# Skywire® 3G HSPA AWS IoT with TLS

**NimbeLink Corp**

**Updated: October 2018**

# Table of Contents



# 1. Introduction

## 1.1 Overview

This document serves as a guide for Amazon AWS connections using the NimbeLink 3G HSPA Skywire. This tutorial will document the configuration of the modem and the Amazon AWS settings, and will demonstrate two different connection examples. Please note that TLS connections only work on the NL-SW-HSPA-B modem running firmware version 12.00.009 or higher.

## 1.2 Orderable Parts

| Orderable Device | Description | Carrier | Network Type |
|---|---|---|---|
| NL-SWDK | Skywire Development Kit | Any | Any |
| NL-SW-HSPA-B | 3G Global HSPA | Any | GSM |

# 2. AWS IoT Setup

## 2.1 Preliminary Setup

Before starting, it is important to note that this guide assumes that the reader already has a valid Amazon AWS account. If this is not the case, Amazon offers a free trial account that can be used to test this guide. For more information about the free account, please follow this link:

https://aws.amazon.com/free/

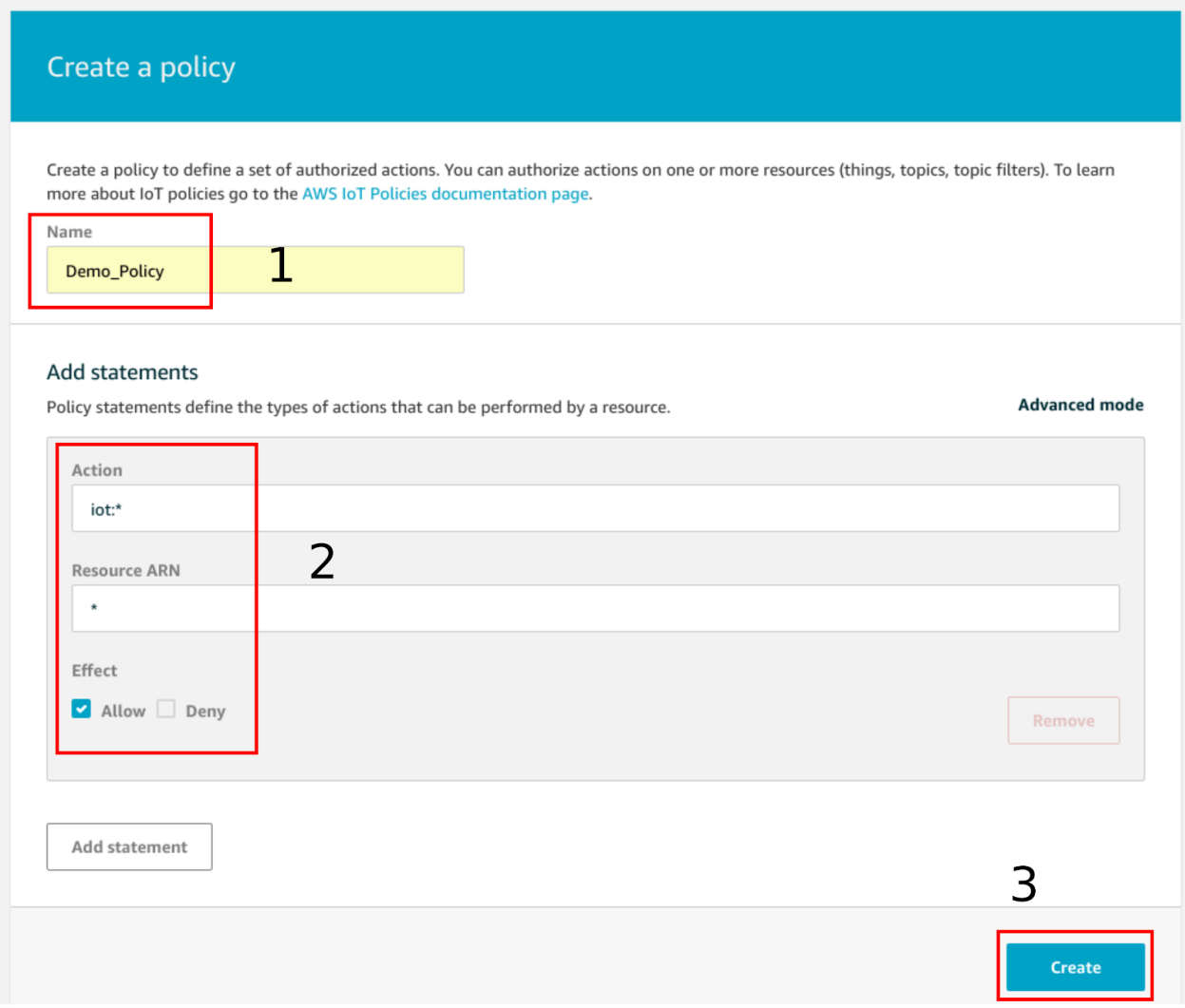
## 2.2 Create a Policy

The first step in the AWS connection process is to create a policy. Login to the AWS IoT console at the following link:

https://console.aws.amazon.com/iot/

and navigate to the 'Secure' > 'Policies' menu. Once there, press the “Create a policy" button located near the center of the screen.

In the next page, choose “iot:*" for the “Action" and “*" for the “Resource ARN" field. Check the “Allow" box, and then click "Add Statement". Finally, click "Create" to create the policy. Refer to the image below.



