Uninstalling programs via PowerShell is a powerful and efficient way to manage software on Windows, particularly for system administrators or users looking to automate cleanup tasks or remove stubborn applications that won't disappear through the standard Control Panel interface. There are several distinct methods to accomplish this, depending entirely on whether the software is a modern UWP (Universal Windows Platform) app, a traditional Win32 desktop application, or a software package installed via a specific manager like Winget.

To ensure a smooth experience and prevent permission errors, **always run PowerShell as an administrator.**

Here are the primary methods for uninstalling programs via PowerShell:

1. Removing UWP/Built-in Apps (`Remove-AppxPackage`)

This method is ideal for removing pre-installed Windows apps (like Xbox, Calculator, or News) or apps installed from the Microsoft Store for the current user.

- **List installed apps:** First, find the exact name of the application.

```powershell
Get-AppxPackage | Select Name, PackageFullName
```

- **Uninstall a specific app for current user:**

```powershell
Remove-AppxPackage -Package "PackageFullName"
```

Example: `Remove-AppxPackage Microsoft.Xbox.TCUI_1.24.10001.0_x64__8wekyb3d8bbwe`.

- **Uninstall for all users:**

```powershell
Get-AppxPackage -AllUsers | Where-Object {$_.Name -eq "AppName"} | Remove-AppxPackage -AllUsers
```

- **Remove provisioned apps (prevent re-installation for new users):**

```powershell
Get-AppxProvisionedPackage -Online | Where-Object {$_.DisplayName -eq "AppName"} | Remove-AppxProvisionedPackage -Online
```

2. Removing Desktop Apps using `Get-Package`

For traditional Win32 desktop applications (like Chrome, Firefox, or Zoom), the `PackageManagement` module is a reliable, modern approach.

- **List installed packages:**

```powershell
Get-Package
```

- **Uninstall a package:**

```powershell
Uninstall-Package -Name "Name of Application"
```

Example: `Uninstall-Package -Name "Google Chrome"`.

3. Using `Winget` (Windows Package Manager)

On Windows 10 and 11, the `winget` tool is available directly in PowerShell to manage applications.

- **List installed applications:**

```powershell
winget list
```

- **Uninstall an application:**

```powershell
winget uninstall "App Name"
```

Example: `winget uninstall "Spotify"`.

4. Registry Method (For stubborn EXE files)

If an application does not appear in `Get-Package` or `winget`, you can find its uninstall string in the Windows Registry and execute it silently.

```powershell
$appName = "Part of App Name"
$app = Get-ChildItem -Path HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall, HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\Current
       Get-ItemProperty |
       Where-Object {$_.DisplayName -match $appName } |
       Select-Object -Property DisplayName, UninstallString

if ($app.UninstallString) {
    # If the string contains an exe, it might need arguments
    $uninstallPath = $app.UninstallString
    Start-Process -FilePath "cmd.exe" -ArgumentList "/c $uninstallPath /quiet" -Wait
}
```

⚠️ Critical Tips and Best Practices

- **Avoid `Win32_Product`:** While `Get-WmiObject -Class Win32_Product` is often recommended online, it is known to be very slow and can trigger a repair/consistency check on all installed MSI packages, which can cause issues.
- **Use `-WhatIf`:** Before running a removal command, add `-WhatIf` to the end of the command to see what *would* happen without actually making changes.
- **Administrative Privilege:** If a command fails, ensure you are running PowerShell as an administrator.

For a complete removal of an application including files and folders, you may need to use `Remove-Item` after the uninstallation, but be careful with this command.