



XKIT



INTEGRATION

How to integrate your Xkit with AWS IoT Hub

April 2017

www.thinxtra.com

INTRODUCTION

This is a documentation to help you create callbacks in the Sigfox backend to view parsed messages from the Xkit, on AWS.

To be able to complete the flow, you will need:

- a registered Thinxtra Xkit on Sigfox backend
- a Sigfox backend account with at least the customer rights
- an AWS account (trial account is enough)

This documentation is separated in the following different sections:

I/ Prepare Amazon Web Services Account

II/ Configure Sigfox callbacks

III/ Check that messages have been received in AWS IoT

IV/ Create Lambda function parse the payload

V/ Push data into DynamoDB

I/ PREPARE AMAZON WEB SERVICES ACCOUNT



STEP 1 Go to <https://aws.amazon.com/> and create your free trial account.

A screenshot of the AWS Home Page. At the top, there's a navigation bar with 'Services', 'Resource Groups', a search bar, and user information ('Tugberk Bekri', 'N. Virginia', 'Support'). Below the navigation is a section titled 'AWS services' with a search bar. It shows 'Recently visited services' including CloudFormation, Lambda, and DynamoDB, and 'All services' including Compute (EC2), Developer Tools (CodeCommit), Internet of Things, and AWS IoT. To the right, there's a 'Featured next steps' section with links to 'Manage your costs' (with a cost monitor icon) and 'Get best practices' (with a clipboard icon).

AWS services

Find a service by name (for example, EC2, S3, Elastic Beanstalk).

Recently visited services

CloudFormation Lambda DynamoDB

CloudWatch

All services

Compute (EC2) Developer Tools (CodeCommit) Internet of Things AWS IoT

Featured next steps

Manage your costs

Get real-time billing alerts based on your cost and usage budgets. [Start now](#)

Get best practices

Use AWS Trusted Advisor for security, performance, cost and availability best practices. [Start now](#)

STEP 2 Click on “Support” in the top right corner, then “Support Center”.

Copy your account number located in the upper right corner.

A screenshot of the AWS Support Center page. At the top, there's a navigation bar with 'Services', 'Resource Groups', a search bar, and user information ('Tugberk Bekri', 'Global', 'Support'). The 'Support' button is highlighted with a red circle. At the bottom right, the account number 'Account Number 610207528298' is displayed, also enclosed in a red circle.

Support Center

Account Number 610207528298

II/ CONFIGURE SIGFOX CALLBACKS

STEP 1 Connect and login to the Sigfox backend at <https://backend.sigfox.com>

STEP 2 Select “Device Type” tab on the top menu.



STEP 3 Select or type in your device type from the “Name” column and click on it.

A screenshot of the "Device type - List" page in the Sigfox Backend. The left sidebar shows "LIST", "DEVICES BEING TRANSFERRED", "GEOLOCATION PAYLOAD", and "BULK CREATIONS". The main area has a title "Device type - List" and a "New" button. It includes search fields for "Name" (containing "Thinxta_Wisol_Test_Module_Wisc") and "Group" (with a "Select a group" dropdown). There are checkboxes for "Include sub groups" and "Display type" set to "All". Below these are buttons for "RESET" and "FILTER". A message "Name : Thinxta_Wisol_Test_Module_WisolOnly" is displayed. At the bottom, there's a "Count : 1 / 177" message, a "page 1" indicator, and a gear icon for settings. A table lists one device entry: "Thinxta Wisol Test Module Wisol Only" with "None" under "Display type", "Hackathon Event" under "Group", "N/A" under "Keep alive", and "Thinxta_Wisol_Test_Module_WisolOnly" under "Name".

Description	Display type	Group	Keep alive	Name
Thinxta Wisol Test Module Wisol Only	None	Hackathon Event	N/A	Thinxta_Wisol_Test_Module_WisolOnly

II/ CONFIGURE SIGFOX CALLBACKS

STEP 4 Click on “Callbacks” on the left side menu.

STEP 5 Click on “New” on the top right corner.

The screenshot shows the Sigfox Device Type configuration interface. At the top, there is a navigation bar with links: SITE, BASE STATION, DEVICE, DEVICE TYPE (which is underlined), USER, GROUP, RADIO PLANNING, and BILLING. To the right of the navigation bar are icons for user, alert, help, and a refresh arrow, with a red oval highlighting the 'New' button. On the left, a sidebar lists several sections: INFORMATION, LOCATION, ASSOCIATED DEVICES, DEVICES BEING TRANSFERRED, STATISTICS, EVENT CONFIGURATION, and CALLBACKS (which is highlighted with a red oval). The main content area displays the title "Device type 'Thinxtra_Wisol_Test_Module_WisolOnly' - Callbacks" and a note: "These callbacks transfer data received from the devices associated to this device type to your infrastructure. For more informations, please refer to the [Callback documentation](#)". A red oval also highlights the "New" button in the top right corner of the main content area.

