Windows 10 IoT Enterprise (17763): Block Device Installation

By Sean D. Liming and John R. Malin
Annabooks – www.annabooks.com

September 2019

Windows 10 Version 17763 Build 1809

System Lockdown Overview

The major effort that goes into creating a custom Windows 10 Enterprise image is focused on locking down the system. As we called out in the book *Starter Guide for Windows 10 IoT Enterprise*, the more you lock the system down, the more challenging it is to support the system in the field. Careful system architecture and planning is important to provide the best support throughout the product lifecycle. There are different areas we look at from look and feel to custom security:

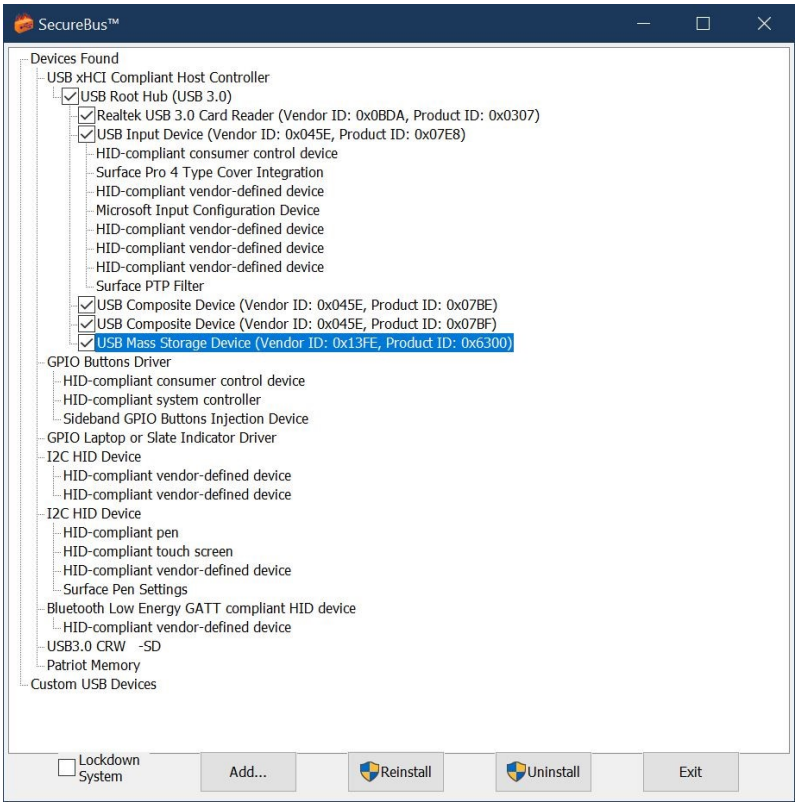
- Splash screen in the BIOS, shell launcher, hiding login screen, etc. go into the look and feel of the system to hide that Windows is running underneath.
- Custom Group Policy, Configurable Code Integrity, Windows Defender, BitLocker, custom security policies, SecureBoot, etc. provide the device security.
- Finally, the Unified Write Filter (UWF) provides some security and system robustness by protecting drives, and Keyboard Filter blocks out Windows hotkeys.

Some of these lockdown features have been in place going all the wall back Windows NT Embedded. Over time more security and lockdown features have been added as Windows matured. The one area that hasn't been addressed is block devices from ports like USB, firewire, and mSATA. These external ports allow anyone to plug something into the system and possibly break into a device. There have been many cases of USB devices with viruses in them like picture frames, programmable keyboard dongles, and the well-known USB flash drives. Another issue is that USB ports allow for hybrid devices. For example, some older USB flash drives acted as a CD-ROM and a flash drive, where the CD-ROM held software to help speed improvement. When your phone is connected via USB, it registers as several different devices in Windows. Just turning off auto-play and auto-run in group policy is not enough. The ability to block devices attaching to a Windows IoT Enterprise system now becomes a critical step in the image architecture. Over the years, we have had a few clients take a hard look at blocking devices. In this paper, we will look at a couple of solutions to block devices.

