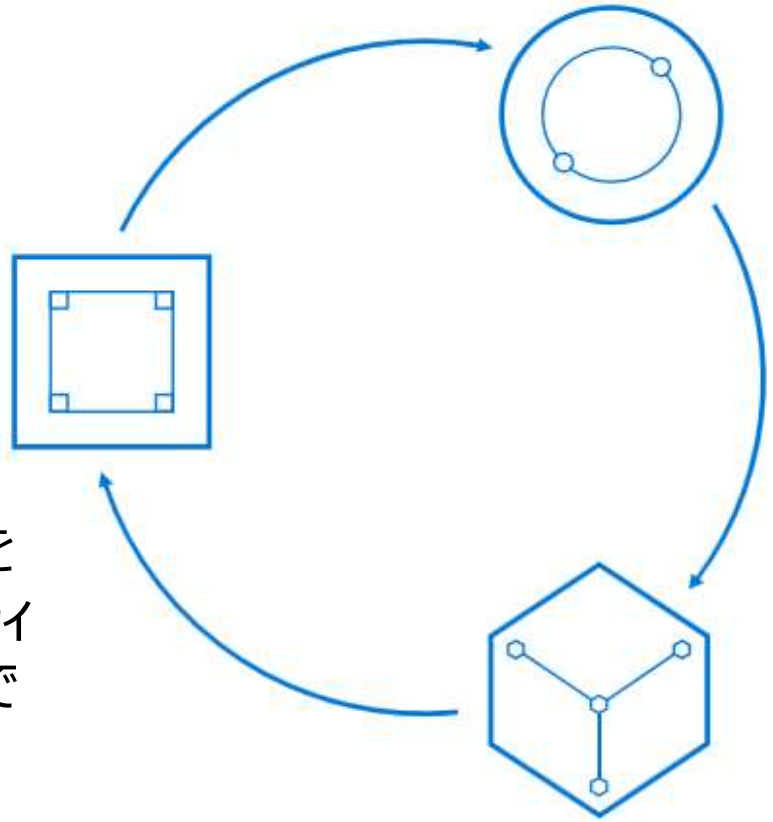
セキュア IoT プラットフォーム
Azure Sphere

アヴネット株式会社

マイクロコントローラデバイス向けのセキュアIoTプラットフォーム

2020年2月 General Availability!



Azure Sphere OS

新しいIoTエクスペリエンスを実現する“信頼されたプラットフォーム”を作るために、10年間のライフタイムを提供する。

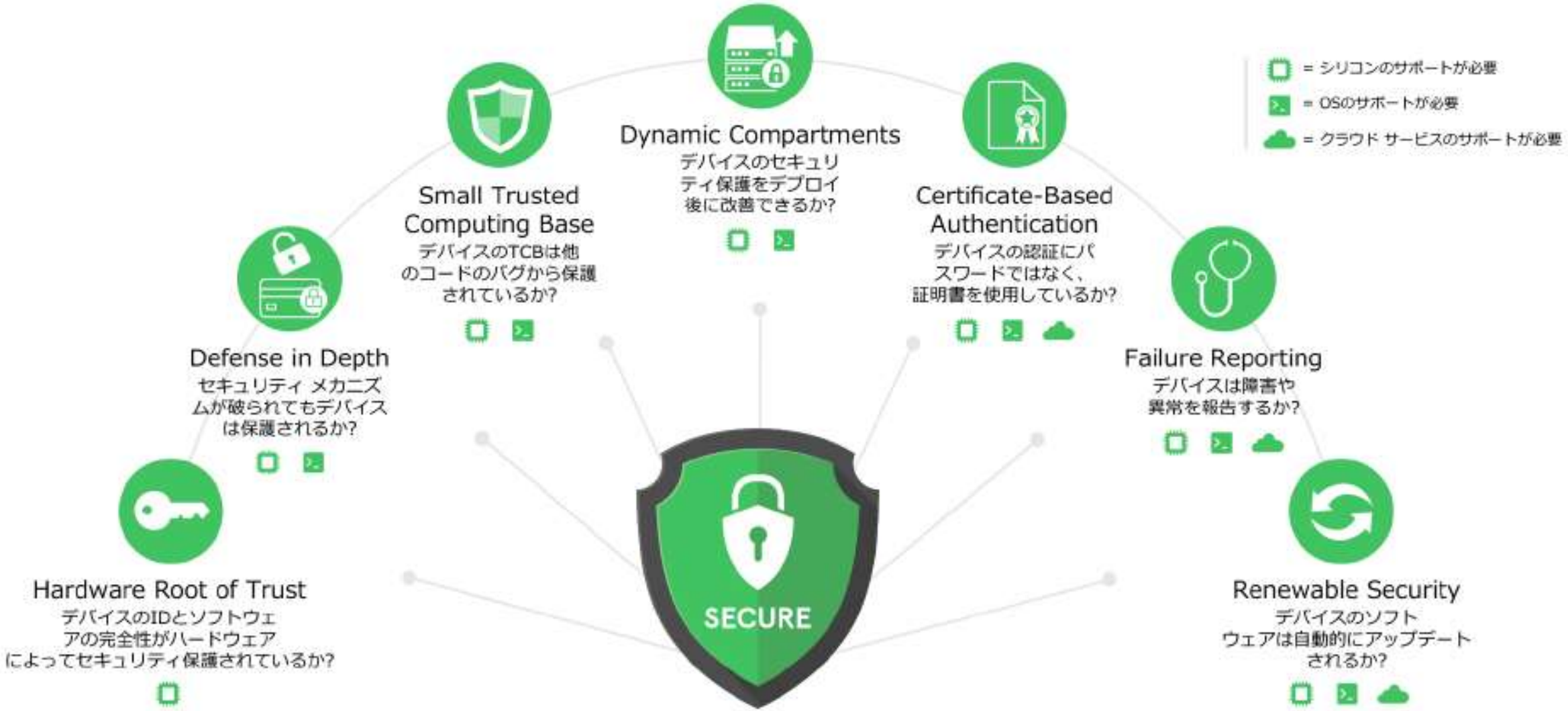
Azure Sphere MCU

接続性と信頼された”Hardware Root of Trust” を提供するマイクロソフトのセキュリティテクノロジーが組み込まれた状態でシリコンパートナーから提供される。

Azure Sphere Security Service

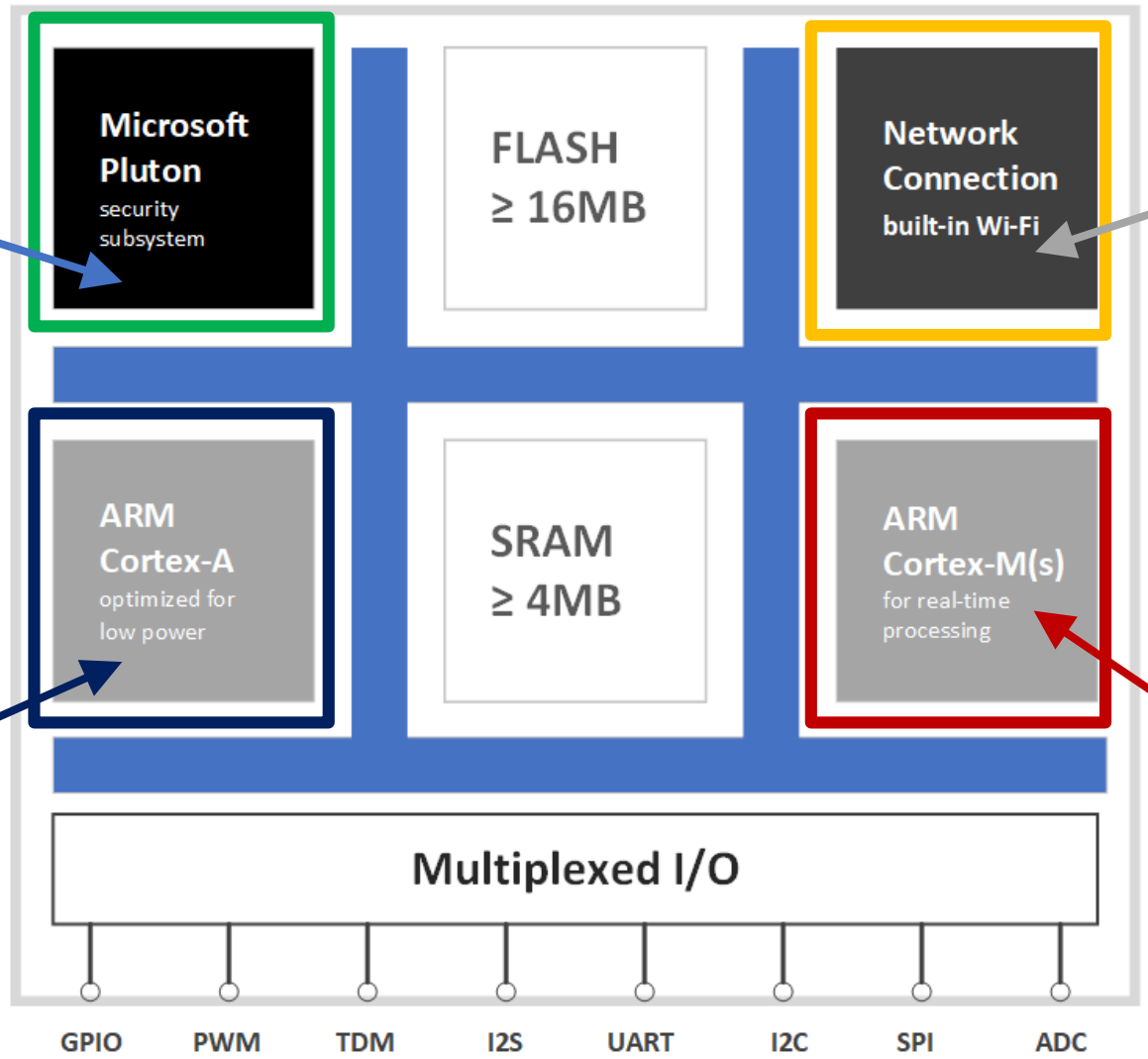
全てのAzure Sphereデバイスをガードする; D2D、D2C通信に対する信頼されたブローカー、新たな脅威の検知、デバイスセキュリティの更新を提供する。

マイクロソフト：セキュアデバイスに求める7つのセキュリティ属性



Azure Sphere MCU Architecture Overview

ハードウェアルートオブトラスト
暗号化エンジン
ランダム発生器
キーストア



ネットワークコネクティビティ

高度なアプリケーション用コア
(HL Core)

リアルタイムアプリケーション用コア
(RT Cores)

Azure Sphere MCU

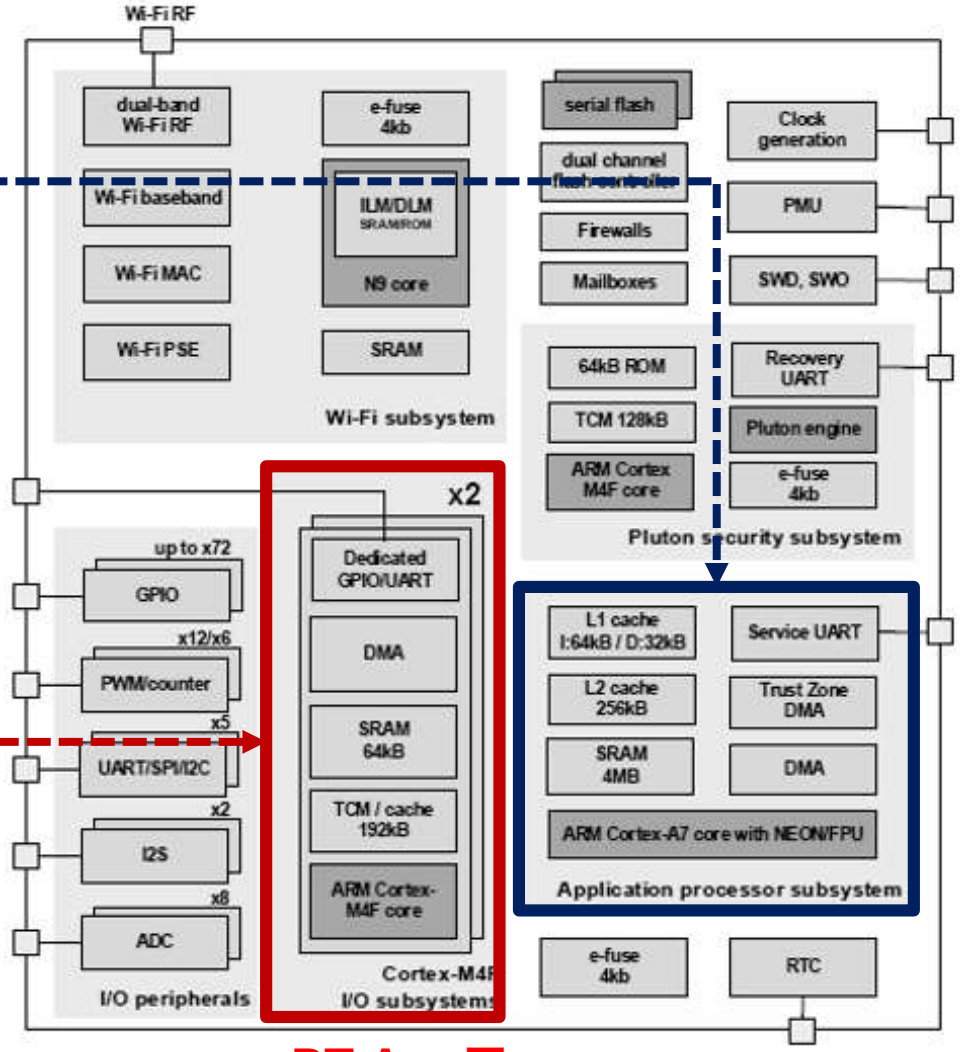
MediaTek MT3620

IoT向けハイレベルセキュリティ　シングルチップトライコアMCU

- POSIXアプリ用のCortex-A7 with NEON/FPU 500MHz
- I/Oアプリ用の2 x Cortex-M4F 192MHz
- 4MB RAM + 16MB FLASH(8MB Runtime + 8MB Backup)
- スタンダードMCUペリフェラル
  - GPIO(up to 76), PWM(12), Counter(6)
  - SPI(5 configurable),I2C(5 configurable),UART(5 configurable)
  - ADC(8 Channels, 12bit SAR, 2M sample/sec)
  - I2S(2 interfaces)
- N9 32bit RISCコア Wifi サブシステムコントローラ
- 802.11 b/g/n準拠
- 動作環境 3.3V、-40°C～85°C

Microsoft Plutonセキュリティサブシステム

- デバイス毎にユニークなランダムキーを生成
- エントロピー監視システム付きのハードウェア乱数発生器
- 暗号鍵はソフトウェアからアクセスが不可能なE-fusesに格納
- AES暗号化、SHAハッシュ、公開鍵アクセラレータ
- ECDSAによるセキュアブート
- ユニット間はファイアウォールによって保護



Azure Sphere MCU

NXP and Microsoft Bring Microsoft Azure Sphere Security to the Intelligent Edge with a New Energy-Efficient Processor

June 12, 2019 at 12:03 PM EDT

News Highlights:

- Companies partner to accelerate development of secure edge-to-cloud connected experiences
- Collaboration includes development of a new crossover applications processor in NXP's i.MX 8 series integrates Microsoft's Azure Sphere security architecture and Pluton Security Subsystem
- Customers will be able to harness the high-performance and energy efficiency of NXP's i.MX 8 applications processors combined with Microsoft's unequalled security and assurance provided by Azure Sphere certified chips

SANTA CLARA, Calif., June 12, 2019 (GLOBE NEWSWIRE) -- NXP Semiconductors N.V. (NASDAQ:NXPI) today announced collaboration with Microsoft to deliver a new Microsoft Azure Sphere certified crossover applications processor, as an extension to their popular i.MX 8 high-performance applications processor series. The collaboration will deliver a secure, ultra-efficient, intelligent embedded processor for edge nodes that seamlessly runs Azure Sphere's security platform while also providing multi-core heterogeneous computing, rich graphics experience, and low-power audio processing capabilities. Limited sampling of the product is planned to begin in Q4 2020.

Azure Sphere MCU

Qualcomm Technologies to Enhance Secure Cellular Connectivity Solutions with Microsoft Azure Sphere for the IoT

— Qualcomm Technologies to Deliver the First Cellular-Enabled Azure Sphere Certified Chips —

OCT 15, 2019 | BARCELONA | Qualcomm products mentioned within this press release are offered by Qualcomm Technologies, Inc. and/or its subsidiaries.

Qualcomm Technologies, Inc., a wholly owned subsidiary of Qualcomm Incorporated, announced today at its 5G Summit in Barcelona, Spain that it is developing the first cellular chip optimized and certified for Microsoft's Azure Sphere Internet of Things (IoT) operating system. Qualcomm Technologies' new Azure Sphere-certified chipset for IoT will include hardware-level security, come preconfigured with the Azure Sphere, and will automatically connect to Azure Sphere security cloud services.

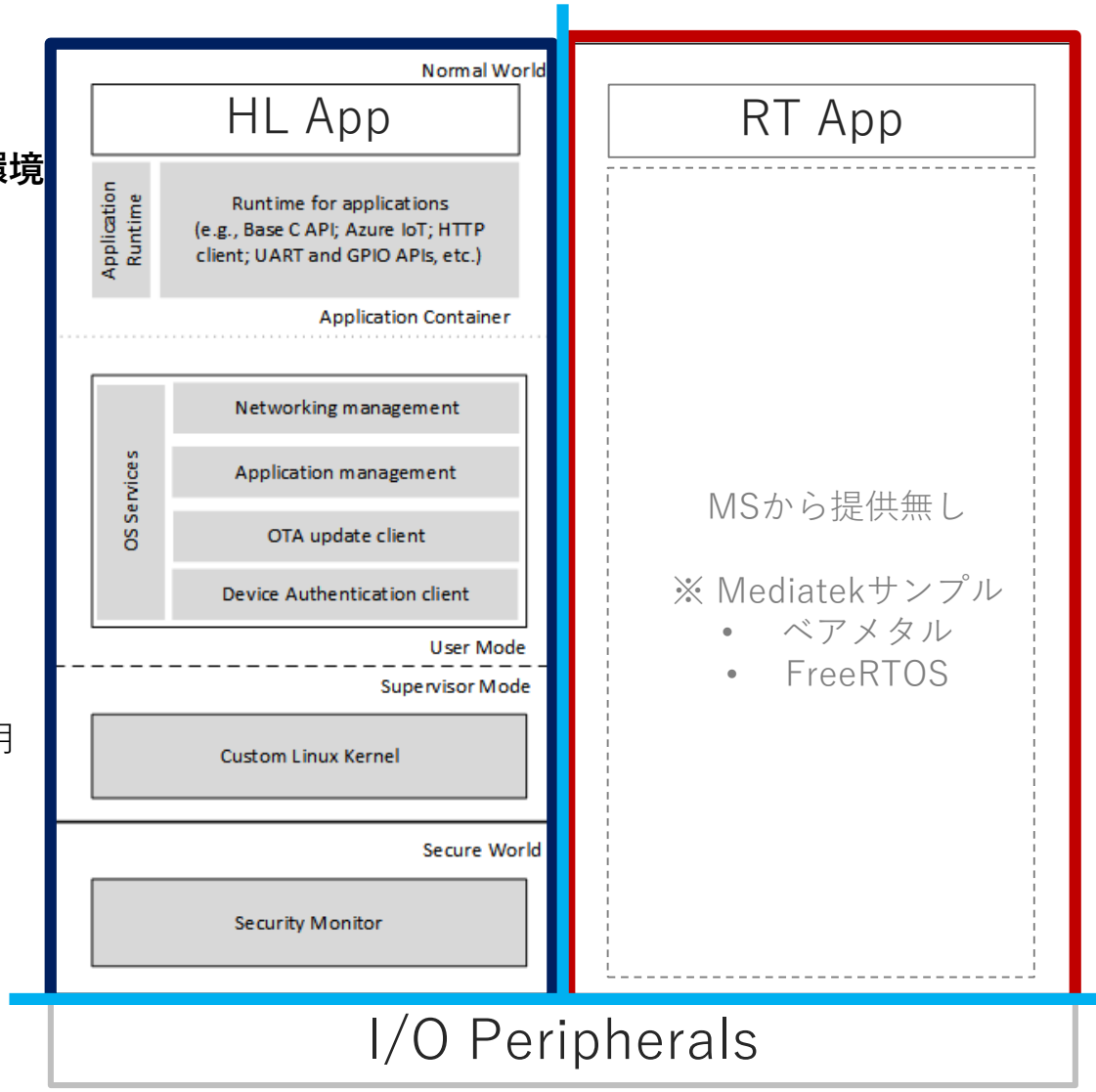
Azure Sphere OS Overview

ARM Trust Zoneによる仮想的に分離された実行環境

セキュリティモニター（セキュアワールド）
- セキュアブート
- ハードウェアファイアウォール
- 重要なシリコンコンポーネントを管理

カスタムLinuxカーネル（ノーマルワールド）
A7ハイレベルアプリケーション
- アプリケーションコンテナ内での実行
- POSIX準拠アプリケーションランタイム
- 1アプリケーションで256kBのメモリ使用

