

```
mirror_mod = modifier_ob.  
set mirror object to mirror.  
mirror_mod.mirror_object  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
  
selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
  
print("please select exactly  
  
-- OPERATOR CLASSES ----  
  
types.Operator):  
on X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
  
context):  
context.active_object is not
```

# Device Update

Hands on Lab with Device Update for  
IoT Hub on Ubuntu

# Overview of Device Update

- A secure and flexible way to keep devices up to date using IoT Hub
- Allows importing, scheduling and reporting on updates
- Integrates with IoT Hub and can have a 1 to many [IoT Hub] deployment
- Leverages existing [and requires] Content Delivery Network – like Windows Update
- Based on an Open-Source Agent written in C and expected to be modified
- Sample Agents: Linux [with or without IoT Edge], Yocto and RTOS
- Device Updates can be package (think apt install xxx) or image based
- Leverages Azure IoT Plug and Play
- <https://docs.microsoft.com/en-us/azure/iot-hub-device-update/device-update-agent-overview>

# How it works

## Agent - workflow

ADU Management service uses IoT Hub Digital Twin properties to orchestrate the agents update workflow



**1** ADU Management sets the "update command" property value:

1. Download
2. Install
3. Apply

To reset, it can use:

- Cancel



**2** ADU agent reads the "update command" property value and executes the desired command

**3** Sets the "update status" property value as it goes:

- Started
- Succeeded
- Failed

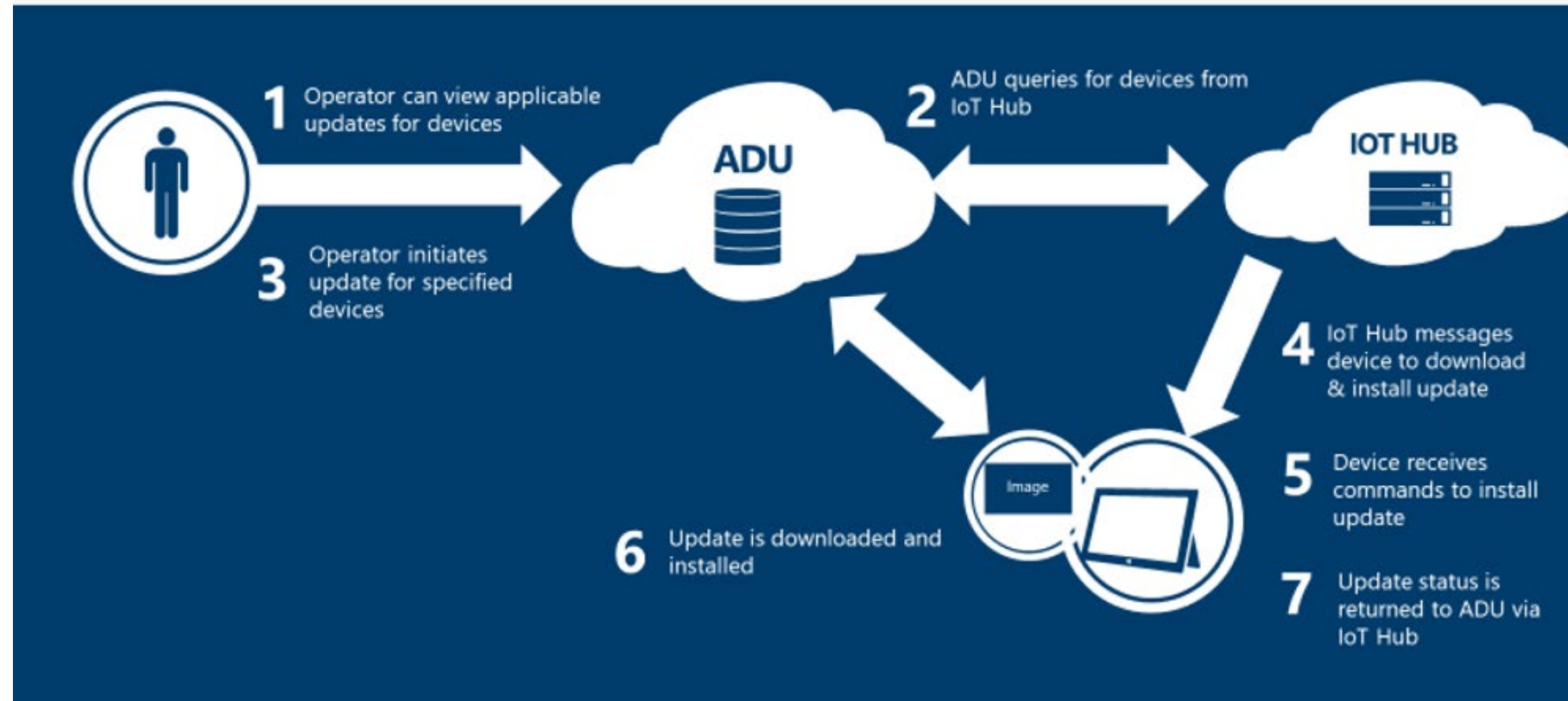
Default value when inactive:

