Skywire® LTE CAT1 AWS IoT with TLS User Manual

NimbeLink Corp

Updated: January 2017

Table of Contents

Table of Contents 2

Introduction 3

Overview 3

Orderable Parts 3

AWS IoT Setup 3

Create a thing 3

Create a policy 5

Create credentials 5

Download the root CA certificate 7

Attach the thing and policy 7

Change format of credentials 8

Create .pfx file 9

Create .jks file 10

Convert .pem files to .der 10

Upload Credentials to Skywire 11

Upload your credentials 11

Upload the root CA certificate 12

Connect to AWS 12

Troubleshooting 22

OpenSSL 22

“Unable to write random state” 22

Skywire Commands 22

"The Certificate does not exist" 22

403 Forbidden 22

400 Bad Request 22

RXTX Mismatch Warning 22

Verify Credentials 23

Verify file format 23

Establish a connection to AWS 23

1. Introduction

1.1 Overview

This document outlines the steps required to establish a secure TLS connection to Amazon's AWS IoT service directly using the Skywire® 4G LTE Cat 1 modem. After going through this tutorial, you should be able to get data from your AWS IoT "thing" by issuing AT commands to the modem.

1.2 Orderable Parts

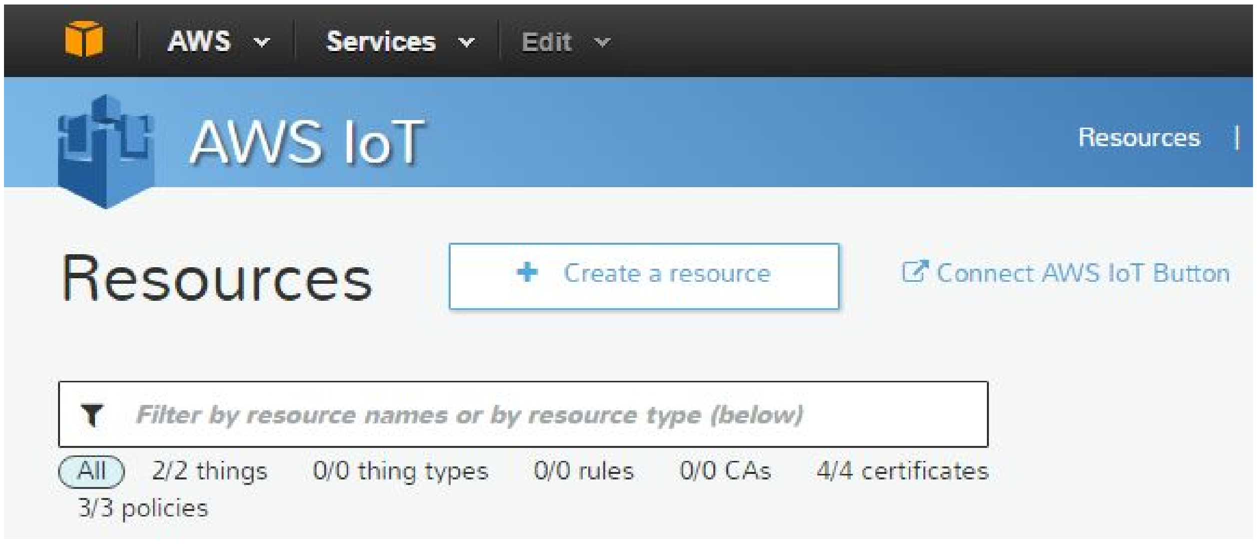
Orderable Device	Firmware Revision	Description	Manufacturer
NL-SW-SWDK	NA	Skywire Development Kit	NimbeLink
NL-SW-LTE-GELS3	4.3.2.0-24916 or higher	Skywire 4G LTE CAT1 Modem	NimbeLink
NL-SW-LTE-GELS3-B	4.3.2.0-25421	Skywire 4G LTE CAT1 Modem	NimbeLink
NL-SW-LTE-GELS3-C	4.3.2.0-26xxx	Skywire 4G LTE CAT1 Modem	NimbeLink