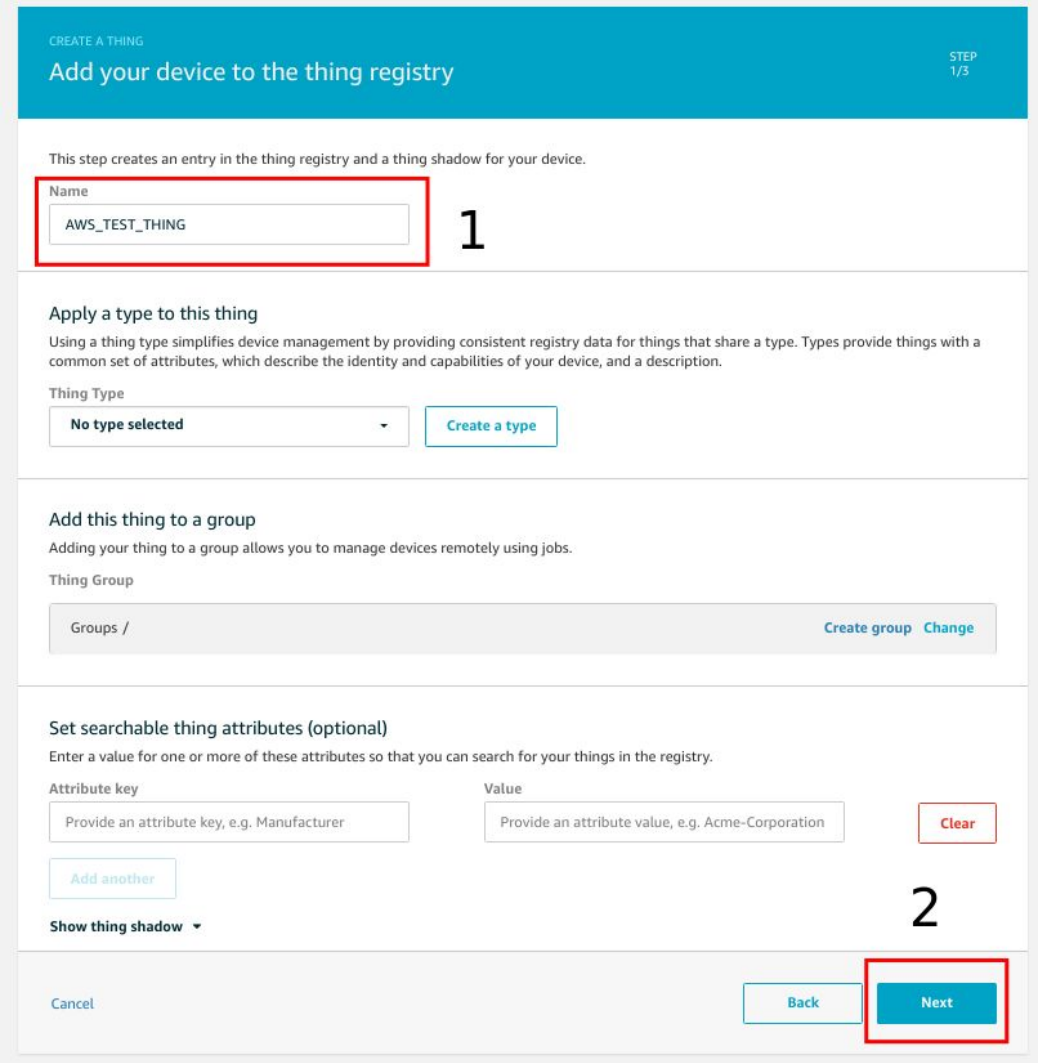

  

## 2.3 Create a "Thing"

Navigate to 'Manage' > 'Things' using the menu on the left-hand side of the dashboard. Next, select ‘Create’ in the top right corner to make a new “thing". Refer to the image below for reference.

After pressing the “Create" button, select the “Create a single thing" option in the next page that loads. In the following page, enter a custom name in the appropriate box, and then press the “Next" button. The webpage should look something like this:

