**📚 What are LOB and Win32 Apps?**

**🏢 1️ LOB Apps (Line-of-Business Applications)**

✅ **What does LOB mean?**  
LOB stands for **Line of Business**. These are applications that are **critical** to running a specific **business process**. Think of them as **custom apps** a company uses to get its work done.

✅ **Examples of LOB apps:**

* A custom **inventory management** app for a warehouse.
* An **employee time-tracking** app for HR.
* A **CRM (Customer Relationship Management)** system used by sales teams.

✅ **Why are they special?**

* Often built **in-house** (custom) or bought from a specialized vendor.
* Not typically available to the public.
* Usually run on company-owned devices.

**💻 2️ Win32 Apps**

✅ **What is Win32?**  
Win32 refers to the **Windows API (Application Programming Interface)** used to build **traditional desktop apps** for Windows. These are the **classic Windows programs** you run on a PC.

✅ **Examples of Win32 apps:**

* **Microsoft Word**, Excel (older versions, non-cloud).
* **Adobe Photoshop**.
* **Legacy custom apps** that companies have used for years.

✅ **Why are they called "Win32"?**

* The **"32"** refers to 32-bit Windows architecture (even though now we also have 64-bit).
* They use the **Windows desktop environment** (with .exe files, for example).

**🤔 So, What’s the Difference?**

| **Feature** | **LOB Apps** | **Win32 Apps** |
| --- | --- | --- |
| **Purpose** | Custom apps for a business | Traditional Windows desktop apps |
| **Where from?** | Built/bought for internal use | Installed like regular software |
| **Technology** | Can be Win32, UWP, web, or others | Specifically built with the Win32 API |
| **Example** | Custom HR app | Microsoft Paint |
| **Distribution** | Managed via MDM (like Intune) | Manual install or managed through tools |

**🎯 Key Point**

* **LOB apps** can be **Win32 apps**, but they can also be web apps, modern apps, or even mobile apps.
* **Win32** is just a **technology type** (classic desktop), while **LOB apps** are about the **purpose** (business-critical apps).

**🔑 Simple Analogy**

Think of a company’s **custom internal software** (LOB apps) as **important tools for work**.  
And think of **Win32 apps** as the classic **Windows programs** like Word or Paint.

* **Intunewin conversion - Compatible version to upload to Intune:**

**📝 What is Intune?**

* **Microsoft Intune** is a cloud-based tool to manage devices and apps for a company.
* It helps deploy **apps (including LOB apps)** to users’ devices.

**🏢 Why Convert Apps to Intunewin Format?**

* Intune **requires certain apps (like Win32 apps)** to be in a special format called **.intunewin**.
* This helps Intune **package** the app and manage how it’s installed.

**🛠️ Steps to Convert an App into .intunewin**

Let’s say you have a classic **Win32 app** (e.g., a setup.exe or setup.msi) that you want to deploy via Intune.

Here’s how to convert it:

**🖥️ 1️ Use the Microsoft Win32 Content Prep Tool**

✅ **Download** the tool:

* Go to [Microsoft’s official page](https://github.com/Microsoft/Microsoft-Win32-Content-Prep-Tool) and download the **IntuneWinAppUtil.exe**.

✅ **What does this tool do?**  
It takes your app’s installation files and **wraps** them into an **.intunewin** package, which is compatible with Intune.

**🗂️ 2️ Prepare Your Files**

📁 Create a folder with:

* The **installer file** (setup.exe or setup.msi).
* Any **supporting files** (DLLs, config files, etc.).

**🔄 3️ Run the Tool**

* Open **Command Prompt** or **PowerShell**.
* Run a command like this:

**IntuneWinAppUtil.exe -c "C:\SourceFolder" -s "setup.exe" -o "C:\OutputFolder"**

Where:

* **-c** = Path to your app’s folder.
* **-s** = Name of the setup file.
* **-o** = Where to save the .intunewin file.

**📦 4️ Result**

* You get a **.intunewin** file in your output folder.  
  This is now ready to be **uploaded to Intune**!

**🟢 Compatible Version for Intune**

* **Intune supports .intunewin files** created with the **latest version** of the Win32 Content Prep Tool.
* You don’t need to worry about “versions” – just use the **latest Content Prep Tool** and your .intunewin file will be **compatible** with Intune.