2. AWS IoT Setup

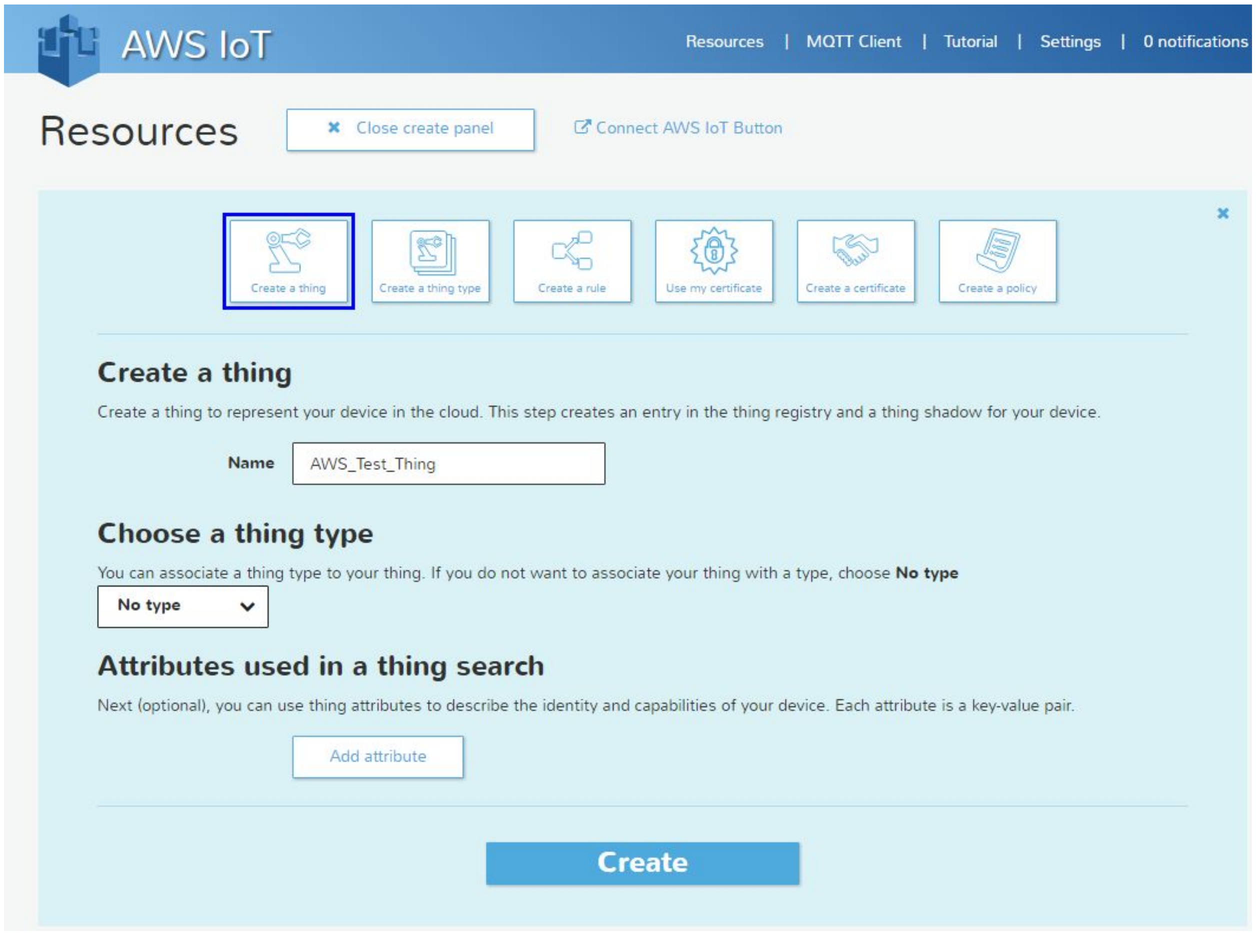
Using the AWS IoT console, we need to create a "thing" and a "policy", and then attach them to credentials that we will have AWS create for us.

2.1 Create a thing

Create a "thing" by clicking on "Create a resource" and selecting "Create a thing".



Set the name (in this case, "AWS_Test_Thing") and click "Create".



Create a thing

Create a thing to represent your device in the cloud. This step creates an entry in the thing registry and a thing shadow for your device.

Name AWS_Test_Thing

Choose a thing type

You can associate a thing type to your thing. If you do not want to associate your thing with a type, choose **No type**