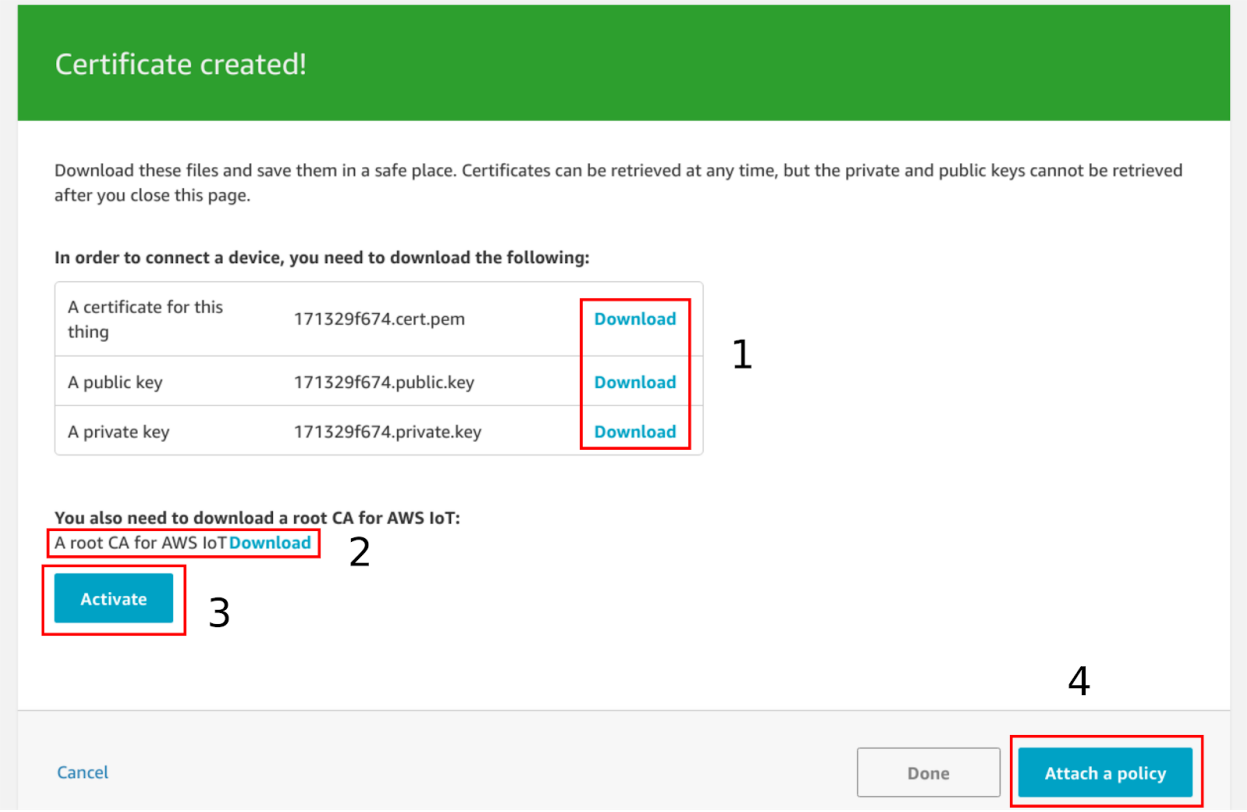


## 2.4 Generate Certificates

After pressing the “Next" button, select the “Create certificate" option in the next web page that loads. Amazon AWS will then generate a client certificate, private key, and a public key for the “thing" that was just created. Download these certificates and save them in a convenient place. Also, be sure to download the Amazon AWS CA certificate as this will be needed for the TLS connection.

Next, press the “Activate" button to assign the generated certificates to the “thing". Finally, click “Attach a policy" to proceed to the next step. Refer to the image below for reference.

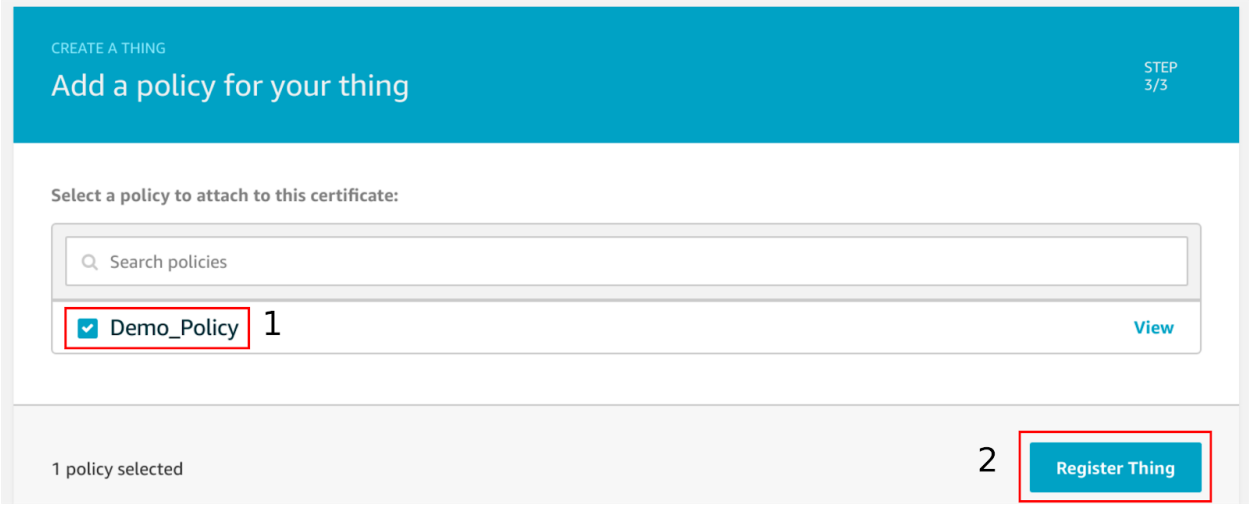



**Note:** The public and private key can only be downloaded from this page. Once this page is navigated from, these files will no longer be available for download.

## 2.5 Attach the Policy to the "Thing"

After advancing to the next page, attach the policy created in Section 2.2 to the “thing" created in Section 2.3. Refer to the image below as an example.




After each of the steps in Section 2 have been completed, proceed to Section 3 for the Skywire configuration instructions.

# 3. Skywire Configuration

## 3.1 Uploading Certificates

The first step in the configuration of the Skywire is to upload the certificates needed for the TLS connection. These three certificates are the private key, client certificate and the CA certificate.