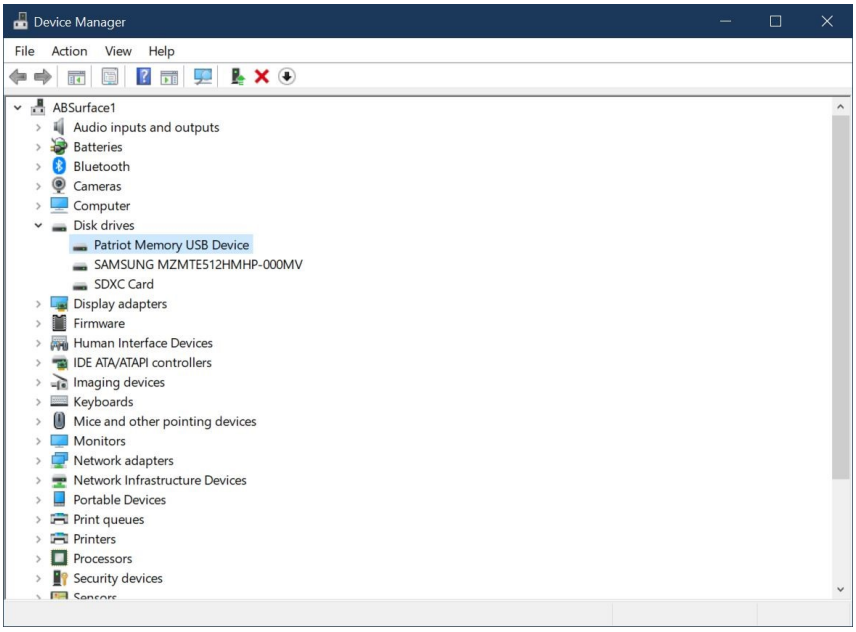
SecureBus™

Many of our medical and government customers were trying to figure out a way to control what users can connect to a device. There was a client-server solution like "Symantec Endpoint Protection", but this was not useful for standalone or non-domain connected devices. We reached out to one of our Windows device driver partners, and it just so happens they were seeing the same client request. We went to work on a solution, and in 2012, SecureBus™ was introduced as a USB device filter solution. We have covered SecureBus in our books and a video online demonstrates its usage. The following discusses the basic usage.

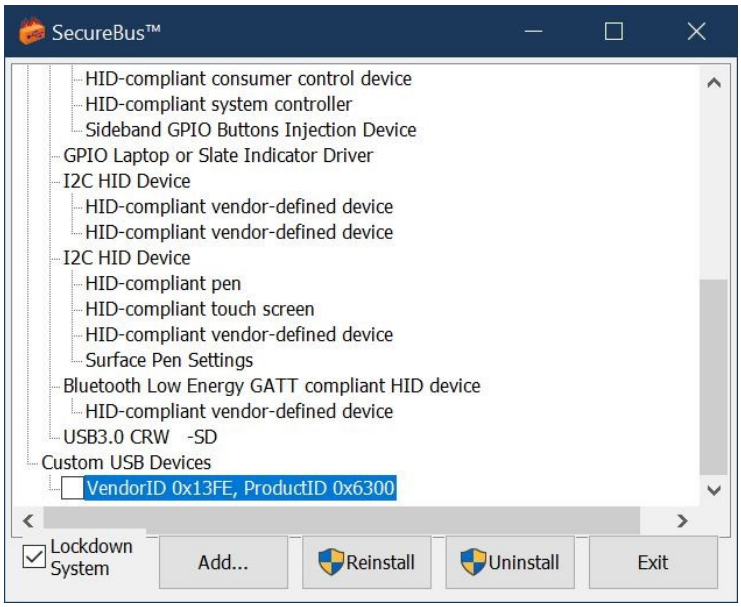
SecureBus comes as a single executable. When the executable is run for the first time, it installs the USB filter driver into the USB stack, and on the next run, the executable opens the utility that lists the currently connected USB devices in the system and the filter controls.