II/ CONFIGURE SIGFOX CALLBACKS

STEP 6 Choose AWS IoT as a new callback.

The screenshot shows the Sigfox Device Type configuration interface. On the left, a sidebar menu includes options like INFORMATION, LOCATION, ASSOCIATED DEVICES, DEVICES BEING TRANSFERRED, STATISTICS, EVENT CONFIGURATION, CALLBACKS (which is selected and highlighted in purple), and BULK CREATIONS. The main content area is titled "Device type 'Thinxtra_Wisol_Test_Module_WisolOnly' - New Callback". It contains instructions for creating callbacks to connect the Sigfox cloud to a server/platform. Two callback types are listed: "Custom callback" (represented by a cloud icon) and "AWS IoT" (represented by a blue shield icon). The "AWS IoT" section is enclosed in a red box, indicating it is the chosen option. Below the "AWS IoT" section, a note states: "AWS IoT is a managed cloud platform that lets connected devices easily and securely interact with cloud applications and other devices. AWS IoT can support billions of devices and trillions of messages, and can process and route those messages to AWS endpoints and to other devices reliably and securely."

You can also work on AWS Kinesis if you have an account. For the sake of this tutorial, we will be using AWS IoT.

II/ CONFIGURE SIGFOX CALLBACKS

STEP 7 Copy the “External Id” given to you in your clipboard.

The screenshot shows the Sigfox Device Type Configuration interface. The left sidebar has a purple background and lists: INFORMATION, LOCATION, ASSOCIATED DEVICES, DEVICES BEING TRANSFERRED, STATISTICS, EVENT CONFIGURATION, CALLBACKS (which is highlighted in purple), and BULK CREATIONS. The main content area has a white background and displays the configuration for a device type named "Device type Thinxtra_Wisol_Test_Module_WisolOnly - Callback new". The "Callbacks" section contains the following fields:

- Config method: CROSS_ACCOUNT (dropdown menu)
- Launch Stack (yellow button)
- External Id: A redacted input field.
- ARN Role: Paste your ARN role id here
- Topic: Enter your topic here
- Region: Select to change value (dropdown menu)
- Custom payload config
- Json Body

STEP 8 Click on “Launch Stack”.

II/ CONFIGURE SIGFOX CALLBACKS

STEP 9 You have been redirected to the AWS CloudFormation console, click on “Next”.

The screenshot shows the AWS CloudFormation 'Create stack' wizard. The top navigation bar includes 'Services', 'Resource Groups', a star icon, a bell icon, 'Tugberk Bekri', 'N. Virginia', and 'Support'. The breadcrumb path shows 'CloudFormation' > 'Stacks' > 'Create Stack'. The main section is titled 'Create stack' and 'Select Template'. On the left, a sidebar lists 'Select Template' (which is selected and highlighted in orange), 'Specify Details', 'Options', and 'Review'. The main content area has two sections: 'Design a template' (with a 'Design template' button) and 'Choose a template' (with three options: 'Select a sample template' (radio button), 'Upload a template to Amazon S3' (with a 'Choose File' button and 'No file chosen'), and 'Specify an Amazon S3 template URL' (radio button selected, with a text input field containing 'https://s3-eu-west-1.amazonaws.com/cloud-formation-script/Sigfox_Cross' and a 'View/Edit template in Designer' link). At the bottom right, there are 'Cancel' and 'Next' buttons, with 'Next' being circled in red.

II/ CONFIGURE SIGFOX CALLBACKS

STEP 10

Enter the following inputs:

- *Stack name*: choose a meaningful name
- *AWSSAccountId*: paste the Id you copied earlier
- *External Id*: paste the Id off of your Sigfox backend
- *Region*: enter the region name that is located your URL
- *Topic name*: choose the topic name you wish to send data to

The screenshot shows the AWS CloudFormation 'Create stack' wizard at the 'Specify Details' step. The 'Stack name' field contains 'ThinextraXKitConnector'. The 'Parameters' section includes fields for 'AWSAccountId' (with a placeholder 'Input your AWS Account ID.'), 'ExternalId' (with a placeholder 'Please copy/paste the External Id assigned to you in the Sigfox console.'), 'Region' set to 'us-east-1', and 'TopicName' set to 'thinextra' (with a placeholder 'Input the name of the topic Sigfox will use to publish data.'). At the bottom right are buttons for 'Cancel', 'Previous', and 'Next'.

Click on “Next”.

II/ CONFIGURE SIGFOX CALLBACKS

- STEP 11** You can customize the next screen optionally, or click “Next”, then the acknowledgment box and “Create”.
- STEP 12** Wait for the stack creation completion. Click on “Outputs”. Copy the value of “ARNRole”, “Region” and “Topic”.