The next two sections will lay out instructions for uploading the certificates in a Linux and a Windows environment. Section 3.1.1 details the Linux instructions, while Section 3.1.2 contains the Windows instructions.

### 3.1.1 Certificate Uploading Using a Linux Environment

First, open a terminal and navigate to the directory that contains the certificates that were downloaded in Section 2.4. Type "**`ls -l`**" to list the contents of the directory on individual lines. Take note of the file sizes of each of the relevant certificates. This information will be needed shortly.

Next, establish a connection to the NL-SW-HSPA-B using a preferred serial console. Once the serial console has been setup properly, issue the following commands to determine if there are any SSL certificates stored on the NL-SW-HSPA-B in non-volatile memory:

```
AT#SSLSECDATA=1,2,0

AT#SSLSECDATA=1,2,1

AT#SSLSECDATA=1,2,2
```

The third argument to the above AT command corresponds to the client certificate, CA certificate and the private key, respectively.

If there are already certificates on the modem, the terminal will output the contents of the certificates to the console. However, if there are not any certificates currently stored in the non-volatile memory, the terminal will respond with something similar to:

```
#SSLSECDATA: 1,0

No data stored

OK
```

If there are existing certificates stored on the modem, it is recommended to delete them. To do so, issue the following commands, which will delete the pre-existing certificates:

**AT#SSLSECDATA=1,0,0**

**AT#SSLSECDATA=1,0,1**

**AT#SSLSECDATA=1,0,2**

After clearing out the certificates, individually upload each of the three requisite certificates by following the process below:

1. Using the information returned by the "**ls -l**" command, determine the number of bytes for the certificate that is to be uploaded.
2. Issue the AT command below:

   **AT#SSLSECDATA=1,1,0,1224**
   - The first "**1**" is the 'storage identifier', and will always be constant for NL-SW-HSPA-B modems.
   - The second "**1**" indicates that this a storage operation involving the non-volatile memory.
   - "**0**" indicates that this operation is writing the client certificate to memory. Change this parameter to "**1**" or "**2**" when writing the CA certificate, and the private key, respectively.
   - Finally, "**1224**" is the size in bytes of the certificate being written to the non-volatile memory. This parameter will also change based on the certificate being written.

3. The modem will respond with "**>**", and will wait for data to be entered. This data will be entered using the Linux terminal in step 4.
4. First, disconnect the serial terminal from the "**/dev/ttyUSB***" port. Then, issue the following command in the Linux terminal, where the name of the file is replaced with the name of the certificate in question, and the destination is replaced with the appropriate path to the serial line.

   **cat 8da6fe87f3-certificate.pem > /dev/ttyUSB0**

   This command will pipe the contents of the certificate to the serial line, which will then be stored in a file on the NL-SW-HSPA-B. After issuing the above command into the linux terminal, return to the AT command line and press "CTRL+Z". This will finalize the writing of the certificate.

   If the upload was successful, the modem will respond with:

   **OK**

Repeat the four steps listed on the previous page until each of the three files have been uploaded. Once the files are confirmed to have been uploaded successfully, proceed to Section 3.2.

### 3.1.2 Certificate Uploading Using a Windows Environment

First open the Windows command prompt and navigate to the directory that contains the certificates that were downloaded in Section 2.4. Type "**dir**" to list the contents of the directory on individual lines. Take note of the file sizes of each of the relevant certificates. This information will be needed shortly.

Next, establish a connection to the NL-SW-HSPA-B using a preferred serial console. Once the serial console has been setup properly, issue the following commands to determine if there are any SSL certificates stored on the NL-SW-HSPA-B in non-volatile memory:

**AT#SSLSECDATA=1,2,0**

**AT#SSLSECDATA=1,2,1**

**AT#SSLSECDATA=1,2,2**

The third argument to the above AT command corresponds to the client certificate, CA certificate and the private key, respectively.

If there are already certificates on the modem, the terminal will output the contents of the certificates to the console. However, if there are not any certificates currently stored in the non-volatile memory, the terminal will respond with something similar to:

**#SSLSECDATA: 1,0**

**No data stored**

**OK**

If there are existing certificates stored on the modem, it is recommended to delete them. To do so, issue the following commands, which will delete the pre-existing certificates:

**AT#SSLSECDATA=1,0,0**

**AT#SSLSECDATA=1,0,1**

**AT#SSLSECDATA=1,0,2**

After clearing out the certificates, individually upload each of the three requisite certificates by following the process below:

1. Using the information returned by the "**dir**" command, determine the number of bytes for the certificate that is to be uploaded.
2. Issue the AT command below:

   **AT#SSLSECDATA=1,1,0,1224**

- The first "**1**" is the 'storage identifier', and will always be constant for NL-SW-HSPA-B modems.
- The second "**1**" indicates that this a storage operation involving the non-volatile memory.
- "**0**" indicates that this operation is writing the client certificate to memory. Change this parameter to "**1**" or "**2**" when writing the CA certificate, and the private key, respectively.
- Finally, "**1224**" is the size in bytes of the certificate being written to the non-volatile memory. This parameter will also change based on the certificate being written.

3. The modem will respond with "**>**", and will wait for data to be entered. This data will be entered using the Windows command prompt in step 4.
4. First, disconnect the serial terminal from the COM port that the modem is connected to. Only one program can have access to the serial line at a time, which is why the terminal emulator must be disconnected during the certificate upload phase.