OSサービス
- ネットワークファイアウォール
- Wi-Fi認証
- アプリケーションデバッガなど



リアルタイム処理

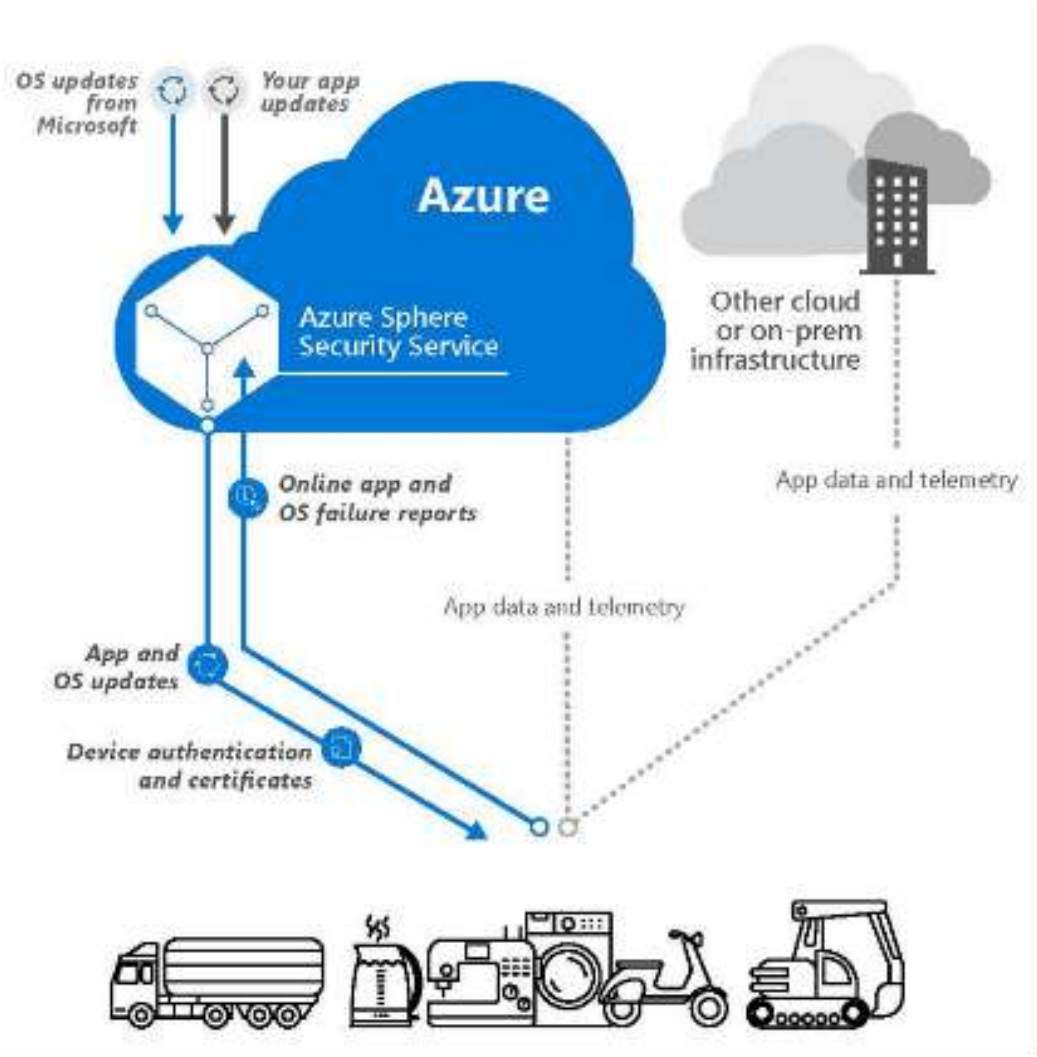
デバイスへの高速応答

I/Oファイアウォール

Azure Sphere Security Service Overview

全ての Azure Sphere搭載デバイスを接続・保護

- **保護する**
  デバイスと顧客を、全ての通信に対する証明書ベースの認証で保護する
- **検知する**
  新たに出現したセキュリティへの脅威を、デバイス上の障害を自動化されたプロセスで検知する
- **対応する**
  完全に自動化されたデバイス上のOSアップデート機能で脅威に対応する
- **許可する**
  Azure Sphere で強化されたデバイスへの、ソフトウェアの容易なアップデートを許可する



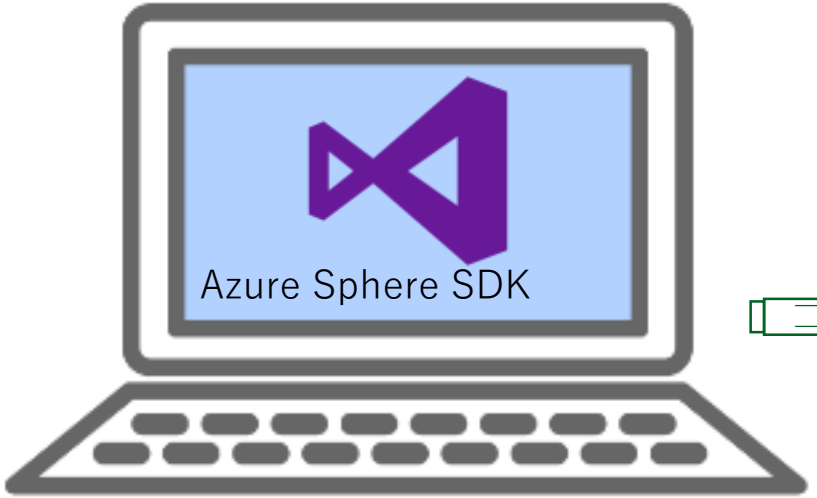
Azure Sphere Development / Visual Studio

プロユースの優れた統合開発環境
コーディングからサイドローディングとラインデバックまで一貫して可能
様々なバスを搭載した開発、検証ボード

Azure Sphere SDK for Visual Studio 2019
- アプリケーションテンプレート for C
- クロスコンパイラ
- コマンドラインツール

Azure Sphere OS が提供するAPIを通じてハードウェアにアクセス

アプリケーション開発のみに集中してアイデアをすばやく形に出来る

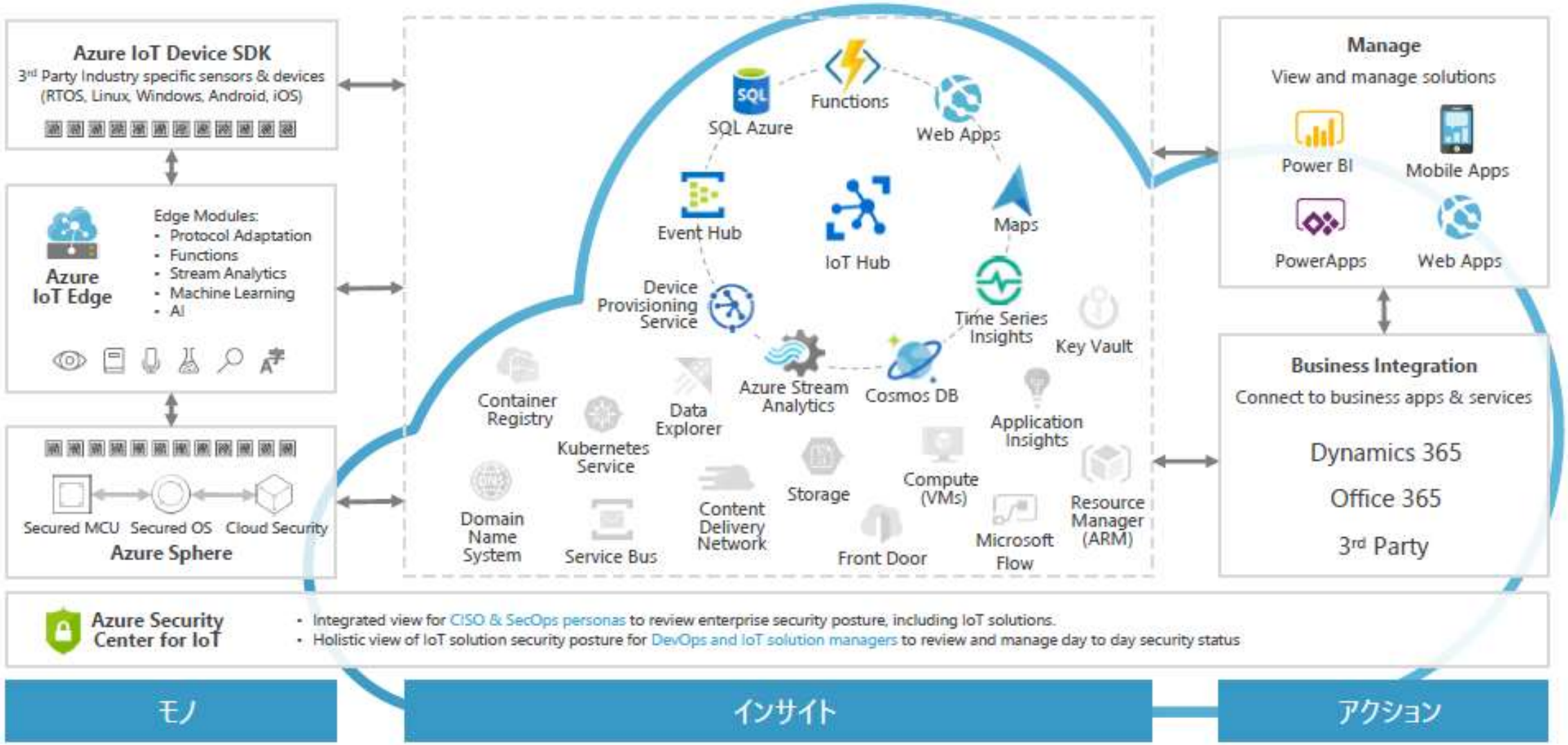


USB ケーブル

Azure Sphere MT3620 Development Kit

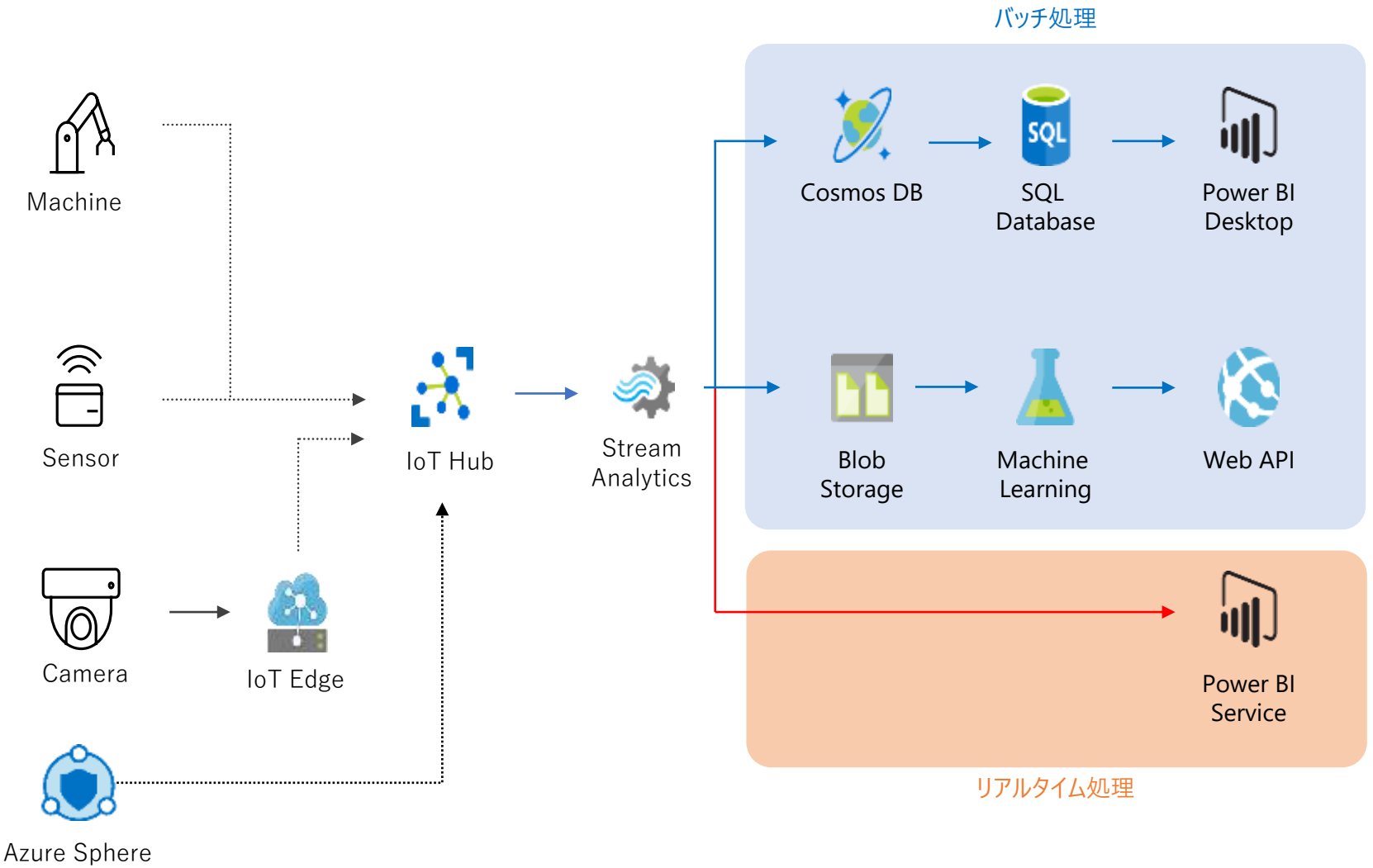
Windows 10 Anniversary Update以降

Internet of Things on Azureクラウド



Azure IoT アーキテクチャ例

Azure IoT は、バッチ処理からストリーム分析まで IoT データの用途に合わせたシステム構成ができます。また IoT Edge のエッジ技術によりエッジ側で処理するシステムを構築することもできます。



Azure IoT Central

Azure Sphere Hands-On

HL App編

はじめに

- アジェンダ

1. Azure Sphere 開発環境とデバイスの準備

2. Azure Sphere 組み込みアプリケーションの開発

   Lab1: Blinkサンプル（Lチカ）

3. Over The Air（OTA）でアプリケーションの配信

4. Azure Sphere IoTアプリケーションの開発

   Lab2: Azure IoTHubにつなげてAzure IoTHubの基本機能を理解する

   Lab3: Azure IoT Centralにつなげて Azure IoT Centralの基本機能を理解する

はじめる前の準備を確認しよう

補足説明

- PCとハンズオンキットの準備

以下の条件を満たしている必要があります。

✓ Windows 10 Anniversary Update 以降を実行している PC、または Ubuntu 18.04 LTS を実行している Linux マシン
✓ 1つ以上の空きがあるUSBポート
✓ Azure Sphere 開発ボード
✓ Visual Studio 2019 (Community/Professional/Enterprise) またはVisual Studio Code

- Azure Sphere用アカウントの準備
  ✓ マイクロソフトアカウントが必要

- Azure IoT （DPS/IoTHub/IoT Central）用アカウントの準備
  ✓ 有効なAzure Subscriptionが必要

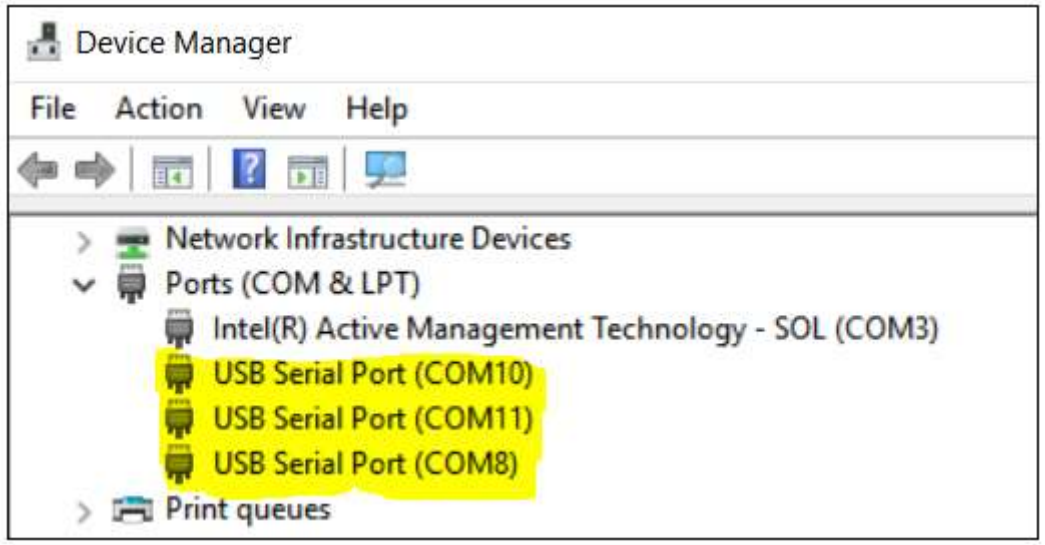
1. Azure Sphere 開発環境の準備

開発ボードの準備をしよう

PC

1. 開発ボードとPCをUSBケーブルで接続します。接続するとドライバーが自動的にダウンロードされてインストールされます。 ドライバーが自動的にインストールされない場合は、[デバイス マネージャー] でデバイス名を右クリックし、[ドライバーの更新] を選択します。

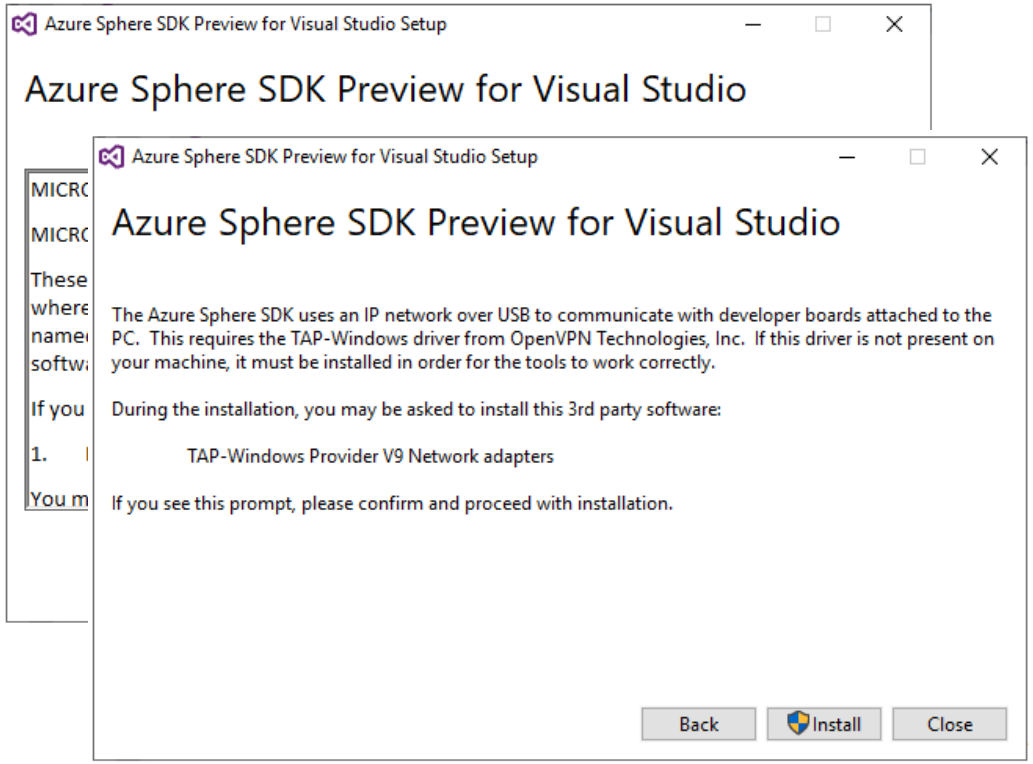
※ 初めて開発ボードを接続する場合には必ずSDKをインストールする前にこの手順を実行します。



2. Azure Sphere SDK preview for Visual Studioをダウンロードします。インストーラはここからダウンロードできます。

※ 本ハンズオンでは講師から配布するインストーラを使用してください。

3. ダウンロードしたAzure_Sphere_SDK_Preview_for_Visual_Studio.exeを実行して SDK をインストールします。



4. インストールが完了するとTAP Windows アダプタが”Azure Sphere”というネットワークデバイスが追加されデバッガ用に”192.168.35.1”のIPアドレスが振られます。

コマンドライン

手動でOSを更新しよう

デバイスで実行されているSphere OSのバージョン18.11以降ではインターネットに接続されると自動的に更新済みバージョンのOSをOTAで受信して更新します。

以降24時間毎に更新されたSphere OSがないか確認され更新の必要があれば自動的に受信して更新されます。これはアプリケーションにより最大24時間、延期できます。