The screenshot shows the AWS CloudFormation console interface. At the top, there's a navigation bar with 'Services', 'Resource Groups', and other account information. Below it, the 'CloudFormation' section is selected, showing a list of stacks. A single stack named 'ThinextraXKitConnector' is listed with a status of 'CREATE_COMPLETE'. The 'Outputs' tab is highlighted with a red circle, and its content is also circled in red. The 'Outputs' table has columns for Key, Value, Description, and Export Name. It contains three entries: ARNRole (with a red box around the Value column), Region (value: us-east-1), and Topic (value: thinextra).

Key	Value	Description	Export Name
ARNRole	Please copy/paste the ARN of the cross account role...		
Region	us-east-1	Please copy/paste the name of the AWS Region into ...	
Topic	thinextra	Please copy/paste the name of IoT topic into the Sig...	

II/ CONFIGURE SIGFOX CALLBACKS

STEP 13

Go back to the Sigfox backend and fill in the following fields:

- ARN Role: paste the value you copied in AWS CloudFormation
- Topic: paste the value you copied earlier
- Region: choose the region from your Cloud Formation stack
- Custom payload config:

```
int1::uint:16:little-endian int2::uint:16:little-endian int3::uint:16:little-endian int4::int:16:little-endian  
int5::int:16:little-endian int6::int:16:little-endian
```

Essentially, this payload configuration separates the bytes of the payload into 6 defined data sections as below. Note that the payload is just separated and not decoded yet.

	Integer 2 bytes		Integer 2 bytes		Integer 2 bytes		Signed Integer 2 bytes		Signed Integer 2 bytes		Signed Integer 2 bytes	
Received frame on the backend	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB
	TEMPERATURE		PRESSURE		PHOTO		X_Acc		Y_Acc		Z_Acc	
Actual measured value												
	Temperature/100 (°C)		Pressure*3 (Pa)		Photo/1000 (V)		X_Acc/250 (g)		Y_Acc/250 (g)		Z_Acc/250 (g)	

Convert the received values on the backend to the actual measured values

II/ CONFIGURE SIGFOX CALLBACKS

STEP 13 (continued)

- Json Body:

```
{  
  "deviceid": "{device}",  
  "time": "{time}",  
  "snr": "{snr}",  
  "station": "{station}",  
  "latitude": "{lat}",  
  "longitude": "{lng}",  
  "rss": "{rss}",  
  "avgSnr": "{avgSnr}",  
  "seqNumber": "{seqNumber}",  
  "data": {  
    "Temperature": "{customData#int1}",  
    "Pressure": "{customData#int2}",  
    "Photo": "{customData#int3}",  
    "x_Accelerator": "{customData#int4}",  
    "y_Accelerator": "{customData#int5}",  
    "z_Accelerator": "{customData#int6}"  
  }  
}
```

The screenshot shows the Sigfox Device Type configuration interface. The top navigation bar includes links for SITE, BASE STATION, DEVICE, DEVICE TYPE (which is currently selected), USER, GROUP, RADIO PLANNING, BILLING, and user profile icons. The left sidebar has sections for INFORMATION, LOCATION, ASSOCIATED DEVICES, DEVICES BEING TRANSFERRED, STATISTICS, EVENT CONFIGURATION, CALLBACKS (which is highlighted in purple), and BULK CREATIONS. The main content area displays the configuration for a device type named "Thinxtra_Wisol_Test_Module_WisolOnly". Under the CALLBACKS section, there is a "Callbacks" configuration panel. It shows the "Config method" set to "CROSS_ACCOUNT" with a "Launch Stack" button. Fields for "External Id" and "ARN Role" are present, along with "Topic" set to "thinxta" and "Region" set to "US_EAST_1". A "Custom payload config" field contains the JSON body provided in the code block above. The JSON body is:

```
{  
  "deviceid": "{device}",  
  "time": "{time}",  
  "snr": "{snr}",  
  "station": "{station}",  
  "latitude": "{lat}",  
  "longitude": "{lng}",  
  "rss": "{rss}",  
  "avgSnr": "{avgSnr}",  
  "seqNumber": "{seqNumber}",  
  "data": {  
    "Temperature": "{customData#int1}",  
    "Pressure": "{customData#int2}",  
    "Photo": "{customData#int3}",  
    "x_Accelerator": "{customData#int4}",  
    "y_Accelerator": "{customData#int5}",  
    "z_Accelerator": "{customData#int6}"  
  }  
}
```

III/CHECK THAT MESSAGES HAVE BEEN RECEIVED IN AWS IOT

- STEP 1** To check if your callback went through, go into “Device” tab, enter your device name .
- STEP 2** Click on “Messages” and check if there is a green arrow on the right side of your callback. It is a confirmation that your callback went through without issues.