**🔑 Key Takeaway**

* Convert apps using **IntuneWinAppUtil.exe**.
* The **.intunewin file** is the **correct format** to upload and deploy Win32 apps with **Intune**.
* **Interactive and Non-Interactive Applications**

**🖥️ What Are Applications?**

Applications (or apps) are programs that **do something for the user**. They can either:

* Work **with the user (interactive)**, or
* Run **on their own (non-interactive)**.

**🔄 1️ Interactive Applications**

These are apps that **require user input** to work.

✅ **What happens?**

* The user **clicks, types, or interacts** with the app.
* The app **responds immediately** based on the user’s action.

✅ **Examples:**

* **Web browsers** (Chrome, Edge)
* **Microsoft Word** (you type, it responds)
* **Games** (you control what happens)
* **Terminal/Command prompt** when waiting for your input

✅ **Key idea:**  
The app waits for the **user to do something** and then reacts.

**🔄 2️ Non-Interactive Applications**

These apps **don’t need user input** to run.

✅ **What happens?**

* They start, **do their job**, and finish – **without asking the user for anything**.
* Often used for **background tasks** or **automated processes**.

✅ **Examples:**

* **Scheduled tasks** (like backups)
* **Batch scripts** or **PowerShell scripts** that run without input
* **Windows Services** (which run in the background)
* **System processes** like antivirus scans

✅ **Key idea:**  
These apps **work automatically** – you don’t have to interact with them.

**🔍 Quick Comparison Table**

| **Feature** | **Interactive App** | **Non-Interactive App** |
| --- | --- | --- |
| **Needs User Input?** | Yes | No |
| **Runs in Background?** | Usually no | Often yes |
| **Examples** | Browsers, Word, Games | Scheduled Tasks, Background Services |
| **Usage** | Real-time user interaction | Automated or background work |

**🎯 Why It Matters?**

* **Interactive apps** are user-friendly and need **user attention**.
* **Non-interactive apps** are great for **automation** or **background work**.

**🗝️ Simple Analogy:**

* **Interactive App:** Like a cashier at a store – waits for you to give money and responds.
* **Non-Interactive App:** Like a **robot** that just stocks shelves without asking.
* **Required and Available App assignments**

**📚 What Are App Assignments?**

When you use **Microsoft Intune** (or other device management tools), you can **assign apps** to devices or users.  
But there are two main ways to assign them:

* **Required**
* **Available**

Let’s break these down.

**🟢 1️ Required App Assignment**

✅ **What it means:**

* The app is **automatically installed** on the user’s device.
* The user **cannot skip or ignore** the installation.

✅ **Why?**

* You want to make sure everyone **must have** the app.
* Examples: security software, corporate apps, VPN client.

✅ **User action needed?**

* **No** – the app installs on its own.

**🟡 2️ Available App Assignment**

✅ **What it means:**

* The app is **optional**.
* The user can **choose to install** it from the **Company Portal app**.

✅ **Why?**

* Useful for apps that some people **might need** but others don’t.
* Examples: optional productivity tools, additional utilities.

✅ **User action needed?**

* **Yes** – the user must **open the Company Portal** and choose to install it.

**🗂️ Quick Comparison Table**

| **Feature** | **Required App** | **Available App** |
| --- | --- | --- |
| **Auto-install?** | Yes | No |
| **User choice?** | No | Yes |
| **Where to find?** | Installs automatically | Company Portal |
| **Typical use case** | Mandatory apps | Optional apps |

**📝 Simple Analogy:**

* **Required App**: Like the company **automatically giving you a work laptop** – you have no choice.
* **Available App**: Like the company **offering you an extra mouse** – you can get it if you want.
* **Groups, Dynamic queries, Users**

**👥 1️ Users**

✅ **What are they?**

* **Users** are **individual accounts** representing real people (employees, students, etc.).
* Each user has a **username** and **password** (and often an email).
* Users can **sign in** to devices, apps, and services.

✅ **In Azure/Intune:**

* A **user** could be “John Smith” with his company email.
* Users can belong to **one or more groups**.

**📂 2️ Groups**

✅ **What are they?**

* **Groups** are **collections of users** (or devices) you can manage together.
* You assign **apps, policies, permissions**, etc. to the **group**, and all members get them.

✅ **Types of Groups:**

* **Security Groups**: Control access to resources like files, apps, etc.
* **Microsoft 365 Groups**: Combine permissions with collaboration tools like Outlook and SharePoint.