   Next, issue the following command in the command prompt, where the name of the file is replaced with the name of the certificate being uploaded, and the "COM10" string is replaced with the proper COM port number.

   **copy 8da6fe87f3-certificate.pem \\.\COM10**

   This command will pipe the contents of the certificate to the serial line, which will then be stored in non-volatile memory on the NL-SW-HSPA-B. After issuing the above command into the command prompt, return to the AT command line and press "**CTRL+Z**". This will finalize the writing of the certificate.

   Repeat the four steps listed on the previous page until each of the three files have been uploaded. After the files have been successfully uploaded, reattach the serial console to the appropriate COM port. Once finished, proceed to Section 3.2.

## 3.2 Verifying the Certificate Uploads

To verify that the file uploads were successful, issue the following AT commands:

```
AT#SSLSECDATA=1,2,0

AT#SSLSECDATA=1,2,1

AT#SSLSECDATA=1,2,2
```

The modem will read the contents of the client certificate, CA certificate and the private key, respectively. An example response to the second command listed above is as follows:

```
CONNECT 1758

-----BEGIN CERTIFICATE-----

MIIE0zCCA7ugAwIBAgIQGNrRniZ96LtKIVjNzGs7SjANBgkqhkiG9w0BAQUFADCB

yjELMAkGA1UEBhMCVVMxFzAVBgNVBAoTDlZlcmlTaWduLCBJbmMuMR8wHQYDVQQL

.                              .                              .

.                              .                              .

4fQRbxC1lfznQgUy286dUV4otp6F01vvpX1FQHKOtw5rDgb7MzVIcbidJ4vEZV8N

hnacRHr2lVz2XTIIM6RUthg/aFzyQkqFOFSDX9HoLPKsEdao7WNq

-----END CERTIFICATE-----
```

For each certificate, visually inspect the text output by the modem after issuing the commands above. If the all three certificates appear to have been uploaded properly, proceed to Section 3.3. If for some reason the certificates appear to be incorrect, try reloading the certificates starting with Section 3.1.

## 3.3 SSL Configuration

The next step is to configure the SSL connection parameters on the Skywire. To do this, issue the following set of commands:

1. Activate the first SSL profile:

   **AT#SSLEN=1,1**

   The first "1" specifies the first SSL profile, and the second "1" enables this profile.

2. Select TLS1.2 protocol for the SSL connection:

   **AT#SSLSECCFG2=1,3**

   The "1" specifies the first SSL profile, and the "3" selects TLS1.2 protocol.

3. Configure the SSL profile with the following commands:

   **AT#SSLSECCFG=1,5,2,1**

   - The "**1**" selects the first SSL profile.
   - The "**5**" selects TLS_RSA_WITH_AES_256_CBC_SHA256 as the cipher suite.
   - The "**2**" specifies server/client authentication mode. This means that the connection requires a CA cert, client cert and a private key.
   - The final "1" specifies that the certificates are in .pem format.

   **AT#SSLCFG=1,1,300,90,100,50,1,2**

   - The first "**1**" specifies the first SSL profile.
   - The second "**1**" specifies the first PDP context.
   - "**300**" indicates the packet size.
   - "**90**" indicates the maximum timeout value while in online mode.
   - "**100**" indicates the default timeout value used by other SSL commands.
   - The third "**1**" sets the "SSLRING" URC mode. This will enable a URC that notifies the user when data has been received through the socket.
   - The "**2**" tells the modem return a verbose error report if it returns "**NO CARRIER**" at any point during the connection.

After the SSL profile has been properly configured, proceed to Section 3.4.

## 3.4 Configure and Activate PDP Context

Next, a PDP context must be defined. To do so, issue the following command, replacing "**[APN]**" with the appropriate APN:

```
AT+CGDCONT=1,"IP","[APN]"
```

To ensure that the previous command was entered properly, issue this command:

```
AT+CGDCONT?
```

The modem should respond with something similar to:

```
+CGDCONT: 1,"IP","[APN]","0.0.0.0",0,0
```

Next, activate the TCP/IP context by issuing the following command:

```
AT#SGACT=1,1
```

To check the status of the activation, use this command:

```
AT#SGACT?
```

The modem should respond with something similar to:

```
#SGACT: 1,1
```

After the PDP context has been activated successfully, proceed to Section 4.

