

Commercial Windows IoT



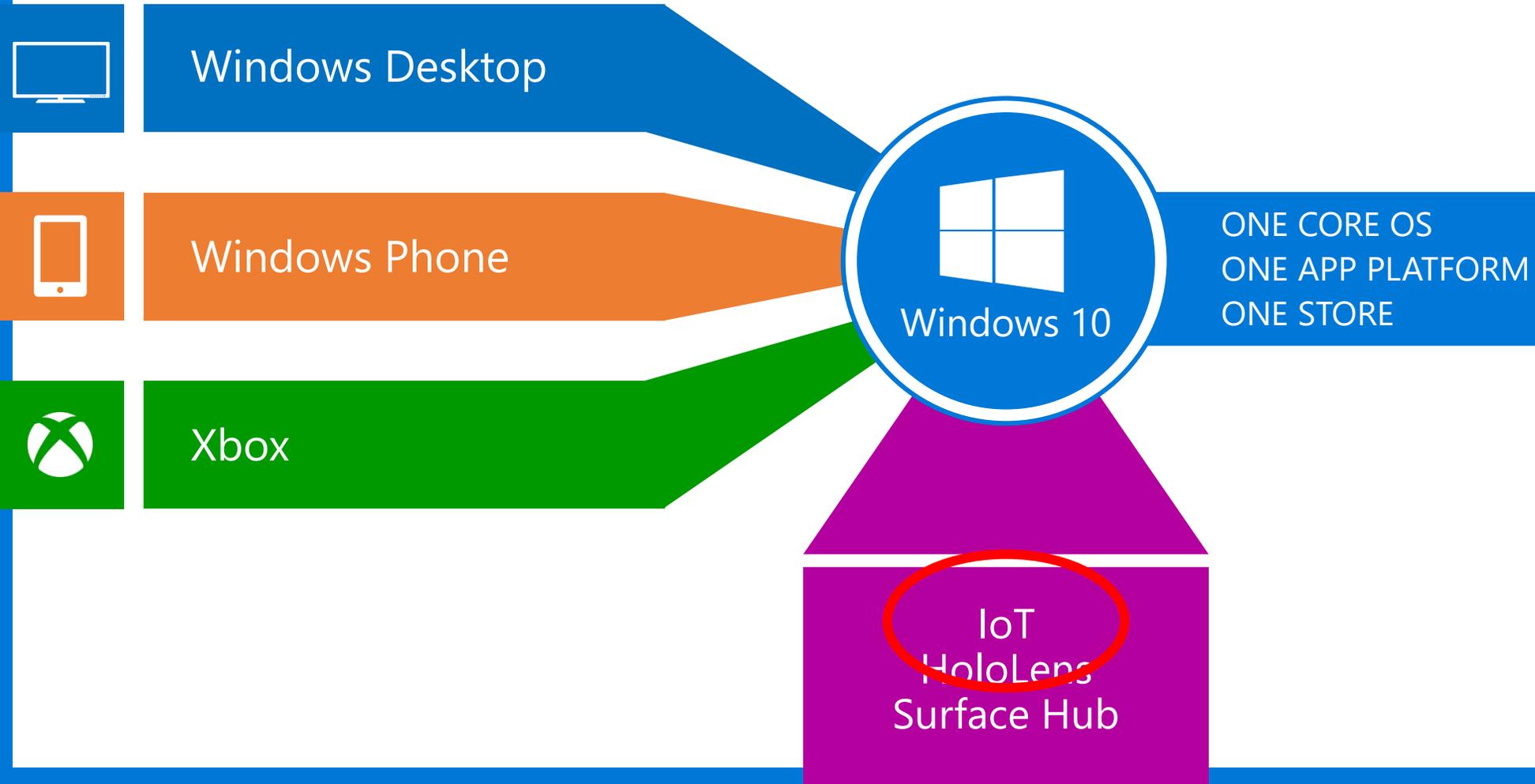
Gunter
Logemann
Sr. Consultant

Commercial Windows IoT

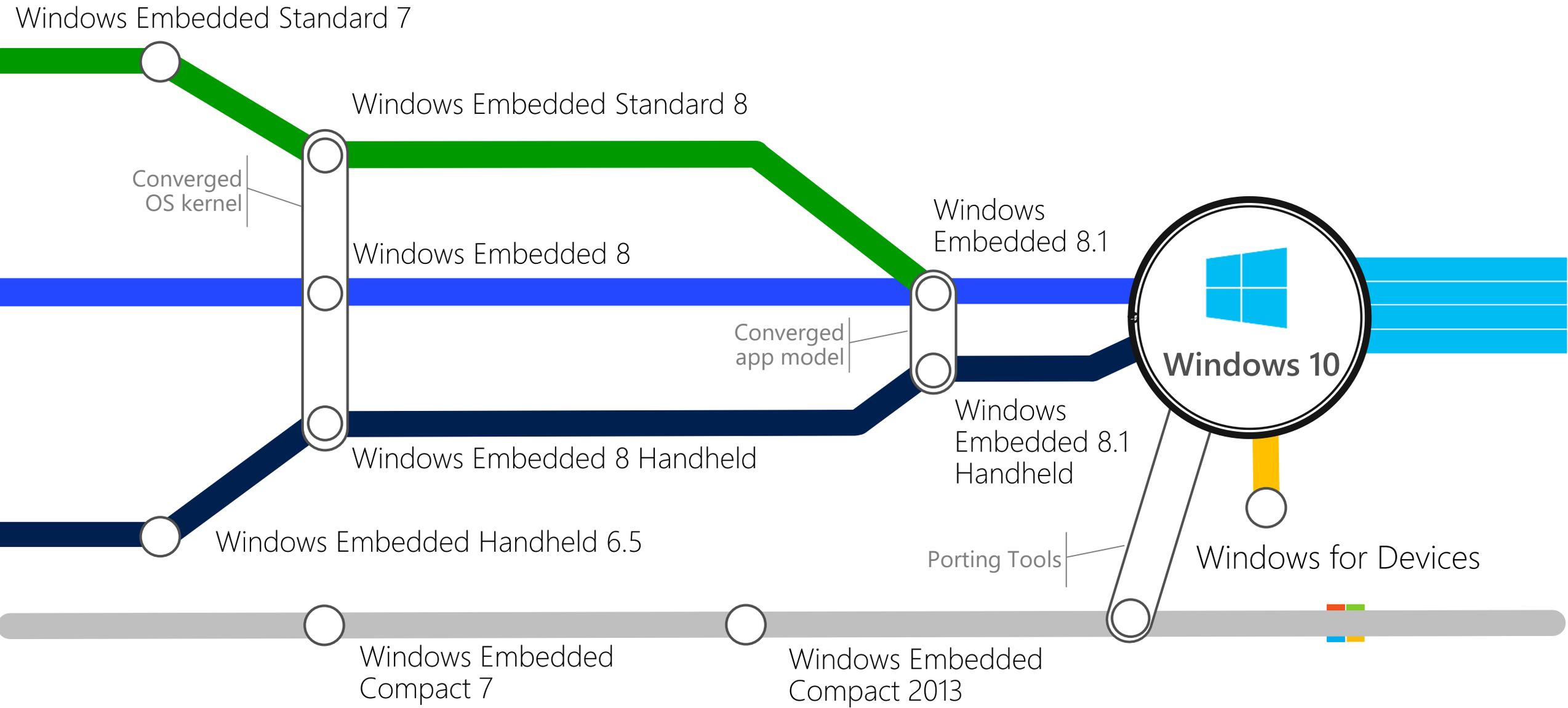


Dr. Holger Kenn
Principal Consultant

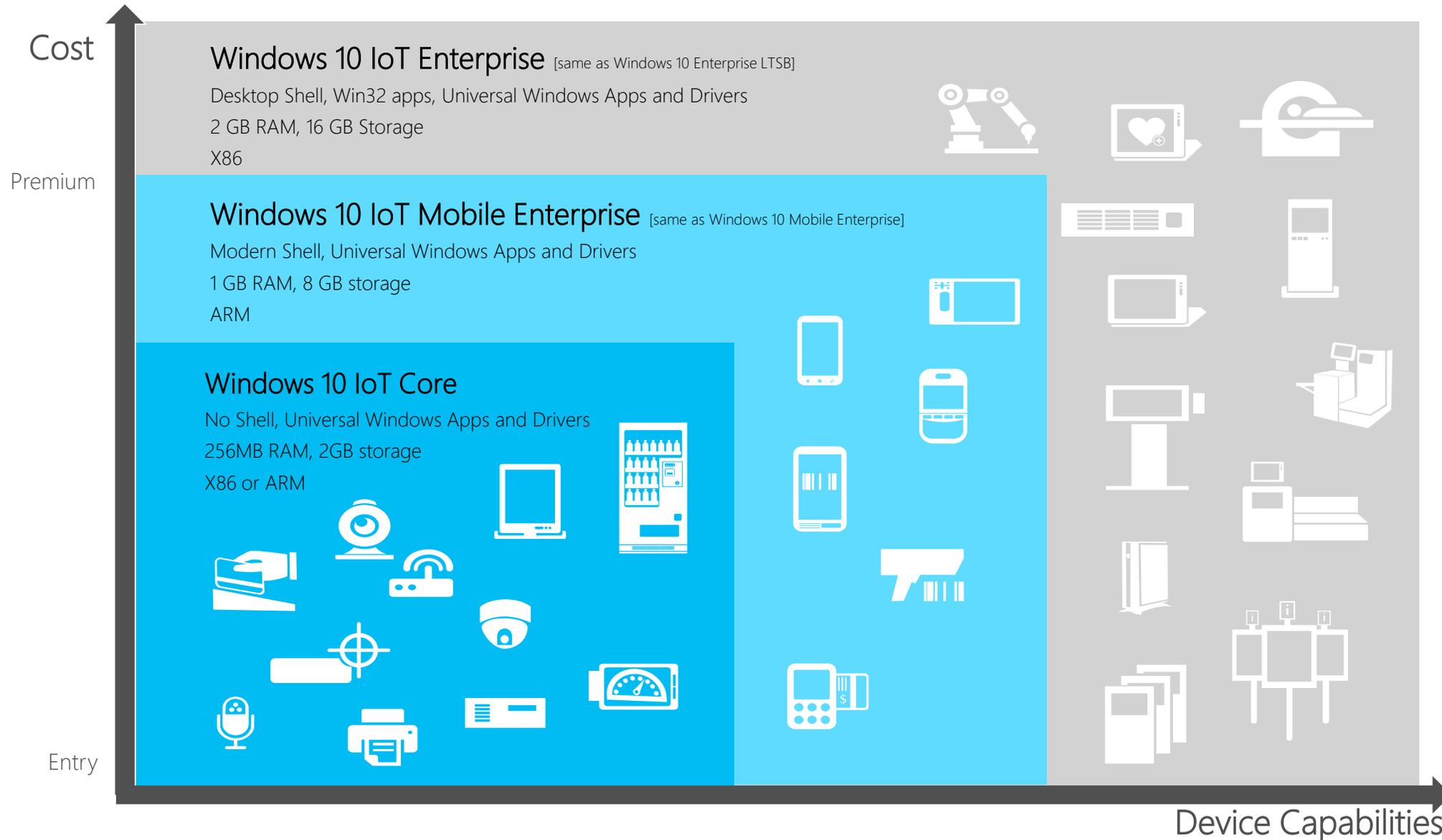
The journey to one Windows...



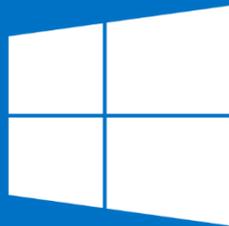
Embedded Platform Convergence Journey



Windows 10 IoT



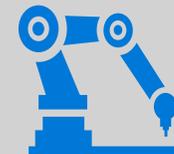
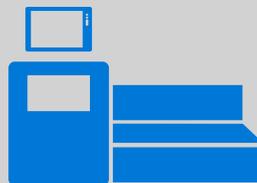
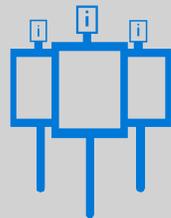
Universal Windows Platform



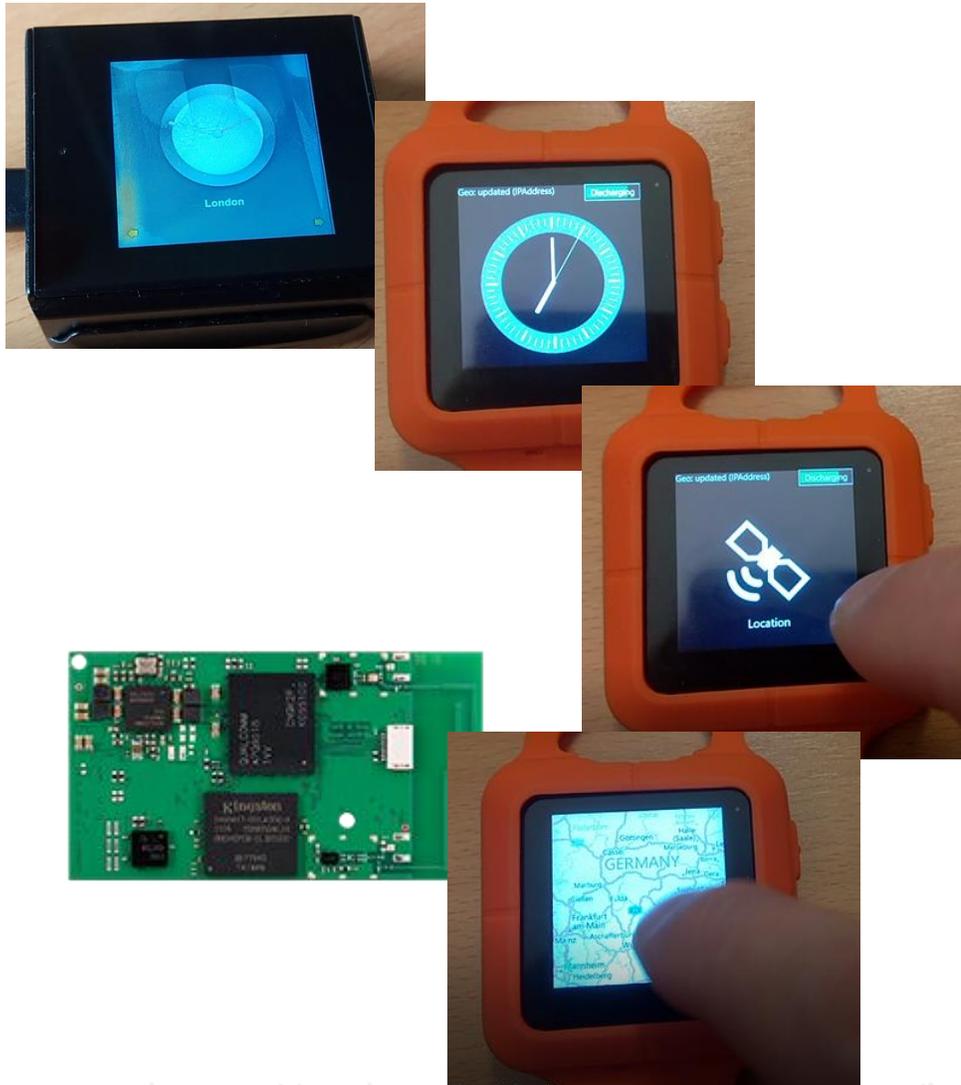
UWP apps

Natural & rich user experience

One management & servicing approach



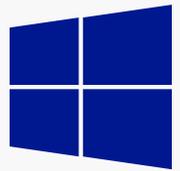
UWP for different devices



- Full support of all App models, using C++, C#, Java Script
- Full support of 3rd party Appmodels that support UWP. E.g. Qt, Xamarin, Unity
- Adaptive Layout and Code allows you to build on App that scales to differnt platforms
- App framework for the Wearable available
- Samples to demonstrate differnt IOT specific features (e.g. Process launcher)

Microsoft IoT

Comprehensive solutions from device to cloud



Windows



Azure IoT

IoT Editions Power a Broad Range of Devices

25 years of history in embedded devices

One Windows platform for all devices

Enterprise-ready, OEM-ready, Maker-friendly

Designed for today's IoT environments

Scalable solutions from free Windows IoT Core to Windows IoT Enterprise on PC-Like Devices