デバイスのSphere OSが18.11よりも古い場合、もしくは手動で更新したい場合には以下の手順で更新できます。

1. 開発ボードがPCが接続されていることを確認してください。

2. Azure Sphere Developer Command Promptを開きます。

3. 最新のAzure Sphere SDK preview for Visual Studioがインストールされていることを確認します。

4. 現在のOSバージョンを確認します。

```
> azsphere device show-os-version
```

5. 次のコマンドでリカバリを実行します。

```
> azsphere device recover
```

```
Starting device recovery. Please note that this may take up to 10 minutes.
Board found. Sending recovery bootloader.
Erasing flash.
Sending images.
Sending image 1 of 16.
Sending image 2 of 16.
. . .
Sending image 16 of 16.
Finished writing images; rebooting board.
Device ID: <GUID>
Device recovered successfully.
Command completed successfully in 00:02:37.3011134.
```

Azure Sphereを要求（Claim）しよう

コマンドライン

Azure SphereデバイスはAzure Sphereテナントから要求（Claim）される必要があります。デバイスを要求するとデバイスに一意に割り当てられたデバイスIDがAzure Sphereテナントと関連付けられます。Azure Sphere Security ServiceはデバイスIDを使ってデバイスを識別、認証します。

一つの組織に一つのAzure Sphereテナントを作成することが推奨されています。

重要！
これは一度限りのオペレーションですので譲渡やなんらかの理由で他のテナントへと移動したいとしても出来ません。

1. Azure Sphere Developer Command Promptを開きます。

2. Azure Sphere テナントへサインインします。

```
> azsphere login
```

サインインすると利用できるテナントが表示されます。
本ハンズオンの場合には”AvnetJapan”というテナントが表示されます。

3. Azure Sphereデバイスを要求（Claim）します。

```
> azsphere device claim
```

Claimを実行すると以下のような結果が表示されます。

```
Claiming device.
Claiming attached device ID '<device ID>' into tenant ID
'd343c263-4aa3-4558-adbb-d3fc34631800'.
Successfully claimed device ID '<device ID>' into tenant ID
'd343c263-4aa3-4558-adbb-d3fc34631800'.
Command completed successfully in 00:00:05.5459143.
```

Azure SphereをWi-Fiに接続しよう

コマンドライン

1. 開発ボードがPCが接続されていることを確認してください。

2. Wi-Fiのアクセスポイントを設定します。

```
> azsphere device wifi add --ssid SSID名 --psk ネットワークキー --targeted-scan
```

現在のところ接続できるアクセスポイントは802.11 b/g/n およびWPA/WPA2がサポートされます。

秘匿されたSSID、ブロードキャストしていないSSIDの場合は指定必要

3. 現在の接続を確認します。

```
> azsphere device wifi show-status
```

```
SSID : SphereDemo
Configuration state : enabled
Connection state : connected
Security state : psk
Frequency : 2442
Mode : station
Key management : WPA2-PSK
WPA State : COMPLETED
IP Address : 192.168.1.15
MAC Address : 52:cf:ff:3a:76:1b
Command completed successfully in 00:00:01.3976308.
```

ハンズオン環境のWi-Fi
SSID名: SphereDemo
ネットワークキー: SphereDemo

デバック状態とフィールド状態とは

補足説明

既定では、アプリケーションのデバッグは許可されていませんのでazsphere device enable-development コマンドを実行し、アプリケーションのOTAを無効化してVisual Studioからのデバックを有効にします。

反対にOTAでアプリケーションを配信する場合にはazsphere device enable-cloud-testコマンドを実行し、デバック機能を無効にしてOTAによるアプリケーション配信を有効にします。

azsphere device enable-development

appdevelopment機能の適用
アプリケーションのOTA無効
デバイスのリブート

azsphere device enable-cloud-test

appdevelopment機能の削除
サイドロードされたアプリケーションの削除
アプリケーションのOTA有効

Azure Sphereをデバック状態にする

コマンドライン

1. 開発ボードがPCが接続されていてかつインターネットに接続されている
   ことを確認してください。

2. Azure Sphereのデバック機能（appdevelopment）を有効にします。

```
> azsphere device enable-development
```

実行すると以下のような表示がされてデバイスがデバック用に準備されます。

```
Getting device capability configuration for application development.
Downloading device capability configuration for device ID '<device ID>'.
Successfully downloaded device capability configuration.
Successfully wrote device capability configuration file 'C:¥Users¥user¥AppData¥Local¥Temp¥tmpD732.tmp'.
Setting device group ID 'a6df7013-c7c2-4764-8424-00cbacb431e5' for device with ID '<device ID>'.
Successfully disabled over-the-air updates.
Enabling application development capability on attached device.
Applying device capability configuration to device.
Successfully applied device capability configuration to device.
The device is rebooting.
Installing debugging server to device.
Deploying 'C:¥Program Files (x86)¥Microsoft Azure Sphere SDK¥DebugTools¥gdbserver.imagepackage' to the attached
device.
Image package 'C:¥Program Files (x86)¥Microsoft Azure Sphere SDK¥DebugTools¥gdbserver.imagepackage' has been
deployed to the attached device.
Application development capability enabled.
Successfully set up device '<device ID>' for application development, and disabled over-the-air updates.
Command completed successfully in 00:00:38.3299276.
```

2. Azure Sphere 組込みアプリケーションの開発

Lab1の概要と目的

- 概要

このラボではマイコンをつかった開発の基本としてAzure Sphere開発ボードに搭載されたLEDの点滅を行うプログラムを開発します。

- 目的

Visual Studioの基本的な使い方、コーディングと実行、デバック方法などを体験します。

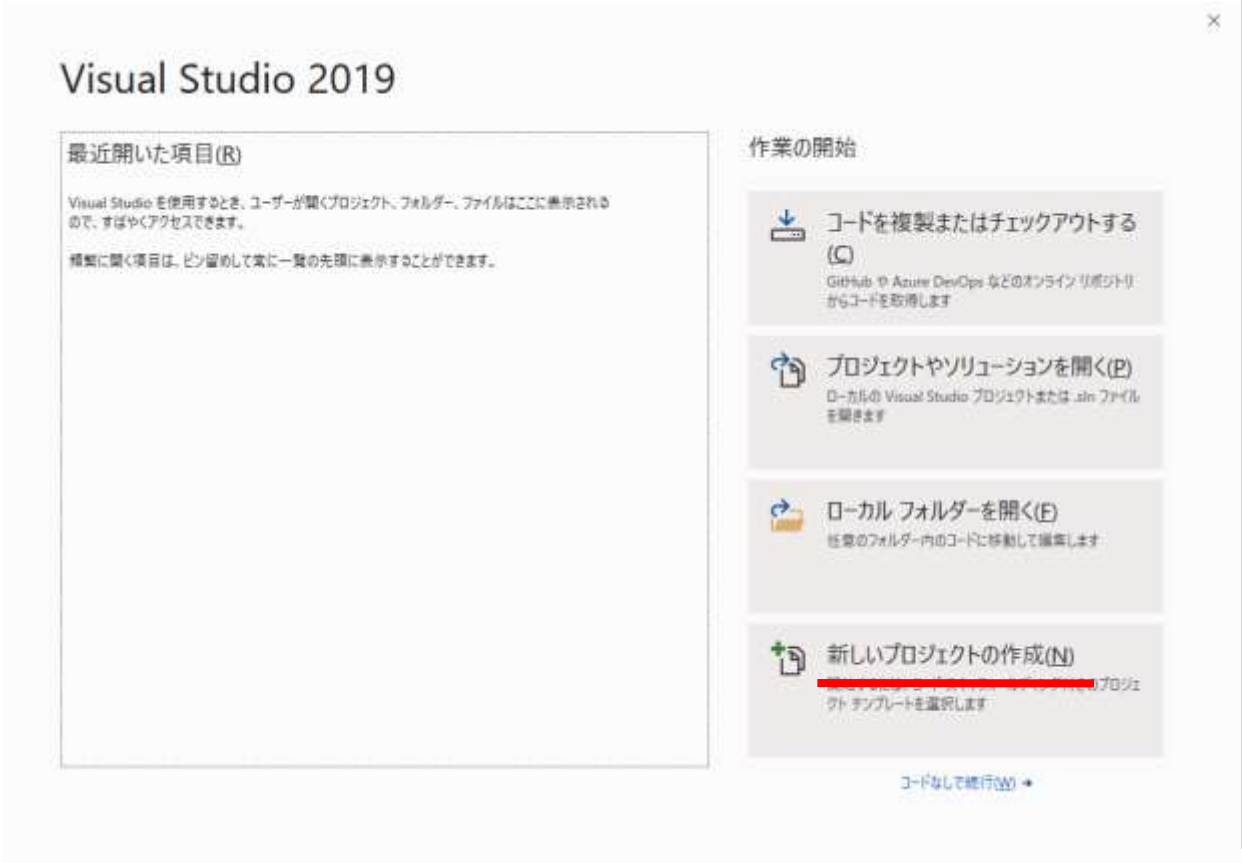
またAzure Sphereの開発環境、アプリケーションマニュフェストの仕組み、提供されるアプリケーションランタイムについて理解します。

Lab1: Blinkサンプルを実行

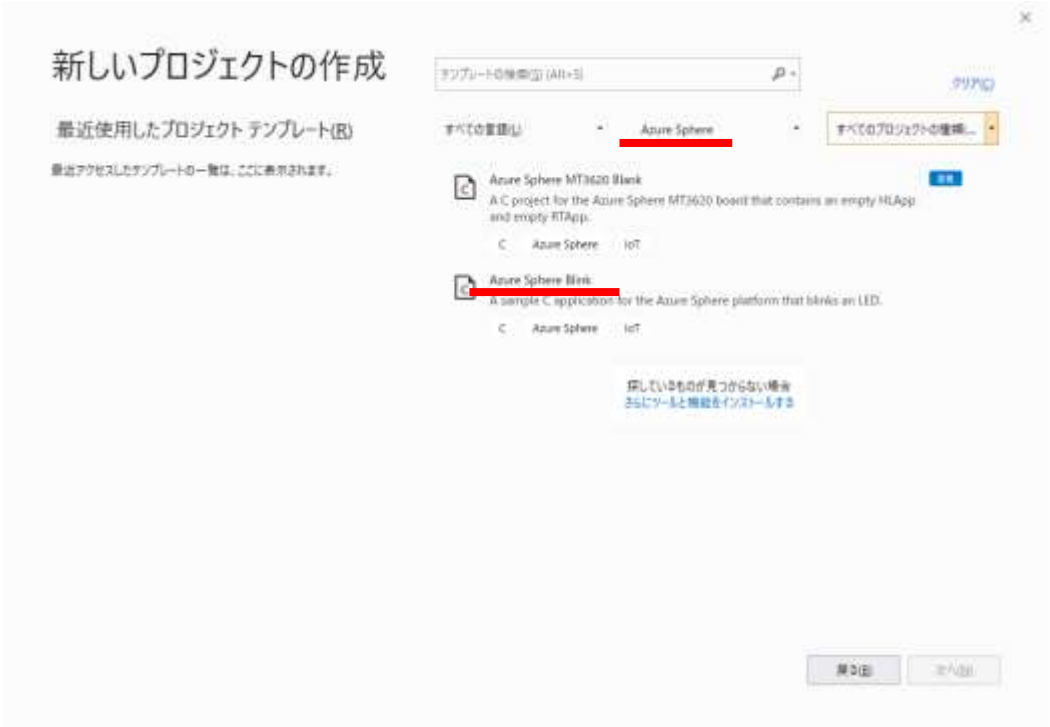
Visual Studio

Visual StudioからLEDサンプルを実行

1. Visual Studio 2019を起動して**新しいプロジェクトの作成** をクリックします。



2. **プラットフォーム**から**Azure Sphere** を選択して**Azure Sphere Blink**を選択して次へクリックします。



Lab1: Blinkサンプルを実行しよう

Visual Studio

Visual StudioからLEDサンプルを実行

1. 項目はデフォルトのままで**作成** をクリックします。

2. GDB Debugger(HL Core)を選択して実行ボタンを押すと開発ボードのLED1が点滅を開始します。これはアプリケーションがWi-Fi経由の配信ではなくサイドローディングされて実行されたことを意味します。

3. Visual Studio 2017の基本的な使い方についてはここから学ぶことができます。

アプリケーションマニュフェストとは

補足説明

アプリケーションマニュフェストにはアプリケーションで使用されるリソースが記述されています。Capabilitiesセクションに列記されていないリソースはアプリケーションから使用することができません。アプリケーションマニュフェストは app_manifest.jsonというファイルですべてのアプリケーションに含まれます。

https://docs.microsoft.com/ja-JP/azure-sphere/app-development/app-manifest

アプリケーションマニュフェストの例

```
{
   "SchemaVersion": 1,
   "Name": "MyTestApp",
   "ComponentId": "072c9364-61d4-4303-86e0-b0f883c7ada2",
   "EntryPoint": "/bin/app",
   "CmdArgs": ["-m", "262144", "-t", "1"],
   "Capabilities": {
      "AllowedConnections" : [ "my-hub.example.net","contoso.azure-devices.net","global.azure-devices-provisioning.net" ],
     "DeviceAuthentication": "77304f1f-9530-4157-8598-30bc1f3d66f0",
      "Gpio": [ 15, 16, 17, 12 ],
      "I2cMaster": [ "ISU2" ],
      "MutableStorage" : { "SizeKB": 64, },
      "SpiMaster": [ "ISU1" ],
      "SystemTime" : true,
      "Uart": [ "ISU0" ],
      "WifiConfig" : true
   }
}
```

Visual Studio

Lab1: LEDの色を変えてみよう

- 問題

LEDの色は割り当てられたGPIOピンの出力を”Low”にすることで点灯させます。
この色を変更してみましょう。

```
22      // the extensible samples here: https://github.com/Azure/azure-sphere-samples
23      Log_Debug(
24          "¥nVisit https://github.com/Azure/azure-sphere-samples for extensible samples to use
25          "starting point for full applications.¥n");
26
27      // Change the GPIO number here and in app_manifest.json if required by your hardware.
28      int fd = GPIO_OpenAsOutput(9, GPIO_OutputMode_PushPull, GPIO_Value_High);
29      if (fd < 0) {
30          Log_Debug(
31              "Error opening GPIO: %s (%d). Check that app_manifest.json includes the GPIO use
32              strerror(errno), errno);
```

ハードウェアユーザーガイド
https://docs.microsoft.com/ja-jp/azure-sphere/hardware/mt3620-user-guide

Azure Sphere Application Runtime

補足説明

Azure Sphere SDKに含まれているアプリケーションランタイムは以下のものがあります。

- 基本API https://docs.microsoft.com/ja-JP/azure-sphere/app-development/baseapis
  - ✓POSIXベース C 開発ライブラリ ・・・ カスタムされたmusl libcライブラリ
  - ✓Azure IoT C SDKライブラリ ・・・ Azure IoTへ接続するためのライブラリ
  - ✓curl ライブラリ ・・・ HTTP/HTTPS転送ライブラリ
  - ✓TLS utilities library ・・・ TLSを介した相互認証のためのライブラリ

for Azure IoT

for Web Service

- Applibs https://docs.microsoft.com/ja-JP/azure-sphere/reference/applibs-reference/api-overview
  - ✓デバイス固有API ・・・ ハードウェアなどを操作するためのAPI

基本APIはAzure Sphere用にいくつかの機能が除外されたサブセットになっています。

Azure Sphere Samples on GitHub

補足説明

Azure Sphere で動かすことができる公式アプリケーションサンプルはGitHub（世界中のデベロッパーが自身の作品を公開することができるウェブサービス）で公開されています。

GitHub

https://github.com/Azure/azure-sphere-samples

その他のAzure Sphere サンプル

補足説明

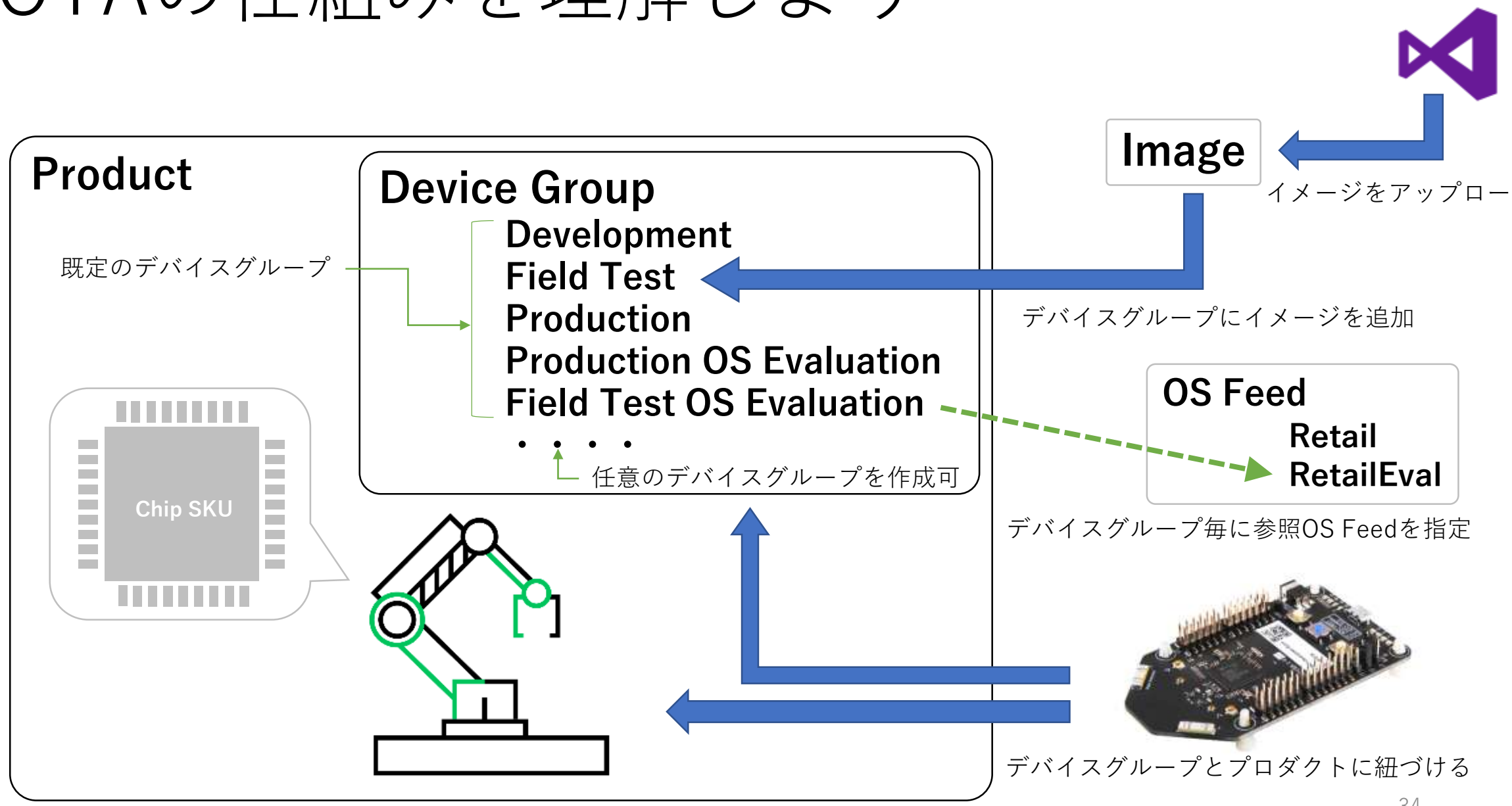
hackstar.ioではハードウェアエンジニア向けに様々な作品が投稿されています。

https://www.hackster.io/microsoft/products/azure-sphere-mt3620-starter-kit

3. Azure Sphere アプリケーションの展開

補足説明

OTAの仕組みを理解しよう



Azure Sphere OSのフィード

補足説明

Azure Sphere OSフィードは２つあります。

https://docs.microsoft.com/ja-JP/azure-sphere/deployment/deployment-microsoft-feeds

- **Retail Azure Sphere OS Feed** (3369f0e1-dedf-49ec-a602-2aa98669fd61)

実稼働環境用のOSが配信されるフィードでエンドユーザーに提供するアプリケーションはこのOSフィードを参照するフィードを使う必要があります。

- **Retail Evaluation Azure Sphere OS Feed** (82bacf85-990d-4023-91c5-c6694a9fa5b4)

アプリケーションの検証用フィードで、Retail Azure Sphere OS Feedがリリースされる14日前にリリースされます。アプリケーションの互換性に問題がないかこの期間で検証することができます。

既定のデバイスグループ

補足説明

- **Development**: アプリケーション開発用。
  OSフィードはRetail Evaluation OS / アプリケーションの更新は無効

- **Field Test**: デバイスのテスト用。
  OSフィードはRetail OS / アプリケーション更新は有効

- **Production**: 運用環境のデバイス用。
  OSフィードはRetail OS / アプリケーション更新は有効

- **Field Test OS Evaluation**: OSと新しいアプリケーションの互換性検証用。
  OSフィードはRetail Evaluation OS / アプリケーション更新は有効

- **Production OS Evaluation**: OS と運用環境アプリケーションの互換性検証用。
  OSフィードはRetail Evaluation OS / アプリケーション更新は有効

