Introduction to Azure IoT

Pierre Cauchois, Senior Software Engineer, Microsoft
Agenda

How is Microsoft making it easy for embedded systems engineers to connect their “things” to “the cloud”?

What do you do once the devices are connected?
Part 1: Connecting Devices to the Cloud
Azure IoT Hub

- Event processing & Analytics (hot and cold path)
- Business logic, Connectivity monitoring
- Device management, provisioning and authorization

IoT Solution & Intelligence

- Storage / Big Data
- Machine Learning
- Dashboards
Device-facing Endpoints

Device-facing Endpoints involve the interaction between devices and cloud services. This diagram illustrates the communication flows and endpoints in an IoT (Internet of Things) context, specifically using Azure IoT Hub as the cloud service. The diagram shows how data flows through different endpoints:

- **Telemetry**: Data collected from devices and sent to the cloud.
- **Commands**: Commands sent from the cloud to devices.
- **Twin**: A model of a device that includes both desired and reported properties, useful for managing device configurations.
- **Properties**: Includes **Desired** and **Reported** properties for synchronization between the device and the cloud.
- **Tags**: Additional metadata for devices.
- **Device Methods**: Operations that can be performed on devices.

These endpoints facilitate two-way communication, ensuring devices can both receive updates and send data to the cloud, allowing for efficient and dynamic device management.
Backend-Facing Endpoints

- Telemetry (D2C) receive
- Commands (C2D) send
- Command feedback
- Event processing (hot and cold path)
- Business logic, Connectivity monitoring
- Twins
- Devices Methods
- Device management
- Device provisioning and authorization
- Operations Monitoring
- IoT Hub management

Azure IoT Hub
SDK Supported Platforms

- Android (Java or Xamarin)
- Arduino
- Debian Linux (v 7.5)
- ESP8266
- Fedora Linux (v 20)
- FreeRTOS
- iOS (Xamarin)
- mbed OS (v 2.0)
- OpenWRT
- Raspbian Linux (v 3.18)

- STM32
- TI RTOS
- Ubilinux (v3.0)
- Ubuntu Linux (v 14.04)
- Windows Desktop (7, 8, 10)
- Windows IoT Core (v 10)
- Windows Server (v 2012 R2)
- Yocto Linux (v 2.1)
- ... and more
## SDK Languages

<table>
<thead>
<tr>
<th><strong>C library:</strong></th>
<th><strong>Node.js library:</strong></th>
<th><strong>Java library:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Microcontrollers</td>
<td>• Node.js (v 0.10+)</td>
<td>• Java (v 1.7+)</td>
</tr>
<tr>
<td>• RTOS</td>
<td>• Node-RED</td>
<td>• Android</td>
</tr>
<tr>
<td>• Linux</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Windows</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>.NET</strong></th>
<th><strong>Python library:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C# libraries supported:</strong></td>
<td>• v 2.7.x</td>
</tr>
<tr>
<td>• Windows Desktop (7,8,10)</td>
<td>• v 3.5.x</td>
</tr>
<tr>
<td>• Universal Windows Platform</td>
<td></td>
</tr>
<tr>
<td>• Windows 10 IoT Core</td>
<td></td>
</tr>
<tr>
<td>• Xamarin (iOS, Android)</td>
<td></td>
</tr>
</tbody>
</table>
Getting an SDK

- Clone on Github
- Look for it on your favoring package manager (npm, maven, nuget, apt...)

- Examples in this presentation will be node.js
  - https://github.com/azure/azure-iot-sdk-node
Giving the device an identity

• Device Identity == Identifier + Secret
• Secret:
  • Symmetric keys stored in the Device Registry + Token generated from this key and used by the device (obviously, the key never travels on the wire)
  • x509 Certs/Keys on the Device + Certificate Thumbprints in the Device Registry
• TPM Support to store secrets

Managing Device Identities

• Each identity supports 2 keys or 2 cert thumbprints for easy roll-over
• Single-device and Bulk APIs to create, update or delete device identities
Giving the Device an Identity

```javascript
var Protocol = require('azure-iot-device-amqp').Amqp;
var Client = require('azure-iot-device').Client;

var connectionString = 'HostName=<hub-name>.azure-devices.net;DeviceId=<some-device-id>;SharedAccessKey=<base64-key>,'

var client = Client.fromConnectionString(connectionString, Protocol);
```

**Device**

```javascript
var iothub = require('azure-iot-hub');
var connectionString = '[IoT Connection String]';
var registry = iothub.Registry.fromConnectionString(connectionString);

var device = { /* device description */ }; registry.create(device, function(err, res) {
  /* do something with the result */
});
/* update and delete work the same way */
/* Bulk APIs also available */
```

**Service**
Sending a Message ("Telemetry" scenario)

Device

```javascript
var message = new Message("foo");
client.sendEvent(message, function (err, res) {/* deal with result */ });
```

Service

```javascript
var EventHubClient = require('azure-event-hubs').Client;
var client = EventHubClient.fromConnectionString(connectionString);
client.open().then(
  client.getPartitionIds.bind(client))
  .then(function (partitionIds) {
    return Promise.map(partitionIds, function (partitionId) {
      return client.createReceiver("$Default", partitionId).then(function(receiver) {
        receiver.on('message', printEvent);
      });
    });
  });
```

Receiving a Message ("Command" scenario)

**Device**

```javascript
client.on('message', function (msg) {
  /* do something with the command */
  client.complete(msg, function (err, res) {
    /* We could also “reject” or “abandon” the message */
    /* do something with the result */
  });
});
```

**Service**

```javascript
var Client = require('azure-iot-hub').Client;

var connectionString = '[IoT Hub Connection String]';

var client = Client.fromConnectionString(connectionString);
client.send(<deviceId>, <message>, function (err, res) {
  /* do something with the result */
});
```

Sending and Receiving Messages

• **MQTT** (and over Websockets)
• **AMQP** (and over Websockets)
• **HTTPS**

• Protocol Gateway:
  • [https://github.com/Azure/azure-iot-protocol-gateway](https://github.com/Azure/azure-iot-protocol-gateway)
Routing messages (new!)