Cloud-Based IoT Services & Solutions

Easy to provision, use and manage

Pay as you go, scale as you need

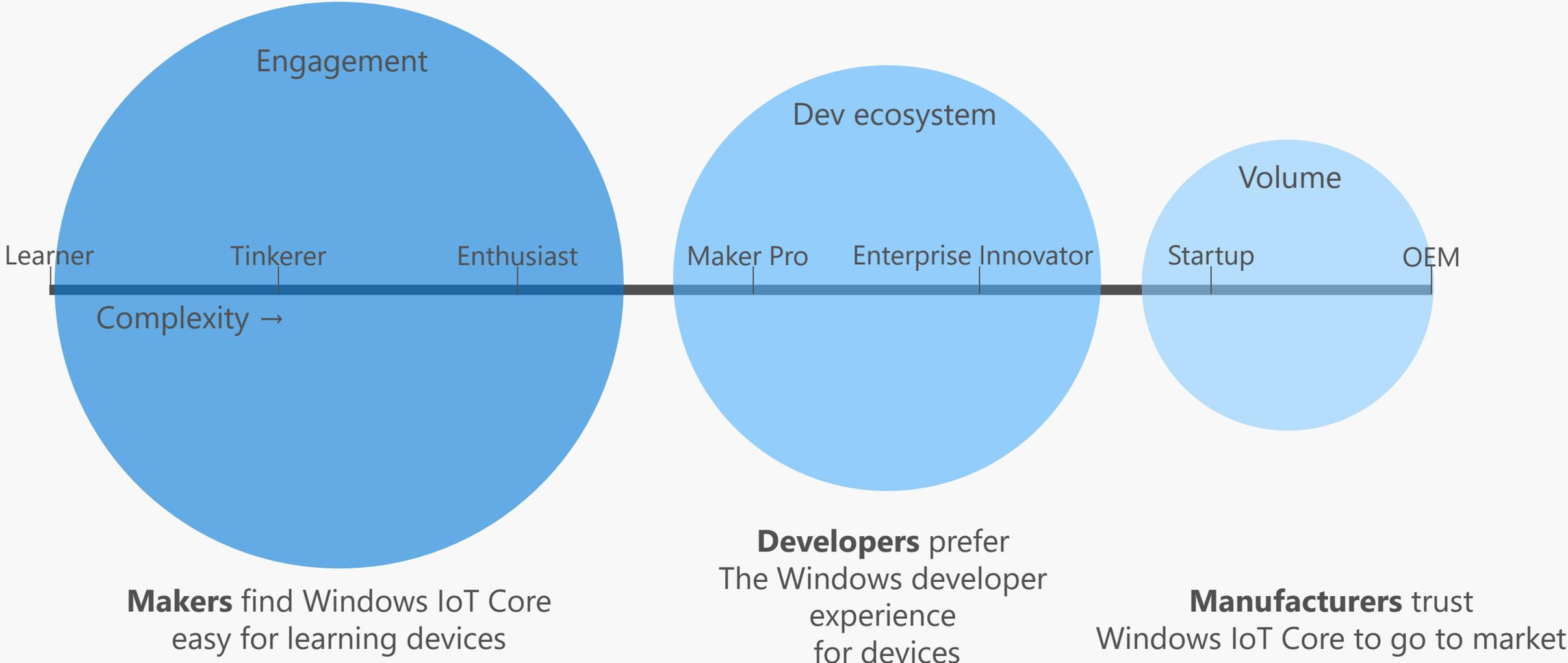
Global reach, hyper scale

End-to-end security & privacy

Windows, Mbed, Linux, iOS, Android, RTOS support

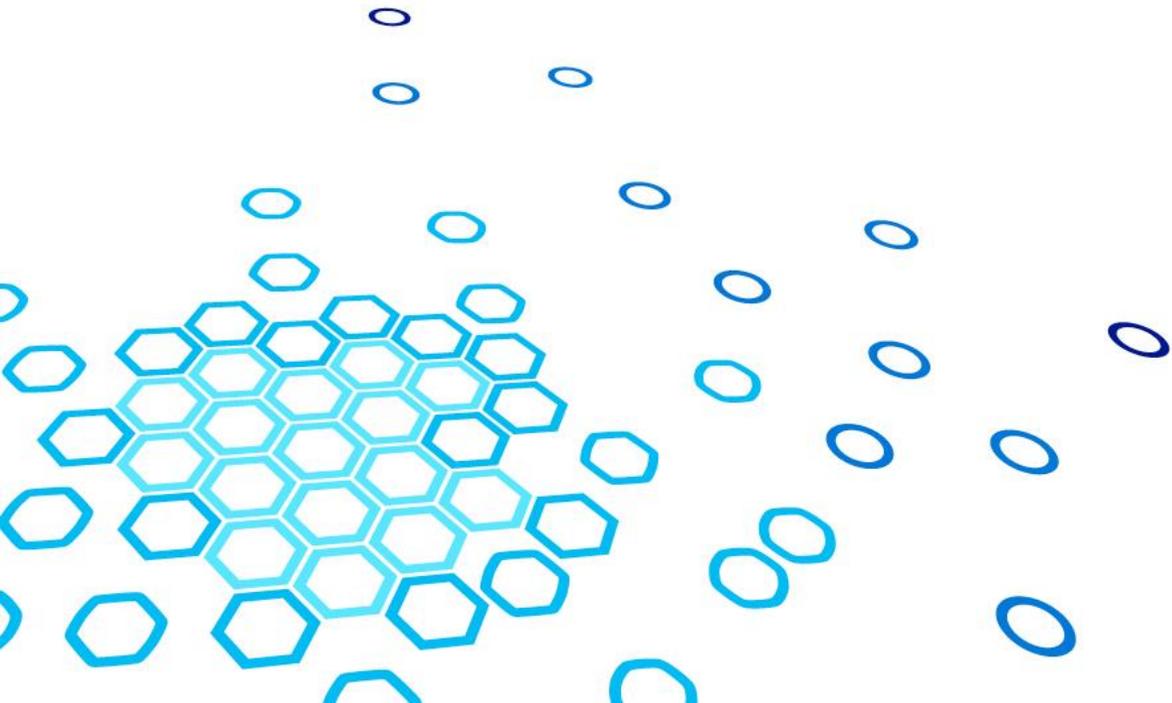
Any developer can build an IoT device

Devices are the new apps!



What is Windows IOT

Enable Embedded Features



Embedded features

Background Applications	Use of the lowlevelDevice capability	Use of the systemManagement capability
Remove limits enforced by the resource manager.	low-level hardware interfaces like GPIO, SPI, and I2C. (IOT Core only)	ProcessLauncher TimeZoneSettings ShutdownManager AllJoyn loopback

Embedded mode is only enabled by default on Windows IoT Core and must be enabled on standard Windows and Windows Mobile

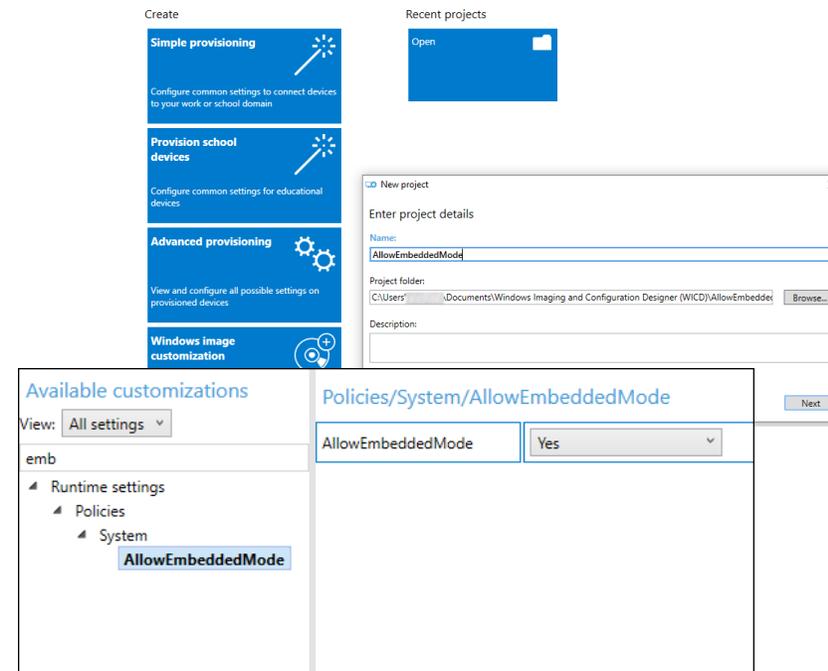
How to enable features

Download and install the ADK

Install Imaging and Configuration Designer (ICD)

Create a provisioning Package that sets AllowEmbeddedMode=Yes

Install the package



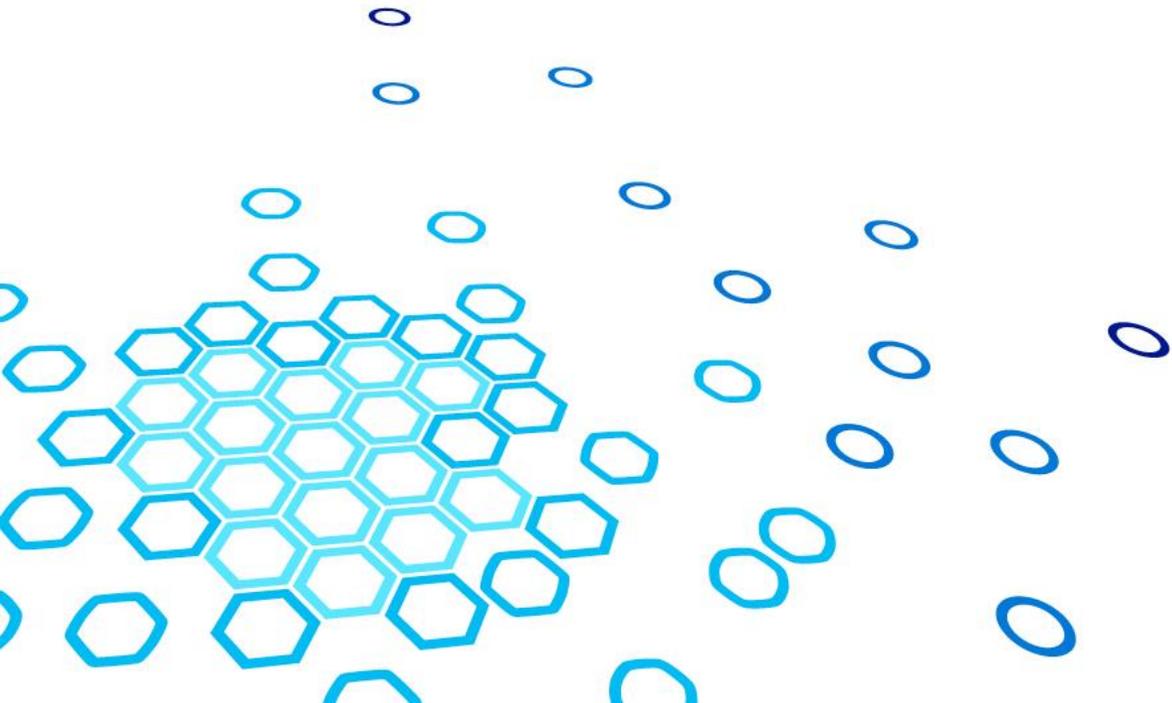
More embedded features...