プロダクトを作成しよう

コマンドライン

> azsphere product create --name <ProductName>

任意のプロダクト名を指定

Image

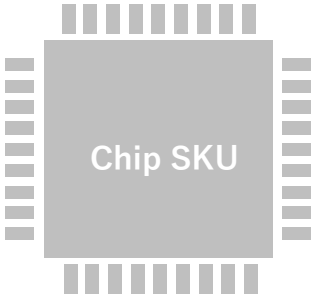
Product

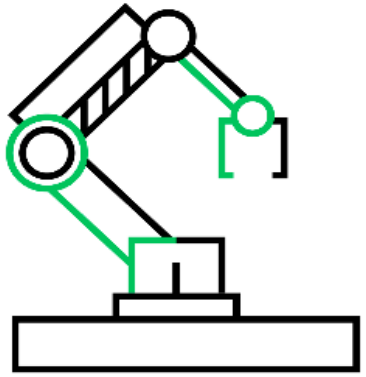
Device Group

Development
Field Test
Production
Production OS Evaluation
Field Test OS Evaluation

OS Feed

Retail
RetailEval

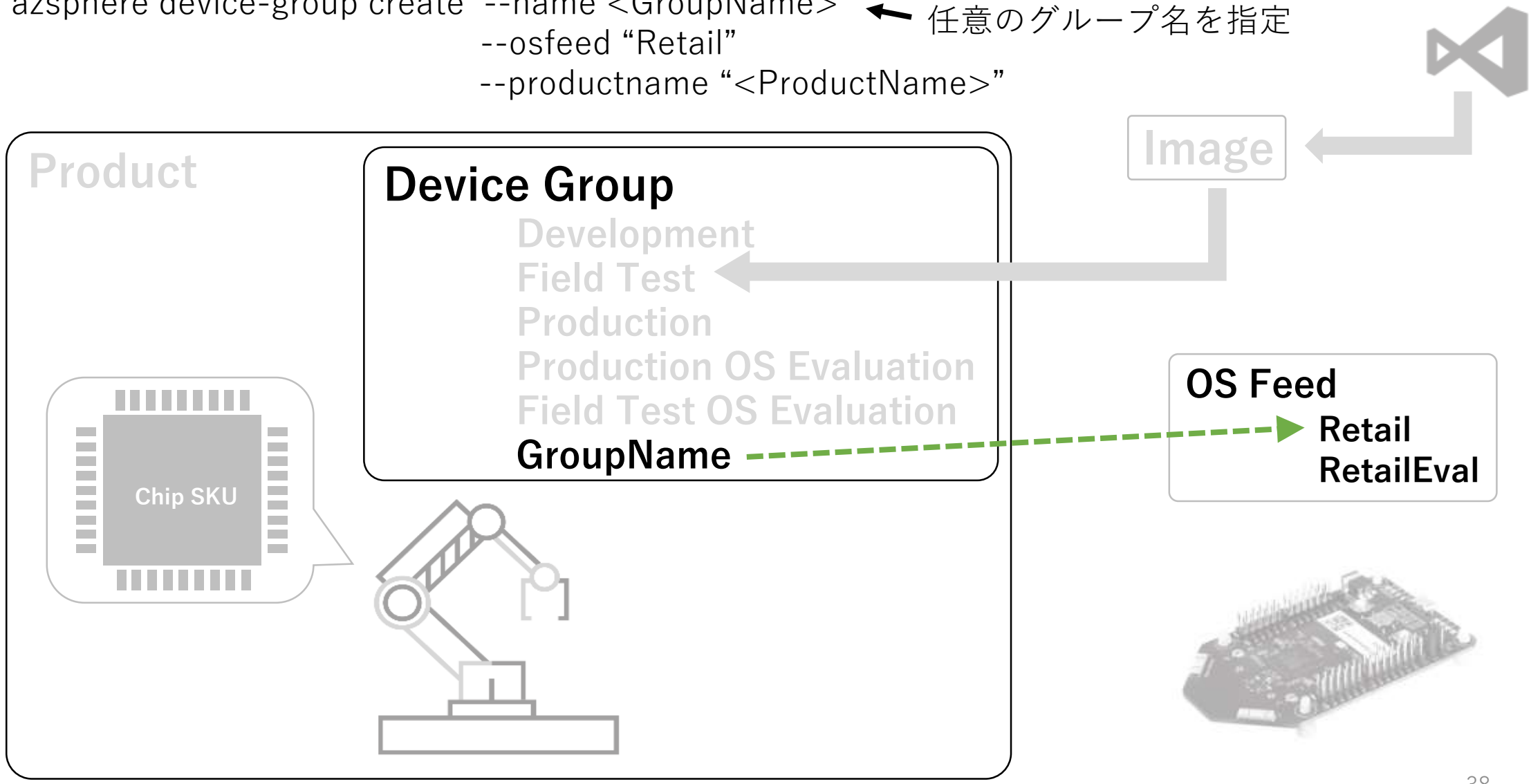




デバイスグループを作成しよう

コマンドライン

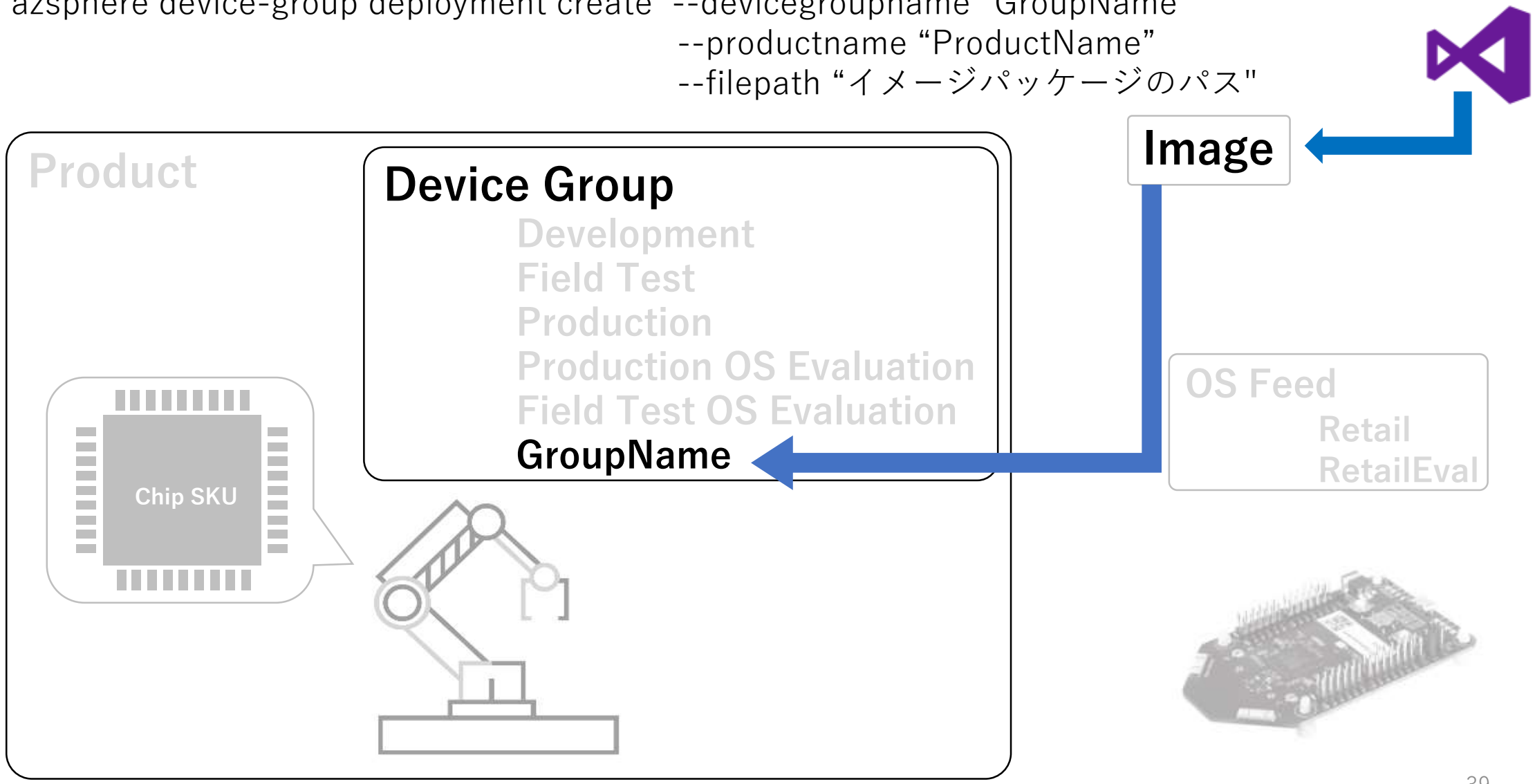
> azsphere device-group create --name <GroupName> ← 任意のグループ名を指定
--osfeed “Retail”
--productname “<ProductName>”



アプリケーションを追加しよう

コマンドライン

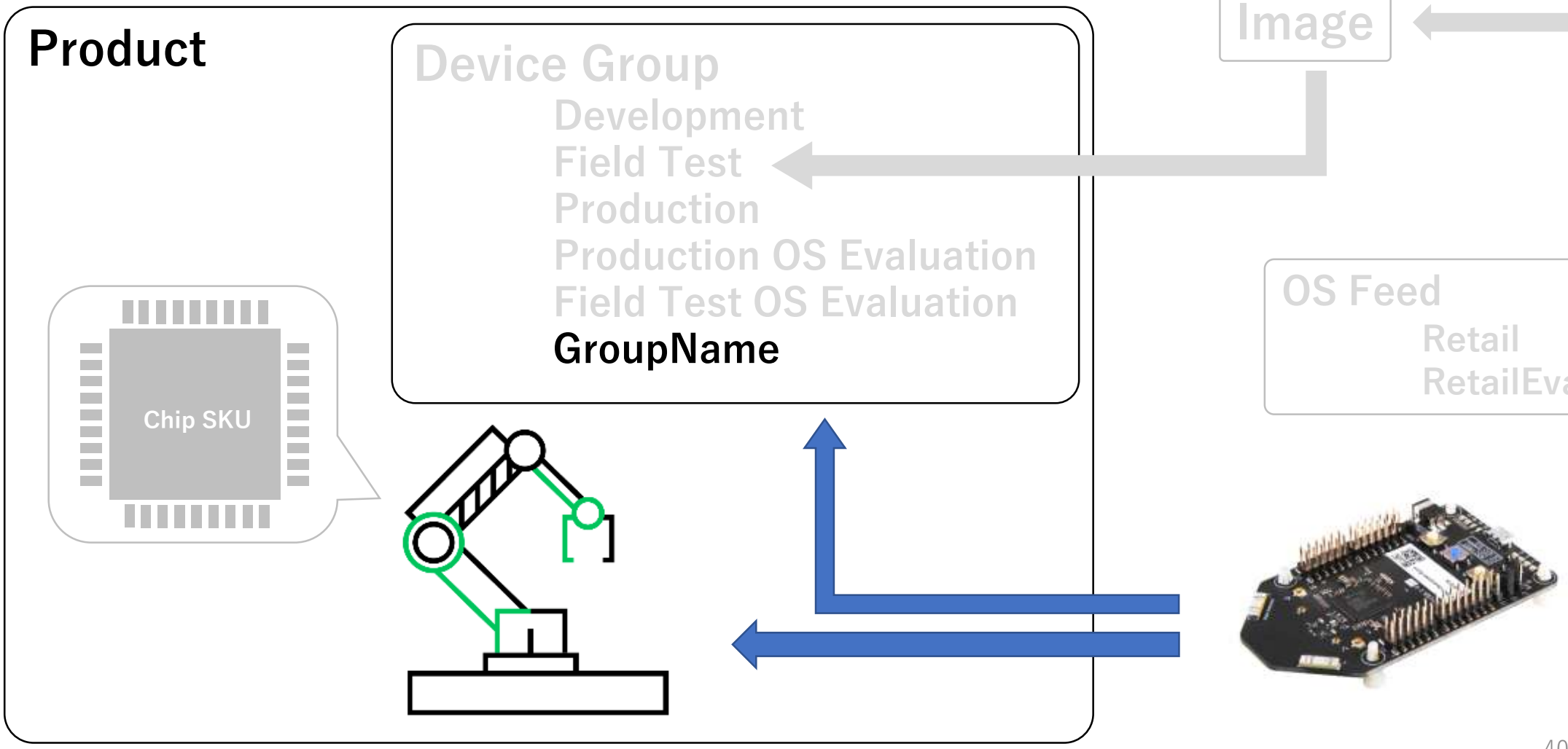
> azsphere device-group deployment create --devicegroupname “GroupName”
--productname “ProductName”
--filepath “イメージパッケージのパス"



補足説明

アプリケーションを配信しよう

> azsphere device enable-cloud-test --devicegroupname “GroupName”
> --productname “ProductName”



OTAの設定を確認する

コマンドライン

- デバイスに既にインストールされているアプリケーション

  > azsphere device image list-installed

- OTAでデバイスにインストールされるべきアプリケーション

  > azsphere device image list-targeted

4. Azure Sphere IoTアプリケーションの開発

Lab2の概要と目的

- 概要

このラボはAzure SphereをAzure IoTへ接続します。

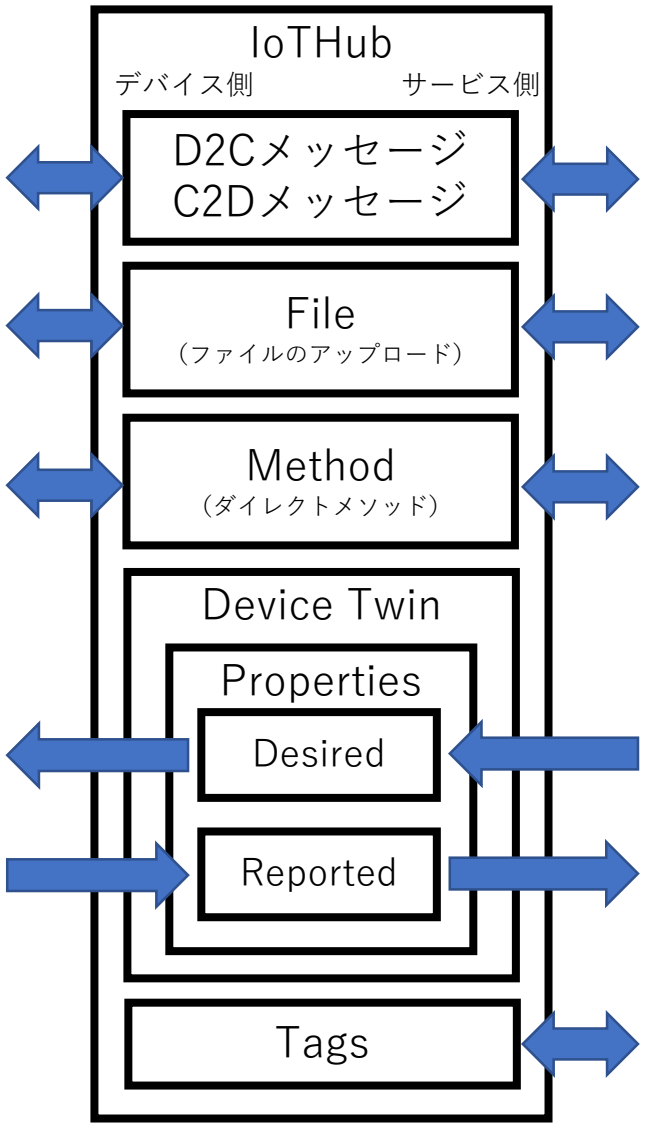
またAzure IoTHubの基本的な機能であるD2CおよびC2Dのテレメトリとデバイスツインの動作を確認します。

- 目的

Azure IoT Device SDKを使ったアプリケーションの動作を理解します。またIoTHubのデータを確認するDevice Explorerの使い方も理解します。