- Idle



# How it works

## Manage and Deploy Updates



# How updates are defined - manifests

```
{
  "manifestVersion": "1",
  "updateId": {
    "provider": "DuTest",
    "name": "DuTestUser",
    "version": "2020.611.534.16"
  },
  "updateType": "microsoft/swupdate:1",
  "installedCriteria": "1.0",
  "files": {
    "000000": {
      "fileName": "image.swu",
      "sizeInBytes": 256000,
      "hashes": {
        "sha256": "IhIIxBJpLfazQ0k/PVi6SzR7BM0jf4HDqw+6gdZ3vp8="
      }
    }
  },
  "createdDateTime": "2020-06-12T00:38:13.9350278"
}
```

# The device side agent is open source

## Device Update for IoT Hub Agent

### Open-source Agent

- Uses Azure IoT device SDK for C
- Integrates with Delivery Optimization SDK for download

IoT Hub Digital Twin to orchestrate the update

### Image Update

#### 1. Linux

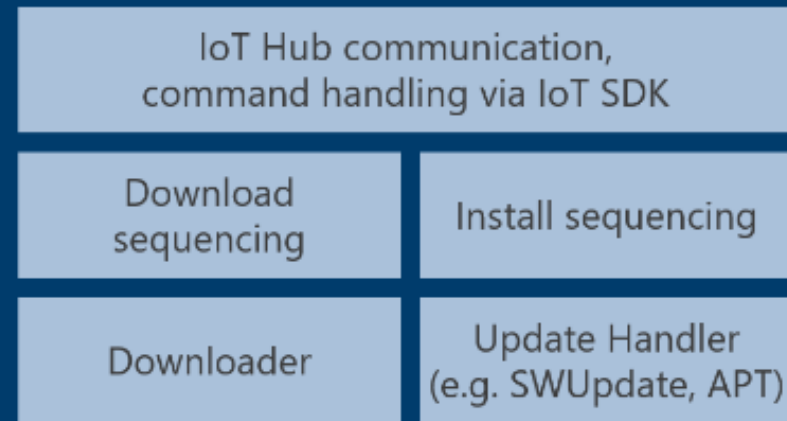
- Reference OS built with Yocto 2.7 (Poky Warrior OS)
- Validated on Raspberry Pi 3
- Integrates with SWUpdate for A/B type install

