**Primitive Driver Documentation:**

**Motivation and Description:**

Prior to Windows 10 Version 1903, certain types of software that used INF-based installation but were not necessarily tied to a particular hardware device were not fully supported by the OS. While these pieces of software leveraged INF files as a manifest for installation, the OS was not directly aware of this scenario and did not have support to handle it natively. Because these pieces of software were not tied to a hardware device, they would install on the whole system regardless of hardware. As a result, there was no guarantee that these pieces of software were properly installed, uninstalled, or handled on OS upgrade. To improve reliability and guarantee proper behavior of these types of software, especially during OS upgrade and reset scenarios, Windows 10 Version 1903’s Plug and Play platform can now handle and manage this type of software package as a top-level entity. The types of software that leverage this new platform support are called *Primitive Drivers.* Primitive drivers continue to use INF-based installation and the underlying platform will leverage the [Driver Store](https://docs.microsoft.com/en-us/windows-hardware/drivers/install/driver-store) to keep track of all relevant files. This allows the underlying Plug and Play platform to gracefully install, uninstall, and maintain driver state on OS upgrade.

Conceptually, there is a change to how these INFs are managed. Previously, [DefaultInstall] (and often, [DefaultUninstall]) were processed by [SetupAPI](https://docs.microsoft.com/en-us/windows-hardware/drivers/install/setupapi) in a script-like fashion, where the INF was used as a manifest and [SetupAPI](https://docs.microsoft.com/en-us/windows-hardware/drivers/install/setupapi) executed the instructions in the relevant sections on the caller’s behalf. Undoing the changes (to perform an uninstallation) requires specifying an INF section that performs the opposite set of instructions as the installation section. An INF leveraging Primitive Drivers, however, does not require an uninstallation section. Primitive drivers leverage the same installation and uninstallation APIs as device drivers, where the uninstallation API will perform the inverse set of operations as the install operation, and the act of installing or uninstalling the driver package will process those sections.

**INF Requirements to leverage Primitive Driver functionality:**

1. Version section must be complete, like PnP drivers
   1. “Provider” directive must be filled in
   2. “Class” directive must be filled in
   3. “ClassGuid” directive must be filled in
2. Driver must comply with [Universal](https://docs.microsoft.com/en-us/windows-hardware/drivers/develop/getting-started-with-universal-drivers) standards
3. No [Manufacturer] section may be present
4. [DefaultInstall] sections must be architecture decorated, and no undecorated versions may be present
   1. ie. [DefaultInstall.amd64]
5. [DefaultUninstall] may not be present in the INF (see legacy compatibility section for an exception)

**Primitive drivers targeting only 1903+:**

Primitive drivers targeted only for Windows 10 Version 1903 and forward should utilize [DiInstallDriver](https://docs.microsoft.com/en-us/windows/desktop/api/newdev/nf-newdev-diinstalldriverw) and [DiUninstallDriver](https://docs.microsoft.com/en-us/windows/desktop/api/newdev/nf-newdev-diuninstalldriverw) to properly install and uninstall their software in/from the driver store. Drivers should also use Dirid 13 to properly specify the Driver Store as the desired destination to be installed. For more information on how to use Dirids, see “[Using Dirids](https://docs.microsoft.com/en-us/windows-hardware/drivers/install/using-dirids)”.

**Legacy Compatibility:**

While [DefaultUninstall] is prohibited in Primitive Drivers, an exception is made for the sake of down-level OS compatibility. An INF directive has been introduced that will cause an OS version supporting Primitive Drivers to ignore the [DefaultUninstall] section. If your driver package needs to support down-level OS versions, include the following syntax to ensure that the platform will appropriately handle such cases:

[DefaultUninstall.NTamd64]

**LegacyUninstall=1**

The [DefaultInstall] and [DefaultUninstall] sections **must still be architecture decorated**; however, by including the LegacyUninstall=1 line, the [DefaultUninstall] section is ignored on Windows 10 Version 1903 and later which allows you to include that section in your INF to be used down-level with a legacy install/uninstall application in order to uninstall the Primitive Driver package.

Beginning on Windows 10 Version 1903, changes have been made to the [InstallHInfSection](https://docs.microsoft.com/en-us/windows/desktop/api/setupapi/nf-setupapi-installhinfsectionw) API in setupapi.dll. If an architecture-decorated [DefaultInstall] or [DefaultUninstall] section is passed into this API for processing, the driver package will be checked to determine if it supports Primitive Driver functionality. If it does, rather than process the specified section in the legacy way, the INF is passed to [DiInstallDriver](https://docs.microsoft.com/en-us/windows/desktop/api/newdev/nf-newdev-diinstalldrivera) or [DiUninstallDriver](https://docs.microsoft.com/en-us/windows/desktop/api/newdev/nf-newdev-diuninstalldriverw), as appropriate. This allows a single installer to leverage Primitive Drivers on compatible OS versions and maintain support for previous OS versions.