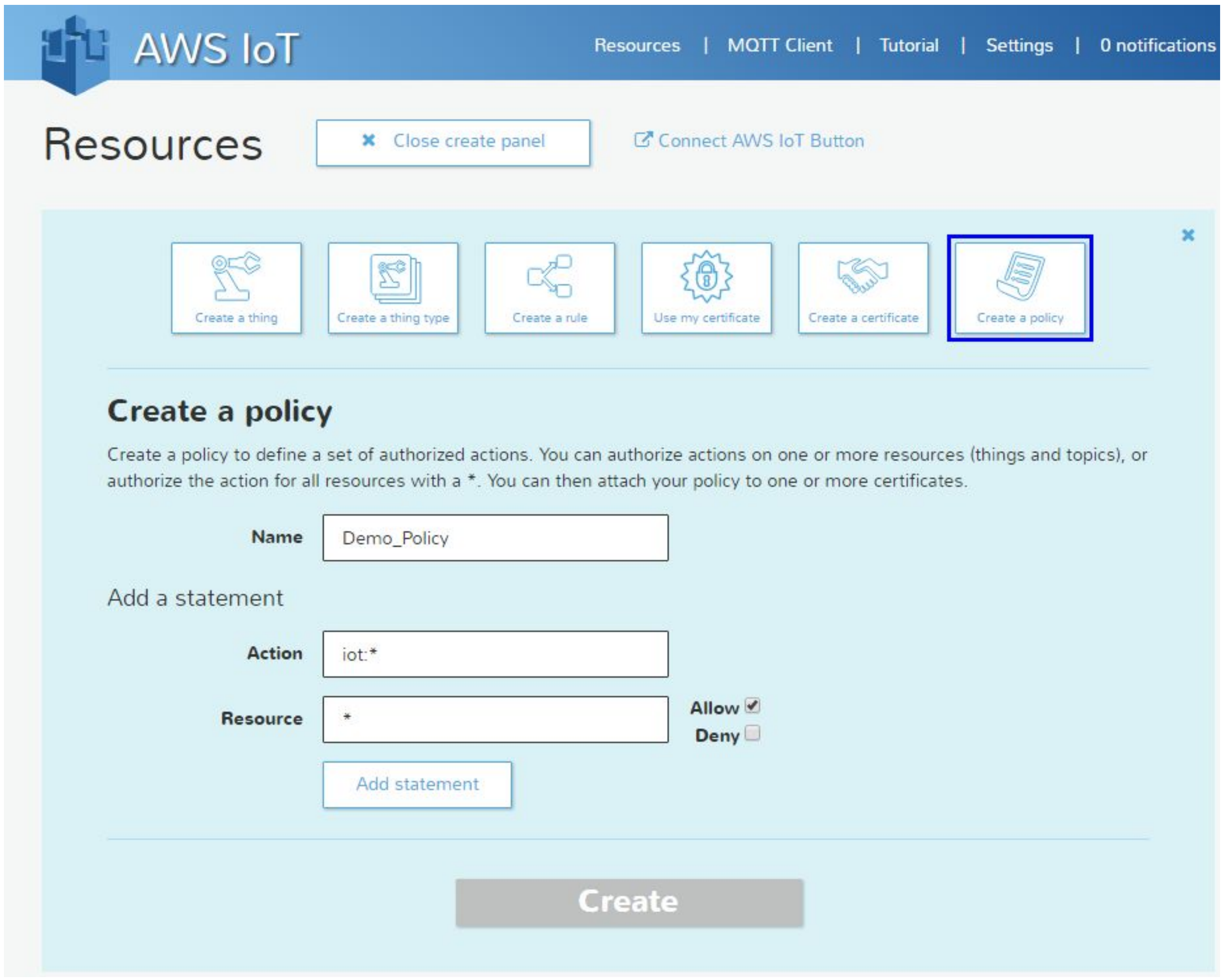
No type

Attributes used in a thing search

Next (optional), you can use thing attributes to describe the identity and capabilities of your device. Each attribute is a key-value pair.

2.2 Create a policy

Create a policy by selecting "Create a resource" and selecting "Create a policy".



Create a policy

Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things and topics), or authorize the action for all resources with a *. You can then attach your policy to one or more certificates.

Name: Demo_Policy

Add a statement

Action: iot:*

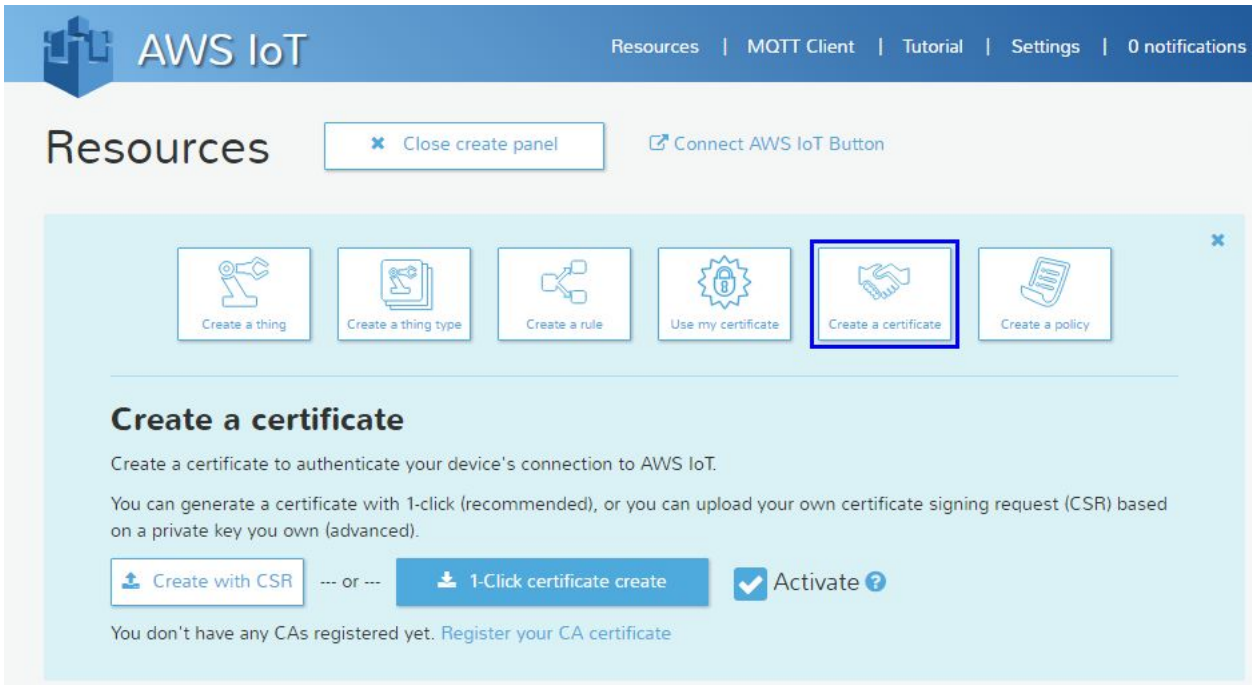
Resource: *

Allow ✓
Deny ☐

Choose iot:* for the Action and * for the Resource field, check the Allow box, and click "Add Statement". Then click "Create" to create the policy.

2.3 Create credentials

Create a certificate and private key pair by selecting "Create a resource" and selecting "Create a certificate".



Create a certificate

Create a certificate to authenticate your device's connection to AWS IoT.