Lab2: IoTHubを理解しよう

補足説明



Azureクラウドと IoTアプリケーション/デバイスとの双方向通信のハブ

主な機能

・デバイスとクラウドとのメッセージ送受信（テレメトリ）

・クラウドからデバイス上のメソッド呼び出し（ダイレクトメソッド）

・デバイスの管理（デバイスツイン）

・デバイスごとの認証

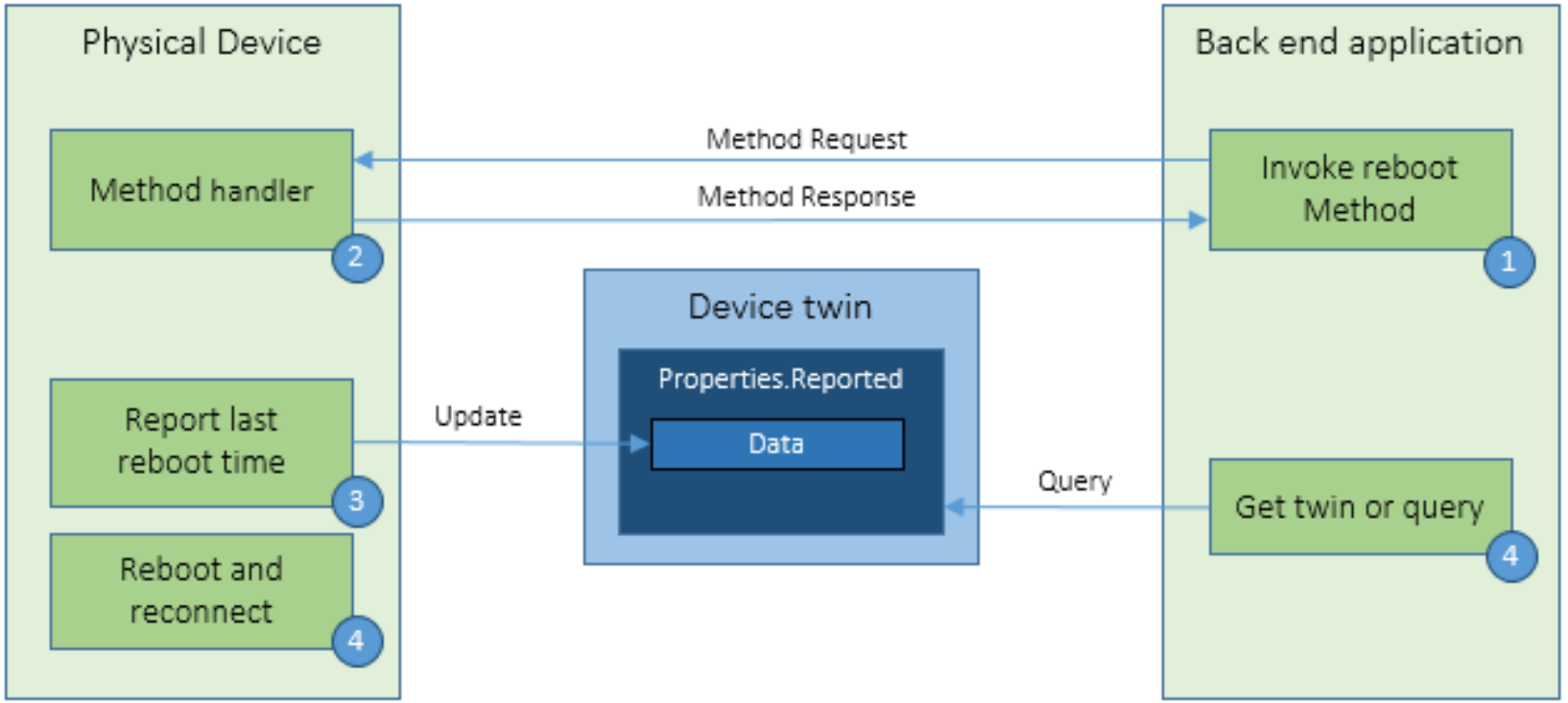
・ファイルアップロード

・メッセージのルーティング

Lab2: IoTHubでデバイス管理の例　その１

補足説明

デバイスの再起動

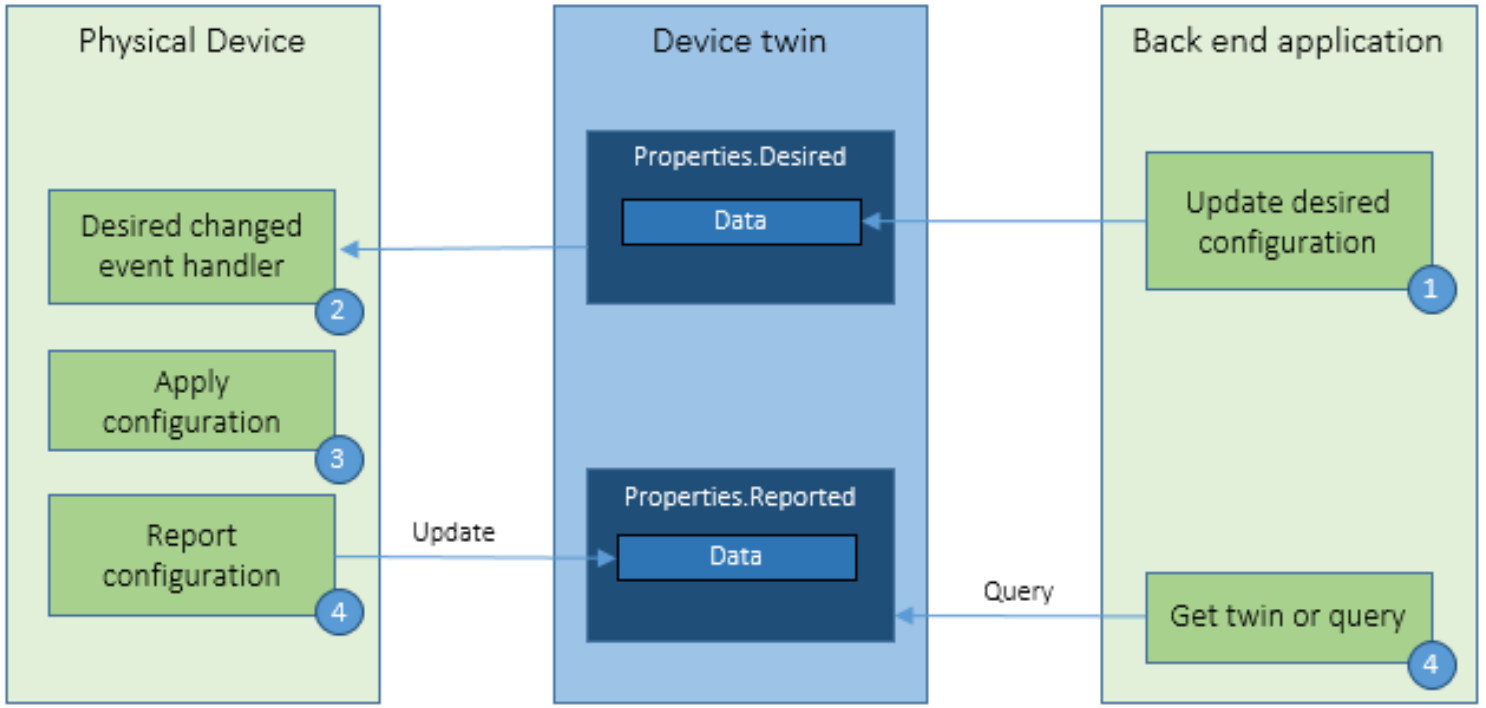


バックエンドアプリが再起動の開始をダイレクトメソッドを呼び出すことで通知して
デバイスアプリはReportedプロパティを使って再起動の状況を更新します。

Lab2: IoTHubでデバイス管理の例　その２

補足説明

ソフトウェアの構成

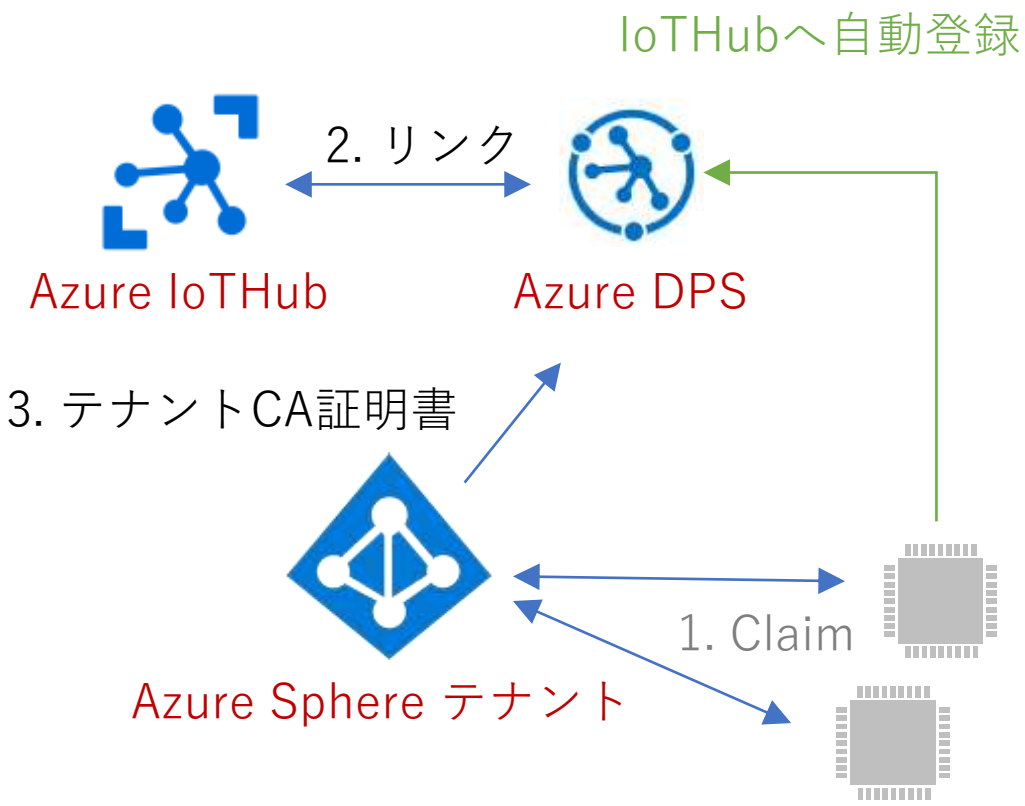


バックエンドアプリはデバイスアプリの構成をDesiredプロパティを使用して通知します。
デバイスアプリはReportedプロパティを使って構成の状況を報告します。

Lab2: Device Provisioning Serviceって？

補足説明

IoTHubへゼロタッチでデバイスをプロビジョニングするIoTHubのヘルパーサービスです。
Azure Sphereと連携するIoTHubを設定するとDPSを使ってAzure Sphereデバイスを
IoTHubへ自動的にプロビジョニングすることができます。



1. デバイスはClaimをすることでAzure Sphere テナントに関連付けがされている
2. Azure DPSとAzure IoTHubを資格情報を使用してリンクする。
3. Azure SphereテナントCA証明書をDPSへアップロードして秘密鍵の保有を証明することで証明書を登録します。

この設定をすることでデバイスはAzure DPSに接続する時にデバイス証明書とテナントCA証明書を送信して受け取ったDPSは検証済み証明書を元にIoTHubに自動的に登録します。

Lab2: Azure IoTの準備をしよう

Azureポータル

コマンドライン

- Azure Sphere 用の Azure IoT ハブを設定する

https://docs.microsoft.com/ja-jp/azure-sphere/app-development/setup-iot-hub

手順１．Azure PortalでIoTHubとDPSを作成してリンクする

手順２．Azure Sphere テナントCA証明書をダウンロードする

手順３．Azure Sphere テナントCA証明書をDPSへアップロードして確認コードを生成する

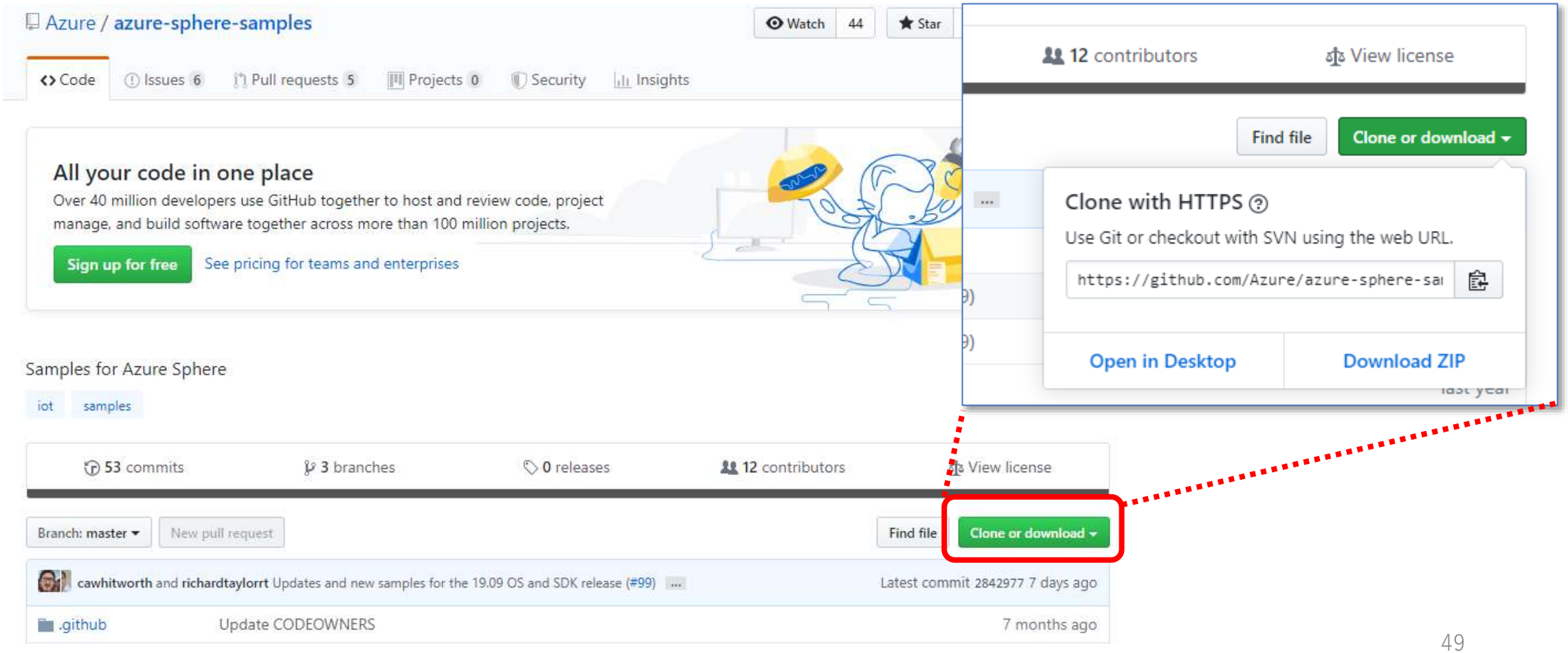
手順４．Azure Sphere テナントCA証明書を検証する

手順５．検証したAzure Sphere テナントCA証明書を使って登録グループを作成する

Lab2: GitHubからサンプルコードをダウンロードしよう

PC

1. GitHubのAzure Sphere Samplesのdownloadリンクからzipファイルでダウンロードします。



Lab2: Azure IoT Hub サンプルを実行しよう

Visual Studio
Azureポータル
コマンドライン

1. Visual Studio 2019を起動して**ファイル>開く>CMake**からGitHubからダウンロードした**AzureIoT/CMakeLists.txt**を開きます。

2. ソリューションエクスプローラからapp_manifest.jsonを開きます。

```
{
  "SchemaVersion": 1,
  "Name": "AzureIoT",
  "ComponentId": "819255ff-8640-41fd-aea7-f85d34c491d5",
  "EntryPoint": "/bin/app",
  "CmdArgs": [],
  "Capabilities": {
    "AllowedConnections": [ "global.azure-devices-provisioning.net" ],
    "Gpio": [ "$SAMPLE_BUTTON_1", "$SAMPLE_BUTTON_2", "$SAMPLE_LED" ],
    "DeviceAuthentication": "00000000-0000-0000-0000-000000000000"
  },
  "ApplicationType": "Default"
}
```

3. アプリケーションの接続先情報を修正します。

・AllowConnctions ⇒ DPSのGlobal device endpointとIoTHubのHostName
・CmgArmgs ⇒ DPSのScopeID
・DeviceAuthentication ⇒ Azure Sphere TenantID

- IoTHub (Azure Portal上からIoTHubの概要で確認)

Hostname : labuser20hub.azure-devices.net
Pricing and scale tier : S1 - Standard
Number of IoT Hub units : 1

- DPS (Azure Portal上からDPSの概要で確認)

Service endpoint : labuser20dps.azure-devices-provisioning.net
Global device endpoint : global.azure-devices-provisioning.net
ID Scope : 0ne0009E740
Pricing and scale tier : S1

- TenantID (Azure Sphere コマンドラインから確認)

```
> azsphere tenant list
```

Lab2: Azure IoT Hub サンプルを実行しよう

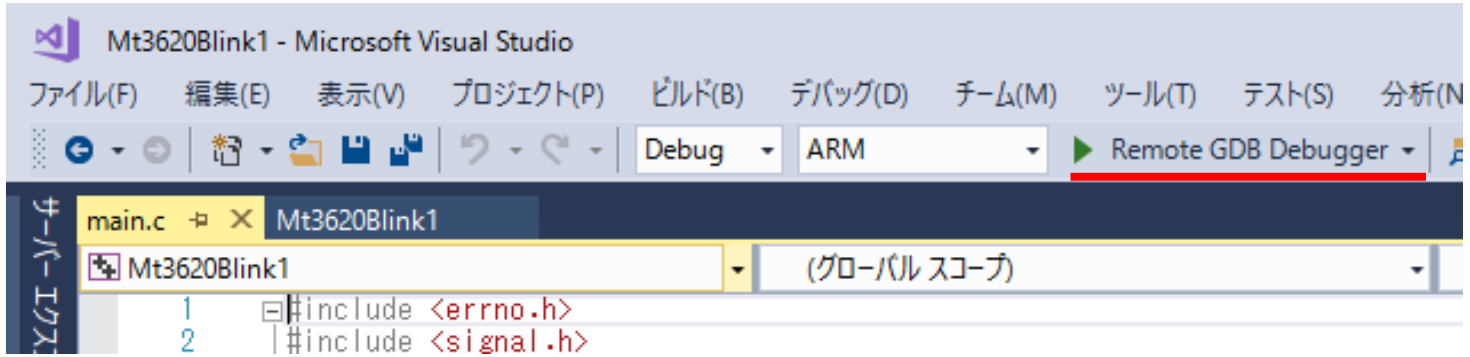
Visual Studio

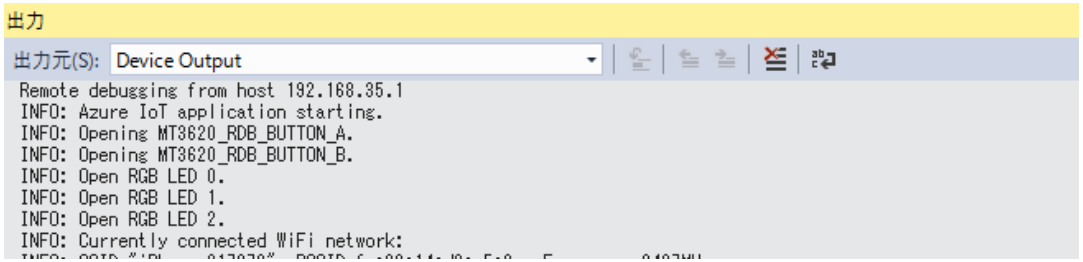
このサンプルアプリケーションは次のような機能を持っています。

Azure IoT Hubとの通信
- IoTHubへ温度(シミュレーションデータ)を送る
- ボタンAを押すと”ButtonPressイベントを送信する
- ボタンBを押すと“Orientation”ステート(シミュレーションデータ)を送る

デバイスのコントロール
- LEDをコントロールする。

1. デバックを開始するとVisual Studioの出力ウィンドウに様々な情報が表示されます。正常に開発キットと通信が開始されるとアプリケーションが実行されます。





Lab2: Azure IoT Hub サンプルを実行しよう

Azureポータル

Device Explorer

Device Explorerを使ってサンプルアプリケーションから送信されるメッセージ、デバイスツインを確認できます。
まずはDevice ExplorerをIoTHubへ接続します。

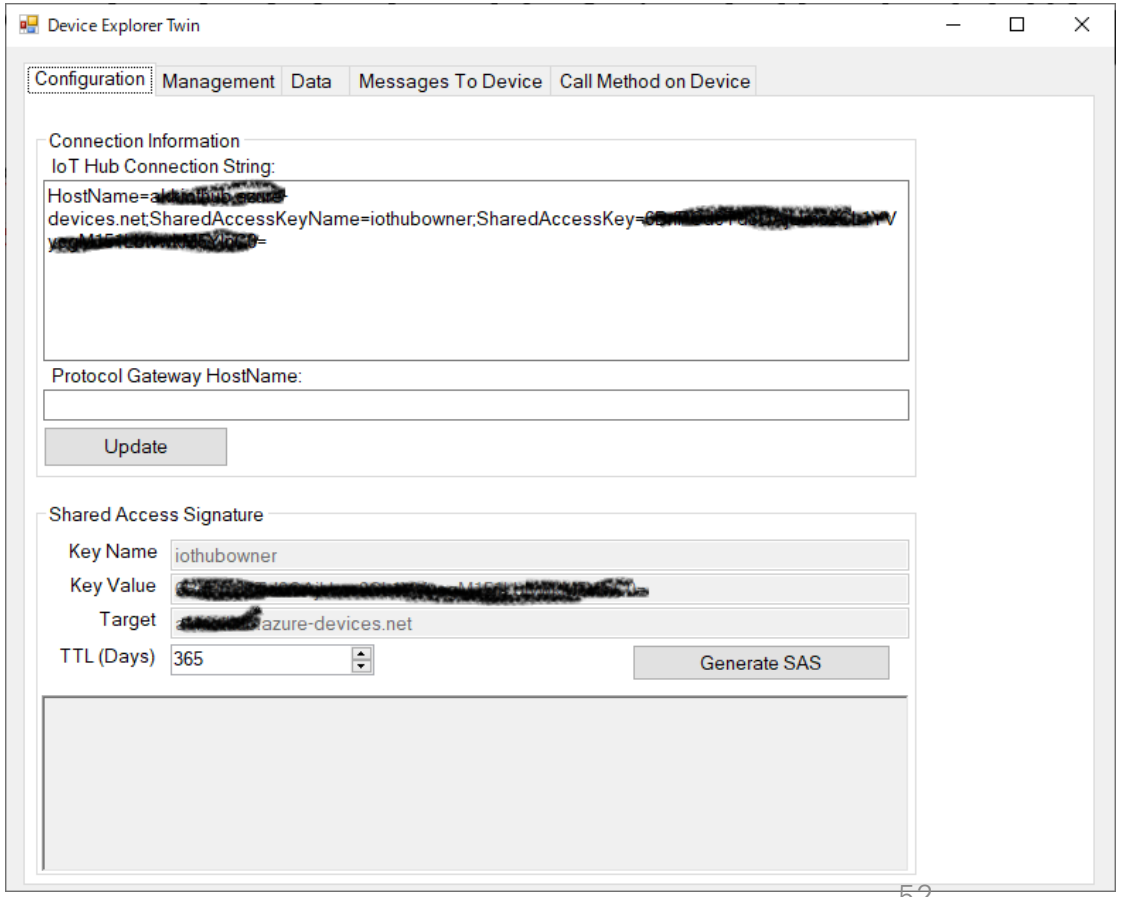
1. Azureポータルへアクセスして作成した準備済みのIoTHubを選択します。

2. IoTHubから**共有アクセスポリシー**>ポリシー>**iothubowner**>**接続文字列-プライマリーキー**をコピーします。

3. Device Explorerを起動してConfigurationタブを開きます。

4. Device Explorerを起動してConfigurationタブを開きます。

5. Connection InformationからIoTHubConnection Stringへコピーした接続文字列を貼り付け**Update**をクリックします。



Lab2: Azure IoT Hub サンプルを実行しよう

Visual Studio

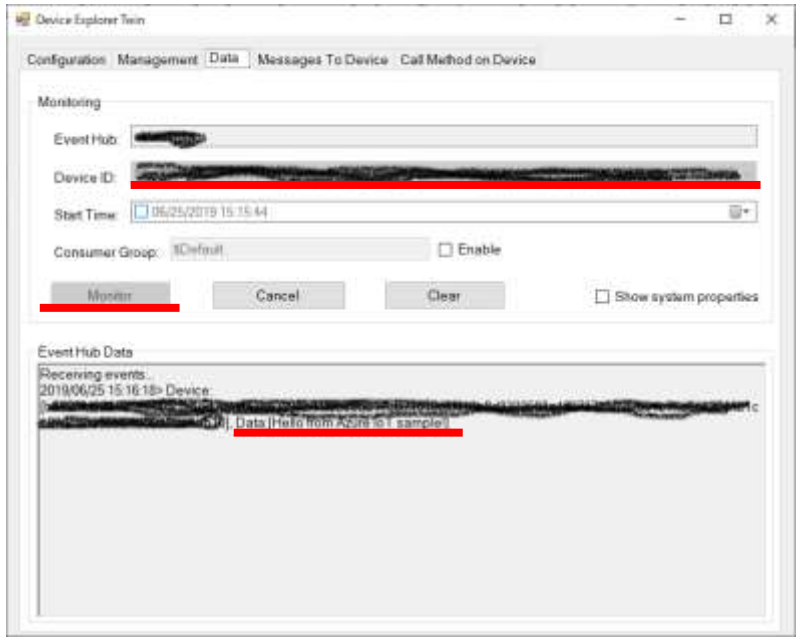
Device Explorer

メッセージ/テレメトリの送受信を確認します。

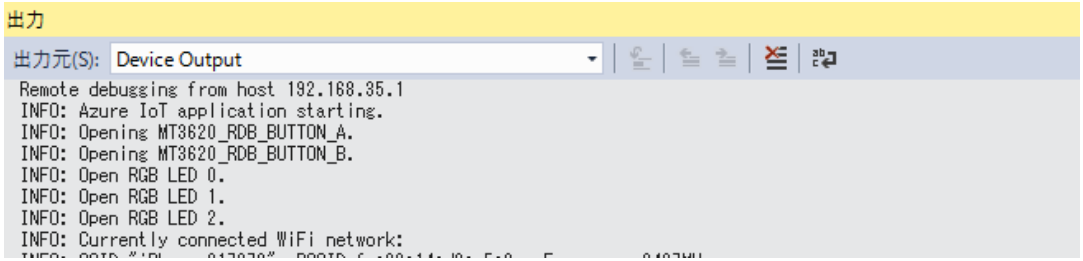
1. Device Explorerを起動して**Data**タブを選択します。