(also apply to windows 10 in general now)

 <p>Write Filters and Overlays</p>	 <p>USB Filter</p>	 <p>Dialog and Notification Filters</p>	 <p>Input Filters</p>	 <p>AppLocker and Layout Control</p>	 <p>Shell and App Launcher</p>
Easily create read only devices. Improve system uptime	Only allow approved USB peripherals	Block Pop-up Dialog Boxes and system notifications	Block hotkeys and edge gestures to prevent system access	Control which apps are visible and can run	Enable single Win32 or Modern app experience on device

IOT Security

Security is not optional for Devices!



Threats for IOT devices

Default Passwords and backdoors in devices

Devices that are not maintained

Unsafe credential store

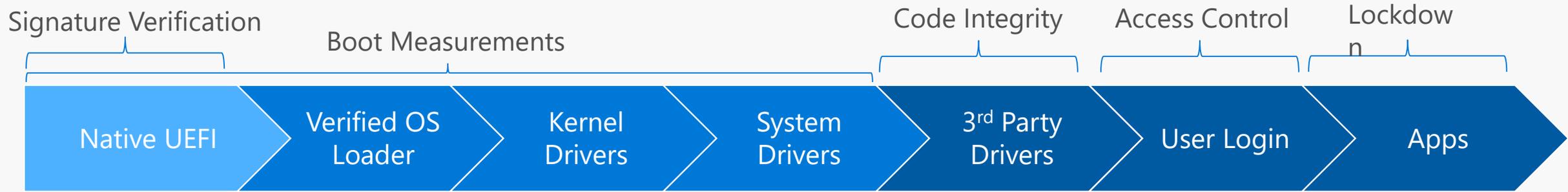
Web-based or app-based configuration

Unpatched bugs

Secrets in the file system

Open ports and services
(I'm looking at you, raspbian!)

Securing Windows IoT Devices



Boot malware resistance with UEFI Secure Boot

Firmware enforces policy and only starts signed OS loader

Secure device identity and health attestation

TPM support across all IoT SKUs brings strong device identity, secure key management and health attestation

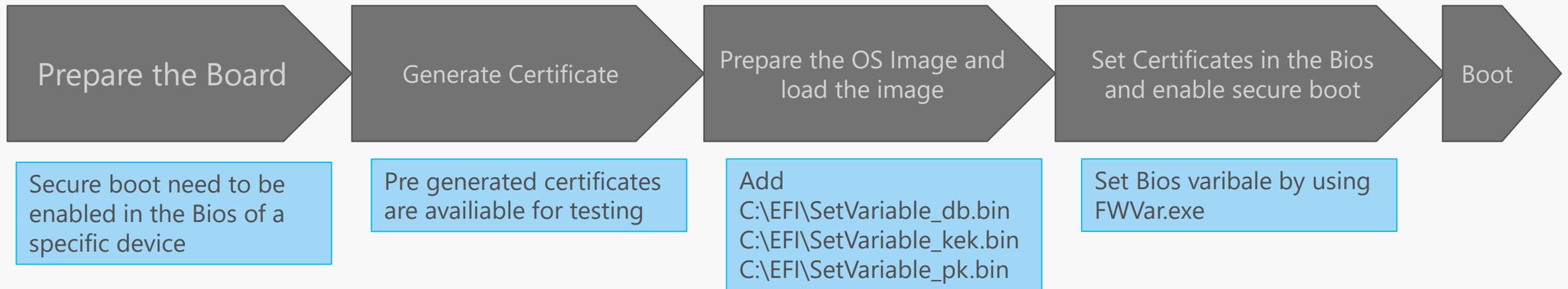
Identity protection and access control

Supported by features like Microsoft Passport (2FA), Windows Hello, Credential Guard (virtualization-based security)

Advanced lock-down capabilities

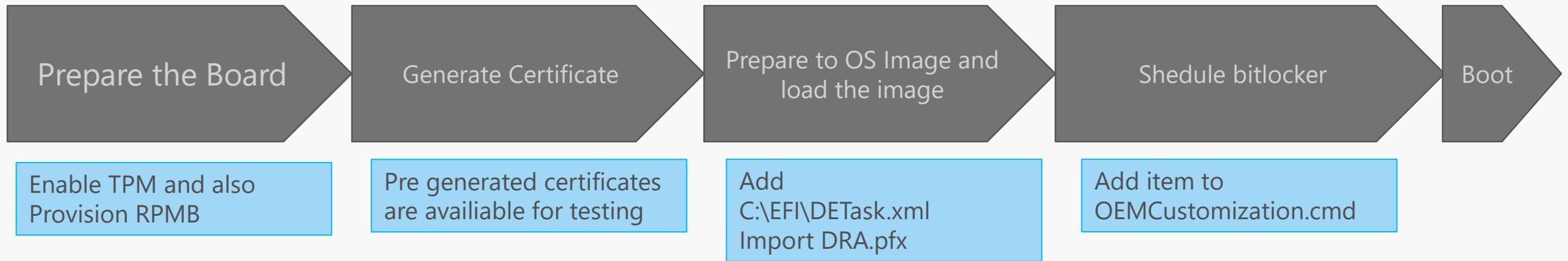
Supported in Windows 10 with AppLocker & Device Guard along with Enterprise Data Protection & BitLocker

Enable Secure Boot



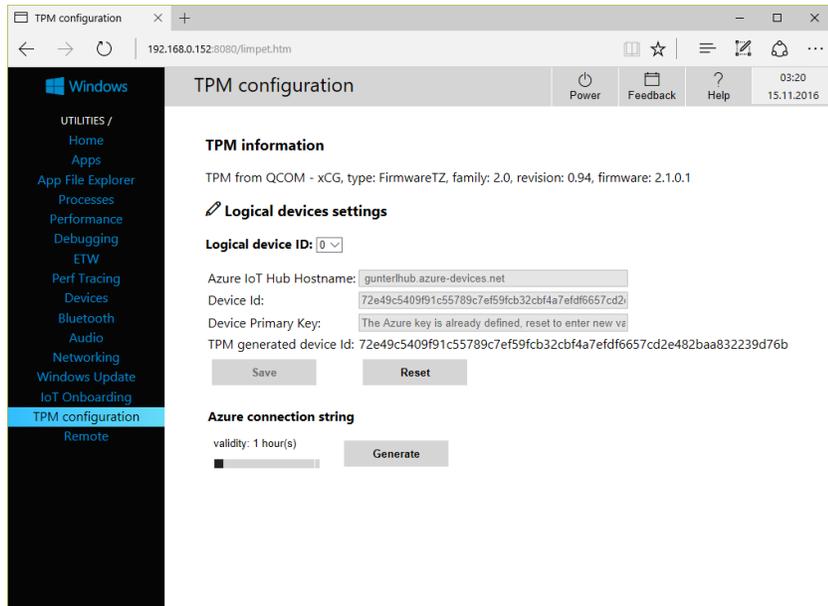
<https://developer.microsoft.com/en-us/windows/iot/docs/securebootandbitlocker>

Enable Bitlocker



Connect to the cloud (Azure IOT Hub)