✅ **Why use Groups?**

* Instead of assigning things **one-by-one to each user**, you assign it **once to the group**, and all members get it.

**🌀 3️ Dynamic Queries (Dynamic Groups)**

✅ **What are they?**

* **Dynamic Groups** automatically **add or remove members** based on **rules/queries** you define.
* Instead of adding users **manually**, the group **updates itself** when users match the criteria.

✅ **Example Queries:**

* Add all users with **Department = HR**.
* Add all devices with **OS = Windows 11**.
* Add all users with **Job Title = Manager**.

✅ **Why use Dynamic Queries?**

* It **automates group membership**.
* Great for **large organizations** where users change roles or devices frequently.

**🔍 Quick Comparison Table**

| **Concept** | **What is it?** | **Example** |
| --- | --- | --- |
| **User** | Individual account | John Smith |
| **Group** | Collection of users/devices | HR Department Group |
| **Dynamic Group** | Group that updates membership automatically | All Windows 11 Devices |

**📝 Simple Analogy:**

* **User:** A **person**.
* **Group:** A **club** where people join.
* **Dynamic Group:** A club with **automatic joining rules** (e.g., “Everyone from HR joins automatically”).

**Process Flow for an Application on Windows client via IME service. (From Polling to detection, to installation , to detection and toast notifications as success/failure):**

**🚀 Process Flow for App Deployment via IME**

**📡 1️ Polling (Check for New Apps)**

* The **IME service** (running on the Windows client) regularly **contacts** the **Intune cloud**.
* It asks:

"Hey Intune, do I have any new apps, policies, or tasks to do?"

* This happens at regular intervals (default ~ every 60 minutes, or triggered manually).

**🔎 2️ Detection (Is the App Already Installed?)**

* IME **checks the device** to see if the **assigned app** is already present.
* It uses **detection rules** (like checking a registry key, a file, or MSI code).
* If the app is already installed, IME says:

"No need to install it again!"

* If it’s **missing** or doesn’t match the detection rules, IME says:

"I need to install this app."

**📦 3️ Download and Installation**

* IME downloads the **app package** from **Intune** (this is usually a **.intunewin** file stored in Azure).
* It starts the **installation process**:
  + Runs the **installer (like setup.exe, MSI)**.
  + **Monitors** the install to see if it succeeds or fails.

**🔎 4️ Post-Installation Detection**

* After the install finishes, IME **runs the detection rule again**.
* This is to confirm:

"Did the app really install properly?"

* If the app **passes the detection rule**, it’s considered **successfully installed**.
* If the detection **fails**, it’s considered a **failed install**.

**🔔 5️ Toast Notifications (Success or Failure)**

* If the install **succeeds**:
  + The user gets a **toast notification** saying something like:

"AppName was successfully installed."

* If the install **fails**:
  + The user gets a **failure notification** saying something like:

"AppName failed to install. Please contact support."

**🗂️ Putting It All Together: Flowchart**

1️ **Polling** – IME asks Intune: "Any new apps?"  
2️ **Detection (Pre-Install)** – Is the app already installed?  
3️ **Download & Install** – Get the package and install it.  
4️ **Detection (Post-Install)** – Check if it’s installed correctly.  
5️ **Notify User** – Show success or failure toast message.

**📝 Simple Analogy**

It’s like the **IME service** is a **robot assistant**:

1. 🕵️‍♂️ **Looks for new orders (polling)**.
2. 🔍 **Checks if you already have the item (detection)**.
3. 📦 **If not, it fetches the item and installs it (install)**.
4. 🔍 **Checks again to make sure the item is really there (detection)**.
5. 🎉 **Tells you if it worked or not (toast notification)**.

* **Registries for LOB and Win32Apps**

**📝 What is the Windows Registry?**

* The **Windows Registry** is like a **big database** where Windows stores **settings and configurations**.
* Apps can **read from** and **write to** the registry to:
  + Store **installation information**.
  + Keep **user preferences**.
  + Record **licensing info**.
  + Indicate that the app is **installed** (which is important for detection!).