Upload large amounts of data at once

Device

```javascript
client.uploadToBlob(<blob_name>, <file_stream>, <file_size>, function (err, result) {
    /* do something with the result */
});
```

Service

```javascript
client.getFileNotificationReceiver(function(err, receiver) {
    receiver.on('message', function(msg) {
        /* msg contains the file location and a token to access it */
        receiver.complete(msg, function(err, res) { /* do something */ });
    });
});
```

Execute code on the device and return the result (“Device Methods”)

**Device**

```javascript
client.onDeviceMethod(<method_name>, function (request, response) {
  /* Do something with the content of the request */
  response.send(<status_code>, <payload>, function(err, res) {
    /* do something with the result */
  });
});
```

**Service**

```javascript
var methodParams = {
  methodName: '<Method Name>',
  payload: '[Method Payload]',
  responseTimeoutInSeconds: 30 //default
};
client.invokeDeviceMethod(<device_id>, methodParams, function (err, res) {
  /* deal with result */
});
```

Maintain a device state and metadata ("Device Twin")

**Device**

```javascript
client.getTwin(function(err, twin) {
    twin.on('properties.desired', function(delta) {
        /* Do something with the delta (new property values) */
    });
    twin.properties.reported.update(patch, function(err) {
        /* Do something if that failed */
    });
});
```

**Service**

```javascript
registry.getTwin(deviceId, function(err, twin) {
    var twinPatch = {
        tags: { city: "Redmond" },
        properties: { desired: { telemetryInterval: 1000 } }
    };
    twin.update(twinPatch, function(err, twin) {
        /* Do something */
    });
});
```

Schedule tasks on devices

Device

/* Same as desired properties update or device method call */

Service

var Protocol = require('azure-iot-device-amqp').Amqp;
var Client = require('azure-iot-device').Client;
var Message = require('azure-iot-device').Message;

var client = Client.fromConnectionString(connectionString, Protocol);
var message = new Message("foo");
client.sendEvent(message, function (err, res) {/* deal with result */});

Query device twins

```javascript
var query = registry.createQuery('SELECT * FROM devices', 100);
var onResults = function(err, results) {
  /* Do something with the results */
  if (query.hasMoreResults) {
    query.nextAsTwin(onResults);
  }
};
query.nextAsTwin(onResults);
```

Monitor Operations and Errors (Service)

var EventHubClient = require('azure-event-hubs').Client;
var client = EventHubClient.fromConnectionString(connectionString, '/messages/operationsMonitoringEvents/*');

client.open().then(client.getPartitionIds.bind(client)).then(function (partitionIds) {
    return Promise.map(partitionIds, function (partitionId) {
        return client.createReceiver('$Default', partitionId).then(function(receiver) {
            receiver.on('message', printEvent);
        });
    });
});
Summary: Azure IoT Hub enables you to:

- Use open source, cross-platform SDKs that lets you:
  - Manage device identities
  - Send and receive millions of messages using industry protocols
  - Route these messages to various endpoints
  - Upload large amounts of data that wouldn’t fit in messages
  - Store and query device state and metadata
  - Execute code or schedule tasks on devices
  - Monitor operations in real time
  - Connect devices to the rest of the Azure stack (stream analytics, storage, databases, machine learning…)
- …
How to try and get started?

• Start for free
• Scale as you need
• Priced according to message processing capacity
• https://azure.microsoft.com/en-us/pricing/calculator/
No Internet Connectivity, No Problem: The Gateway SDK

- [https://github.com/azure/azure-iot-gateway-sdk](https://github.com/azure/azure-iot-gateway-sdk)
- Open-Source & Cross Platform (duh!)
- Modular, with connectors to mainstream programming languages
- Easily add support for local communication protocols
- Supports OPC-UA

Earlier Breakout Session: Building Modular IoT Gateways with OSS: [http://sched.co/9mAh](http://sched.co/9mAh)
The Device Catalog

- [https://catalog.azureiotsuite.com/](https://catalog.azureiotsuite.com/)
- Browse/Search using various criteria and applications
- Hundreds of certified devices
- Fully-fledged certification program
Part 2: What to do with these connected devices?
Build your own IoT Solution

• Process Data in Realtime with Stream Analytics
• Build Dashboards with Power BI
• Get Smart with Azure Machine Learning
• Store into Azure Storage or Azure Data Lake
• Run Big Data Analytics with Azure Data Lake Analytics or HDInsight
• Go serverless with Azure Functions
• …
Use a Preconfigured Solution: Azure IoT Suite

Predictive maintenance
Anticipate maintenance needs and avoid unscheduled downtime by connecting and monitoring your devices for predictive maintenance.

Remote monitoring
Connect and monitor your devices to analyze untapped data and improve business outcomes by automating processes.

https://www.azureiotsuite.com
Remote Monitoring
Predictive Maintenance

There’s more: Connected Vehicle Platform

- Announced at CES 2017
- https://aka.ms/mcvp
Security Program for Azure IoT


Questions?
Thank You!

- pierreca@microsoft.com
- @pierreca
- https://azure.com/iot
- https://azure.com/iotdev
- https://blogs.microsoft.com/iot