📦 PackingIntunewin - Portable IntuneWin App Packaging Tool

A user-friendly, **fully portable** PowerShell tool for automatically creating Microsoft Intune `.intunewin` packages from EXE files.

🌍 Language / Sprache

- US **English** (this file)
- DE **Deutsch** → README.de.md

🎯 Goal

Make packaging `.exe` files into `.intunewin` apps as simple and automated as possible - **without manual configuration or fixed paths**.

✨ Features

- 🚀 **Fully portable** - works from any location
- 🔄 **Automatic download** of Microsoft IntuneWinAppUtil.exe
- 🗑️ **Intelligent uninstallation** - Registry analysis & EXE parameter testing
- 📝 **Automatic batch creation** - install.cmd & uninstall.cmd
- 📊 **App overview** - shows available apps with status
- 🔍 **Error validation** - checks EXE count and folder structure
- 💾 **Complete metadata** - optimized for Microsoft Intune
- 🎨 **Colored output** - clear progress indication

📁 Folder Structure

```
PackingIntunewin/
│
├── Create-IntuneWinApp.ps1             # 🚀 Main script (portable)
├── README.md                           # 📖 This documentation (English)
├── README.de.md                        # 📖 German documentation
│
├── apps/                               # 📥 Input folder for your apps
│   ├── Chrome/                         # Example: Chrome app
│   │   └── chrome-installer.exe        # Your EXE file
│   ├── VLC/                           # Example: VLC app
│   │   └── vlc-installer.exe          # Your EXE file
│   └── YourApp/                       # 👈 Your app folder
│       └── yourapp.exe                # 👈 Your EXE file (exactly one!)
│
└── IntunewinApps/                      # 📤 Output folder (created
automatically)
    ├── tools/                          # 🔧 Microsoft tools (downloaded
```

```
automatically)
    │   └── IntuneWinAppUtil.exe        # Microsoft Intune Win32 Content Prep
Tool
    │
    ├── Chrome/                          # 📦 Packaged Chrome app
    │   ├── install.cmd                  # Automatically generated
    │   ├── uninstall.cmd               # Automatically generated
    │   ├── chrome-installer.intunewin   # 🎯 Ready package for Intune
    │   └── metadata.json                # Complete metadata
    │
    └── YourApp/                         # 📦 Your packaged app
        ├── install.cmd                  # Automatically generated
        ├── uninstall.cmd               # Automatically generated
        ├── yourapp.intunewin            # 🎯 Ready package for Intune
        └── metadata.json                # Complete metadata
```

🚀 Quick Start

1. Clone or download repository

```
git clone https://github.com/yourusername/PackingIntunewin.git
cd PackingIntunewin
```

2. Prepare your app

```
# Create a folder for your app in the 'apps' directory
mkdir "apps\MyApp"

# Copy your EXE file into it (exactly one EXE per folder!)
copy "C:\Downloads\my-app-installer.exe" "apps\MyApp\"
```

3. Run the script

```
# Start script from the main folder
.\Create-IntuneWinApp.ps1
```

4. Select and configure app

The script guides you through the process:

- 📋 Shows available apps
- ❓ Asks for app name and reboot requirements
- 🔍 Automatically determines uninstallation information
- 📦 Creates the ready `.intunewin` package

🔧 Automatic Features

Tool Download Process

```
🔧 Checking IntuneWinAppUtil.exe...
⚠️ IntuneWinAppUtil.exe not found, downloading from GitHub...
🌐 Searching for latest version...
📥 Downloading: IntuneWinAppUtil.exe (Version: v1.8.4)
☑️ Download successful! File size: 0.89 MB
```

App Status Overview

```
📋 Available apps in 'apps' folder:
   ☑️ Chrome
   ☑️ VLC
   ❌ Broken-App (No EXE)
   ⚠️ Multi-EXE-App (Multiple EXE)
```

Smart Uninstallation Detection

```
🔧 Determining uninstallation information...
🔍 Searching for uninstallation information for 'Chrome'...
☑️ Registry entry found!
☑️ Found: Google Chrome
📝 Using QuietUninstallString: MsiExec.exe /X{GUID} /quiet
```