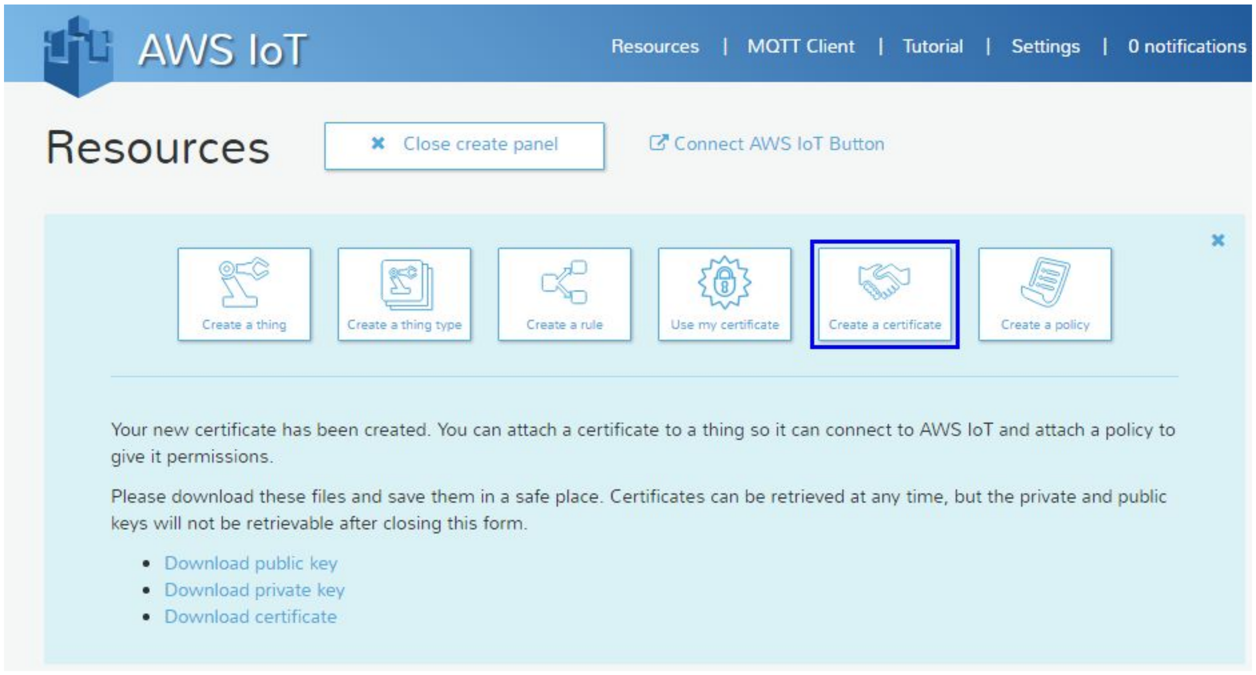
You can generate a certificate with 1-click (recommended), or you can upload your own certificate signing request (CSR) based on a private key you own (advanced).

You don't have any CAs registered yet. Register your CA certificate

Make sure to check the "Activate" box and then click the "1-Click certificate create" button.

After creating the credentials, download the public key, private key, and certificate.

<u>**Important**</u>: you can only download the public and private keys when you create the credentials. They will not be available for download once you leave this screen!



Your new certificate has been created. You can attach a certificate to a thing so it can connect to AWS IoT and attach a policy to give it permissions.

Please download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys will not be retrievable after closing this form.

- Download public key
- Download private key
- Download certificate

2.4 Download the root CA certificate

Download the AWS IoT root CA certificate, found at the following location:

https://www.symantec.com/content/en/us/enterprise/verisign/roots/VeriSign-Class%203-Public-Primary-Certification-Authority-G5.pem

In this example, the name of the CA certificate is left as "VeriSign-Class%203-Public-Primary-Certification-Authority-G5.pem".

Note: for this example, the ".txt" file extension was removed from the filename when downloading the file.