Step 1: Provision the TPM



```
tpm = new TpmDevice(0);
// reset TPM to clean previous
try
{
    Debug.WriteLine("Reset TPM...");
    tpm.Destroy();
}
catch (Exception ex)
{
    Debug.WriteLine("TPM was not initialized!");
}
Debug.WriteLine("TPM initialized");
string id = tpm.GetDeviceId();

//HWID is unique for this device.
string hwid = tpm.GetHardwareDeviceId();
Debug.WriteLine("TPM Hardware ID:" + hwid);

string hmackey = CryptoKeyGenerator.GenerateKey(32);
Debug.WriteLine("TPM hmackey:" + hmackey);

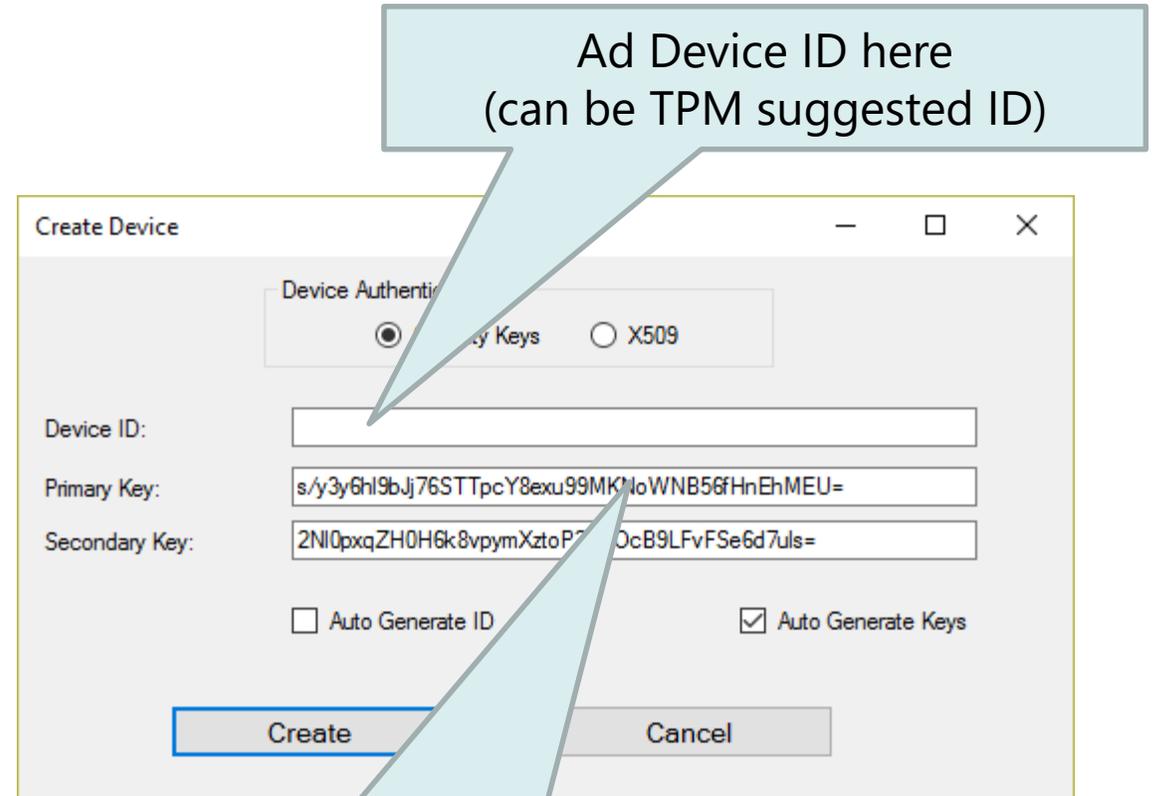
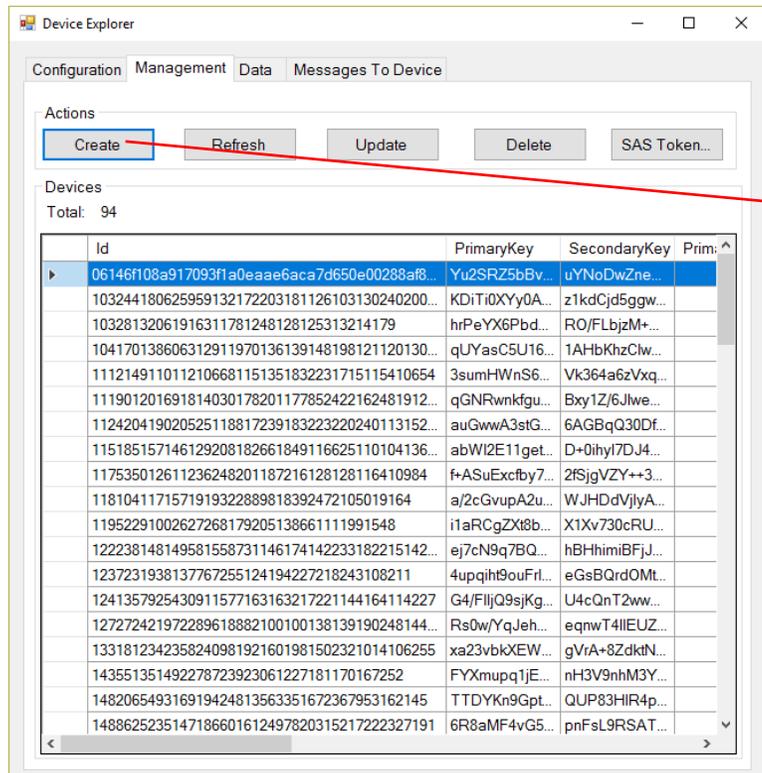
//provision the device.
tpm.Provision(hmackey, "gunterlhub.azure-devices.net", hwid);
```

Use Portal or
provisioning app

Connect to the cloud (Azure IOT Hub)

Step 2: Create the device on IOT Hub

Manual Way using
Device Explorer



Ad HMAC Key here
(generated by the provisioning App)

Connect to the cloud (Azure IOT Hub)

Step 3: Use the IOT Hub in your application (test connection)

```
public static async Task<bool> TestHubConnection(bool sendRestartMessage, string restartMessage)
{
    try
    {
        TpmDevice myDevice = new TpmDevice(0); // Use logical device 0 on the TPM
        string hubUri = myDevice.GetHostName();
        string deviceId = myDevice.GetDeviceId();
        string sasToken = myDevice.GetSASToken();
        if ((hubUri.Length == 0) || (sasToken.Length == 0)) return false;
    }
    catch (Exception ex)
    {
        return false;
    }

    if (sendRestartMessage)
    {
        return await SendDeviceToCloudMessageAsync(restartMessage);
    }
    return true;
}
```

Connect to the cloud (Azure IOT Hub)

Step 3: Use the IOT Hub in your application (send message)

```
public static async Task<bool> SendDeviceToCloudMessageAsync(string str)
{
    try
    {
        TpmDevice myDevice = new TpmDevice(0); // Use logical device 0 on the TPM
        string hubUri = myDevice.GetHostName();
        string deviceId = myDevice.GetDeviceId();
        string sasToken = myDevice.GetSASToken();

        var deviceClient = DeviceClient.Create(
            hubUri,
            AuthenticationMethodFactory.
                CreateAuthenticationWithToken(deviceId, sasToken), TransportType.Amqp);

        var message = new Message(Encoding.ASCII.GetBytes(str));

        await deviceClient.SendEventAsync(message);
        return true;
    }
    catch (Exception ex)
    {
        return false;
    }
}
```

Connect to the cloud (Azure IOT Hub)

Step 3: Use the IOT Hub in your application (receive message)

```
public static async Task<string> ReceiveCloudToDeviceMessageAsync()
{
    while (true)
    {
        TpmDevice myDevice = new TpmDevice(0); // Use logical device 0 on the TPM by default
        string hubUri = myDevice.GetHostName();
        string deviceId = myDevice.GetDeviceId();
        string sasToken = myDevice.GetSASToken();

        var deviceClient = DeviceClient.Create(hubUri, AuthenticationMethodFactory.CreateAuthenticationWithToken(deviceId, sasToken), TransportType.Amqp);

        Message receivedMessage = null;

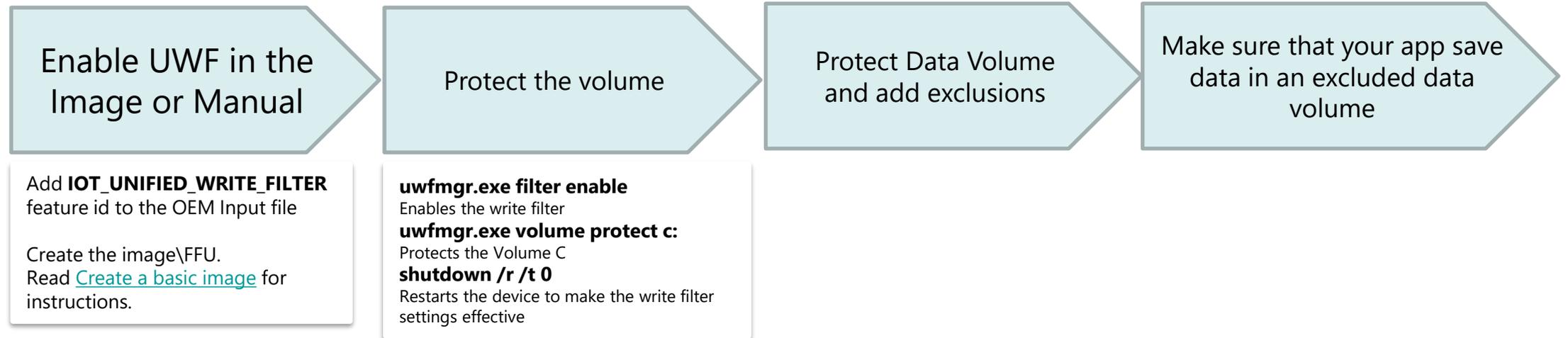
        receivedMessage = await deviceClient.ReceiveAsync();

        if (receivedMessage != null)
        {
            var messageData = Encoding.ASCII.GetString(receivedMessage.GetBytes());
            await deviceClient.CompleteAsync(receivedMessage);
            deviceClient.Dispose();
            return messageData;
        }

        await Task.Delay(100);
    }
}
```