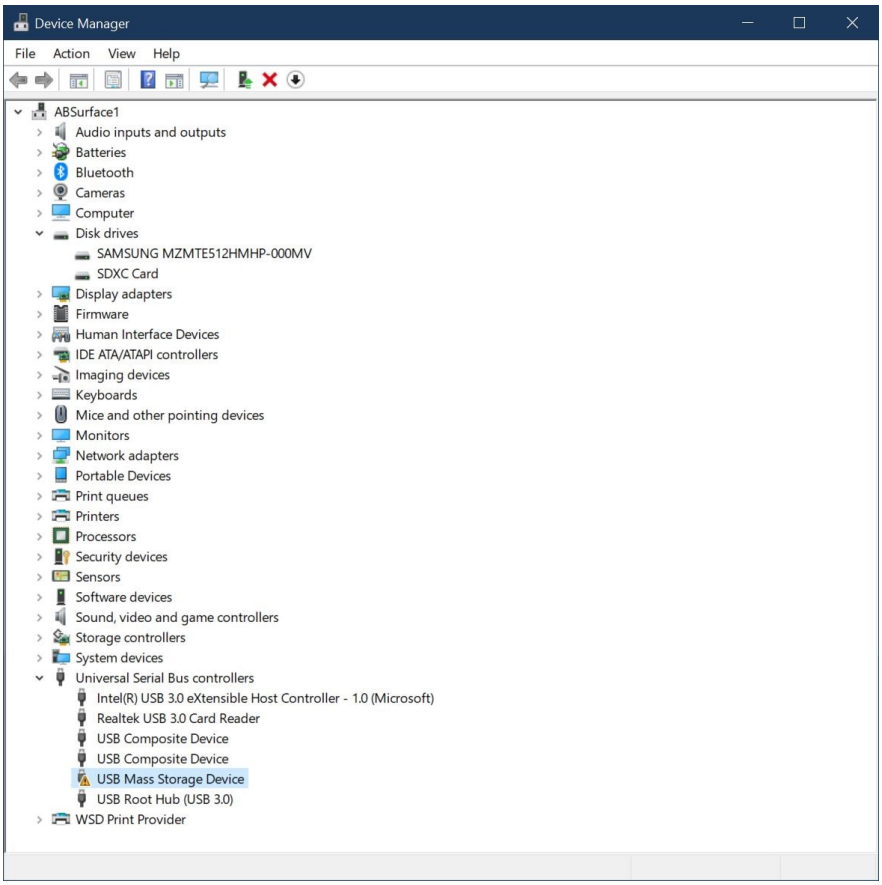
The picture above highlights a USB flash disk connected to the system. The flash disk can be seen in device manager.



SecureBus controls devices by PnP ID. Simply uncheck the device, check the Lockdown System, and reboot. When the system boots back up, the device is blocked from connecting to the system, and Device Manager shows a bang (!) next to the driver.



The SecureBus driver sits in the root hub, which all other hubs go through. Anything that is not on the list or has not been installed before will be blocked. If a device has already been connected, it can be connected again in the future, since it already exists in the registry; but one can simply uncheck the unwanted devices from the list.



PnP IDs can be manually added to the list without having to connect the device to the system. SecureBus requires a license key and has a per-unit royalty.

Group Policy: Device Installation Restrictions

In 2013, Windows 8.1 Embedded Industry was released. The new OS introduced a new lockdown feature called the USB Filter. One problem with this new feature was the lack of documentation on how to use it. When Windows 10 Enterprise LTSB 2015 was released, the USB filter feature was gone. In its place was a set of Group Policy Device Installation Restrictions that controlled device installation. Until the release of Windows 10 Enterprise LTSC 2019, enabling these policies and the Unified Write Filter (UWF) at the same time could result in a blue screen of death (BSOD). It appears the UWF filter driver was not getting excluded from the list of devices. With the release of Windows 10 Enterprise LTSC 2019, the BSOD issue appears to have been fixed.

Unlike SecureBus, the Device Installation Restrictions are for all devices, not just USB. If the system has a PCIe slot, the Device Installation Restrictions can prevent the driver from loading for any random PCIe devices plugged into the slot. There are 10 policies in this group, and using gpedit.msc, they can be found in the following path:

Computer Configuration : Administrative Templates : System : Device Installation : Device Installation Restrictions

The root registry for these policies is:

HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions

We have grouped the policies to help provide a discussion on different setup scenarios. The following lists the policy, the corresponding registry key, and the explanation:

Top Level Policies

These policies are the basic on and off for the Device Installation Restrictions.

