Build Mobile Apps for IoT Devices and IoT Apps for Mobile Devices

Jan Metzner
Solutions Architect, Amazon Web Services
Easy customer setup

Think user experience

Almost “magical”
Build trust
More engagement
Easy customer setup

Intel Edison

Bluetooth

WiFi

User

Mobile Application

WiFi Router

AWS IoT
Sending data to AWS IoT

Node.js

// Certificates for secure communications
var KEY = fs.readFileSync(pathToPrivateKey);
var CERT = fs.readFileSync(pathToCert);
var TRUSTED_CA_LIST = [fs.readFileSync(pathToROOTCA)];

// Connect to the MQTT broker
var client = mqtt.connect(options);

Node.js

// Set connections parameters
var options = {
  port: 8883,
  host: '<AWS IoT Endpoint>',
  protocol: 'mqtts',
  ca: TRUSTED_CA_LIST,
  key: KEY,
  cert: CERT,
  secureProtocol: 'TLSv1_2_method',
  protocolId: 'MQIsdp',
  clientId: 'Edison',
  protocolVersion: 3
};
Sending data to AWS IoT

```javascript
// Connect to AWS IoT
client.on('connect', function () {
    // Do stuff here when connection is established
});

// Receive messages
client.on('message', function (topic, message) {
    // Do stuff here when a message is received
});

// Subscribe to an MQTT topic
client.subscribe(topic);

// Publish data on MQTT topic
client.publish(topic, JSON.stringify(myMsg));
```
Bluetooth on the AWS IoT Device

Node.js

// Bluetooth Low Energy Library
var bleno = require('bleno');

// When starting up
bleno.on('stateChange', function(state) {
    if (state === 'poweredOn') {
        bleno.startAdvertising('ASFA device', ['f00df00dffb48d2b060d0f5a71096e0']);
    }
});

// Reads data and update WiFi configuration
BLEConnectCharacteristic.prototype.onWriteRequest = function(data, offset, withoutResponse, callback) {
    updateWPASupplicant(JSON.parse(data));
    callback(this.RESULT_SUCCESS);
};
// Initializes a Bluetooth adapter on Android
final BluetoothManager bluetoothManager = (BluetoothManager)
getSystemService(Context.BLUETOOTH_SERVICE);
    mBluetoothAdapter = bluetoothManager.getAdapter();

// Automatically connects to the device upon successful start-up initialization.
mBluetoothLeService.connect(mDeviceAddress);

// Write data
characteristic.setValue(this.chunks[0].toString());
mBluetoothGatt.writeCharacteristic(characteristic);
Monitor the system

*Using the web or mobile*

- Different users
- Access
- Tools
The rule pushes ALL data to an Amazon DynamoDB table.
Telemetry dashboard

- User
- Smartphone
- Serverless Web Dashboard
- SDK JavaScript
- Amazon S3
- Amazon Cognito
- SDK Android
- Amazon DynamoDB
Telemetry dashboard

Rule

```json
{
    "sql": "SELECT * FROM 'things/data'",
    "ruleDisabled": false,
    "actions": [
        {
            "dynamoDB": {
                "roleArn": "arn:aws:iam::xxxxxxxxxxx:role/iot-role",
                "tableName": "thingsData",
                "hashKeyField": "topic",
                "hashKeyValue": "things/data",
                "rangeKeyField": "timestamp",
                "rangeKeyValue": "${devicetimestamp}"}
            }
        }
    ]
}
```
Control the connected object

*Whenever, wherever you are*

- 4G
- Bluetooth
- Wi-Fi

**Multiple technologies**

**Not always online**

**Visibility**
AWS IoT Thing Shadow: Desired state

Ask for desired state to activate the pump
POST /things/Edison/state

```json
{
    "state": {
        "desired": {
            "pump": "1"
        }
    }
}
```
AWS IoT Thing Shadow: Desired state

```json
Shadow
{
  "state": {
    "pump": "1"
  },
  "version": "3",
  "metadata": {
    "color": <time-stamp>
  }
}
MQTT  $aws/things/Edison/shadow/update
```
AWS IoT Thing Shadow: Reported state

Send SNS Mobile Push Notification when pump has been activated

"reported": { "pump": 1 }
AWS IoT Thing Shadow: Reported state

MQTT $aws/things/Edison/shadow/update

{
    "state":{
        "reported":{
            "pump":"1"
        }
    }
}
AWS IoT Thing Shadow: Reported state

Rule

```json
{
    "sql": "SELECT * FROM '$aws/things/Edison/shadow/update/delta'
WHERE state.desired.pump = 1 AND state.reported.pump = 1",
    "ruleDisabled": false,
    "actions": [
        {
            "sns": {
                "roleArn": "arn:aws:iam::xxxxxxxxxxx:role/iot-role",
                "targetArn": "arn:aws:sns:us-east-1:xxxxxxxxxxx:ReInventDemo"
            }
        }
    ]
}
```
Self-regulated IoT systems