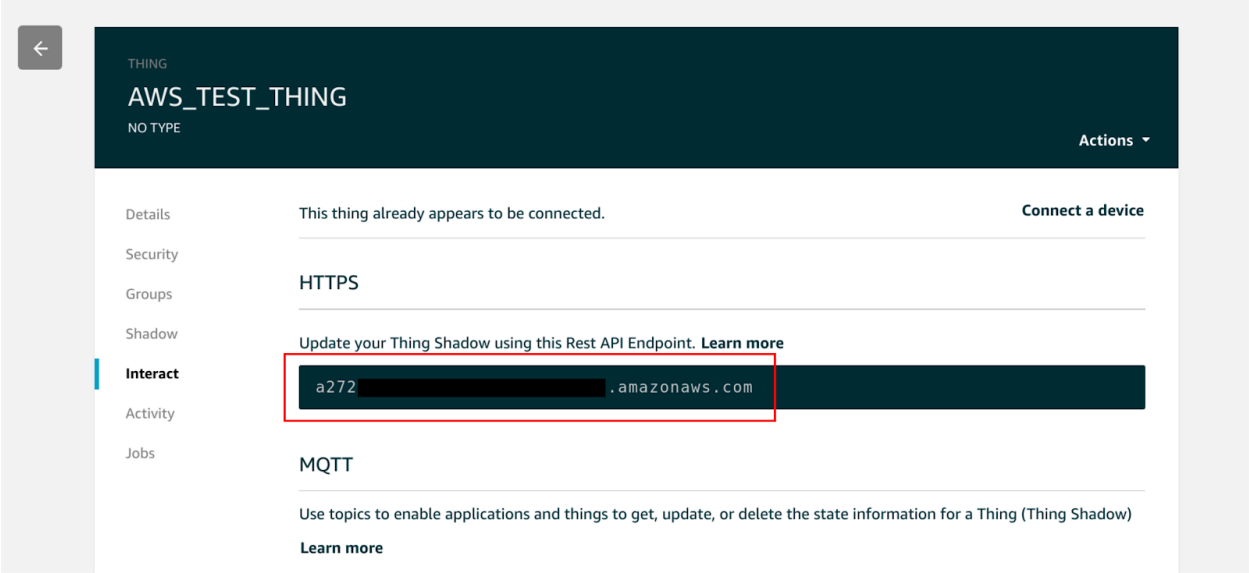
# 4. Connect to Amazon AWS

## 4.1 Opening an SSL Socket

After the Skywire has been configured properly, it is ready to establish a connection to the AWS server.

First, find the endpoint for the "thing" that was created on the AWS website. To do this, navigate to the "Things" page using the menu on the left-hand side of the AWS console page. Click on the "thing" and then navigate to the "Interact" menu. The correct menu should look something like the image on the next page:




In the image above, the URL for the device endpoint has been enclosed in a red rectangle. Record whatever URL shows up in this page, as it will be needed in the SSL socket connection command.

To open the SSL socket, issue the command below. Make sure to replace the device endpoint in the AT command below with the endpoint that is unique to the AWS account being used for this example.

**AT#SSLD=1,8443,"a272...amazonaws.com",0,1,60**

- The first "**1**" selects the first secure socket profile.
- "**8443**" specifies the port being connected to.
- **"a272...amazonaws.com"** is the URL being connected to.
- "**0**" specifies closure mode, and must be zero at all times.
- The second "**1**" specifies that the connection will use command mode as opposed to online mode. Using command mode makes it easier to interact with the modem during a connection.
- "**60**" specifies the timeout value for the connection.

After issuing this command, the modem will attempt to connect to the AWS endpoint. If the connection is successful, the modem will return the following:

**OK**

The above text indicates that the SSL handshake was successful and that the socket has been opened. If the modem responds with anything other than the above text, try issuing the command again.

After the SSL socket has been successfully been opened. Proceed to Section 4.2.

## 4.2 Sending an HTTP Request

After successfully opening an SSL socket, use the following command to send an HTTP request to the AWS endpoint:

**AT#SSLSEND=1,60**

Where "**1**" is the number of the socket used for the SSL connection, and "**60**" is the timeout value for the send command.

After issuing this AT command, the modem will wait for text to be entered into the terminal. This text can either be pasted or typed into the terminal. See the text below for a sample HTTP POST request using the "**AT#SSLSEND**" command.

Also, take note of the text in red below. Any of the red text indicates key presses, and are not to be typed explicitly. These key press sequences are crucial in order to format the POST command properly. Also, bold text signifies commands issued to the modem, and text pasted into the terminal.

```
AT#SSLSEND=1,60
> POST /things/AWS_TEST_THING/shadow HTTP/1.1 "CTRL+M, CTRL+J"
Host: a272...amazonaws.com:8443 "CTRL+M, CTRL+J"
Content-Type: application/json "CTRL+M, CTRL+J"
Content-Length: 123 "CTRL+M, CTRL+J, CTRL+M, CTRL+J"

{"state":{"desired":{"string1":"TLS Connect to AWS","string2":"Using the
built-in stack","string3":"of the NL-SW-HSPA-B"}}}
"CTRL+Z"
OK
SSLSRING: 1,189
```

As can be seen in the text above, the POST command was pasted into the terminal, and the modem responded with "**OK**" indicating that the transmission succeeded. Also note that the "**SSLSRING: 1,189**" URC indicates that an HTTP response was received. Section 4.3 will detail how to read this response.

Another important item to note is the "**Content-Length: 123**" line. In this case, the value of "**123**" indicates that 123 bytes of data are being sent through the socket. This helps the endpoint know how many bytes to consider as data. It is crucial to ensure that this number is updated whenever the data JSON is edited.

## 4.3 Reading an HTTP Response

To read an HTTP response, issue the following command:

```
AT#SSLRECV=1,1000,10
```

Where "1" indicates the socket identifier, "1000" instructs the modem to read 1000 bytes, and "10" indicates the timeout for the read command.