- Allow administrators to override Device Installation Restriction policies

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions
Value Name: AllowAdminInstall
Type: REG_DWORD
Value Data: 0 - Disable, 1 – Enable

By enabling this setting all Administrator accounts can install devices. Disabling the setting will make Administrator accounts adhere to the other policy settings.

- Prevent installation of removable devices

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions
Value Name: DenyRemovableDevices
Type: REG_DWORD
Value Data: 0 - Disable, 1 – Enable

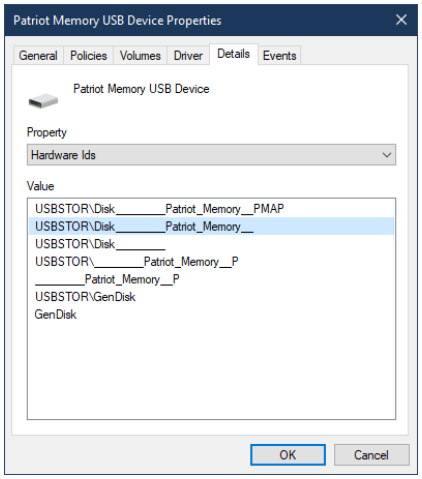
When enabled, all removable devices are blocked. Disabling the setting removes the restriction.

- Prevent installation of devices not described by other policy settings

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions
Value Name: DenyUnspecified
Type: REG_DWORD
Value Data: 0 - Disable, 1 – Enable

You can think of this setting as the master on/off for the Device Installation Restriction. Enabled this setting will block any device from being installed that is not allowed by other settings. It will also prevent device drivers from being updated. There are two settings that allow devices to be installed either by Device ID or by Device Class. The next two groups discuss these settings.

Control by Device ID
There are two policies that either allow or prevent a device from being installed by Device ID. The Device ID isn't the PnP ID but a Hardware ID that you can find listed in the device properties.

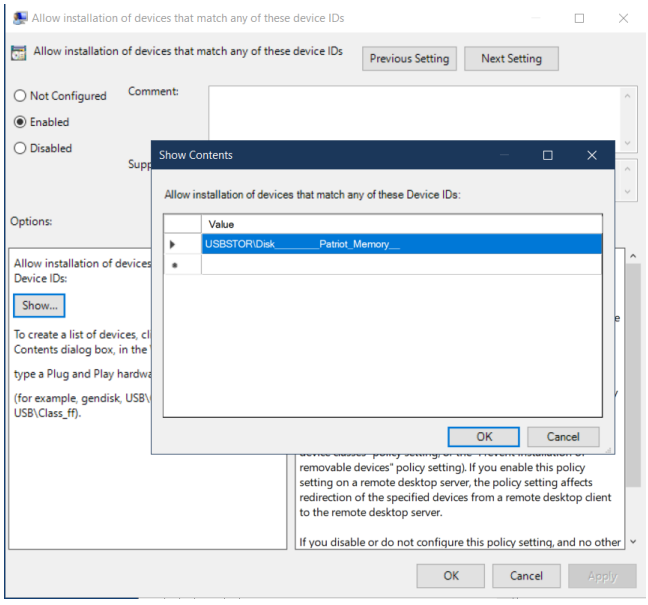


- Allow installation of devices that match any of these device IDs

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions
Value Name: AllowDeviceIDs
Type: REG_DWORD
Value Data: 0 - Disable, 1 – Enable

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions\AllowDeviceIDs
Value Name: 1
Type: REG_SZ
Value Data: *"Hardware ID of the Device"*

When enabled, you can list all the devices that are allowed to be connected to the system using the Device ID. The list is a numbered item under the AllowDeviceIDs path. The Value Names start with 1 and continue upward in integer values.