**🏢 LOB Apps and Win32 Apps – How They Use the Registry**

**🔹 Win32 Apps (Traditional Desktop Apps)**

* When you install a **Win32 App**, it often:
  + Writes a **registry key** (e.g., under HKEY\_LOCAL\_MACHINE\Software or HKEY\_CURRENT\_USER).
  + Might also register **Uninstall info** under  
    HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows\CurrentVersion\Uninstall.
* Registry entries can include:
  + App version
  + Install path
  + Publisher
  + License status

These entries are crucial because tools like **Intune** use them to detect whether a Win32 app is installed.

**🔹 LOB Apps**

* LOB (Line-of-Business) Apps can also be **Win32 Apps**, so they **might use the same registry practices**.
* However, **LOB apps** could also be:
  + **Modern/UWP apps** (which store data differently, often in isolated storage).
  + **Web apps** (which don’t touch the registry at all).
* For **LOB Win32 Apps** specifically:
  + You can define **custom detection rules in Intune** to look for a **specific registry key or value** to confirm the app is installed.

**🧐 How Does Intune Use Registry for Win32/LOB Apps?**

When you deploy a **Win32 or LOB app** using Intune:

* You often configure a **detection rule** in Intune.
* One of the most common detection methods is:
  + **Check the registry** for a specific **key/value** that the app writes when it’s installed.

✅ **Example**:

* App installer writes this during installation:

HKEY\_LOCAL\_MACHINE\Software\MyCompany\MyApp

Value: Installed = 1

* Intune’s detection rule checks:

"Does this registry key exist and is the value set to 1?"

If yes → Intune says **"App is installed"**  
If no → Intune might **reinstall or flag failure**

**📑 Summary Table**

| **Feature** | **Win32 Apps** | **LOB Apps** |
| --- | --- | --- |
| **Registry Use?** | Yes, often | Yes (if Win32), or no (if web/UWP) |
| **Common Keys** | HKLM\Software, HKCU, Uninstall keys | Same as Win32 apps, or none for web/UWP |
| **Detection Rule?** | Registry key/value is commonly used | Same for Win32-based LOB apps |
| **Key Purpose** | Stores install info, settings, user preferences | Same, or none if non-Win32 |

**🔑 Key Takeaway**

* **Win32 apps** heavily use the **registry**.
* **LOB apps** may **use the registry** if they are Win32-based, or **not use it** if they’re modern/UWP or web apps.
* **Detection rules** in tools like **Intune** often rely on **registry keys/values** to verify if the app is installed.

**Specific Registries with Application GUID, which give you the status of Installation/Uninstallation.**

**📚 What’s a GUID?**

* **GUID** = **Globally Unique Identifier**.  
  It’s like a **unique ID** for an application (e.g., {1234-ABCD-5678-EFGH}).
* When an app installs, it **registers itself in the registry** using a **GUID**.
* This helps Windows (and Intune, SCCM, etc.) know which apps are installed and where.

**🔍 Key Registry Locations for Install/Uninstall Info**

**📁 1️ 32-bit Apps on 64-bit Windows:**

HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall

**📁 2️ 64-bit Apps:**

HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall

**📁 3️ Per-user installations (sometimes used):**

HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Uninstall

**🏷️ What’s Inside These Keys?**

Under each of these **Uninstall** registry keys, you’ll find **subkeys named by GUIDs** or app names.

✅ Inside each **subkey**, you’ll see **values** like:

* DisplayName – the app’s name.
* DisplayVersion – version number.
* InstallDate – when it was installed.
* UninstallString – the command to uninstall.
* QuietUninstallString – silent uninstall command.
* InstallLocation – where it’s installed.
* **Optional status indicators** (sometimes written by the app).

**🔑 How Do Detection Rules Work?**

When deploying an app (Win32 or LOB) in **Intune**, you can:  
Find the app’s **GUID** in the registry (under Uninstall).  
Use that key to **detect if the app is installed**.  
You can check for **specific values**, like DisplayVersion or even custom values (if the app writes its own “status” key).

**🎯 Example**

Let’s say your app’s GUID is {ABC12345-6789-DEF0-1234-56789ABCDEF0}.

✅ To check if it’s installed, look under:

HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\{ABC12345-6789-DEF0-1234-56789ABCDEF0}

If the key **exists**, and DisplayName matches your app, it’s installed.

✅ **Optional**: Some apps write a custom **“Status”** value:

* InstallStatus = Installed (for success)
* InstallStatus = Failed (for failure)

**🧠 Key Takeaway**

* **Registry Uninstall keys (with GUIDs)** tell Windows (and tools like Intune) if an app is installed or not.
* Intune can **detect app presence** using:
  + **Key existence**
  + **Specific values** (like DisplayVersion, InstallStatus)
* **Win32 and LOB apps** often use these keys with **GUIDs**.