The screenshot shows the Sigfox Device Management interface. The top navigation bar includes tabs for SITE, BASE STATION, DEVICE (which is highlighted with a red box), DEVICE TYPE, USER, GROUP, RADIO PLANNING, and BILLING. On the far right of the top bar are icons for user profile, alert, help, and refresh. Below the top bar, a sidebar on the left lists categories: INFORMATION, LOCATION, MESSAGES (which is highlighted with a red box), EVENTS, STATISTICS, and EVENT CONFIGURATION. The main content area is titled "Device 7FACC - Messages". It features a search bar with "From date" and "To date" fields, a dropdown for "Type" set to "All", and buttons for "RESET", "FILTER", and "CSV". Below the search bar is a table header with columns: Time, Delay (s), Header, Data / Decoding, Location, Base station, RSSI (dBm), SNR (dB), Freq (MHz), Rep, and Callbacks. The table contains two data rows. The first row corresponds to the timestamp 2017-03-09 10:24:36 and has three entries under the "Callbacks" column, each marked with a green upward-pointing arrow icon. The second row corresponds to the timestamp 2017-03-09 09:24:29 and also has three entries under the "Callbacks" column, each marked with a green upward-pointing arrow icon.

Time	Delay (s)	Header	Data / Decoding	Location	Base station	RSSI (dBm)	SNR (dB)	Freq (MHz)	Rep	Callbacks
2017-03-09 10:24:36	1.7	0000	000000000000000000000000000000001c1400	↗	360A 239B 34B5	-128.00 -67.00 -50.00	16.00 76.43 93.20	920.7857 920.8144 920.7857	2 3 3	↗ ↗ ↗
2017-03-09 09:24:29	1.3	0000	000000000000000000000000000000001b1300	↗	239C 239B 34B5	-115.00 -67.00 -52.00	29.00 76.72 91.57	920.8630 920.7971 920.8374	3 3 3	↗ ↗ ↗

III/CHECK THAT MESSAGES HAVE BEEN RECEIVED IN AWS IOT

STEP 3 Go back into AWS IoT. Go into “Services” in the upper left corner.

STEP 4 Click on AWS IoT, then “Get started”.

STEP 5 On the left side menu, you will find “Test”, click on it.

The screenshot shows the AWS IoT MQTT client interface. At the top, there is a navigation bar with 'Services' (highlighted with a red circle), 'Resource Groups', and a star icon. To the right are notifications, user 'Tugberk Bekri', region 'N. Virginia', and support links. The main area has a title 'MQTT client' with a question mark icon and a status 'Connected as iotconsole-1489016820482-0'. On the left, a sidebar lists 'AWS IoT' with icons for Dashboard, Connect, Registry, Security, Rules, and Test (highlighted with a red circle). The main content area is titled 'Subscriptions' and contains a 'Subscribe to a topic' section. It includes a description, a 'Subscription topic' input field containing 'thinextra', a 'Max message capture' input field with value '100', and 'Quality of Service' radio buttons set to '0'. A 'Subscribe to topic' button is at the bottom.

III/ CHECK THAT MESSAGES HAVE BEEN RECEIVED IN AWS IOT

STEP 6 Enter the subscription topic you previously chose in your Sigfox callback. Click on “Subscribe to topic”.

STEP 7 Send a message through your Xkit. Click on your subscription topic. You will see your undecoded message.

The screenshot shows the AWS IoT MQTT client interface. On the left, there's a sidebar with icons for Dashboard, Connect, Registry, Security, Rules, and Test. The main area is titled "MQTT client" and shows a subscription named "thinxtra". A message has been received, and its content is displayed in a code block. The message payload is as follows:

```
{
  "deviceId": "2BF340",
  "time": "1489636095",
  "snr": "23.02",
  "station": "239B",
  "latitude": "-34.0",
  "longitude": "151.0",
  "rssi": "-120.00",
  "avgSnr": "avgSnr",
  "seqNumber": "195",
  "data": {
    "Temperature": "2803",
    "Pressure": "33612",
    "Photo": "3203",
    "x_Accelerator": "-7",
    "y_Accelerator": "7",
    "z_Accelerator": "241"
  }
}
```

The "thinxtra" subscription name and the first few lines of the message content are highlighted with red boxes.

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

In this section, you will create a Lambda function that will decode the payload and push it in a Dynamo Database.

By referring back to the “X-kit configuration”, we know that we need to:

- divide the Temperature data by 100
- multiply the Pressure data by 3
- divide the Photo data by 1000
- divide each of the Accelerator data by 250
- decode the Unix timestamp to a “human-readable” time

STEP 1 Go into “Services” in the top left corner and type in “IAM”.

STEP 2 Click on “Roles” on the left side menu and click on “Create New Role”.



IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

STEP 3 Add a name for your new role. Select “AWS Lambda” as a Role Type.

