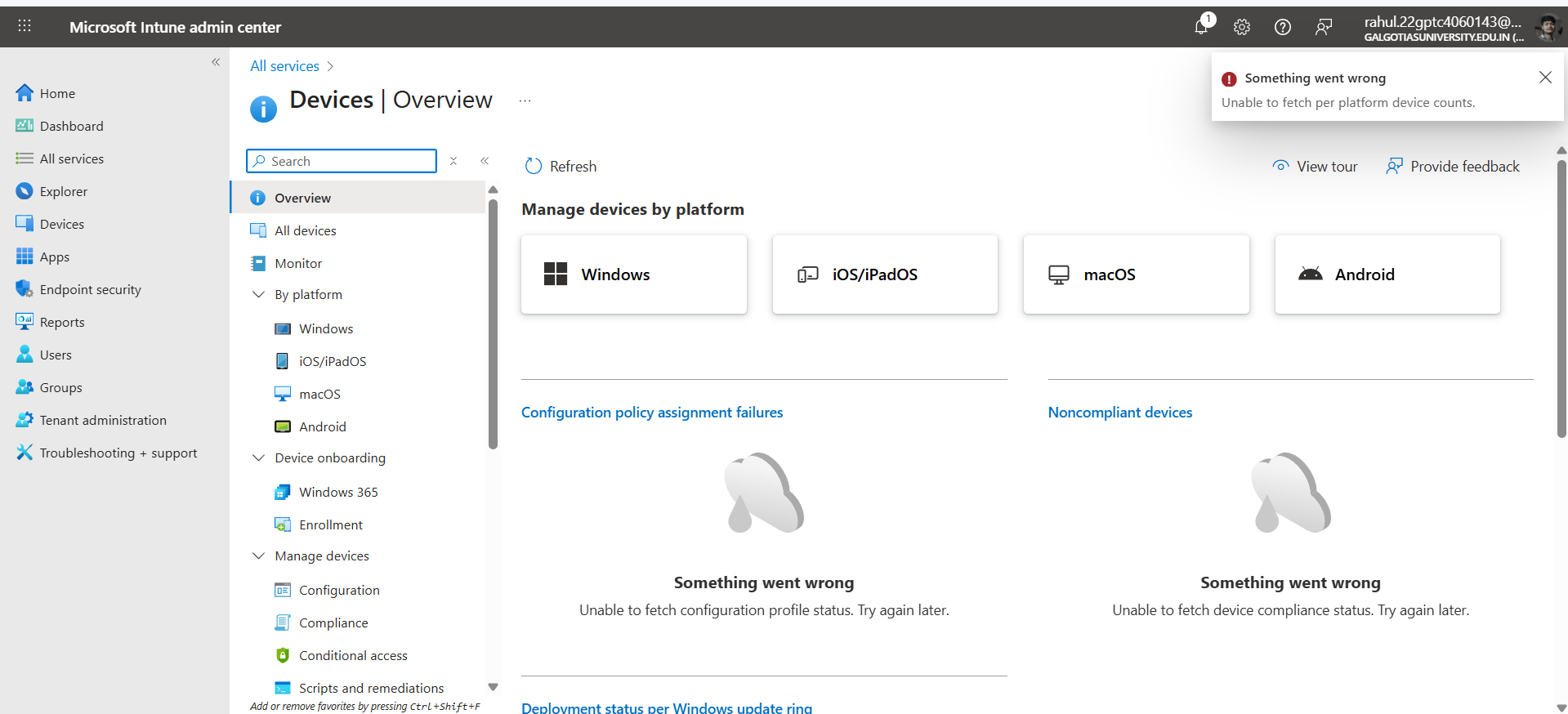
To deploy windows apps using Intune we will first need to create the app package using the Win32 content prep tool, add that to intune and assign it to the desired devices or users.

1. **Prepare your app**

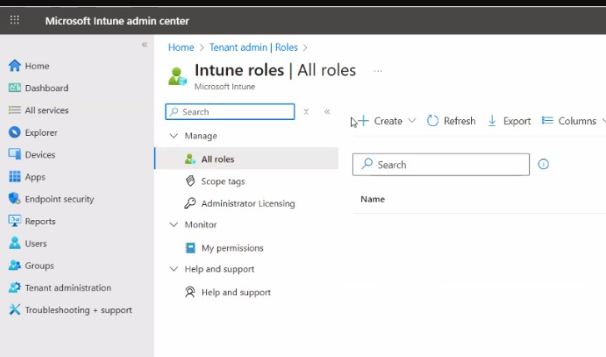
A white paper with a corner curled up

AI-generated content may be incorrect.