In the case of the sample HTTP POST command in Section 4.2, the HTTP response was:

```
#SSLRECV: 478
HTTP/1.1 200 OK
content-type: application/json
content-length: 289
date: Thu, 02 Aug 2018 20:21:35 GMT
x-amzn-RequestId: e99e4a88-05a9-294c-feb2-604eb0fc41fc
connection: keep-alive

{"state":{"desired":{"string1":"TLS Connect to AWS","string2":"Using the
built-in stack","string3":"of the
NL-SW-HSPA-B"}},"metadata":{"desired":{"string1":{"timestamp":1533241295},"st
ring2":{"timestamp":1533241295},"string3":{"timestamp":1533241295}}},"version
":21,"timestamp":1533241295}
```

## 4.4 Closing an SSL Socket

To close an SSL socket, issue the following command:

```
AT#SSLH=1
```

Where "1" is the socket identifier.

# 5. Working Examples

Section 5 contains two examples involving an HTTP POST and and HTTP GET operation, and the AWS cloud.

The certificate upload procedure will be demonstrated for both a Linux and a Windows environment, however note that the remainder of the process after the certificate upload is identical for these two environments.

## 5.1 Initial Setup

This section details the configuration of the Skywire that is common to both of the HTTP examples. Commands entered into the Windows command prompt and Linux terminal will be colored red, while commands issued to the NL-SW-HSPA-B will be colored black. Additionally, commands issued to the modem by the user will be in boldface, while responses from the modem will be in regular font.

Proceed to Section 5.1.1 for Linux certificate upload instructions, or Section 5.1.2 for Windows certificate upload instructions.

### 5.1.1 Linux Certificate Upload

First, upload the certificates needed for the SSL connection:

```
AT#SSLSECDATA=1,1,0,1224
>
cat 8da6fe87f3-certificate.pem.crt > /dev/ttyUSB0

"CTRL+Z" pressed here
OK

AT#SSLSECDATA=1,1,1,1758
>
cat VeriSign-Class\ 3-Public-Primary-Certification-Authority-G5.pem > /dev/ttyUSB0

"CTRL+Z" pressed here
OK

AT#SSLSECDATA=1,1,2,1675
>
cat 8da6fe87f3-private.pem.key > /dev/ttyUSB0

"CTRL+Z" pressed here
OK
```

Once the certificates have been uploaded successfully, proceed to Section 5.1.3.

### 5.1.2 Windows Certificate Upload

First, upload the certificates needed for the SSL connection:

```
AT#SSLSECDATA=1,1,0,1224
>
copy 8da6fe87f3-certificate.pem \\.\COM10

"CTRL+Z" pressed here
OK

AT#SSLSECDATA=1,1,1,1758
>
copy VeriSign-Class\ 3-Public-Primary-Certification-Authority-G5.pem \\.\COM10

"CTRL+Z" pressed here
OK

AT#SSLSECDATA=1,1,2,1675
>
copy 8da6fe87f3-private.pem.key > \\.\COM10

"CTRL+Z" pressed here
OK
```

Once the certificates have been uploaded successfully, proceed to Section 5.1.3.

### 5.1.3 Connection Settings Configuration

After the files have been successfully uploaded, configure the SSL profile.

```
AT#SSLEN=1,1
OK

AT#SSLSECCFG=1,5,2,1
OK

AT#SSLCFG=1,1,300,90,100,50,1,2
OK

AT#SSLSECCFG2=1,3
OK
```

Next, configure and activate the PDP context. Make sure to change "**[APN]**" to the appropriate APN. In this example, the APN was set to "**m2mglobal**".

```
AT+CGDCONT=1,"IP","[APN]"
OK

AT#SGACT=1,1
OK

AT+CREG?
+CREG: 0,1

OK
```

Finally, establish a connection with the AWS server.

```
AT#SSLD=1,8443,"a272...amazonaws.com",0,1,60
OK
```

After completing the configuration steps, proceed to either Section 5.2 or Section 5.3 for an HTTP POST example, or an HTTP GET example, respectively.

## 5.2 HTTP POST Example

After properly initializing the modem in Section 5.1, enter the commands below to execute an HTTP POST request. For help with POST command formatting, refer to the next page.

**Note:** the text in red indicates CTRL key sequences, and are not to be typed into the terminal verbatim. Instead, press these CTRL key sequences in the order listed. Bold text indicates commands, as well as text that was pasted into the terminal.

```
AT#SSLSEND=1,60
> POST /things/AWS_TEST_THING/shadow HTTP/1.1 "CTRL+M, CTRL+J"
Host: a272...amazonaws.com:8443 "CTRL+M, CTRL+J"
Content-Type: application/json "CTRL+M, CTRL+J"
Content-Length: 123 "CTRL+M, CTRL+J, CTRL+M, CTRL+J"

{"state":{"desired":{"string1":"TLS Connect to AWS","string2":"Using the
built-in stack","string3":"of the NL-SW-HSPA-B"}}}

"CTRL+Z"

OK
SSLSRING: 1,189

AT#SSLRECV=1,1000,10
#SSLRECV: 478
HTTP/1.1 200 OK
content-type: application/json
content-length: 289
date: Thu, 02 Aug 2018 20:21:35 GMT
x-amzn-RequestId: e99e4a88-05a9-294c-feb2-604eb0fc41fc
connection: keep-alive

{"state":{"desired":{"string1":"TLS Connect to AWS","string2":"Using the
built-in stack","string3":"of the
NL-SW-HSPA-B"}},"metadata":{"desired":{"string1":{"timestamp":1533241295},"stri
ng2":{"timestamp":1533241295},"string3":{"timestamp":1533241295}}},"version":21
,"timestamp":1533241295}

OK
```