The screenshot shows the 'Select Role Type' step of the 'Create Role' wizard. On the left, a sidebar lists steps: Step 1 : Set Role Name, Step 2 : Select Role Type (which is active), Step 3 : Establish Trust, Step 4 : Attach Policy, and Step 5 : Review. The main area is titled 'Select Role Type' and shows the 'AWS Service Roles' section. It lists several services with their descriptions and 'Select' buttons. The 'AWS Lambda' option is highlighted with a red box around its row. Other listed services include Amazon EC2, AWS Directory Service, Amazon Redshift, and Amazon API Gateway. Below this section are two other options: 'Role for Cross-Account Access' and 'Role for Identity Provider Access'. The top navigation bar includes 'Services', 'Resource Groups', a search bar, and account information for 'Tugberk Bekri'.

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

STEP 4 Attach the following policies:

Policy Name
 AmazonDynamoDBFullAccess
 AWSLambdaDynamoDBExecutionRole
 AWSIoTDataAccess
 AWSIoTFullAccess

STEP 5 Press “Next” and “Create Role.

You have now created a special IAM role that will allow your function to use the AWS IoT service and write into a Dynamo DB

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

STEP 6 Go into your account tab in the upper right corner, click on “My Security Credentials”, then click on “Access Keys” and “Create a new access key”.

The screenshot shows the AWS IAM 'Your Security Credentials' page. On the left, there's a sidebar with links like Dashboard, Groups, Users, Roles, Policies, Identity providers, Account settings, Credential report, and Encryption keys. The main area has sections for Password, Multi-Factor Authentication (MFA), and Access Keys (Access Key ID and Secret Access Key). The 'Access Keys' section is highlighted with a red box. Below it, a note says you can have a maximum of two access keys. A table lists one existing access key: Created (Mar 14th 2017), Deleted (grey bar), Access Key ID (redacted), Last Used (2017-03-16 10:58 UTC+1100), Last Used Region (us-east-1), Last Used Service (execute-api), Status (Active), and Actions (Make Inactive | Delete). At the bottom, a 'Create New Access Key' button is highlighted with a red box. In the top right, there's a dropdown menu with options: My Account, My Organization, My Billing Dashboard, My Security Credentials (highlighted with a red box), and Sign Out. The top navigation bar includes Services, Resource Groups, a search bar, and user information (Tugberk Bekri) with a notification icon.

STEP 7 Save your access key and secret key in a safe place. You will need them in the next step.

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

STEP 8 You will now need to compute your IoT endpoint. To do so, you will need to download the Command Line Interface corresponding to your running OS. Follow the option that suits you best.

Option A: if you have pip (a package manager for Python), follow the steps at:

<http://docs.aws.amazon.com/cli/latest/userguide/installing.html>

Option B: if you don't have pip, following the steps at:

- For Linux, Unix, macOS: <http://docs.aws.amazon.com/cli/latest/userguide/awscli-install-bundle.html>
- For Windows: <http://docs.aws.amazon.com/cli/latest/userguide/awscli-install-windows.html#install-msi-on-windows>

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

STEP 9 Now that your AWS Command Line Interface is set up, go into your console/terminal and type in:

```
aws configure
```

STEP 10 Your console will ask you for your access key. Add it and press enter.

STEP 11 Your console will then ask you for your Secret Access Key. Add it and press enter.

STEP 12 Your console will then ask you for your Region. Add it and press enter.

STEP 13 Type in the following:

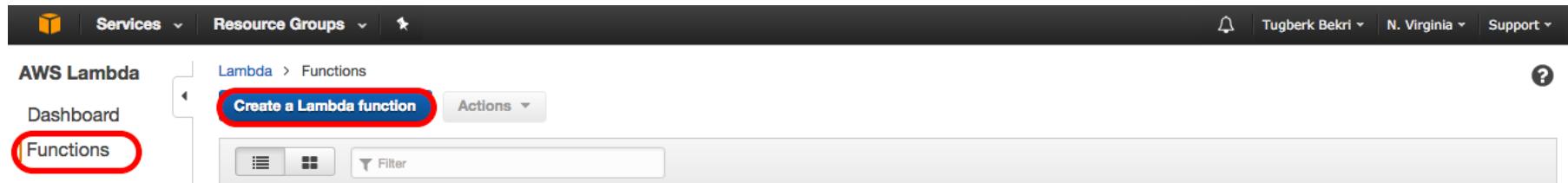
```
aws iot describe-endpoint
```

STEP 14 Copy your AWS IoT endpoint.