**📝 Summary Table**

| **What** | **Registry Path** |
| --- | --- |
| **32-bit app info** | HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall\{GUID} |
| **64-bit app info** | HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\{GUID} |
| **Per-user install info** | HKCU\Software\Microsoft\Windows\CurrentVersion\Uninstall\{GUID} |
| **Key values** | DisplayName, DisplayVersion, UninstallString, etc. |

**Log File locations**

**📚 Log File Locations in Intune/Windows (IME, Win32, LOB Apps)**

**🗂️ 1️ Intune Management Extension (IME) Logs**

When you deploy **Win32 apps** or **LOB apps via Intune**, the **IME agent** handles the installation on Windows. Its logs are super important for troubleshooting.

**📁 Location (IME logs):**

C:\ProgramData\Microsoft\IntuneManagementExtension\Logs

🔹 **Key Log Files:**

* IntuneManagementExtension.log –  
  This is the **main log** showing all IME activity: polling, detection, app install/uninstall, errors, and more.
* AgentExecutor.log –  
  Shows **detailed logs** of app installation processes (Win32 apps).

**🗂️ 2️ Company Portal / MDM Enrollment Logs**

If the device is enrolled through Intune and using the **Company Portal app**, you can check:

**📁 Location:**

C:\Users\<Username>\AppData\Local\Packages\Microsoft.CompanyPortal\_8wekyb3d8bbwe\LocalState\DiagOutputDir

🔹 Logs here capture **Company Portal activities**.

**🗂️ 3️ Windows Event Viewer Logs**

Windows logs app installation and MDM-related events into **Event Viewer**.

**📁 Location in Event Viewer:**

Applications and Services Logs -> Microsoft -> Windows -> DeviceManagement-Enterprise-Diagnostics-Provider

Look for logs like:

* **App Install Status**
* **Device Management (MDM) Events**

**🗂️ 4️ Win32 App Installer Logs (Custom App Logs)**

Some **Win32 apps** write their own logs (if the installer supports it).

**Common Locations:**

* **Temp Folder** (during install):

C:\Windows\Temp

C:\Users\<Username>\AppData\Local\Temp

* **Custom Logs** (depends on the app):  
  Check the installer’s documentation for a **log file path** or **switch** (e.g., /log "C:\InstallLogs\myapp.log").

**🗂️ 5️ Setup Logs (MSI Installer)**

If your app is installed via an **MSI**, you can enable **MSI logging**.

**📁 Location:**

C:\Windows\Temp\MSI\*.log

Or enable via registry:

HKLM\Software\Policies\Microsoft\Windows\Installer

Logging = voicewarmupx

**📝 Summary Table**

| **Log Type** | **Location** |
| --- | --- |
| **IME Logs (Intune)** | C:\ProgramData\Microsoft\IntuneManagementExtension\Logs |
| **Company Portal Logs** | C:\Users\<Username>\AppData\Local\Packages\Microsoft.CompanyPortal\_8wekyb3d8bbwe\LocalState\DiagOutputDir |
| **Event Viewer Logs** | Applications and Services Logs -> Microsoft -> Windows -> DeviceManagement-Enterprise-Diagnostics-Provider |
| **Temp Logs (Installer)** | C:\Windows\Temp or C:\Users\<Username>\AppData\Local\Temp |
| **MSI Logs** | C:\Windows\Temp\MSI\*.log |

**🎯 Key Takeaways**

✅ **IME logs** are the main place to check for Intune/Win32 app installs.  
✅ **Event Viewer** gives you extra details about MDM and app deployment events.  
✅ **Custom installers** may have their own logs.  
✅ **MSI installs** can write logs if enabled.

**Company Portal**

**📚 What is Company Portal?**

✅ **Company Portal** is an **app** used by organizations to:

* **Enroll devices** into management (e.g., Intune)
* **Access work resources** (apps, documents, policies)
* **Install company-approved apps** (like LOB apps or Win32 apps)
* **View compliance status** (to ensure devices meet security requirements)
* **Request support** (helpdesk contact, device info)

✅ It’s typically used with **Microsoft Intune** (or **Microsoft Endpoint Manager**) to help manage and secure **corporate and BYOD (Bring Your Own Device)** devices.