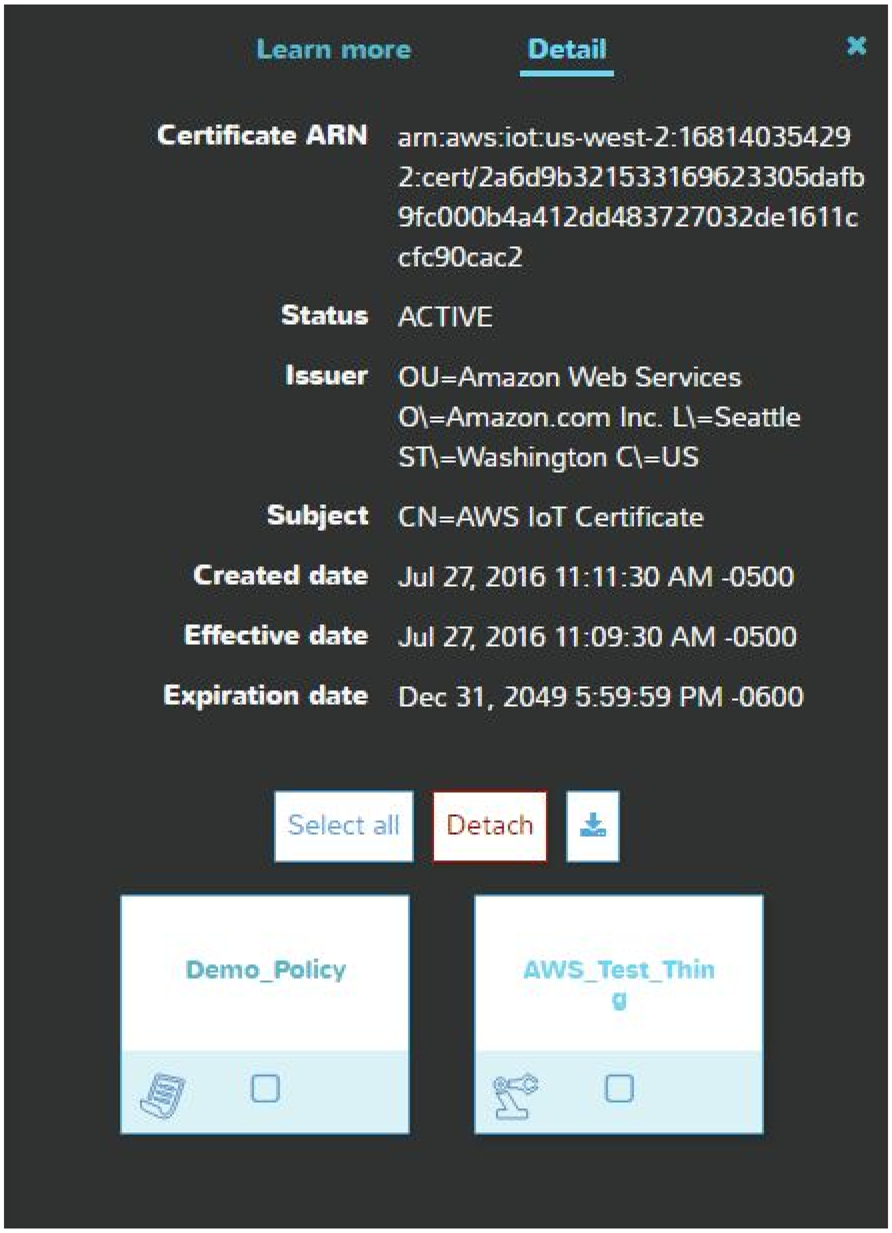
2.5 Attach the thing and policy

Attach your thing and policy to your credentials by doing the following:

Select the certificate and go to "Actions" -> "Attach a thing" and enter the name of the thing you created (AWS_Test_Thing in this example)

Select the certificate and go to "Actions" -> "Attach a policy" and enter the name of the policy you created (Demo_Policy in this example)

Clicking on your certificate should bring up a details window that looks similar to the following:



Certificate ARN	arn:aws:iot:us-west-2:168140354292:cert/2a6d9b321533169623305dafb9fc000b4a412dd483727032de1611ccfc90cac2
Status	ACTIVE
Issuer	OU=Amazon Web Services O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US
Subject	CN=AWS IoT Certificate
Created date	Jul 27, 2016 11:11:30 AM -0500
Effective date	Jul 27, 2016 11:09:30 AM -0500
Expiration date	Dec 31, 2049 5:59:59 PM -0600

3. Change format of credentials

AWS IoT provides certificates in PEM format, but the Skywire 4G LTE Cat 1 modem requires certificates in DER format, and uploading the credentials to the modem requires a Java keystore (.jks) file. We will use OpenSSL and the Java SDK built-in keytool to create the required files.

OpenSSL can be downloaded for windows here:

https://slproweb.com/products/Win32OpenSSL.html

OpenSSL version 1.0.2.h for Windows 64-bit machines is used for this demo. OpenSSL is also available on many Linux distributions and can be installed with your distribution's package manager.

The Java JDK can be downloaded from Oracle:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Java JDK version 1.8.0_101 is used in this example.

3.1 Create .pfx file

Use OpenSSL to create .pfx file from certificate and private key. This file is required by the Java keytool to create a Java keystore file. You may need to add the OpenSSL \bin folder to your PATH variable if the installation didn't add it automatically. To do this, right click on "Computer" in Windows Explorer and click on "Properties" and then "Advanced system settings". Click on "Environment Variables" and find the system variable named PATH. Select "Edit" and append your OpenSSL \bin path to the "Variable value" field. Next, perform the following steps to create the file:

- Open the Command Prompt **as an administrator**
- Navigate to the folder containing your certificate and keys
- Set the location of the RANDFILE that is required by OpenSSL:

```
set RANDFILE=.rnd
```

- Create the .pfx file:

```
openssl pkcs12 -export -out my_pfx_out.pfx -inkey
2a6d9b3215-private.pem.key -in
2a6d9b3215-certificate.pem.crt -name "my_keystore"
-certfile
"VeriSign-Class%203-Public-Primary-Certification-Autho
rity-G5.pem"
```

Note: the file names in bold will have to be changed to the names of your private key and certificate files

Important: remember the password you entered when creating the .pfx file. We will use it later when uploading the files to the modem. The password used in this example is "123456".

The "-name" parameter will also be used later as the alias associated with the keystore we created. Its value in this example is "my_keystore".

3.2 Create .jks file

Now that we have the .pks file, we can use the Java keytool to create a .jks file. You will need to have the Java JDK installed for this step. If the installation doesn't add the Java SDK bin folder to your PATH environment variable, add it manually so the keytool can be run from the folder with your keys and certificates.

Run the keytool using the following command to create a .jks file:

```
keytool -importkeystore -srckeystore my_pfx_out.pfx
-srcstoretype pkcs12 -srcalias "my_keystore" -destkeystore
my_jks_store.jks -deststoretype jks -deststorepass 123456
-destalias "my_jks"
```

The command will prompt you for the password for the .pfx file that was generated in the previous step. In this example, "123456" is used as the password for both the .pfx and .jks files.

3.3 Convert .pem files to .der

The last files we need to create are DER versions of our certificate and private key. OpenSSL allows us to convert keys and certificates from PEM to DER format with the following commands (**bolded** names should be changed to your file names):

Convert the certificate:

```
openssl x509 -outform der -in
2a6d9b3215-certificate.pem.crt -out
2a6d9b3215-certificate.der
```

Convert the private key:

```
openssl rsa -in 2a6d9b3215-private.pem.key -inform PEM -out
2a6d9b3215-private.der -outform DER
```

Convert the root CA certificate:

```
openssl x509 -outform der -in
"VeriSign-Class%203-Public-Primary-Certification-Authority-
G5.pem" -out
"VeriSign-Class%203-Public-Primary-Certification-Authority-
G5.der"
```

You should now have .der versions of the root CA certificate as well as your certificate and private key. In this example, the output files are:

- VeriSign-Class%203-Public-Primary-Certification-Authority-G5.der
- 2a6d9b3215-certificate.der
- 2a6d9b3215-private.der

4. Upload Credentials to Skywire

We now have all the files required to upload to the Skywire modem.