- Prevent installation of devices that match any of these device IDs

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions
Value Name: DenyDeviceIDs
Type: REG_DWORD
Value Data: 0 - Disable, 1 – Enable

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions
Value Name: DenyDeviceIDsRetroactive
Type: REG_DWORD
Value Data: 0 - Disable, 1 – Enable

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions\DenyDeviceIDs
Value Name: 1
Type: REG_SZ
Value Data: *"Hardware ID of the Device"*

This setting is the opposite of the previous setting. When enabled, any listed devices with the matching device ID in the list will be blocked from installing. There is an additional checkbox to retroactively disable devices of the same device ID if already installed.

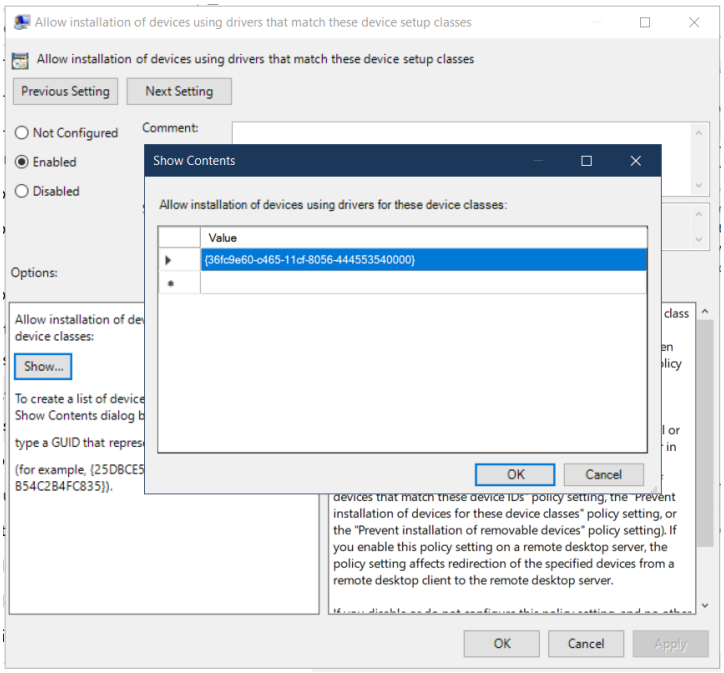
Control by Device Class
There are two policies that either allow or prevent a device from being installed by Device Class. The Device Class is the high-level grouping of devices like a keyboard or mouse. Microsoft defines the class types with GUIDs. There are two tables of Class GUIDs on Microsoft Docs: System-Defined Device Setup Classes Available to Vendors and System-Defined Device Setup Classes Reserved for System Use.

- Allow installation of devices using drivers that match these device setup classes

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions
Value Name: AllowDeviceClasses
Type: REG_DWORD
Value Data: 0 - Disable, 1 – Enable

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions\AllowDeviceClasses
Value Name: 1
Type: REG_SZ
Value Data: *"Device Class"*

When enabled, you can add a list of device class GUIDs. The list is a numbered item under the AllowDeviceClasses path. The Value Names start with 1 and continue upward in integer values.



Some classes are dependent on others. For example, enabling a USB keyboard will require adding two classes to the list: Keyboard {4d36e96b-e325-11ce-bfc1-08002be10318} and USB Bus Devices (hubs and host controllers) {36fc9e60-c465-11cf-8056-444553540000}. Allowing a Device Class to be enabled might be a hole in security, as enabling a class will allow any device of that class to be connected to the system. Device ID provides more granular control.

- Prevent installation of devices using drivers that match these device setup classes

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions
Value Name: DenyDeviceClasses
Type: REG_DWORD
Value Data: 0 - Disable, 1 – Enable

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions
Value Name: DenyDeviceClassesRetroactive
Type: REG_DWORD
Value Data: 0 - Disable, 1 – Enable

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions\DenyDeviceClasses
Value Name: 1
Type: REG_SZ
Value Data: *"Device Class"*

This setting is the opposite of the previous setting. When enabled, any listed devices with the matching device class in the list will be blocked from installing. There is an additional checkbox to retroactively disable devices of the same device class if already installed.

Other Policy Settings
The last three policy settings provide user messaging and auto-restart if a driver has been installed.