**🖥️ What does Company Portal do?**

* When a user signs in to **Company Portal**:
  1. The device **enrolls into Intune MDM (Mobile Device Management)**.
  2. **Compliance policies** are applied (e.g., require password, encryption).
  3. The user can see a **list of available apps** (assigned via Intune).
  4. Users can **install apps** (e.g., Win32 apps, LOB apps, Microsoft 365).
  5. It shows if the device is **compliant** or **non-compliant**.
  6. Users can **remove work profiles** or **reset** the device.

**📱 Where is Company Portal available?**

* **Windows 10/11**
* **iOS/iPadOS**
* **Android**
* **macOS**

**🔍 Example Scenario:**

1️ Your company wants employees to use a secure laptop for work.  
2️ The IT team sets up **Intune** and publishes the **Company Portal app**.  
3️ When you open Company Portal on your laptop:

* You sign in with your **work account**.
* The device enrolls in **Intune**.
* Compliance policies are checked.
* You see a list of **available apps** (like MS Teams, Outlook, custom LOB apps).
* You click **install** on an app, and it installs via Intune/IME.

**📝 Quick Summary Table**

| **Feature** | **Description** |
| --- | --- |
| **What is it?** | An app for users to enroll devices and access company resources |
| **Used for?** | Device enrollment, app installation, compliance status |
| **Platform?** | Windows, iOS, Android, macOS |
| **Works with?** | Microsoft Intune/Endpoint Manager |

**How to Sync once app assignments are done. (Intune Device Sync/ Company Portal Local side Sync)**

**🔄 How to Sync App Assignments After Deploying in Intune**

**1️ Intune Device Sync (From the Cloud Side)**

* Intune automatically **syncs devices** at regular intervals (default every 8 hours), but you can also **force a sync manually**.
* When a device syncs, it **contacts Intune** to check for new assignments, apps, policies, etc.
* This is usually done by the **device itself**.

**How to manually trigger sync:**

* **From Intune Admin Portal:**
  + Go to **Devices > All devices**
  + Select the device you want
  + Click **Sync** button
  + This triggers the device to check Intune immediately.
* **From the Device (Windows):**
  + Open **Settings > Accounts > Access work or school**
  + Select the connected work account
  + Click **Info**
  + Click **Sync** button
  + This tells the device to check in with Intune now.

**2️ Company Portal Local Side Sync (User-Initiated on Device)**

* The **Company Portal app** on Windows or mobile devices allows the user to **manually sync** to get the latest apps and policies.
* Useful if you want to speed up app installs or policy updates without waiting for the automatic sync.

**How to sync from Company Portal:**

* Open **Company Portal app**
* Go to **Settings** or **Menu**
* Click **Sync** or **Check for updates**
* This triggers the device to sync with Intune immediately.

**📝 Summary Table**

| **Sync Type** | **Where to Trigger** | **What Happens** |
| --- | --- | --- |
| **Intune Device Sync** | Intune Portal (admin) or Device Settings | Device contacts Intune to get latest assignments & policies |
| **Company Portal Sync** | Company Portal app (user) | User forces device to sync and get latest apps/policies |

**🎯 Key Tips:**

* Sometimes you may want to **tell users** to open Company Portal and hit **Sync** to speed things up.
* IT admins can also **trigger sync** from Intune portal for targeted devices.
* After sync, the **Intune Management Extension (IME)** picks up Win32 app installs or updates.

**Breakdown of events in log files.**

**📂 Breakdown of Events in Intune Management Extension Logs**

**🔍 Where to Find the Logs?**

C:\ProgramData\Microsoft\IntuneManagementExtension\Logs\IntuneManagementExtension.log