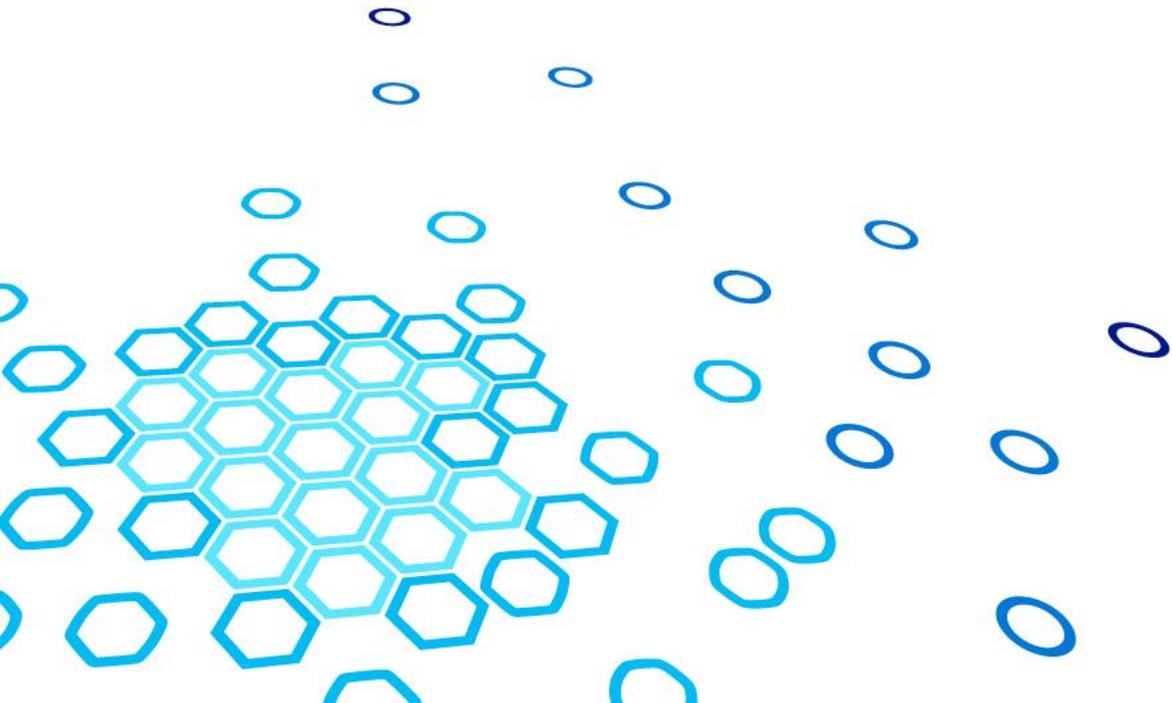
Unified Write Filter

- The Unified Write Filter (UWF) is a feature to protect physical storage media from data writes.
- UWF intercepts all write attempts to a protected volume and redirects those write attempts to a virtual overlay.
- improves the reliability and stability of your device and reduces



Productization

Provisioning and Update





Configure Windows Update

Windows 10 IoT (Core) Pro SKU provides the capability to control and schedule the windows update. This enables the policies related to Update such as:

[Update/AllowAutoUpdate](#), [Update/ScheduledInstallDay](#), [Update/ScheduledInstallTime](#), [Update/UpdateServiceUrl](#).

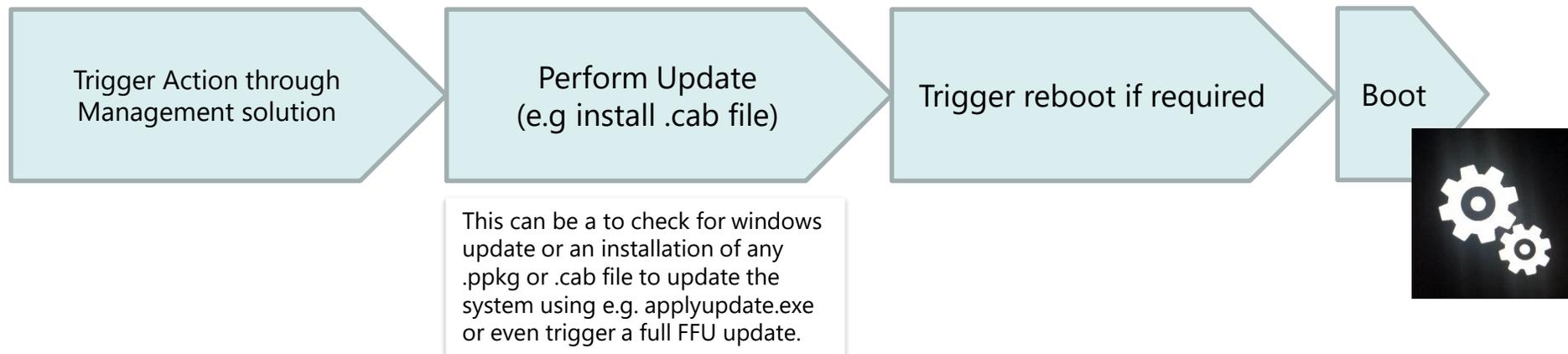


Managing Windows 10 IoT Core Devices

Windows 10 IoT Core devices can be managed using a traditional OMA DM MDM server that supports certificate based enrollment or using Azure IoT Hub's Device Management (currently in preview).

Learn more about MDM and Windows 10 [here](#).

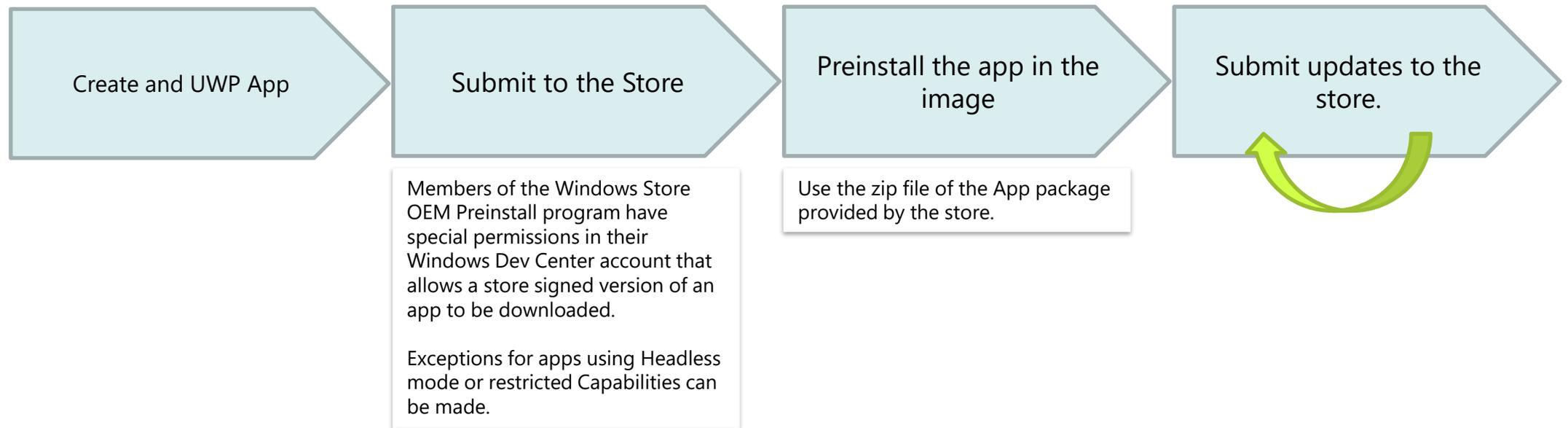
Learn more about Azure IoT Hub Device Management [here](#).



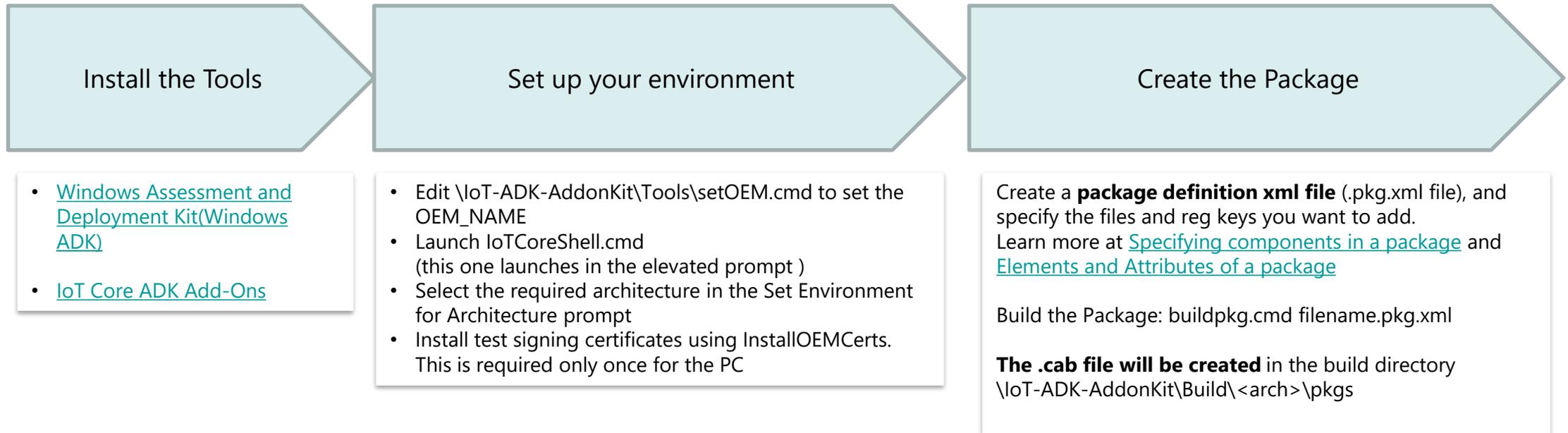
Installing and Servicing apps on Windows 10 IoT Core (Pro) using the Store

Microsoft makes it easy for OEMs to install and service UWP apps on Windows 10 IoT Core through the Universal Store.