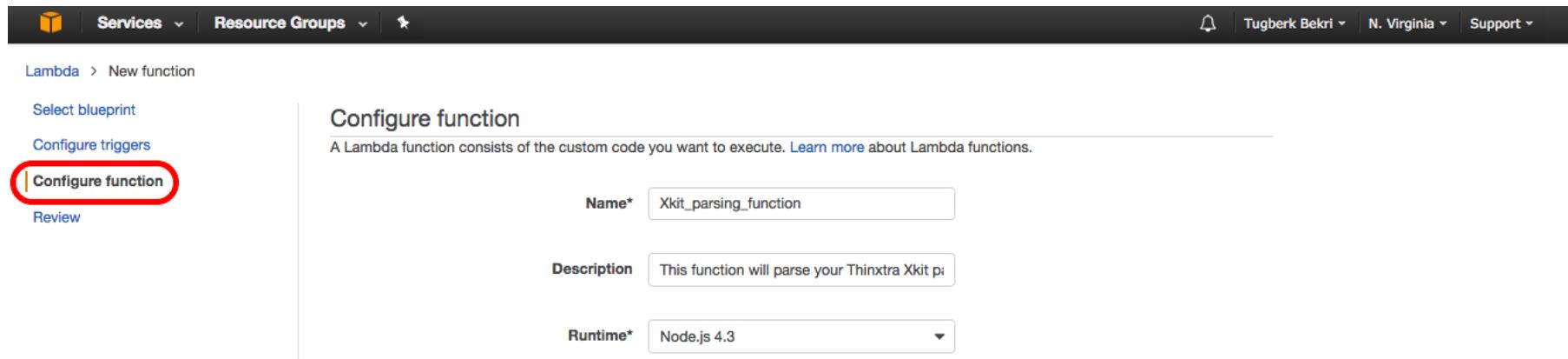
IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

STEP 15 Go back into your AWS IoT console. Go into “Services” at the top right and search for “AWS Lambda”.

STEP 16 Go into “Functions” and “Create a Lambda function”.



STEP 17 Click on “Configure function” on the left-side menu. Add a new name for your function and select “Node.js 4.3”.



IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

STEP 18 Write your function in the code section as followed.

You can download the full code at: <http://www.thinxtra.com/download/6078/>

```
console.log('Loading event');
var AWS = require('aws-sdk');
var dynamodb = new AWS.DynamoDB();
var iotData = new AWS.IotData({endpoint: 'your_AWS_endpoint_you_copied_earlier'});

//make sure you replace your_AWS_endpoint_you_copied_earlier with the value you copied just before

exports.handler = function(event, context) {
    console.log("Request received:\n", JSON.stringify(event));
    console.log("Context received:\n", JSON.stringify(context));

    var tableName = event.device;
    var time = event.time;
    var Temperature = event.data.Temperature;
    var Pressure = event.data.Pressure;
    var Photo = event.data.Photo;
    var x_Accelerator = event.data.x_Accelerator;
    var y_Accelerator = event.data.y_Accelerator;
    var z_Accelerator = event.data.z_Accelerator;
    var Accelerator = event.data.Accelerator;
```

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

```
function returnTime(value){  
    return new Date((value)*1000 + 39600000);  
}  
  
//replace 39600000 with the Epoch time offset according to your timezone. Here it is set to Sydney time  
offset.  
  
timedecode = '' +returnTime(time).toLocaleString();  
Temperature = (Temperature)/100;  
Pressure = (Pressure)*3;  
Photo = (Photo)/1000;  
x_Accelerator = (x_Accelerator)/250;  
y_Accelerator = (y_Accelerator)/250;  
z_Accelerator = (z_Accelerator)/250;  
Accelerator = Math.sqrt(x_Accelerator * x_Accelerator + y_Accelerator * y_Accelerator + z_Accelerator *  
z_Accelerator);  
  
var payloadObj={ "state": {  
    "reported": {  
        "device": event.device,  
        "time": event.time,  
        "timedecode": timedecode,  
    }  
};
```

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

```
"snr": event.snr,  
"avgSnr": event.avgSnr,  
"station": event.station,  
"Temperature": Temperature,  
"Pressure": Pressure,  
"Photo": Photo,  
"x_Accelerator": x_Accelerator,  
"y_Accelerator": y_Accelerator,  
"z_Accelerator": z_Accelerator,  
"Accelerator": Accelerator  
},  
}  
};  
  
var paramsUpdate = {  
    thingName : event.deviceId,  
    payload : JSON.stringify(payloadObj)  
};  
  
//This function will update the Device Shadow State  
  
iotData.updateThingShadow(paramsUpdate, function(err, data) {
```

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

```
if (err){  
    console.log("Error in updating the Thing Shadow");  
    console.log(err, err.stack);  
}  
});  
  
//This function will store the message in a dynamoDB table  
dynamodb.putItem({  
    "TableName": event.deviceId,  
    "Item": {  
        "deviceId": {  
            "S": event.deviceId  
        },  
        "time": {  
            "S": event.time  
        },  
        "timedecode": {  
            "S": timedecode.toString()  
        },  
        "Temperature": {  
            "S": Temperature.toString()  
        },  
        "Pressure": {  
            "S": Pressure.toString()  
        }  
    }  
});
```

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

```
        "S": Pressure.toString()
    },
    "Photo": {
        "S": Photo.toString()
    },
    "x_Accelerator": {
        "S": x_Accelerator.toString()
    },
    "y_Accelerator": {
        "S": y_Accelerator.toString()
    },
    "z_Accelerator": {
        "S": z_Accelerator.toString()
    },
    "Accelerator": {
        "S": Accelerator.toString()
    }
}
```

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

```
function(err, data){  
    if (err) {  
        context.fail('ERROR: Dynamo failed: ' + err);  
    } else {  
        console.log('Dynamo Success: ' + JSON.stringify(data, null, ' '));  
        context.succeed('SUCCESS');  
    }  
};  
};
```

STEP 19 Set role as “Choose an existing role”, choose the IAM role name you’ve created previously.