**🔑 Key Events You’ll See in the Log**

| **Event Type** | **Description** | **Example Log Entry/Message** |
| --- | --- | --- |
| **Polling** | IME contacts Intune to check for new tasks | "Checking for new assignments from Intune..." |
| **Download Start** | IME starts downloading app package | "Downloading content from <URL>..." |
| **Download Complete** | App package download finished | "Download succeeded" |
| **Pre-Install Detection** | IME checks if the app is already installed | "Checking detection rules for <AppName>..." |
| **Install Start** | IME starts app installation | "Starting install of <AppName>..." |
| **Install Progress** | Updates on installation progress | "Running installer with command line..." |
| **Install Success** | Installer finished successfully | "Installation succeeded for <AppName>" |
| **Post-Install Detection** | IME rechecks app presence after install | "Verifying detection rule post-install for <AppName>" |
| **Install Failure** | Installer failed or detection failed | "Installation failed with error code <ErrorCode>" |
| **Retry Attempts** | IME retries install if failure detected | "Retrying installation attempt 2 of 3..." |
| **User Notifications** | Toast notifications shown to user | "User notified: <AppName> installed successfully" |
| **Cleanup** | IME removes temporary files or old versions | "Cleaning up installation files..." |
| **Heartbeat/Status Update** | IME reports status back to Intune | "Reporting installation status to Intune..." |

**📋 What Each Event Means**

* **Polling**: IME asking Intune, “Do you have any apps or updates for me?”
* **Download**: Fetching the app installer package.
* **Detection**: Checking if the app is already installed before and after installation.
* **Install Start/Progress**: Running the installer, e.g., MSI or EXE.
* **Success/Failure**: Whether the install succeeded or failed.
* **Retry**: IME tries again if something went wrong.
* **User Notification**: Windows shows a toast message saying install success/failure.
* **Cleanup**: Removing temporary files to save space.
* **Heartbeat**: Sending status updates back to Intune.

**⚙️ Example Log Snippet**

csharp

CopyEdit

[2025-05-27 09:00:00] INFO - Checking for new assignments from Intune...

[2025-05-27 09:01:00] INFO - Downloading content from https://intunestorage.blob.core.windows.net/...

[2025-05-27 09:02:30] INFO - Download succeeded

[2025-05-27 09:03:00] INFO - Checking detection rules for MyApp...

[2025-05-27 09:03:10] INFO - App not found, starting install

[2025-05-27 09:05:00] INFO - Running installer with command line "setup.exe /quiet"

[2025-05-27 09:06:00] INFO - Installation succeeded for MyApp

[2025-05-27 09:06:30] INFO - Verifying detection rule post-install for MyApp

[2025-05-27 09:07:00] INFO - Reporting installation status to Intune

[2025-05-27 09:07:10] INFO - User notified: MyApp installed successfully

[2025-05-27 09:07:30] INFO - Cleaning up installation files

**🎯 Why This Helps**

* You can **track each step** during app deployment.
* **Identify where issues happen** (e.g., download failed, install failed).
* See if **detection rules work properly**.
* Verify if the **user got notified**.
* Understand retries or cleanup operations.

**Introduction to Cmdlets**

**🛠️ What are Cmdlets?**

* **Cmdlets** (pronounced “command-lets”) are **small, lightweight commands** built into **PowerShell** (Microsoft’s powerful command-line shell).
* They perform **specific tasks** like managing files, services, users, or devices.
* Cmdlets are designed to be simple but powerful, with consistent naming like Verb-Noun (e.g., Get-Process, Install-Module).

**⚡ Why Use Cmdlets?**

* Automate repetitive tasks (like installing apps, checking device status).
* Manage Windows systems and Intune without clicking through GUIs.
* Combine commands in scripts for complex workflows.
* Quickly get information or perform actions remotely.

**📚 Basic Examples of Cmdlets**

| **Cmdlet** | **What it Does** |
| --- | --- |
| Get-Process | Lists all running processes |
| Get-Service | Lists all Windows services |
| Install-Module | Installs a PowerShell module |
| Get-IntuneDevice | (Example) Gets info about Intune devices (if module installed) |

**🖥️ Cmdlets in Intune Management**

Microsoft provides **PowerShell modules** (like Microsoft.Graph.Intune) with cmdlets to:

* **Get device/app status**
* **Deploy apps**
* **Manage policies**
* **Query users and groups**

Example:

Get-IntuneManagedDevice

Gets a list of devices managed by Intune.

**🔑 Key Points**

* Cmdlets are **case-insensitive** but usually written in PascalCase.
* They **accept parameters** to specify what you want, e.g., Get-Process -Name notepad.
* They output objects you can **filter, sort, or pipe** to other cmdlets.

**🎯 Summary**

| **Concept** | **Description** |
| --- | --- |
| Cmdlets | Small PowerShell commands |
| Purpose | Automate system and Intune tasks |
| Format | Verb-Noun (e.g., Get-Process) |
| Use Case | Managing apps, devices, users |

Bottom of Form