All store signed apps on Windows 10 devices are capable of receiving updates directly from the store.



Create and Install Packages manually



Create a package with files and reg keys

```
<?xml version="1.0" encoding="utf-8"?>
<Package xmlns="urn:Microsoft.WindowsPhone/PackageSchema.v8.00"
  Owner="OEMName"      OwnerType="OEM"
  ReleaseType="Test"   Platform="PlatformName"
  Component="ComponentName" SubComponent="SubName">
  <Components>
    <OSComponent>
      <Files>
        <File Source="$_RELEASEDIR\test_file1.dll"/>
        <File Source="$_RELEASEDIR\toBeRenamed.dat"
          DestinationDir="$(runtime.system32)\test" Name="test.dat"/>
      </Files>
      <RegKeys>
        <RegKey KeyName="$(hklm.software)\OEMName\test">
          <RegValue Name="StringValue" Value="Test string" Type="REG_SZ"/>
          <RegValue Name="DWordValue" Value="12AB34CD" Type="REG_DWORD"/>
          <RegValue Name="BinaryValue" Value="12,AB,CD,EF" Type="REG_BINARY"/>
        </RegKey>
        <RegKey KeyName="$(hklm.software)\OEMName\EmptyKey"/>
      </RegKeys>
    </OSComponent>
  </Components>
</Package>
```

Create an Appx package

Use appx2pkg.cmd or newappxpkg.cmd tool to generate the .pkg.xml file for a given appx file.

This tool expects the appx dependencies in the sub directory named "dependencies" in the folder containing the appx file.

<https://msdn.microsoft.com/en-us/windows/hardware/commercialize/manufacture/iot/iot-core-manufacturing-guide>

The image shows a file explorer window on the left and a command prompt window on the right. The file explorer shows a folder structure with the following items:

- Add-AppDevPackage.resources
- Dependencies
- Add-AppDevPackage
- Appx.IOTBrowser.pkg
- IOTBrowser_1.0.0.0_ARM
- IOTBrowser_1.0.0.0_ARM.appxsym
- IOTBrowser_1.0.0.0_ARM

The command prompt shows the following commands and output:

```
IoTCore arm 10.0.0.0
C:\iot-adk-addonkit\Tools>newappxpkg.cmd "C:\test\IOTBrowser_1.0.0.0_ARM_Test\IOTBrowser_1.0.0.0_ARM_Test\IOTBrowser_1.0.0.0_ARM.appx" Appx.Browser
Creating Appx.Browser package
Creating package xml file
Authoring Appx.Browser.pkg.xml
C:\iot-adk-addonkit\Source-arm\Packages\Appx.Browser ready

IoTCore arm 10.0.0.0
C:\iot-adk-addonkit\Tools>buildpkg Appx.Browser
Processing Appx.Browser.pkg.xml

IoTCore arm 10.0.0.0
C:\iot-adk-addonkit\Tools>
```

The command prompt output is circled in green. A blue arrow points from the "Dependencies" folder in the file explorer to the "newappxpkg.cmd" command in the command prompt. Another blue arrow points from the "buildpkg Appx.Browser" command in the command prompt to the "PCCON.Appx.Browser" folder in the file explorer. The file explorer shows the following files and folders:

- AppInstall (Windows Command Script, 1,82 KB)
- AppxConfig (Windows Command Script, 233 bytes)
- arm_pcon.appx.browser0_62884447771337a_10.0.0.0_none_b10132a... (MANIFEST File)
- IOTBrowser_1.0.0.0_ARM (APPX File, 842 KB)
- Microsoft.NET.Native.Framework.1.3 (APPX File)
- Microsoft.NET.Native.Runtime.1.4 (APPX File, 212 KB)
- Microsoft.VCLibs.ARM.14.00 (APPX File, 662 KB)
- update (Security Catalog, 5,85 KB)
- update.mum (MUM File, 3,95 KB)
- arm_pcon.appx.browser.deploymentsent_62884447771337a_10.0.0.0_n... (MANIFEST File)
- IOTBrowser_1.0.0.0_ARM (Security Certificate, 730 bytes)

Download and install the Package

```
public async Task StartDownloadandInstall(Uri url)
{
    Uri source = url; // new Uri("http://www.pcon.de/test/appupdate/update.main.zip");

    StorageFile destinationFile = await ApplicationData.Current.LocalFolder.CreateFileAsync(
        "download.zip", CreationCollisionOption.GenerateUniqueName);

    BackgroundDownloader downloader = new BackgroundDownloader();
    DownloadOperation download = downloader.CreateDownload(source, destinationFile);
    await download.StartAsync();

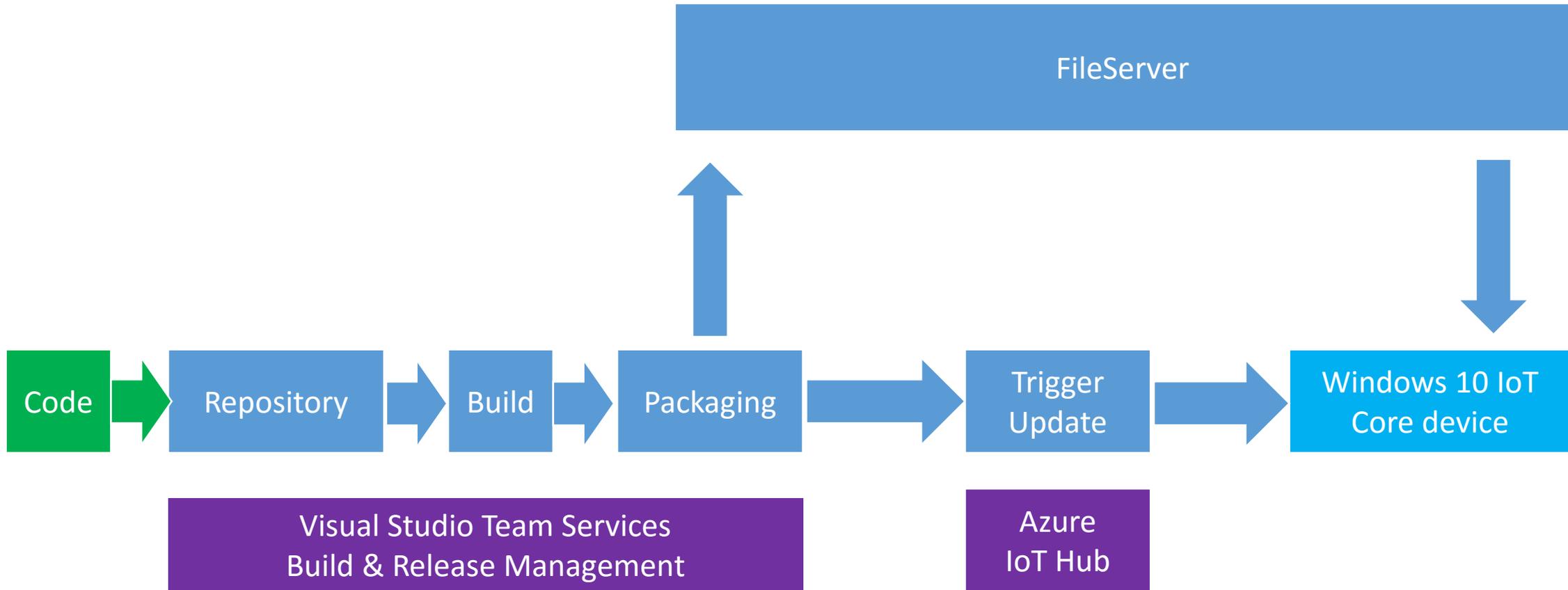
    await UnzipFile(download.ResultFile.Path);

    StorageFolder localFolder = ApplicationData.Current.LocalFolder;
    StorageFolder t = null;
    try {
        t = await localFolder.GetFolderAsync("installer");
    }
    catch {
        t = null;
    }
    if (t!=null) {
        await t.DeleteAsync();
    }
    StorageFolder f = await localFolder.GetFolderAsync("update.main");
    await f.RenameAsync("installer");
    string path = localFolder.Path + "\\installer\\AppInstall\\appinstall.cmd";
    string s = "";

    // REG ADD "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\EmbeddedMode\ProcessLauncher" /v AllowedExecutableFilesList /t REG_MULTI_SZ /d
    "c:\windows\system32\applyupdate.exe\0c:\windows\system32\deployappx.exe\0c:\installer\appinstall.cmd\0c:\Data\Users\DefaultAccount\AppData\Local\Packages
    \15c8ba7d-b8cc-46ee-84f1-ef0f27753fbe_0wy2ejr5nfw9j\LocalState\installer\AppInstall\appinstall.cmd\0"

    await App.ViewModel.RunProcess(path, s);
}
```

Example:
End-To-End deployment to Windows 10 IoT Core Devices
via **VSTS** & **IoT Hub**



OEM license requirements

The process of licensing Windows 10 IoT Core product and the OEM license agreement is provided at [Windows 10 IoT Core Commercialization](#).