*Be notified, create autonomous systems*

Complex and automated IoT applications

Create high value for your customers
Self-regulated systems

Intel Edison  🔄  AWS IoT  🔄  Rule  🔄  Desired state  🔄  AWS Lambda

Rule

Triggers Lambda function when humidity is too low
Self-regulated systems

{  
"sql": "SELECT * FROM 'things/data' WHERE humidity < 20",
"ruleDisabled": false,
"actions": [
  {
"lambda": {
  "functionArn": "arn:aws:lambda:us-east-1:xxxxxxxxxxx:function:pumpAlert"
  }
  }
]
}
AWS IoT SDKs and APIs
// Libraries
#include "mqtt_interface.h"
#include "iot_version.h"

// Connecting to MQTT broker
MQTTConnectParams connectParams;
connectParams.MQTTVersion = MQTT_3_1_1;
connectParams.pClientID = "CSDK-test-device";
connectParams.pHostURL = HostAddress;
connectParams.port = port;
iot_mqtt_connect(connectParams);

// Subscribing to a topic
MQTTSubscribeParams subParams;
subParams.mHandler = MQTTcallbackHandler;
subParams.pTopic = "sdkTest/sub";
subParams.qos = qos;
iot_mqtt_subscribe(subParams);
AWS IoT SDK for Javascript

// Enable AWS SDK for JavaScript support using a service model file
var myService = new AWS.Service({apiConfig:
    require('./path/to/service-model.json'),
    endpoint: "service endpoint"});

// Initialize SDK
var aws = require('aws-sdk');
var iot = new aws.Service({apiConfig:
    require('./iot-service-model.json'),
    endpoint: "iot.us-east-1.amazonaws.com" });

var iotData = new aws.Service({apiConfig:
    require('./iot-data-service-model.json'),
    endpoint: "data.iot.us-east-1.amazonaws.com" });

// Publish message on MQTT topic
var params = {
    "topic": "foo/bar",
    "payload" : "hello world"
};
iotData.publish(params, function(err, data) {
    console.log(err, data);});
// Use TLS1.2 to connect to AWS IoT
SSLContext sslContext = SSLContext.getInstance("TLSv1.2");
sslContext.init(keyManagerFactory.getKeyManagers(), null, new SecureRandom());
mqttConnectOptions.setSocketFactory(sslContext.getSocketFactory());

// Use Android MQTT Paho Library to establish connection
mqttConnectOptions.setCleanSession(true);
mqttConnectOptions.setConnectionTimeout(AWSIoTConstants.IoTTimeout);
mqttConnectOptions.setKeepAliveInterval(AWSIoTConstants.IoTKeepalive);
if (AWSIoTConstants.lastWillTestament != "" && AWSIoTConstants.LastWillTopic != null)
{
    mqttConnectOptions.setWill(AWSIoTConstants.LastWillTopic,
                                AWSIoTConstants.lastWillTestament.getBytes(),
                                AWSIoTConstants.IoTQoS, true);
}
try
{
    connectionListener.setMQTTClient(mqttClient, mqttConnectOptions);
    mqttClient.connect(mqttConnectOptions, null, connectionListener);
    instance = this;
}
Amazon Cognito and AWS Identity and Access Management (IAM) policies

{  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
      ],
      "Condition": {
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "graph.facebook.com"
        }
      }
    }
  ]
}

Policy variables

cognito-identity.amazonaws.com:amr
cognito-identity.amazonaws.com:aud
cognito-identity.amazonaws.com:sub
AWS IoT CLI and Web Console

Additional Access to AWS IoT available:
- AWS CLI
- AWS IoT Web Console
Common Design Architectures with AWS IoT
Using Smartphone as a hub

- **No connectivity**: Very limited resources / Saving costs

- **Security**: Use Amazon Cognito to securely send data to AWS IoT or to the AWS cloud

- **Hub**: Use Smartphone’s capabilities (WiFi/4G)
Building Automation / Mobile Control of IoT

- **Dynamic Automation**: Dynamically respond to Events happening in the Factory
  
  *ex. Opening doors and requesting assistance when emergency button is pushed*

- **Mobile Control**: Simplify mobile control of IoT/Factory automation devices with AWS IoT rules
Display complex metrics...

... Using processing power from the AWS Cloud
Learn your user preferences…

... And anticipate their needs

Millions of sources producing terabytes of data

IoT Devices

Mobile Application

Serverless Web Dashboard

AWS IoT

Amazon Kinesis

Amazon S3

Amazon Machine Learning
Demo

Documented in our mobile Blog:
• http://bit.ly/1lagagFU
• http://bit.ly/1Pej60L
• http://bit.ly/1RAHuYf

Code
https://github.com/awslabs/aws-device-farm-sample-app-for-android
Demo

Documented in our mobile Blog:
http://bit.ly/1lagaFU
http://bit.ly/1Pej60L
http://bit.ly/1RAHuYf