2. **Device ID**のドロップダウンメニューから自分の**Sphereデバイスを選択**して**Monitor**をクリックします。

3. ボタンBを押すと**Event Hub Dataウィンドウ**にIoTHubへ送られたメッセージが表示されます。



4. Visual Studio の出力タブで送られたデータが確認できます。



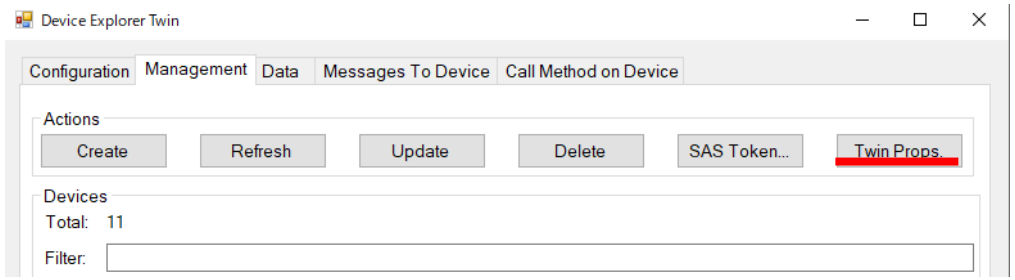
Lab2: Azure IoT Hub サンプルを実行しよう

Visual Studio

Device Explorer

デバイスツインの確認

1. Device Explorerを起動して**Management**タブを開きTwinPropsをクリックします。

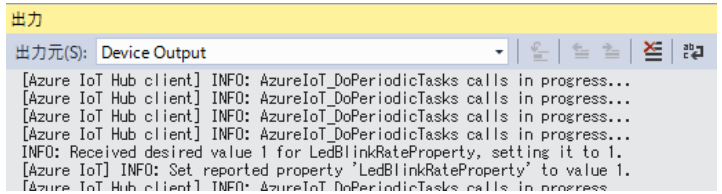


2. 自分の**Sphereデバイスを選択**して**Refresh**をクリックするとTwinプロパティが更新されます。

3. Device Twinの右側ウィンドウでLedBlinkRatePropertyのdesiredセクションに新しい設定を書き込みSendをクリックするとデバイスのLEDが点灯します。

```
"properties": {
    "desired": {
      "StatusLED": {
         "value": true
      },
      "$metadata": {
        "$lastUpdated": "2019-01-30T22:18:19.612025Z",
```

4. Visual Studioの出力タブで設定変更メッセージを確認します。

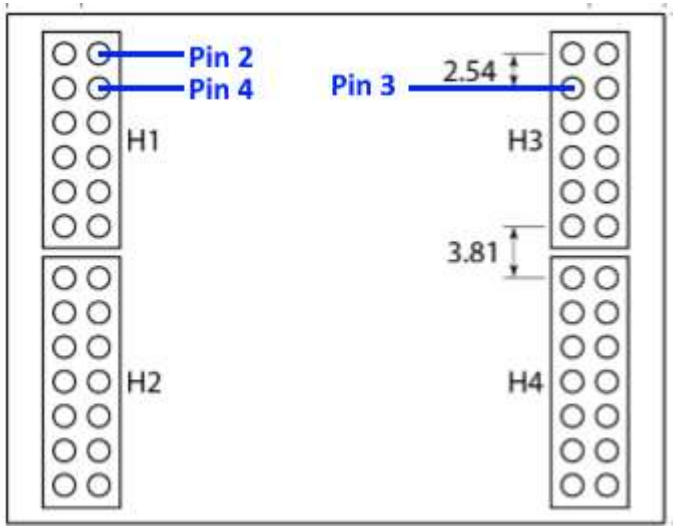


Lab2: センサーを接続しよう

開発ボード

1. 開発ボードの電源を切った状態でセンサーを接続します。 https://docs.microsoft.com/ja-JP/azure-sphere/hardware/mt3620-user-guide

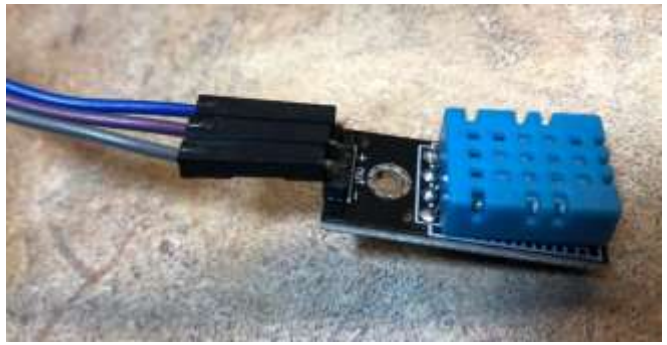
	MT3620	DHT11	Wire
Ground	Header 1, pin 2	-	grey
Data	Header 1, pin 4	out	purple
3.3V	Header 3, pin 3	+	blue



接続イメージ

H1: グランドとデータ

H3: 3.3V



DHT11: センサー側

Lab2: 実際のセンサーデータを送ろう

Visual Studio

1. 配布したDHTlib.cとDHTlib.hをAzureIoTフォルダをコピーします。

2. CMakeLists.txtとmain.c、App_manifest.jsonを編集します。

〇app_manifest.json

・GPIO 0を追加

```
"Gpio": [ "$SAMPLE_BUTTON_1", "$SAMPLE_BUTTON_2",
"$SAMPLE_LED", "$MT3620_RDB_HEADER1_PIN4_GPIO" ],
```

〇CMakeLists.txt

・ADD_EXECUTABLEにDHTlib.cを追加

```
ADD_EXECUTABLE(${PROJECT_NAME} main.c epoll_timerfd_utilities.c
parson.c DHTlib.c)
```

〇main.c

・43行目に以下のコードを挿入

```
#include “DHTlib.h“
```

・以下のコードをコメントアウト(78行目付近)

```
//static void SendSimulatedTemperature(void);
```

・79行目に以下のコードを挿入

```
static void SendRealTemperature(void);
```

・以下のコードをコメントアウト（191行目付近）

```
//        SendSimulatedTemperature();
```

・192行目に以下のコードを挿入

```
                SendRealTemperature();
```

・以下のコードをコメントアウト（536行目付近）

```
//void SendSimulatedTemperature(void)
//{
//    static float temperature = 30.0;
//    float deltaTemp = (float)(rand() % 20) / 20.0f;
//    if (rand() % 2 == 0) {
//        temperature += deltaTemp;
//    } else {
//        temperature -= deltaTemp;
//    }
//
//    char tempBuffer[20];
//    int len = snprintf(tempBuffer, 20, "%3.2f", temperature);
//    if (len > 0)
//        SendTelemetry("Temperature", tempBuffer);
//}
```

・552行目に以下のコードを挿入

```
void SendRealTemperature(void)
{
 DHT_SensorData* pDHT = DHT_ReadData(0);
  if (pDHT != NULL) {
  char tempBuffer[20];
  int len = snprintf(tempBuffer, 20, "%3.2f", pDHT->TemperatureCelsius);
  if (len > 0)
   SendTelemetry("Temperature", tempBuffer);
  }
}
```

Lab3の概要と目的

- 概要

このラボではIoT Centralへテレメトリデータを送信すると共にクラウドからデバイス（LED）のコントロールを行います。

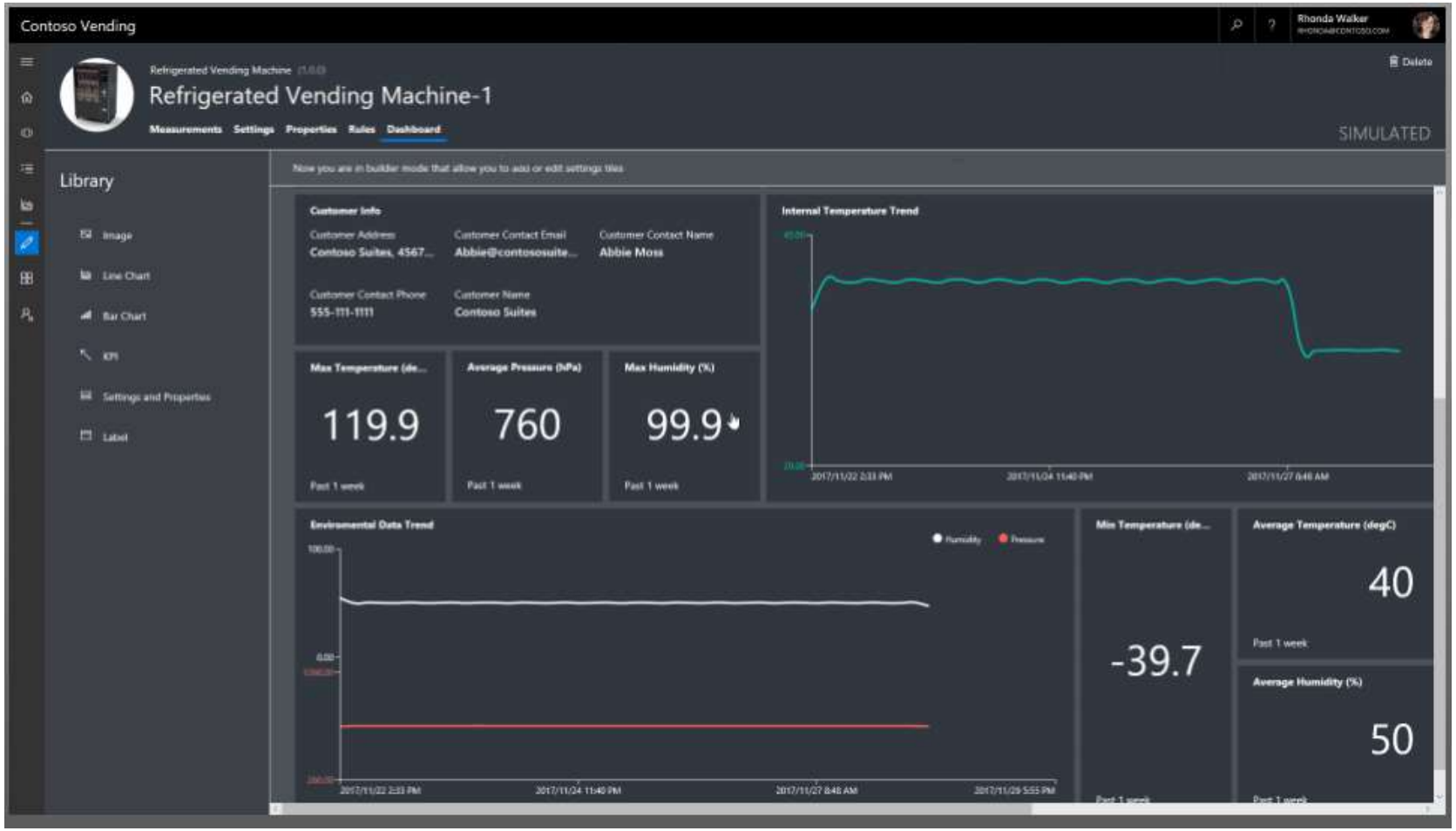
- 目的

Azure IoT Centralとの接続方法を理解します。またAzure IoT Centralのダッシュボードの使い方とデバイスからのテレメトリデータを可視化する方法について理解します。

補足説明

Lab3: Azure IoT Centralって？

プログラミング不要のSaaS型のIoTソリューションです。クラウドに不慣れな方でも簡単にIoTを実現できます。



デバイスの接続と管理

テレメトリデータ取り込み
コマンド送信、コントロール

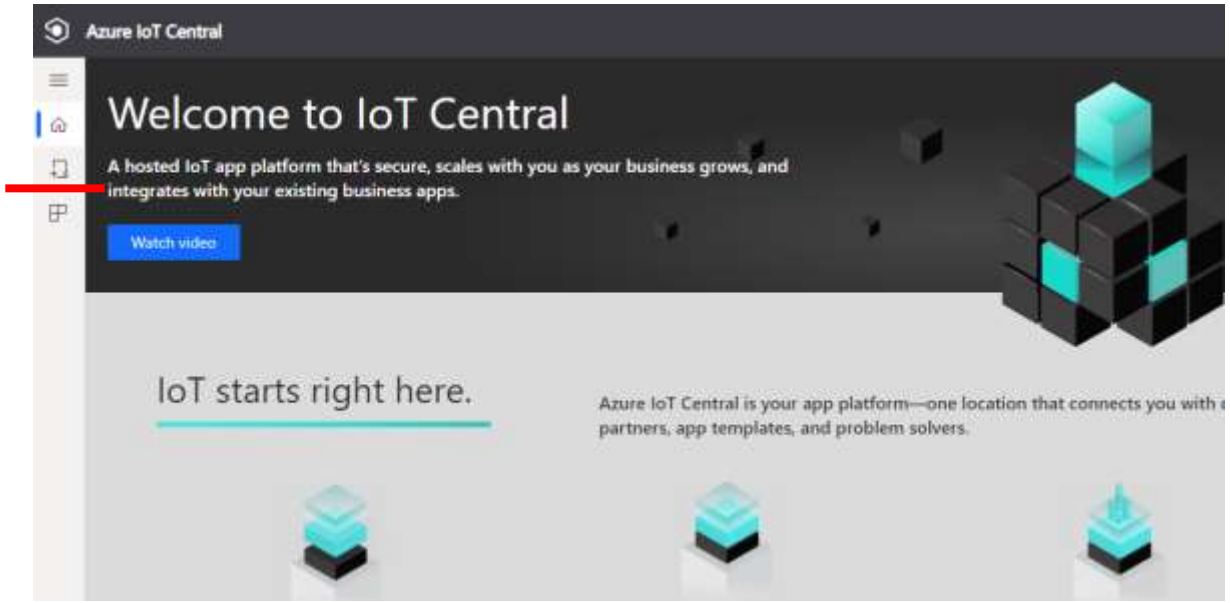
設定した監視ルールを元に
アクションをトリガ

ダッシュボード表示による
データの可視化

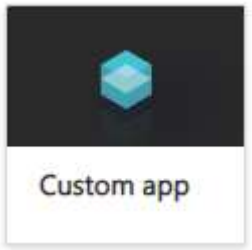
接続台数5台までは無料

Lab3: Azure IoT Central アプリケーションを作成しよう

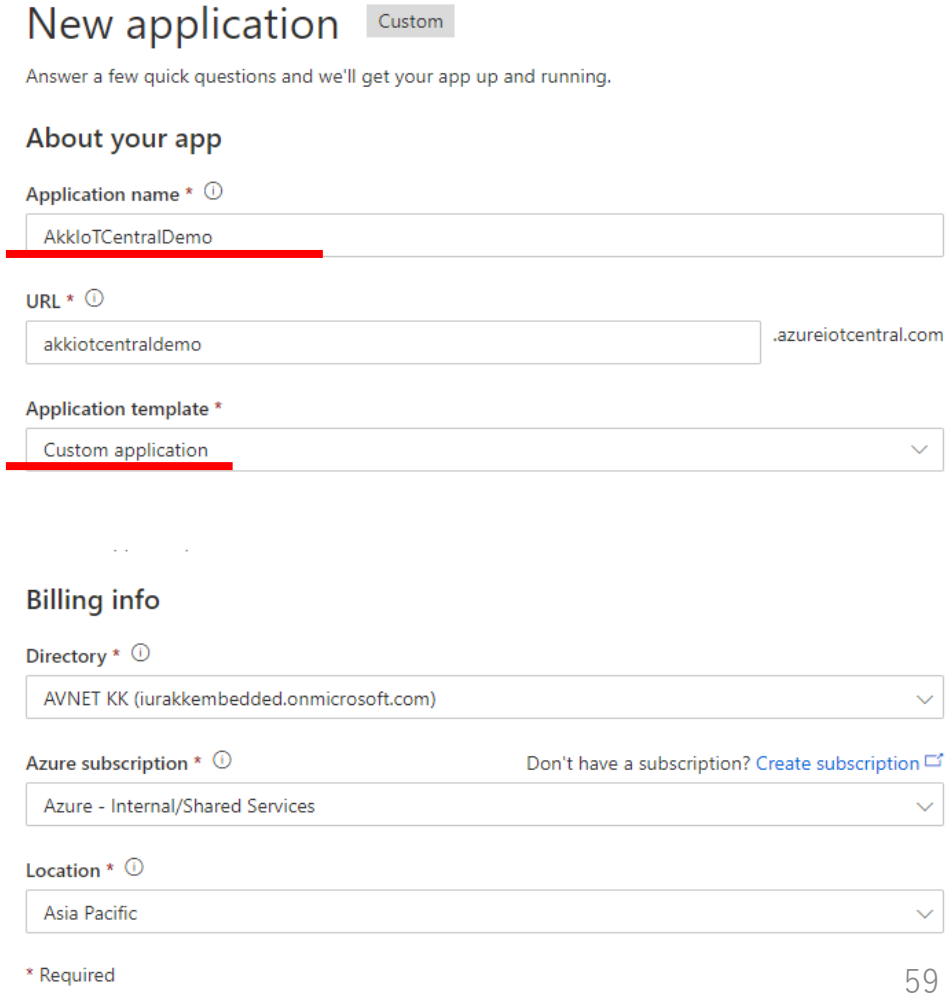
1. IoT CentralアプリケーションマネージャからBuildをクリックします。