We will be using Gemalto's "cmd_IpCertMgr.jar" Java application to upload the required files. Move both the "cmd_ipcertmgr.jar" and "rxtxserial.dll" files into the folder that has your certificate and private key.

Note: you will need the correct version of the "rxtxserial.dll" library for your machine architecture (32-bit vs 64-bit).

4.1 Upload your credentials

To write your client certificate and private key, send the following command:

```
java -jar cmd_IpCertMgr.jar -serialPort COM19 -serialSpd
115200 -cmd writecert -certfile 2a6d9b3215-certificate.der
-keyfile 2a6d9b3215-private.der -certIndex 0 -imei
356278070014784 -alias my_jks -keypass 123456 -keystore
my_jks_store.jks -storepass 123456
```

Note: the values in **bold** will have to be changed based on your files and modem. The "-serialPort" option should be set to the COM port being used by your Skywire modem. The "-imei" option should be set to the IMEI number shown on your Skywire modem. The "-certfile" and "-keyfile" options should be set to the file names specific to your certificate and private key. If you used the same alias names and passwords as this example, those options can be left as is. Otherwise they should be changed to whatever values you chose in the previous steps.

If the files upload successfully, the response should look like the following:

```
SECURE CMD READY: SEND COMMAND…
SECURE CMD END OK
```

4.2 Upload the root CA certificate

To write the root CA certificate, send the following command:

```
java -jar cmd_IpCertMgr.jar -serialPort COM19 -serialSpd
115200 -cmd writecert -certfile
VeriSign-Class%203-Public-Primary-Certification-Authority-G
5.der -certIndex 1 -imei 356278070014784 -alias my_jks
-keypass 123456 -keystore my_jks_store.jks -storepass
123456
```

Note: the values in **bold** will have to be changed based on your modem. The "-serialPort" option should be set to the COM port being used by your Skywire modem. The "-imei" option should be set to the IMEI number shown on your Skywire modem.

If the file uploads successfully, the response should look like the following:

```
SECURE CMD READY: SEND COMMAND…
SECURE CMD END OK
```

5. Connect to AWS

Now that the files are loaded onto the Skywire modem, we can connect through TeraTerm (or your terminal program of choice) and issue the commands to connect to AWS IoT. We will be using the Skywire Development Kit in this example.

Before issuing the following commands, make sure to **power cycle** the modem to ensure that the certificates are properly stored. After connecting to the Skywire modem through TeraTerm, issue the following sequence of AT commands:

Note: commands are in **bold**, responses and unsolicited messages are in plain text, and comments regarding the commands are in *italics*.

Get modem info

ati1

Cinterion

ELS31-V

REVISION 4.3.2.0

```
A-REVISION 4.3.2.0-24916
L-REVISION 3.7.6
OK
```

Verify that the certificates were written to the modem

at^sbnr=is_cert

```
^SBNR: 0, size: "862", issuer: "/OU=Amazon Web Services O=Amazon.com Inc.
L=Seattle ST=Washington C=US", serial number:
"fd9b08cc5e803708cd7a729177c0b14afb003778", subject: "/CN=AWS IoT Certificate",
signature: "sha256RSA", thumbprint algorithm: "sha1", thumbprint:
"ebec0265cd3cbca38abee38a2f8176eb6a6a992a"

^SBNR: 1, size: "1239", issuer: "/C=US/O=VeriSign, Inc./OU=VeriSign Trust
Network/OU=(c) 2006 VeriSign, Inc. - For authorized use only/CN=VeriSign Class 3
Public Primary Certification Authority - G5", serial number:
"18dad19e267de8bb4a2158cdcc6b3b4a", subject: "/C=US/O=VeriSign,
Inc./OU=VeriSign Trust Network/OU=(c) 2006 VeriSign, Inc. - For authorized use
only/CN=VeriSign Class 3 Public Primary Certification Authority - G5", signature:
"sha1RSA", thumbprint algorithm: "sha1", thumbprint:
"4eb6d578499b1ccf5f581ead56be3d9b6744a5e5"

^SBNR: 2, size: "0", issuer: "", serial number: "", subject: "", signature: "", thumbprint
algorithm: "", thumbprint: ""

^SBNR: 3, size: "0", issuer: "", serial number: "", subject: "", signature: "", thumbprint
algorithm: "", thumbprint: ""

^SBNR: 4, size: "0", issuer: "", serial number: "", subject: "", signature: "", thumbprint
algorithm: "", thumbprint: ""

^SBNR: 5, size: "0", issuer: "", serial number: "", subject: "", signature: "", thumbprint
algorithm: "", thumbprint: ""

^SBNR: 6, size: "0", issuer: "", serial number: "", subject: "", signature: "", thumbprint
algorithm: "", thumbprint: ""

^SBNR: 7, size: "0", issuer: "", serial number: "", subject: "", signature: "", thumbprint
algorithm: "", thumbprint: ""

^SBNR: 8, size: "0", issuer: "", serial number: "", subject: "", signature: "", thumbprint
algorithm: "", thumbprint: ""
```

^SBNR: 9, size: "0", issuer: "", serial number: "", subject: "", signature: "", thumbprint algorithm: "", thumbprint: ""

^SBNR: 10, size: "0", issuer: "", serial number: "", subject: "", signature: "", thumbprint algorithm: "", thumbprint: ""