📊 Generated Files

Installation Script (install.cmd)

```
@echo off
echo Installing YourApp...
yourapp.exe /silent
if %errorlevel% neq 0 (
    echo Installation failed with exit code %errorlevel%
    exit /b %errorlevel%
)
echo Installation completed successfully
exit /b 0
```

Uninstallation Script (uninstall.cmd)

```
@echo off
echo Uninstalling YourApp...
MsiExec.exe /X{12345678-1234-1234-1234-123456789012} /quiet
if %errorlevel% neq 0 (
    echo Uninstallation failed with exit code %errorlevel%
    exit /b %errorlevel%
)
echo Uninstallation completed successfully
exit /b 0
```

Metadata File (metadata.json)

```
{
  "AppName": "YourApp",
  "Installer": "yourapp.exe",
  "InstallCommand": "install.cmd",
  "UninstallCommand": "uninstall.cmd",
  "UninstallMethod": "Registry",
  "UninstallString": "MsiExec.exe /X{GUID} /quiet",
  "ExitCode": 0,
  "RebootRequired": false,
  "DetectionType": "Registry (configure manually)",
  "CreatedOn": "2025-07-03 14:30",
  "CreatedBy": "username",
  "ScriptVersion": "2.0",
  "WorkingDirectory": "C:\\Path\\To\\PackingIntunewin"
}
```

🎯 Microsoft Intune Configuration

After creation, you receive all required information:

```
📋 Information for Microsoft Intune:
═══════════════════════════════════════
App Name: YourApp
Install Command: install.cmd
Uninstall Command: uninstall.cmd
Return Codes: 0 (Success), 3010 (Reboot required)
═══════════════════════════════════════
```

🛠️ Advanced Features

Multi-Tier Uninstallation Detection

The script uses **intelligent detection mechanisms**:

1. **Registry Analysis** (preferred)
    - Searches `HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\*`
    - Searches `HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall\*`
    - Searches `HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\*`
2. **EXE Parameter Testing** (fallback)
    - Tests common parameters: `/uninstall`, `/remove`, `/u`, `/x`
    - Combined with silent flags: `/silent`, `/quiet`, `/s`, `/q`
3. **Standard Fallback**
    - Uses `/uninstall /silent` as last resort

Comprehensive Error Handling

- ❌ **No EXE**: Clear error message when EXE file is missing
- ❌ **Multiple EXEs**: Warning when more than one EXE file exists
- ❌ **Folder not found**: Helpful path display
- 🌐 **Network errors**: Graceful fallback with manual instructions

💡 Tips & Best Practices

App Preparation

- ☑️ **One EXE per folder**: Exactly one EXE file per app folder
- ☑️ **Descriptive names**: Use clear folder names (e.g., "Chrome", "VLC")
- ☑️ **Test silent parameters**: Check beforehand if your EXE supports silent installation

Portability

- 📁 **Copy entire folder**: Copy the complete `PackingIntunewin` folder
- 🔄 **No path adjustments**: The script works from any location
- 💾 **USB stick compatible**: Perfect for mobile use

Intune Integration

- 📝 **Detection Rules**: Manually configure detection rules in Intune
- 🔍 **Registry detection**: Use metadata for registry-based detection
- 🔄 **Exit codes**: Consider the documented return codes

🆘 Common Issues & Solutions

"IntuneWinAppUtil.exe could not be downloaded"

- 🌐 Check internet connection
- 🔒 Check firewall/proxy settings
- 📥 Manual download from Microsoft GitHub

"No EXE files found"

- 📁 Check the app folder under `apps\YourApp\`
- ☑️ Ensure exactly one `.exe` file is present
- 📋 Use the app overview for diagnosis

"Uninstallation not found"

- 🔍 The script automatically tests various parameters
- ✏️ You can manually adjust the `uninstall.cmd`
- 📖 Consult your software's documentation

🤝 Contributing

Improvement suggestions and pull requests are welcome!

1. Fork the repository
2. Create feature branch
3. Commit changes
4. Create pull request

📄 License

This project is licensed under the MIT License.

🔗 Useful Links

- Microsoft Intune Win32 Content Prep Tool
- Microsoft Intune Documentation
- PowerShell Documentation

💡 **The tool is fully portable and can be used anywhere!** 🚀

🤖 AI Assistance

This project was developed with assistance from AI (GitHub Copilot) to enhance code quality, documentation, and user experience.