STEP 20 In the advanced settings, set your timeout to 5min.

▼ Advanced settings

These settings allow you to control the code execution performance and costs for your Lambda function. Changing your resource settings (by selecting memory) or changing the timeout may impact your function cost. [Learn more](#) about how Lambda pricing works.

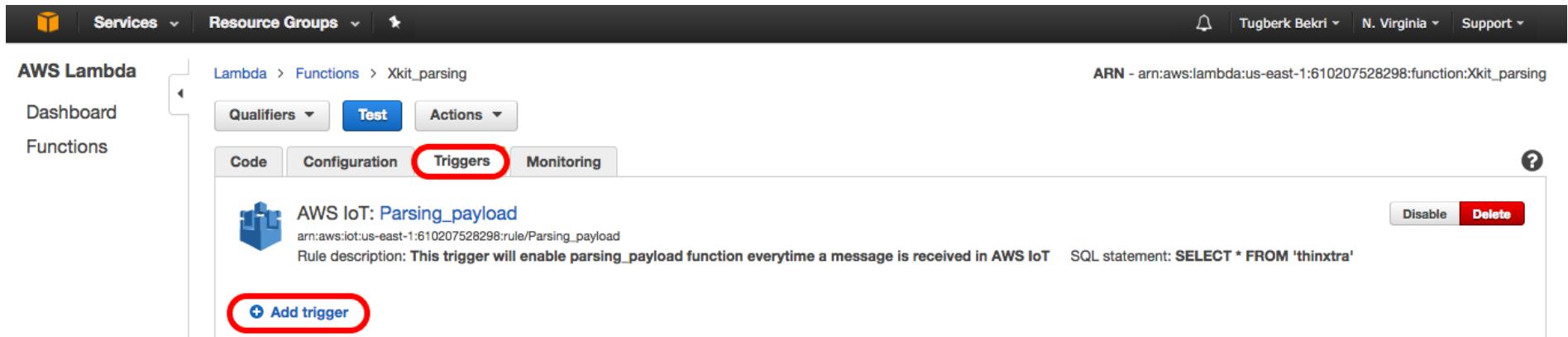
The screenshot shows the 'Advanced settings' section of the AWS Lambda configuration interface. It includes fields for 'Memory (MB)' (set to 128) and 'Timeout' (set to 5 min 0 sec). The 'Timeout' field is highlighted with a red oval.

Memory (MB)	128
Timeout	5 min 0 sec

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

STEP 21 Click on “Create”.

STEP 22 Now click on your function. Click on “Triggers” tab, then “Add a trigger”.



STEP 23 Choose “AWS IoT” as an input.

STEP 24 Choose “Custom IoT rule” as IoT Type.

IV/ CREATE A LAMBDA FUNCTION TO PARSE THE PAYLOAD

STEP 25 Add a rule name. Enter the following SQL statement and click on “Submit”.

```
SELECT * FROM 'your_topic'
```

Make sure to change `your_topic` with the topic you entered in your callback in the Sigfox backend.

Add trigger

Configure your Lambda function `Xkit_parsing` to respond to events from the selected trigger. Click on the box below to select your trigger type.

AWS IoT  → Lambda 

Warning: Altering the description or SQL statement of an existing rule will overwrite it. 

For more information about IoT rules and SQL statements, please see the [AWS IoT documentation](#). Lambda will add the necessary permissions for AWS IoT to invoke your Lambda function. [Learn more](#) about the Lambda permissions model.

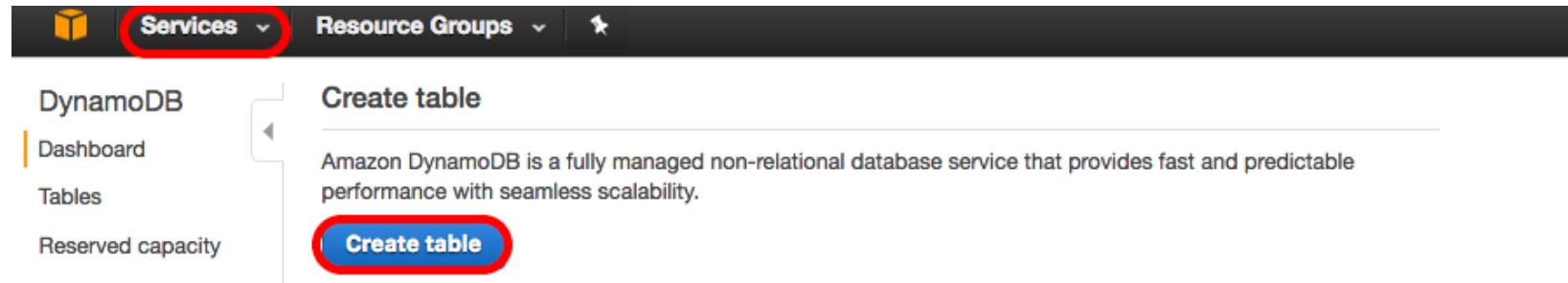
IoT Type	Custom IoT rule	
Rule name	parsing	
Rule description	<input type="text"/>	
SQL statement	SELECT * FROM 'thinxtre'	