As part of signing the Windows 10 IoT Core OEM license agreement, you are required to meet these system requirements for the Windows 10 IoT Core device.

SMBIOS Support

The following are the minimum required fields in SMBIOS for IoTCore

- (Table 1, offset 04h) System Manufacturer
- (Table 1, offset 05h) System Product Name
- (Table 1, offset 19h) System SKU
- (Table 1, offset 1Bh) System Family

And finally.... Go To market

Understand licensing requirements

- See [Windows 10 IoT Core Commercialization](#) to understand the licensing requirements

Create a retail image

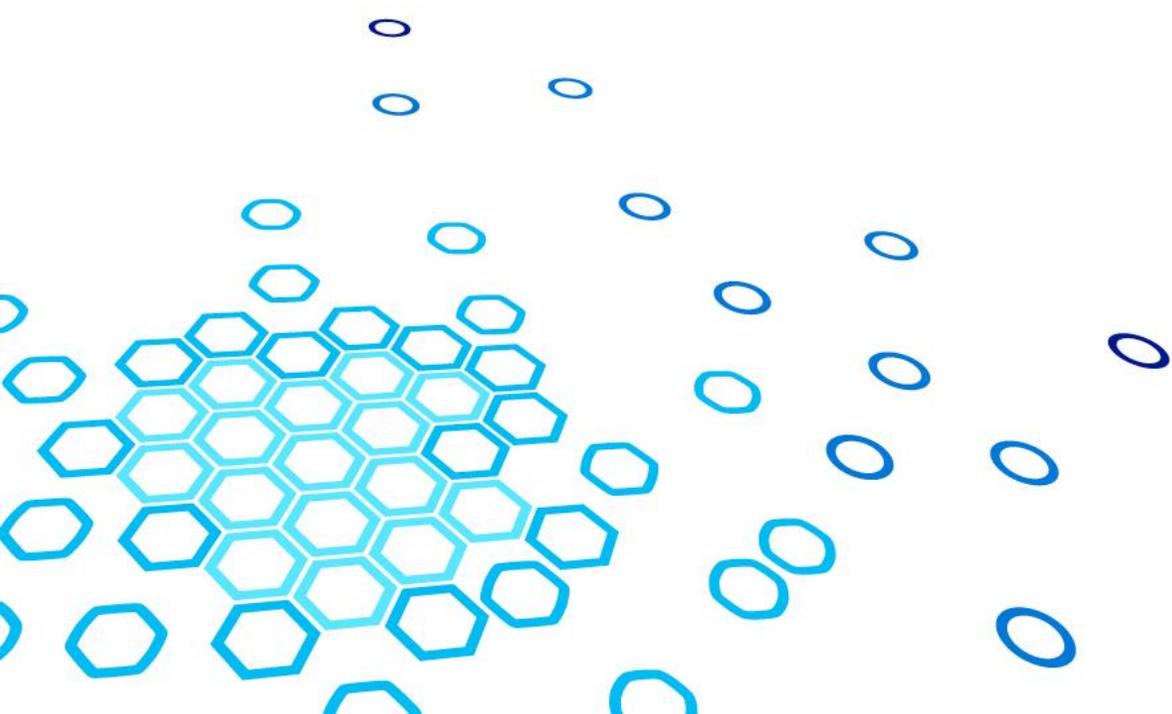
- Learn how to create a custom image at [Windows 10 IoT Core manufacturing guide](#)
- Learn how to lockdown your device at [Building Secure Devices](#)
- Learn how to configure your device for servicing at [Service IoTCore](#)

Arrange for device manufacturing

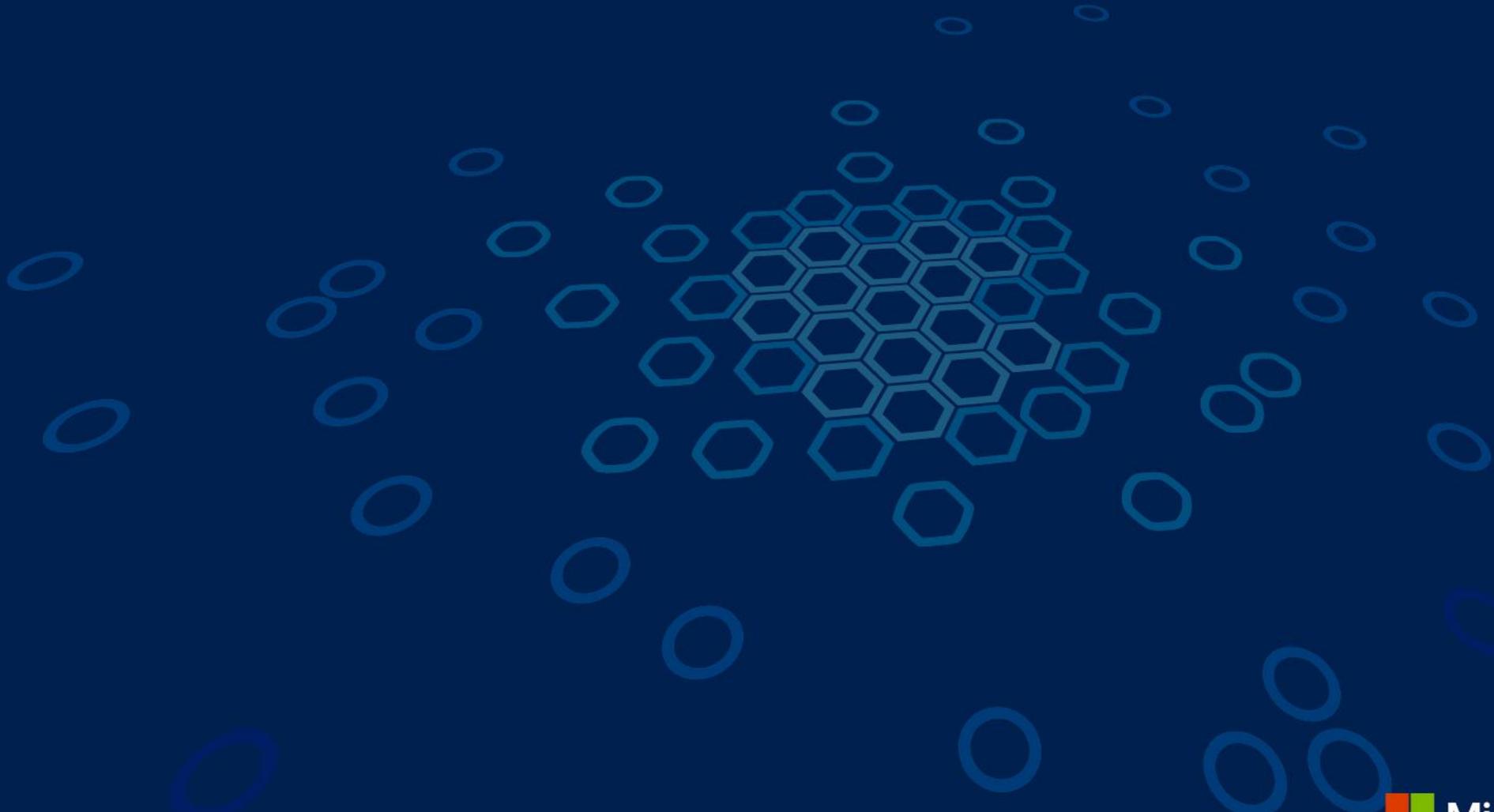
Get contact info for parts suppliers (SVs/IHVs/ODMs). Contact the supplier directly and follow their process to get components and licenses as necessary. Work directly with your SoC partner to locate an appropriate manufacturer.

- See [Supported boards and SoCs](#)

Vielen Dank



© 2016 Microsoft Corporation. All rights reserved. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.



Technical Summit 2016
connecting.technologies.

#msts2016 

Technical Summit 2016
connecting.technologies.

#msts2016 

Vielen Dank

© 2016 Microsoft Corporation. All rights reserved. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.