OK

Configure modem to automatically update timezone info

at+ctzu=1

OK

at^sind="nitz",1

^SIND: "nitz",1,"",+00

OK

at+cmer=3,0,0,2

OK

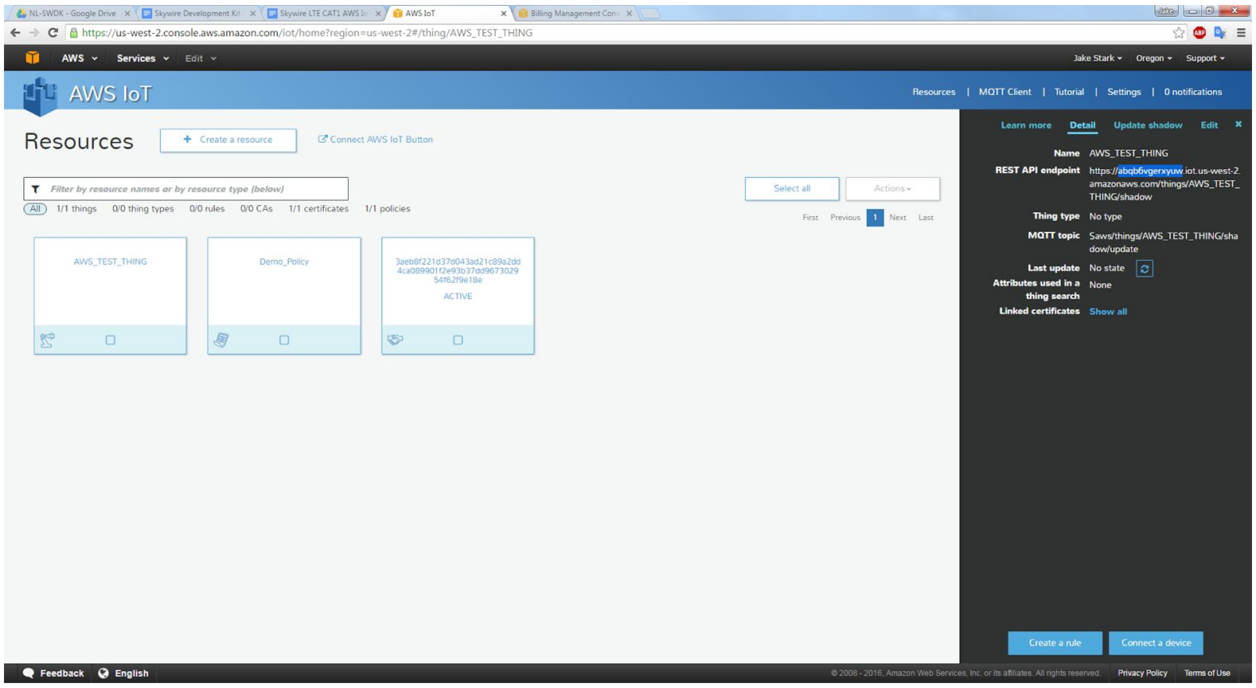
at^sind="is_cert",1

^SIND: "is_cert",1,0,"","","","","",""

OK

Configure socket parameters for connecting to AWS

Note: the string of characters before ".iot.us-west-2.amazonaws.com" (ACVWOJPAQ6LEI in this example) is specific to your AWS account. Find it by going to the resource page under AWS IoT, clicking on your "Thing", and copying the string from the Rest API endpoint url as seen below.



Name	AWS_TEST_THING
REST API endpoint	https://abqb6vgerxyuw.iot.us-west-2.amazonaws.com/things/AWS_TEST_THING/shadow
Thing type	No type
MQTT topic	$aws/things/AWS_TEST_THING/shadow/update
Last update	No state
Attributes used in a thing search	None
Linked certificates	Show all

at^siss=0,srvType,"Socket"

OK

at^siss=0,conId,3

OK

at^siss=0,address,"socktcps://ACVWOJPAQ6LEI.iot.us-west-2.amazonaws.com:8443"

OK

at^siss=0,secopt,1

OK

Activate the PDP context

at^sica=1,3

OK

Verify that the modem has an IP address

at+cgpaddr

+CGPADDR: 1,"254.128.0.0.0.0.0.0.0.0.0.0.63.200.115.165.1"

+CGPADDR: 3,"166.157.157.127"

OK

6. AWS IOT: HTTP GET Example

Now that your Skywire is set up, we can send and receive data from AWS. This section covers an HTTP GET to read data from AWS.

<u>Note</u>: commands are in **bold**, responses and unsolicited messages are in plain text, and comments regarding the commands are in *italics*.

Open the socket connection to AWS

at^siso=0

OK

+CIEV: "is_cert",0,"/C=US/O=Symantec Corporation/OU=Symantec Trust Network/CN=Symantec Class 3 Secure Server CA - G4","25608F1A38647F23CBEC4421982984B1","/C=US/ST=Washington/L=Seattle/O=Amazon.com, Inc./CN=*.iot.us-west-2.amazonaws.com","sha256RSA","sha1","2B91E3B6FEC136FD31F9276C1ECB0508DA4783E8"

+CIEV: "is_cert",0,"/C=US/O=VeriSign, Inc./OU=VeriSign Trust Network/OU=(c) 2006 VeriSign, Inc. - For authorized use only/CN=VeriSign Class 3 Public Primary Certification Authority - G5","513FB9743870B73440418D30930699FF","/C=US/O=Symantec Corporation/OU=Symantec Trust Network/CN=Symantec Class 3 Secure Server CA - G4","sha256RSA","sha1","FF67367C5CD4DE4AE18BCCE1D70FDABD7C866135"

^SISW: 0,1

Send the "write" command to the modem and specify the number of bytes (46) to write