Enable trigger 

V/ PUSH DATA INTO DYNAMODB

STEP 1 Go to services and click on “DynamoDB”. Click on “Create table”.



STEP 2 Add your deviceld (Xkit Id on the Sigfox backend) as a Table name.

It is very important to name your table with your deviceld since it is how we configured our Lambda function.

V/ PUSH DATA INTO DYNAMODB

STEP 3 Add ‘deviceld’ as a primary key and ‘time’ as a sort key.

Click on “Create”.

The screenshot shows the 'Create DynamoDB table' wizard. At the top, there's a navigation bar with 'Services', 'Resource Groups', a search bar, and account information for 'Tugberk Bekri' in 'N. Virginia'. Below the navigation bar is the title 'Create DynamoDB table' with a 'Tutorial' and a help icon. A descriptive text explains that DynamoDB is schema-less and requires a table name and primary key. The primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

The main configuration area has 'Table name*' set to '2BF340' (highlighted with a red box). Under 'Primary Key*', 'deviceld' is listed as a 'Partition key' of type 'String'. A checkbox 'Add sort key' is checked, and 'time' is listed as a 'String' sort key. In the 'Table settings' section, a checkbox 'Use default settings' is checked, and a list of three items is shown: 'No secondary indexes.', 'Provisioned capacity set to 5 reads and 5 writes.', and 'Basic alarms with 80% upper threshold using SNS topic "dynamodb".' At the bottom, a note states that additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console. The bottom right features 'Cancel' and 'Create' buttons, with 'Create' being highlighted.

V/ PUSH DATA INTO DYNAMODB

STEP 4 Once you have created your table, click on it and go into the items Tab.

STEP 5 Send a message from your Xkit, you should be seeing the parsed payload coming in.

The screenshot shows the AWS DynamoDB console. On the left, the navigation bar includes 'Services' (selected), 'Resource Groups', and a user dropdown for 'Gildas Seimblle'. Below that, 'DynamoDB' is selected, followed by 'Dashboard', 'Tables' (which is currently selected), and 'Reserved capacity'. A sidebar on the left lists 'Create table' and 'Actions' with a dropdown menu. The main area shows a table titled '2BEE65' with three items. The table has columns: deviceld, time, Accelerator, Photo, Pressure, Temperature, timedecode, x_Accelerator, y_Accelerator, and z_Accelerator. The first item is selected.

	deviceld	time	Accelerator	Photo	Pressure	Temperature	timedecode	x_Accelerator	y_Accelerator	z_Accelerator
<input checked="" type="checkbox"/>	2BEE65	1490836760	1.0137849870...	0.356	100854	27.56	Thu Mar 30 2017 12:19:20 GMT+...	0.06	0.004	1.012
<input type="checkbox"/>	2BEE65	1490848265	1.0001999800...	0.439	100698	28.21	Thu Mar 30 2017 15:31:05 GMT+...	0	0.02	1
<input type="checkbox"/>	2BEE65	1490848513	0.9104108962...	0.395	100704	28.21	Thu Mar 30 2017 15:35:13 GMT+...	-0.06	0.028	0.908

V/ PUSH DATA INTO DYNAMODB

STEP 6 Click on your deviceld. You should be able to see your latest full decoded message.

```
Accelerator String : 0.9529491067208153
deviceid String : 2BF340
Photo String : 1.074
Pressure String : 100764
Temperature String : 29.2
timedecode String : Thu Mar 16 2017 15:44:43 GMT+0000 (UTC)
timestamp String : 1489639483
x_Accelerator String : -0.032
y_Accelerator String : 0.028
z_Accelerator String : 0.952
```

You have now sent your message through the Sigfox backend to AWS IoT Hub. You parsed your payload via a Lambda function and pushed the data to a DynamoDB table.

If you wish to move further with the integration and visualize the message in a more graphic matter, you can push your DynamoDB data to AWS ElasticSearch service coupled with a third-party Visualisation dashboard tool.



www.thinxtra.com/xkit/

More infos on AWS website:

aws.amazon.com/fr/blogs/iot/connect-your-devices-to-aws-iot-using-the-sigfox-network/