#### 2. RTOS

- Azure RTOS and Device Update samples to be created

### Package Update

1. Validated updating IoT Edge Runtime
2. Integrates with Apt for package updating

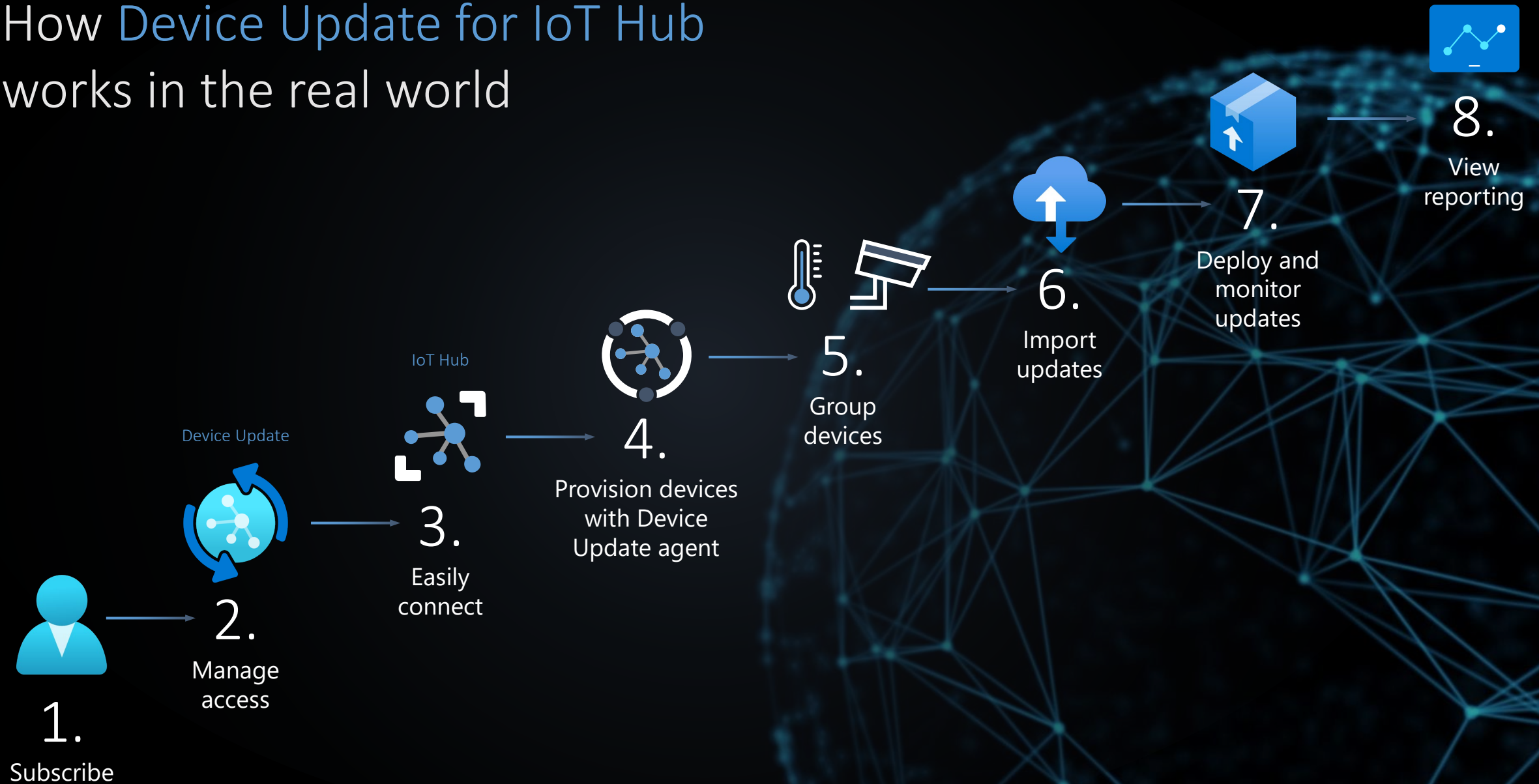


# Several Versions for Azure Device Update

- Public Preview
  - 0.6.0
  - 0.7.0
- Public Preview Refresh - 0.8.0
  - Recently released
  - Adds important new features