2. サインインしてCustom appをクリックします。

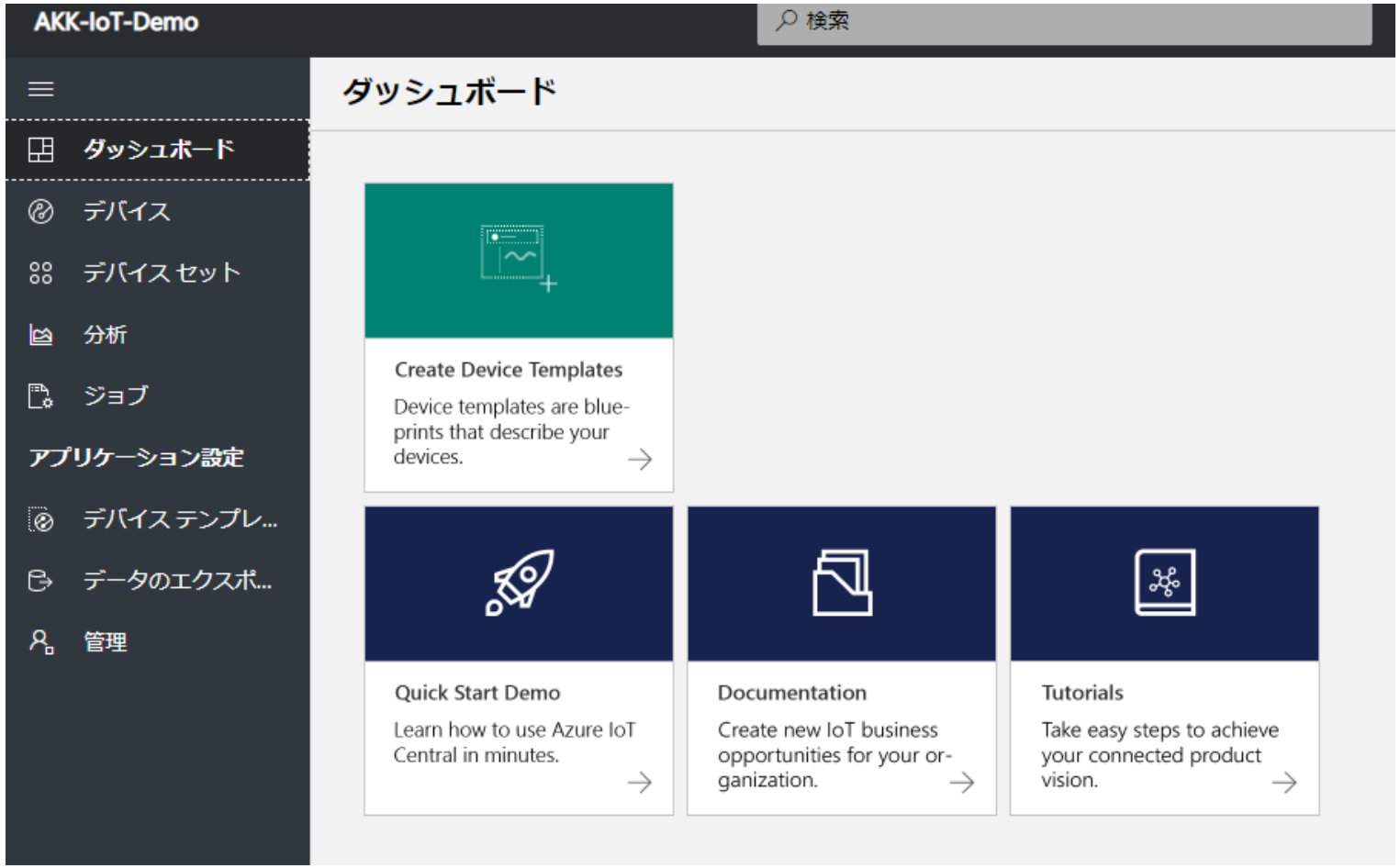


3. Application nameは任意、Application Templateは”custom application”を選択します。その他はデフォルト



Lab3: Azure IoT Central アプリケーションを作成しよう

4. アプリケーションが作成されるとAzure IoT Centralアプリケーションのポータルが表示されます。



Lab3: デバイステンプレートを作成しよう

IoT Central

1. デバイステンプレートを作成するをクリックします。

2. **デバイステンプレート** > **新しいテンプレート** > **カスタム**を選択します。

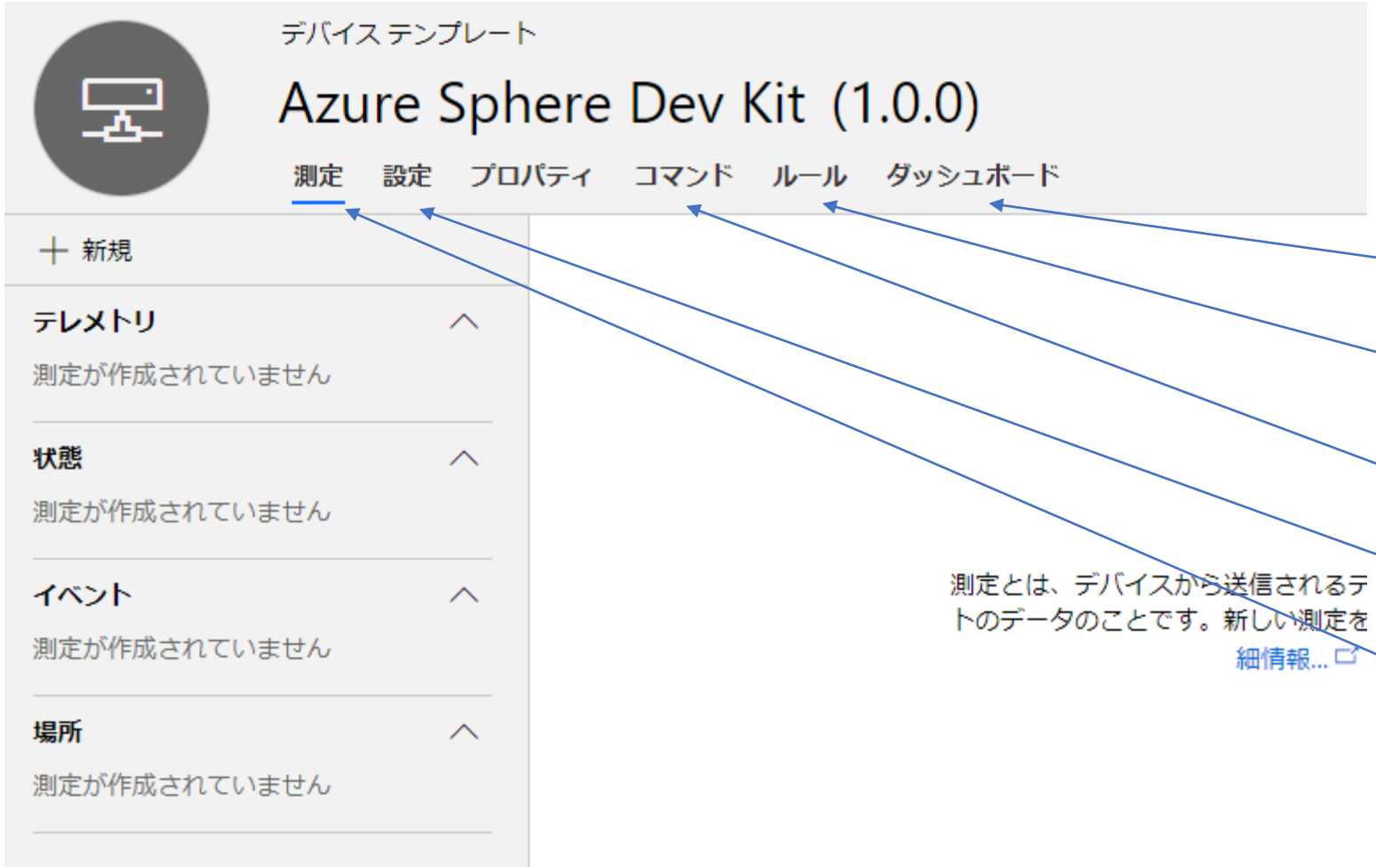
3. デバイステンプレートの名前を任意で入力します。

4. 作成をクリックするとデバイステンプレートが作成されます。

補足説明

Lab3: デバイステンプレートを理解しよう

デバイスが持っているテレメトリや状態、イベントなどの種類を定義します。ここに同じデータを上げてくるデバイスを登録していくことで個々のデバイスに対しては個別に定義する必要がありません。



ダッシュボードのタイルを作成

トリガーアクションを定義

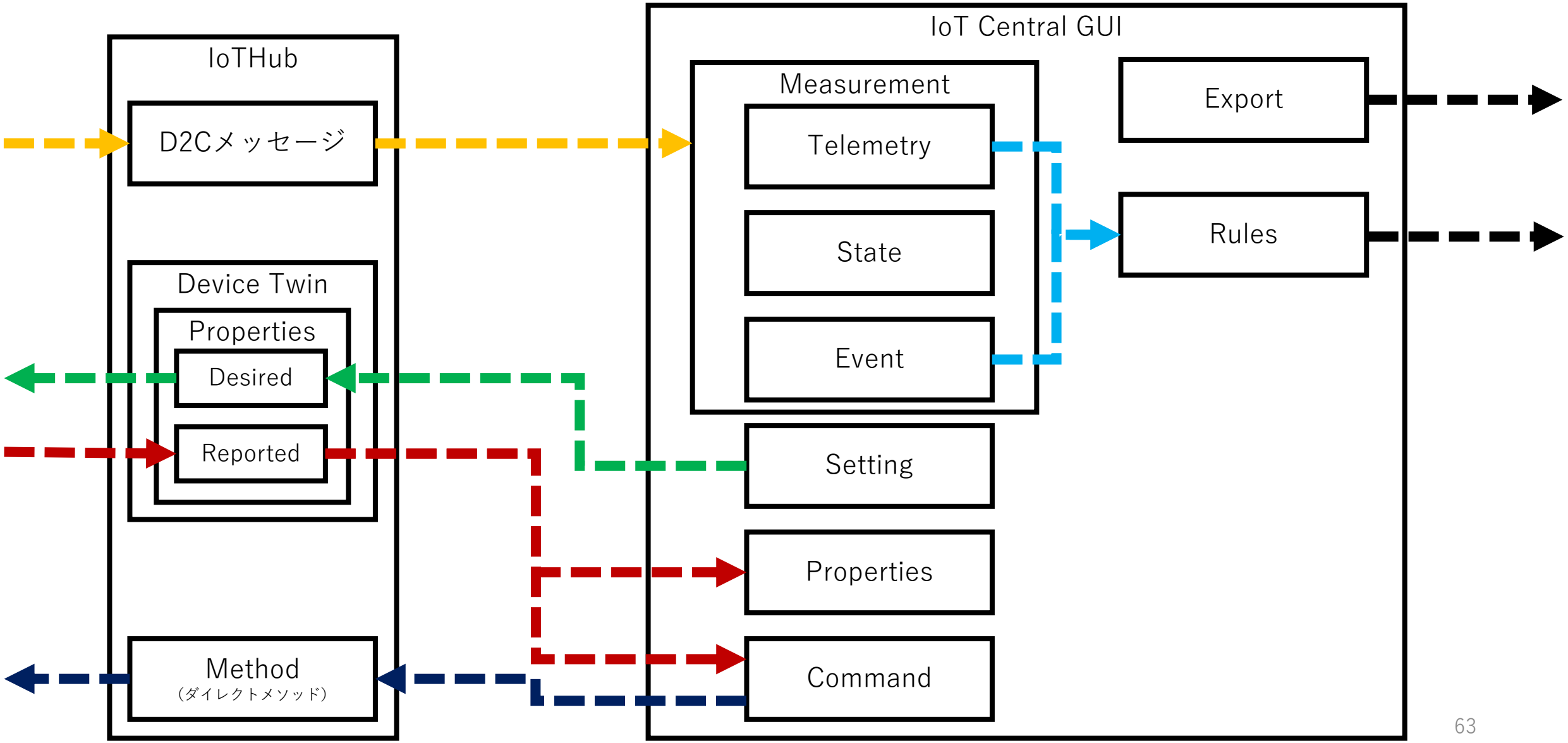
デバイスへコマンドを送って管理

デバイスの設定

デバイスからのデータ可視化

補足説明

IoT Centralを理解しよう

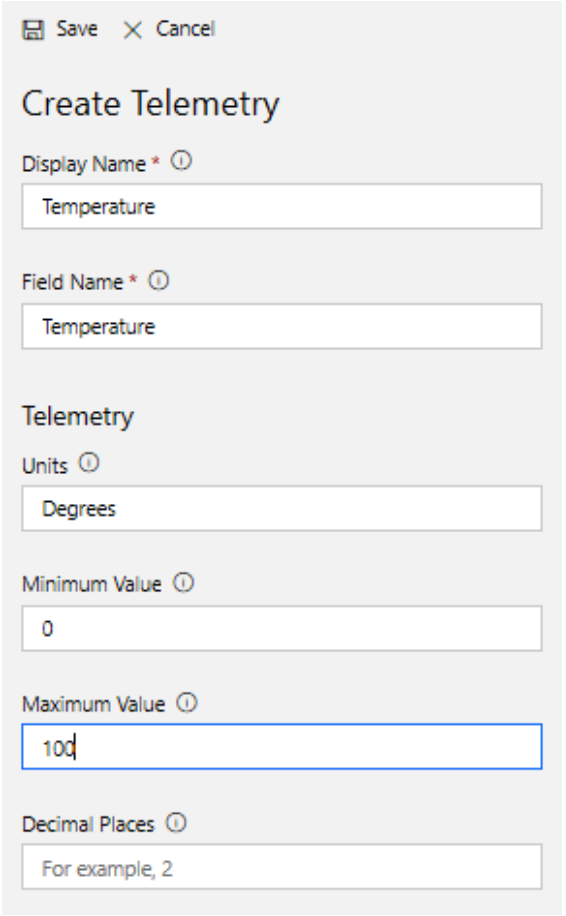


Lab3: 測定データを表示する設定をしよう

IoT Central

1. デバイステンプレートの**測定**から+**新規**をクリックして**テレメトリ**を作成します。

2. Display Nameに”温度”、Field Nameに ”Temperature”、表示する温度の最低最大値などを入力して保存をクリックします。

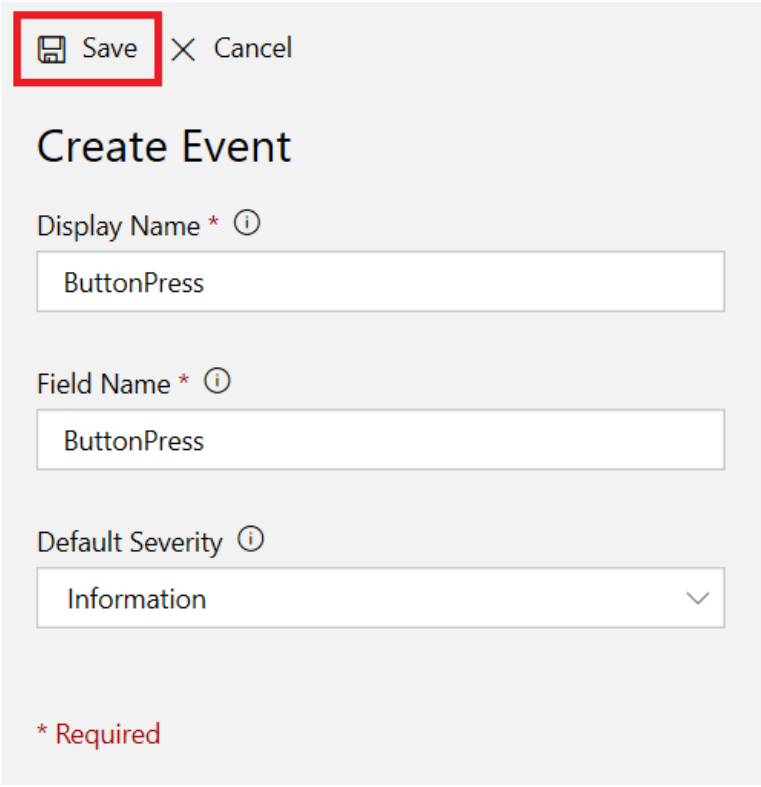


Lab3: ボタンイベントを表示する設定をしよう

IoT Central

1. デバイステンプレートの**測定**から+**新規**をクリックして**イベント**を作成します。

2. Display Nameに”ボタン”、Field Nameに ”ButtonPress”、既定の重要度は”Information”を選択して保存をクリックします。

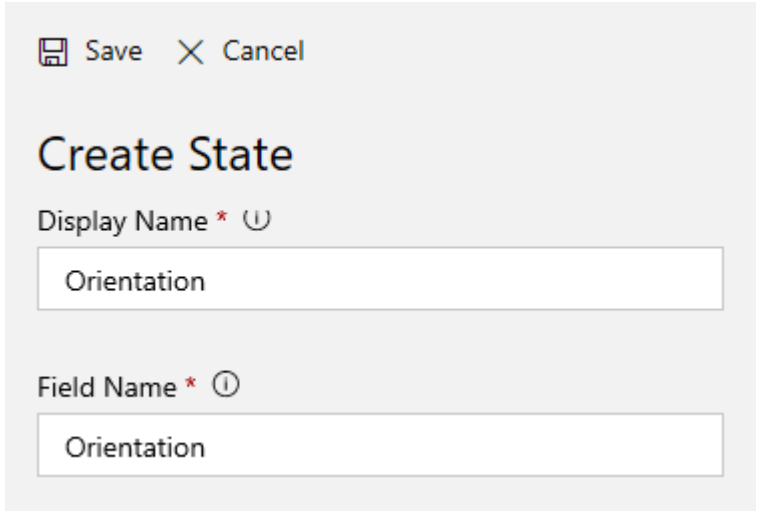


Lab3: 状態を表示する設定をしよう

IoT Central

1. デバイステンプレートの**測定**から+**新規**をクリックして**状態**を作成します。

2. Display Nameに”向き”、Field Nameに ” Orientation”を入力して保存をクリックします。

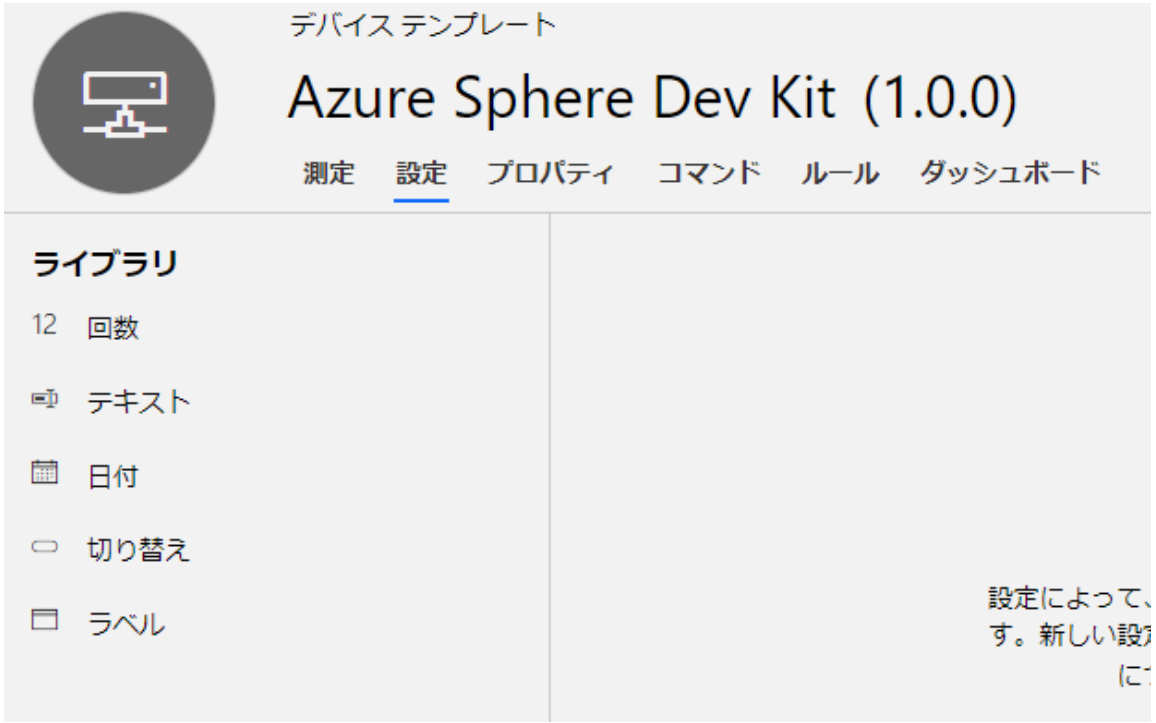


3. 値は”Up”と”Down”の2つを作成します。

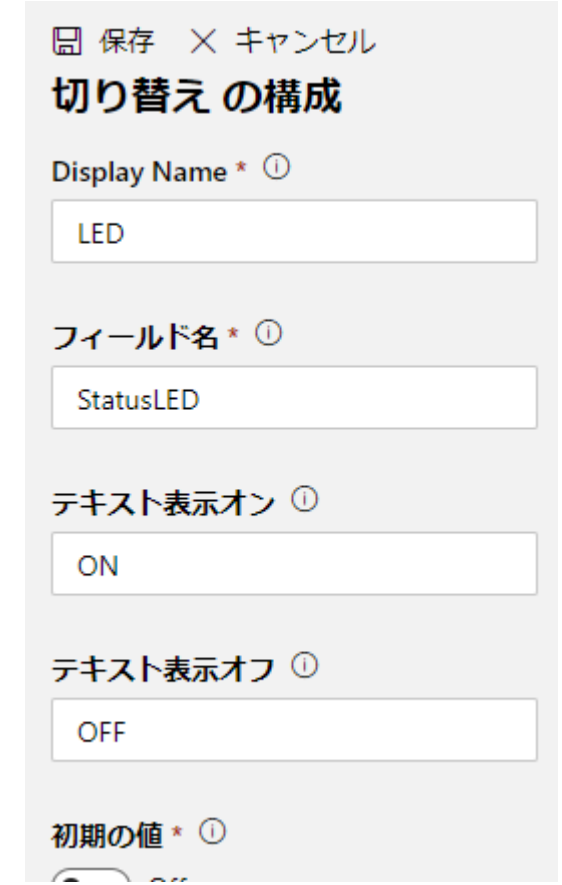
Lab3: LEDをコントロールする設定をしよう

IoT Central

1. デバイステンプレートの**設定**から切り替えをクリックします。



2. Display Nameに”LED”、Field Nameに ”StatusLED”、その他を適時入力して保存をクリックします。



Lab3: デバイスを作成しよう

コマンドライン

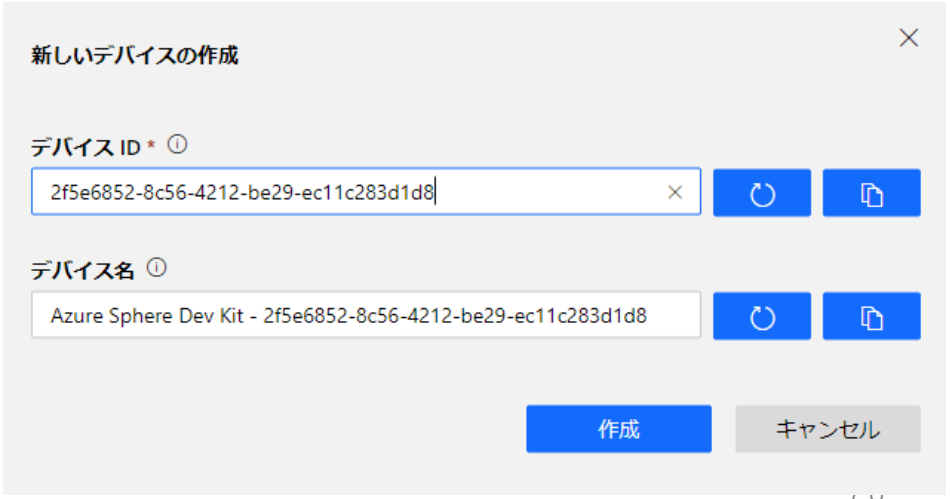
IoT Central

1. **デバイス**メニューから作成した**テンプレート**を選択して＋をクリックして**実際**をクリックします。

2. Azure Sphereのデバイスを貼り付けます。

デバイスIDは azsphere device show-attached で取得できますが
入力するデバイスIDは小文字で入力する必要があるのでPowerShellで変換します。