Properly formatting the POST command can be challenging. Below are a few pointers for correct POST command formatting and entry:

- It is usually impossible to paste the entire POST command all at once. Try pasting the command in line-by-line as opposed to all at once.
- Press "**`CTRL+M, CTRL+J`**" after each line of the POST command. This sequence enters in a newline and carriage return character after each line.
- Press "**`CTRL+M, CTRL+J, CTRL+M, CTRL+J`**" after the "**`Content-Length: 123`**" line. In other words, insert two newline and carriage return sequences after this line.
- Replace the "**`AWS_TEST_THING`**" identifier with the unique name assigned during the "thing" creation in Section 2.3.
- Replace the AWS endpoint "**`a272...amazonaws.com:8443`**" with the unique endpoint associated with the Amazon AWS account in use.
- If the contents of the JSON are changed for any reason, the integer argument of the "**`Content-Length: 123`**" line must be modified.

  In other words, if the JSON is made larger or smaller, the total number of bytes being sent must be recalculated, and the "**`Content-Length: x`**" line must be updated with the new length.

## 5.3 HTTP GET Example

After properly initializing the modem in Section 5.1, enter the commands below to execute an HTTP GET request.

**Note:** the text in red indicates CTRL key sequences, and are not to be typed into the terminal verbatim. Instead, press these CTRL key sequences in the order listed. Bold text indicates commands, as well as text that was pasted into the terminal.

```
AT#SSLSEND=1,60
> GET /things/AWS_TEST_THING/shadow HTTP/1.1
"CTRL+M, CTRL+J, CTRL+M, CTRL+J, CTRL+Z"

OK
SSLSRING: 1,189

AT#SSLRECV=1,1000,10
#SSLRECV: 588
HTTP/1.1 200 OK
content-type: application/json
content-length: 399
date: Thu, 02 Aug 2018 20:22:56 GMT
x-amzn-RequestId: 321e9ff5-3d54-e658-cf82-1ba91296786a
connection: keep-alive

{"state":{"desired":{"string1":"TLS Connect to AWS","string2":"Using
the built-in stack","string3":"of the
NL-SW-HSPA-B"},"delta":{"string1":"TLS Connect to
AWS","string2":"Using the built-in stack","string3":"of the
NL-SW-HSPA-B"}},"metadata":{"desired":{"string1":{"timestamp":15332412
95},"string2":{"timestamp":1533241295},"string3":{"timestamp":15332412
95}}},"version":21,"timestamp":1533241376}

OK
```

As per the HTTP POST example, be sure to replace the "**AWS_TEST_THING**" string with whatever unique name was assigned to the AWS "thing" in Section 2.3.

# 6. Troubleshooting

## 6.1 HTTP Response Codes

### 6.1.1 403 Forbidden

If a connection can be established, but the AWS response to the "**GET**" command is "**403 Forbidden**", make sure that the current AWS policy is set to allow all IoT actions. This can be done through the AWS IoT Console.

### 6.1.2 400 Bad Request

If a connection can be established, but the AWS response to the "**GET**" or "**POST**" command is "**400 Bad Request**", make sure the syntax of the "**GET**" or "**POST**" command is correct.

## 6.2 Verify Credentials

If for some reason the credentials for the AWS connection do not work, OpenSSL can be used to check their validity. This process is helpful for narrowing down the source of the connection issue.

### 6.2.1 Testing AWS Credentials using OpenSSL

To test credentials with OpenSSL, first ensure that OpenSSL is properly installed on a Linux or Windows system. Next, navigate to the directory that contains the certificates that are being tested. Issue the following command to attempt a connection to AWS:

```
openssl s_client -servername a272...amazonaws.com -connect a272..amazonaws.com:8443
-CAfile VeriSign-Class\ 3-Public-Primary-Certification-Authority-G5.pem -cert
8da6fe87f3-certificate.pem.crt -key 8da6fe87f3-private.pem.key -certform PEM
-keyform PEM
```

Be sure to replace any of the text in bold with unique certificate names, and the unique AWS endpoint URL.

If the connection is successful, the terminal should respond with "CONNECTED" followed by some information about the connection, including the server certificate. To further test the connection, type the following command, followed by the sequence "**CTRL+M, CTRL+J**":

```
GET /things/AWS_TEST_THING/shadow HTTP/1.1
```

Remember to replace the "**AWS_TEST_THING**" string with the unique "thing" name assigned in Section 2.3.

The server should respond with something similar to:

```
HTTP/1.1 200 OK
content-type: application/json
content-length: 399
date: Thu, 02 Aug 2018 20:22:56 GMT
x-amzn-RequestId: 321e9ff5-3d54-e658-cf82-1ba91296786a
connection: keep-alive

{"state":{"desired":{"string1":"TLS Connect to AWS","string2":"Using
the built-in stack","string3":"of the
NL-SW-HSPA-B"},"delta":{"string1":"TLS Connect to
AWS","string2":"Using the built-in stack","string3":"of the
NL-SW-HSPA-B"}},"metadata":{"desired":{"string1":{"timestamp":1533241
295},"string2":{"timestamp":1533241295},"string3":{"timestamp":153324
1295}}},"version":21,"timestamp":1533241376}
```

If a valid connection can be established, then it is safe to say that the certificates are indeed valid, and thus are not the source of the problem.