# How Device Update for IoT Hub works in the real world

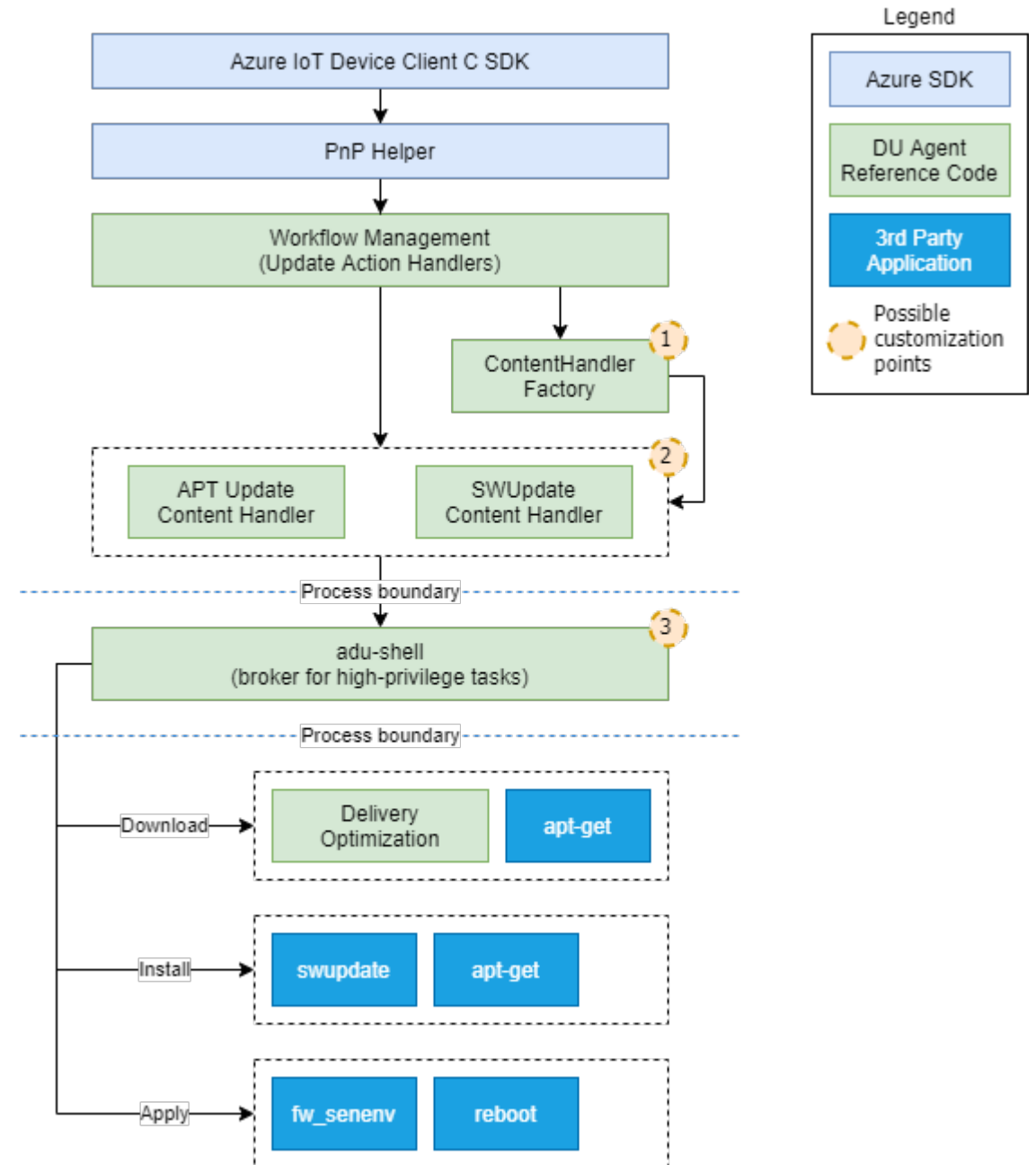




# Agent Architecture

- <https://github.com/Azure/iot-hub-device-update>
- APT or Image Handler
  - APT leverages existing package repositories
  - Image delivered via included CDN
  - BASH Scripts
- Delivery Optimization
  - Uses port 80 and can be proxied
- IoT Hub Protocol and supports Nested Edge
- Uses module of device identity in IoT Hub and twin properties
- On Linux, uses the Identity Service

```
apt install aziot-identity-service
apt install deviceupdate-agent
apt install deliveryoptimization-plugin-apt
```



# Update Manifest Version 4

- Enables the following features:
  - Multi Step Ordered Execution (MSOE)
  - Multi Component Updating
  - Goal State Deployment
  - Detached Update Manifest
- <https://github.com/Azure/iot-hub-device-update/blob/main/docs/agent-reference/update-manifest-v4-schema.md>

# Multicomponent Updating - Proxy Updates

- Targeting specific update files to different apps/components on the device
- Targeting specific update files to sensors connected to IoT devices over a network protocol (e.g., USB, CANbus etc.).
- <https://github.com/Azure/iot-hub-device-update/blob/main/docs/agent-reference/multi-component-updating.md>

# Multi-Step Ordered Execution (MSOE) Support

- Multi-Step Ordered Execution (MSOE) Support
- Parent and child updates
- Pre and post install steps
- <https://github.com/Azure/iot-hub-device-update/blob/main/docs/agent-reference/update-manifest-v4-schema.md>

# Update Types

- Built in UpdateTypes
  - microsoft/apt:1
  - microsoft/swupdate:1
- Update Content Handler extension and custom UpdateTypes
- <https://github.com/Azure/iot-hub-device-update/blob/main/docs/agent-reference/update-manifest-v4-schema.md>

# Further Reference

- <https://github.com/Azure/iot-hub-device-update/tree/main/docs/agent-reference>
- <https://github.com/Azure/iot-hub-device-update/blob/main/docs/agent-reference/whats-new.md>
- <https://github.com/Azure/iot-hub-device-update/blob/main/docs/agent-reference/how-to-build-agent-code.md>
- <https://docs.microsoft.com/en-us/azure/iot-hub-device-update/understand-device-update>