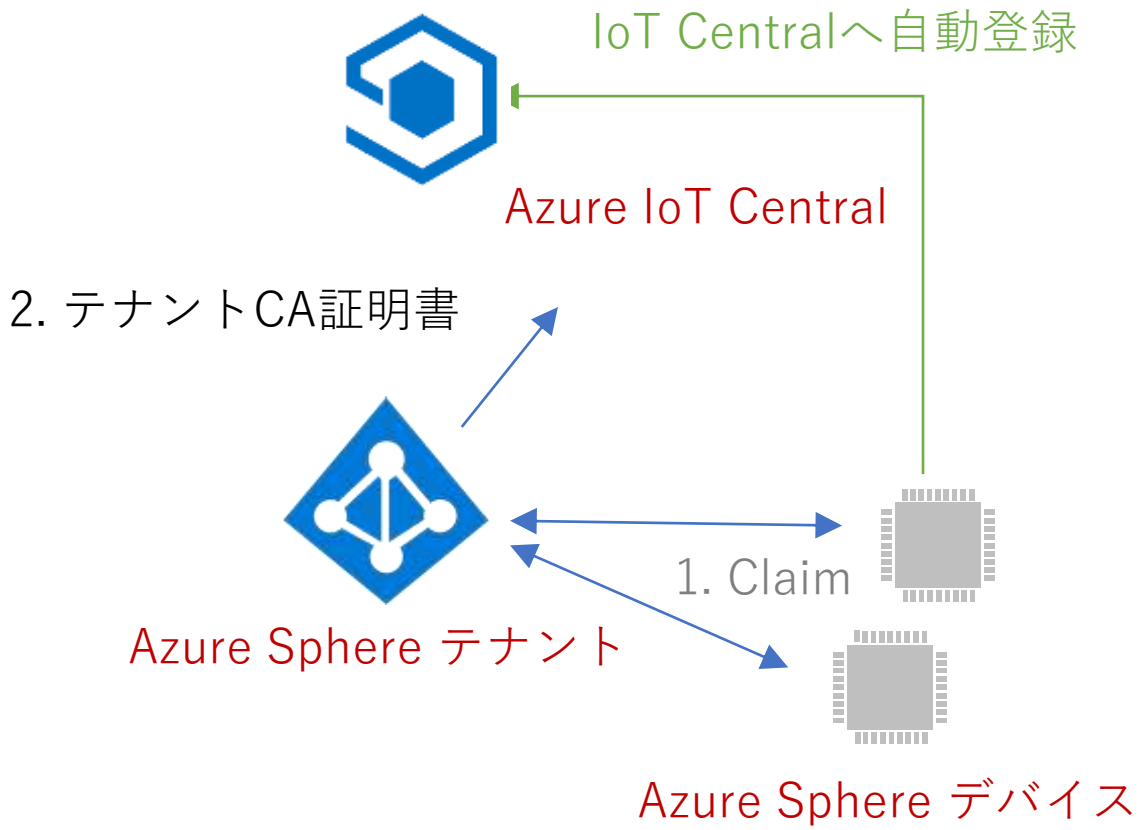
powershell -Command ((azsphere device show-attached)[0] -split ': ')[1].ToLower()



Lab3: Azure IoT CentralとAzure Sphere

補足説明

Azure Sphere テナントの証明書を登録することで簡単にデバイスを接続できます。



1. デバイスはClaimをすることでAzure Sphere テナントに関連付けがされている

2. Azure SphereテナントCA証明書をIoT Centralへアップロードして秘密鍵の保有を証明することで証明書を登録します。

この設定をすることでデバイスはAzure IoT Centralに接続すると自動で登録されるようになります。

Lab3: Azure IoT CentralをAzure Sphereと連携しよう

IoT Central

コマンドライン

- Azure Sphereと連携する Azure IoT Centralを設定する
  https://docs.microsoft.com/ja-JP/azure-sphere/app-development/setup-iot-central

手順１．~~Azure IoT Centralアプリケーションを作成する（前のステップで作成済み）~~

手順２．Azure Sphere テナントCA証明書をダウンロードする

手順３．Azure Sphere テナントCA証明書をIoT Centralへアップロードして確認コードを生成する

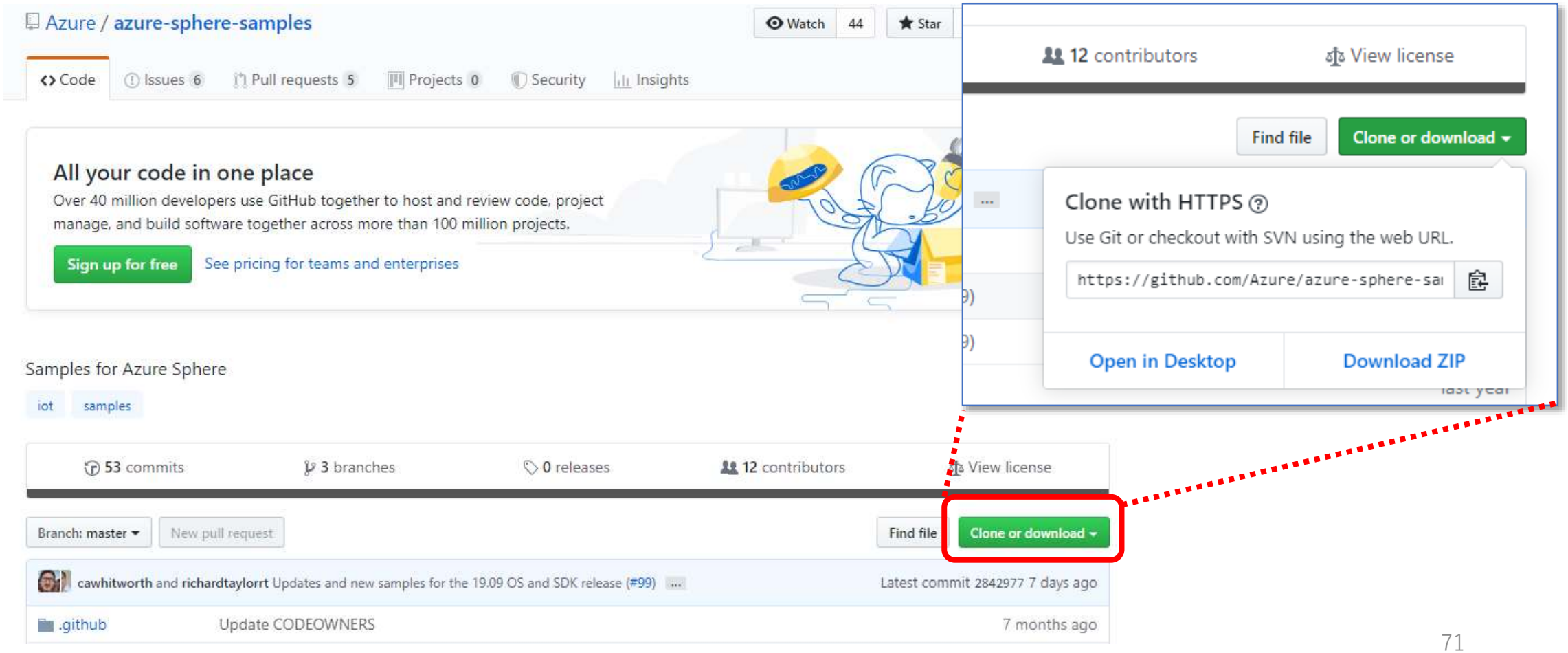
手順４．Azure Sphere テナントCA証明書を検証する

手順５．検証したAzure Sphere テナントCA証明書を使ってテナントIDを検証する

Lab3: GitHubからサンプルコードをダウンロードしよう

PC

1. GitHubのAzure Sphere Samplesのdownloadリンクからzipファイルでダウンロードします。



Lab3: Azure IoT Central サンプルを実行しよう

Visual Studio

Azureポータル

コマンドライン

1. Visual Studio 2017を起動して**ファイル>開く>プロジェクト/ソリューション**からGitHubからダウンロードした**AzureIoT.sln**を開きます。

2. ソリューションエクスプローラからapp_manifest.jsonを開きます。

```
{
  "SchemaVersion": 1,
  "Name": "AzureIoT",
  "ComponentId": "819255ff-8640-41fd-aea7-f85d34c491d5",
  "EntryPoint": "/bin/app",
  "CmdArgs": [],
  "Capabilities": {
    "AllowedConnections": [ "global.azure-devices-provisioning.net" ],
    "Gpio": [ "$SAMPLE_BUTTON_1", "$SAMPLE_BUTTON_2", "$SAMPLE_LED" ],
    "DeviceAuthentication": "00000000-0000-0000-0000-000000000000"
  },
  "ApplicationType": "Default"
}
```

3. アプリケーションの接続先情報を修正します。

・AllowConnctions ⇒ Azure IoT Central内のIoTHub、DPS
・CmgArmgs ⇒ Azure IoT CentralのScopeID
・DeviceAuthentication ⇒ Azure Sphere TenantID

4. ダウンロードしたZipファイルの中、Sample/AzureIoT/Toolsにある ShowIoTCentralConfig.exeをコマンドプロンプトから実行します。

```
azure-sphere-samples-master¥Samples¥AzureIoT¥Tools>ShowIoTCentralConfig.exe
Tool to show Azure IoT Central configuration for Azure Sphere applications

Are you using a Work/School account to sign into your IoT Central Application (Y/N) ?y
Getting your IoT Central applications
You have one IoT Central application 'akk-iotcentral'.
Getting the Device Provisioning Service (DPS) information.
Getting a list of IoT Central devices.

Find and modify the following lines in your app_manifest.json:
"CmdArgs": [ “xxxxxxxxx" ],
"AllowedConnections": [ "global.azure-devices-provisioning.net", "iotc-3xxxxxxxxxxxxxxxxx3.azure-devices.net" ],
"DeviceAuthentication": "--- YOUR AZURE SPHERE TENANT ID--- ",

Obtain your Azure Sphere Tenant Id by opening an Azure Sphere Developer Command Prompt and typing the following command:
'Azsphere tenant show-selected'
```

- TenantID (Azure Sphere コマンドラインから確認)

  > azsphere tenant list

Lab3: Azure IoT Central サンプルを実行しよう

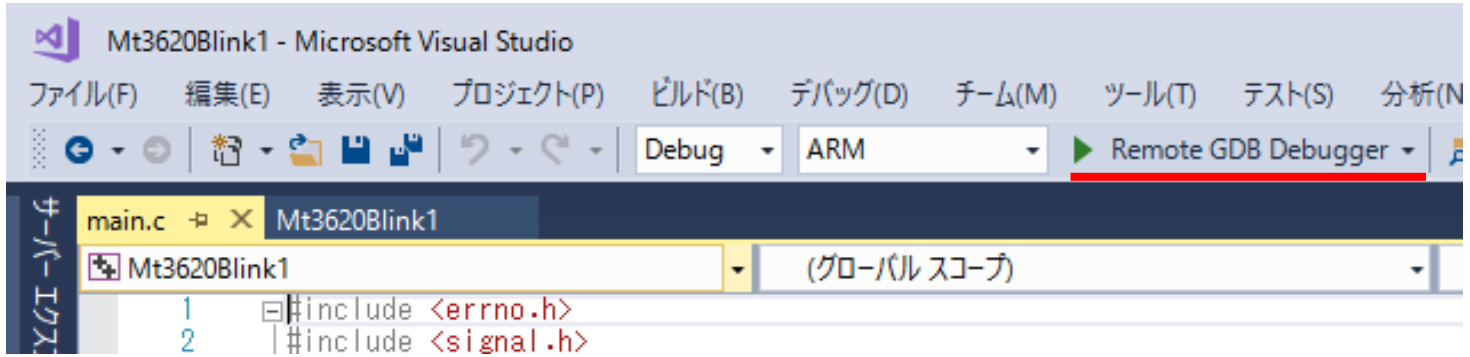
Visual Studio

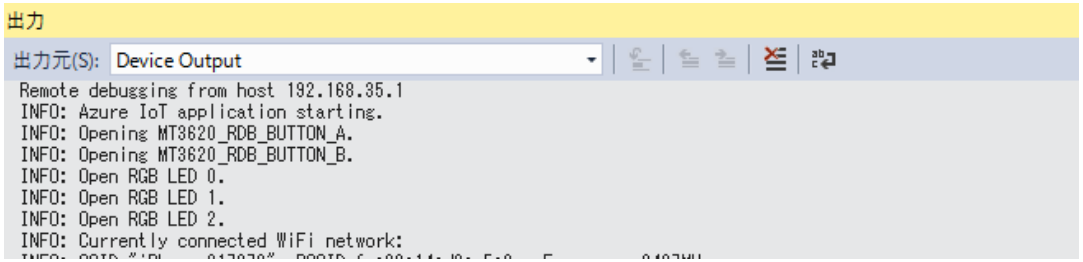
このサンプルアプリケーションは次のような機能を持っています。

Azure IoT Centralとの通信
- IoT Centralへ温度(シミュレーションデータ)を送る
- ボタンAを押すと”ButtonPressイベントを送信する
- ボタンBを押すと“Orientation”ステート(シミュレーションデータ)を送る

デバイスのコントロール
- LEDをコントロールする。

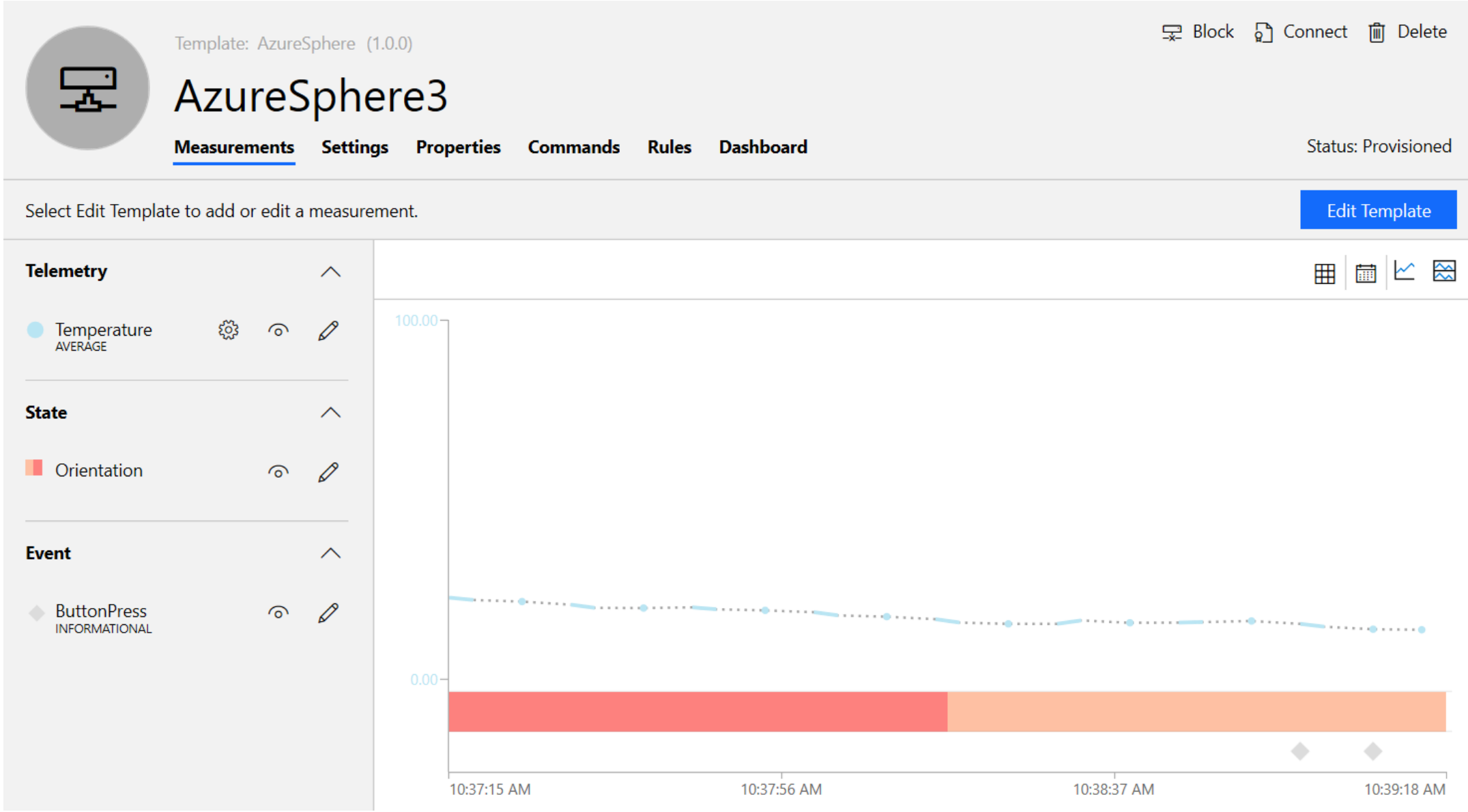
1. デバックを開始するとVisual Studioの出力ウィンドウに様々な情報が表示されます。正常に開発キットと通信が開始されるとアプリケーションが実行されます。





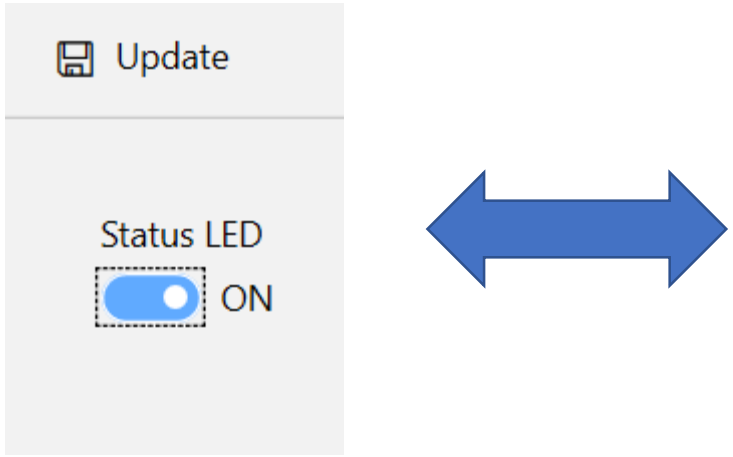
IoT Central

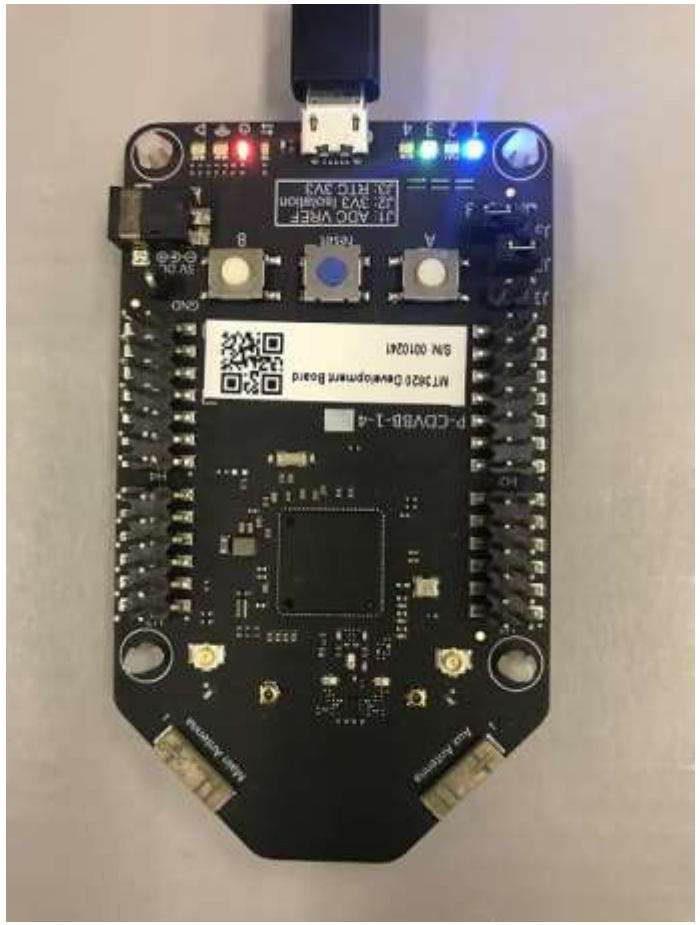
Lab3: Azure IoT Central でデータを確認しよう



Lab3: Azure IoT Central でLEDを切り替えよう

1. 作成したトグルを切り替えてUpdateボタンをクリックするとしばらくして開発ボード上のLEDが連動します。





いろいろ困ったら・・・

- Azure Sphere 公式ドキュメント
  https://docs.microsoft.com/ja-jp/azure-sphere/

- Azure Sphere公式フォーラム
  https://social.msdn.microsoft.com/Forums/en-US/home?forum=azuresphere（MSDN）
  https://stackoverflow.com/search?q=azuresphere（StackOverflow）

- 新機能リクエスト
  https://feedback.azure.com/forums/915433-azure-sphere

- Azureサポートプラン
  ✓プレミアムサポートサービス（有償）