- Display a custom message when the installation is prevented by a policy setting

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions\DeniedPolicy
Value Name: DetailText
Type: REG_SZ
Value Data: *"Message to the user"*

When enabled, the setting will enable a message to be displayed when a device is blocked. This may or may not be useful, as most companies want to hide any messages that might indicate Windows is running in the system.

- Display a custom message title when device installation is prevented by a policy setting

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions\DeniedPolicy
Value Name: SimpleText
Type: REG_SZ
Value Data: *"Title of the Message"*

When enabled, the setting adds a title to the message defined in the previous setting.

- Time (in seconds) to force reboot when required for policy changes to take effect

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions
Value Name: ForceReboot
Type: REG_DWORD
Value Data: 0 - Disable, 1 – Enable

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions
Value Name: RebootTime
Type: REG_DWORD
Value Data: "*Time in seconds*"

When enabled, the system will force a reboot when a device driver change takes effect.

Group Policy Scenarios and Setup

Most of our clients are using a few of these settings to lock out devices from being connected. Here are a few scenarios to help with your setup:

Note: The settings will not appear in gpedit.msc if you used reg.exe to add the setting directly. It would be better to set up a custom Group Policy first and then control the individual registry keys using reg.exe. Please see *Start Guide for Windows 10 IoT Enterprise* for more information on creating a custom Group Policy.

- Scenario 1: The system has a single Administrator account with a defined USB device to be connected.

In this scenario, the Administrator account needs to be blocked from adding devices so the following settings are configured:

  - Allow administrators to override Device Installation Restriction policies - Disabled
  - Prevent installation of devices not described by other policy settings - Enabled
  - Allow installation of devices that match any of these device IDs – Enabled and the device ID of the USB device is added to the list.

- Scenario 2: The system has an Administrator account and User account. The User account is the main account in the system. The User is denied installation of devices. The Administrator can install devices.

This simple scenario has the following settings:

  - Allow administrators to override Device Installation Restriction policies - Enabled
  - Prevent installation of devices not described by other policy settings – Enabled
  - (Optional) Prevent installation of removable devices - Enabled

- Scenario 3: The system has a single Administrator account. Only mouse and keyboards can be connected to the system.

This is a case where the device class fits.

- Allow administrators to override Device Installation Restriction policies - Disabled
- Prevent installation of devices not described by other policy settings – Enabled
- Allow installation of devices using drivers that match these device setup classes – Enabled with the following classes in the list for keyboard, mouse, and USB Bus Device:

    - {4d36e96b-e325-11ce-bfc1-08002be10318}
    - {4d36e96f-e325-11ce-bfc1-08002be10318}
    - {36fc9e60-c465-11cf-8056-444553540000}

Comparison

The following table provides a comparison between the two options:

	SecureBus	**Group Policy: Device Installation Restrictions**
Devices Blocked	USB only	All devices
Devices individually blocked by using	PnP ID	Device ID
Device Blocked by Device Class	No	Yes
Provide Message to the user that the device is blocked	No	Yes
Utility to add/remove devices	Yes (as well as API available)	Gpedit.msc / Device Manager
Can block device drivers from updating	No	Yes
Works when UWF is enabled	Yes	Only with LTSC 2019 (version 17763) release
Cost	Per unit royalty	Free in Windows

In either case, any device already installed that you don't want to have in the system will have to be uninstalled or blocked. For example, the USB device, keyboard, and mouse used during Windows installation will have to be removed. In SecureBus, this can be accomplished from within the SecureBus utility. For Group Policy Device Installation Restrictions, you will have to go into Device Manager to uninstall each device.

Most embedded PCs only expose USB ports besides the usual serial, video, audio, etc. When you have to replace a hard drive or update a device driver, SecureBus is a simpler solution than the all device block in the group policy solution.

Summary

Security is at the top of everyone's mind when building an embedded/IoT device. Carefully architecting the Windows IoT image is import to secure the system and make sure the system can still be supported in the field. We have covered look and feel solutions, Keyboard filter, Unified Write Filter, custom security, etc. in previous books and articles. One more thing to add to the list is device blocking to prevent any backdoor access to your system.