1. **Adding the App to Intune**

****

1. **Assigning the app**

******

1. **Managing the deploypment**

***A screenshot of a computer

AI-generated content may be incorrect.***

**Interactive and Non-Interactive Applications**

|  |  |  |
| --- | --- | --- |
| Feature | Interactive Applications | Non-Interactive Applications |
| User Input Required | Yes – prompts, dialogs, GUI | No – runs silently or in background |
| Execution Context | Foreground (user session) | Background (SYSTEM or service context) |
| UI Presence | GUI-based | CLI or headless |
| Automation Compatibility | Low – needs user interaction | High – ideal for automated deployments |
| Deployment Tools Compatibility | May fail in SCCM, Intune, MDT without switches | Works seamlessly with deployment tools |
| Examples | Setup wizards, Office installers, Adobe Reader | PsExec, Sysmon, MSI with /quiet, PowerShell scripts |
| Error Risk in SYSTEM Context | High – may hang or crash | Low – designed for background execution |
| Packaging Strategy | Requires silent switches or wrapper scripts | Native silent execution or scriptable |
| Monitoring Tools | Use Process Explorer to inspect UI threads | Use ProcMon to trace background activity |

**Process Flow for an Application on Windows client via IME service**

**Intune Management Extension (IME) – Application Deployment Flow**

**1. Polling**

* **What happens**: IME service regularly checks with Intune servers for new or updated app assignments.
* **Why it matters**: Ensures the client device stays up-to-date with the latest deployment instructions.

***2*. Detection**

* **Before Installation**:
  + IME evaluates detection rules to check if the app is already installed.
  + Prevents redundant installations.
* **After Installation**:
  + Detection rules are re-run to confirm successful deployment.
* **Detection Methods**:
  + Registry key/value presence
  + File existence
  + Running process check

**3. Installation**

* **Trigger**: If app is not detected and assigned, IME starts installation.
* **Steps**:
  + **Download & Unpack**: .intunewin package is downloaded and extracted to a staging folder.
  + **Execute Installer**: Admin-defined command (e.g., msiexec) runs to install the app.
  + **Monitor Progress**: IME tracks installation duration; if it exceeds timeout, it's marked as failed.

**4. Post-Installation Detection & Notifications**

* **Detection Recheck**: Confirms the app is properly installed.
* **Toast Notifications**:
  + Sent to user indicating success or failure.
  + Can include icons, text, and actions.
* **Device Restart**:
  + May occur based on Intune policy to finalize installation.

**Summary**

|  |  |  |
| --- | --- | --- |
| **Step** | **Purpose** | **Key Mechanism** |
| Polling | Sync with Intune | Periodic server check |
| Detection | Verify app presence | Registry, file, process |
| Installation | Deploy app | Download, execute, monitor |
| Post-Detection | Confirm success | Re-evaluate rules |
| Notifications | Inform user | Toast messages |
| Restart | Apply changes | Policy-driven |

**Registries with respect to LOB and Win32Apps**

**What Are LOB and Win32 Apps?**

|  |  |  |
| --- | --- | --- |
| Type | Meaning | Use Case |
| LOB (Line-of-Business) Apps | Typically packaged as .msi, .appx, or .msix | Used for internal business applications that are simple to deploy |
| Win32 Apps | Packaged using the .intunewin format | Ideal for complex apps like .exe installers, Office 365, SAP, or apps needing custom scripts |

**How They Fit into Intune Deployment**

* **LOB Apps** are straightforward and often used for MSI-based installations.
* **Win32 Apps** require conversion using the **IntuneWinAppUtil.exe** tool to wrap the installer and supporting files into a .intunewin format.
* Win32 Apps allow:
  + Custom detection rules
  + Return code handling
  + Reboot behavior control
  + Script integration (PowerShell, batch, etc.)
* **Registry and Troubleshooting Context**

Your syllabus also mentions:

* **Registry references** for installation status
* **Detection logic** using GUIDs and registry keys
* **Toast notifications** for success/failure
* **IME service flow** (Intune Management Extension) for Win32 app lifecycle: polling → detection → install → verification

**Why This Matters for You**

Given your work with MSI packaging, COM Add-ins, and Active Setup, understanding the distinction between LOB and Win32 apps helps you:

* Choose the right packaging strategy
* Implement advanced deployment logic
* Troubleshoot installations using logs and registry keys