<u>Note</u>: the number of bytes you send may be different depending on your thing name. Replace "AWS_Test_Thing" with your thing name. The number of bytes includes the 2

carriage return and 2 linefeed characters that must be sent after the GET command. Send the character sequence CTRL+M, CTRL+J, CTRL+M, CTRL+J after entering the GET command.

```
at^sisw=0,46
^SISW: 0,46,0
GET /things/AWS_Test_Thing/shadow HTTP/1.1



OK


+CIEV: nitz,16/07/27,14:26:49,-20,1


^SISW: 0,1


^SISR: 0,1
```

Issue the "read" socket command to read up to 1000 bytes on connection number 0. This will read the header information of the response from AWS

```
at^sisr=0,1000
^SISR: 0,189
HTTP/1.1 200 OK
content-type: application/json
content-length: 130
date: Wed, 27 Jul 2016 19:26:50 GMT
x-amzn-RequestId: 127e9f5b-61e6-43d0-94fd-0e929bb9daa8
connection: Keep-Alive


OK
```

^SISR: 0,1

^SIS: 0,0,48,"Remote peer has closed the connection"

Issue the "read" command again to read the response body. The data can still be read even though the remote peer has closed the connection

at^sisr=0,1000

^SISR: 0,130

{"state":{"reported":{"test1":123}},"metadata":{"reported":{"test1":{"timestamp":1469645422}}},"version":1,"timestamp":1469647610}

OK

^SISR: 0,2

+CIEV: nitz,16/07/27,14:27:00,-20,1

Close the socket connection

at^sisc=0

OK

7. AWS IOT: HTTP POST Example

This section covers doing an HTTP POST to send data to AWS.

Note: commands are in **bold**, responses and unsolicited messages are in plain text, and comments regarding the commands are in *italics*.

Open the socket connection to AWS

at^siso=0

OK

+CIEV: "is_cert",0,"/C=US/O=Symantec Corporation/OU=Symantec Trust Network/CN=Symantec Class 3 Secure Server CA - G4","25608F1A38647F23CBEC4421982984B1","/C=US/ST=Washington/L=Seattle/O=

```
Amazon.com,
Inc./CN=*.iot.us-west-2.amazonaws.com","sha256RSA","sha1","2B91E3B6FEC136FD3
1F9276C1ECB0508DA4783E8"

+CIEV: "is_cert",0,"/C=US/O=VeriSign, Inc./OU=VeriSign Trust Network/OU=(c) 2006
VeriSign, Inc. - For authorized use only/CN=VeriSign Class 3 Public Primary
Certification Authority -
G5","513FB9743870B73440418D30930699FF","/C=US/O=Symantec
Corporation/OU=Symantec Trust Network/CN=Symantec Class 3 Secure Server CA -
G4","sha256RSA","sha1","FF67367C5CD4DE4AE18BCCE1D70FDABD7C866135"

^SISW: 0,1
```

Format your HTTP POST

Here is the HTTP POST we are sending to AWS:

```
POST /things/AWS_Test_Thing/shadow HTTP/1.1
Host: ACVWOJPAQ6LEI.iot.us-west-2.amazonaws.com:8443
Content-Type: application/json
Content-Length: 114

{"state":{"desired":{"attribute1":123,"attribute2":"test1"},"reported":{"attribute1":
456,"attribute2":"string1"}}}
```

Note: In the HTTP POST, the "Content-Length" header is the length of the JSON packet being sent, and does not include carriage return or linefeed characters, nor does it include the headers.

Note: The "Host" header's address is the same as you typed in the AT^SISS command in Section 5. You must include this header.

Send the "write" command to the modem and specify the number of bytes (268) to write.

Note: the number of bytes you send may be different depending on your thing name. Replace "AWS_Test_Thing" with your thing name. The number of bytes includes the carriage return (CTRL-M) and linefeed (CTRL-J) characters that must be sent after each line. In addition, after sending the last header (Content-Length: 114), this number *includes the second set of carriage return and linefeed characters that must be sent to create the blank line between the headers and the body.*

```
at^sisw=0,268

^SISW: 0,268,0


POST /things/AWS_Test_Thing/shadow HTTP/1.1

Host: ACVWOJPAQ6LEI.iot.us-west-2.amazonaws.com:8443

Content-Type: application/json

Content-Length: 114


{"state":{"desired":{"attribute1":123,"attribute2":"test1"},"reported":{"attribute1":
456,"attribute2":"string1"}}}
OK


+CIEV: ...


+CIEV: ...


^SISW: 0,1


+CIEV: nitz,17/01/09,17:24:59,-24


+CIEV: nitz,17/01/09,17:25:23,-24


^SISW: 0,1


^SISR: 0,1
```

Issue the "read" socket command to read up to 1000 bytes on connection number 0. This will read the header information of the response from AWS

```
at^sisr=0,1000
^SISR: 0,189
HTTP/1.1 200 OK
content-type: application/json
content-length: 340
date: Mon, 09 Jan 2017 23:25:33 GMT
x-amzn-RequestId: a6c94da8-5f33-8d46-8e0b-6397bb37ce2a
connection: Keep-Alive



OK


^SISR: 0,1


^SIS: 0,0,48,"Remote peer has closed the connection"
```

Issue the "read" command again to read the response body. The data can still be read even though the remote peer has closed the connection.

```
at^sisr=0,1000
^SISR: 0,340
{"state":{"desired":{"attribute1":123,"attribute2":"test1"},"reported":{"attribute1":456,"attribute2":"string1"}},"metadata":{"desired":{"attribute1":{"timestamp":1484004333},"attribute2":{"timestamp":1484004333}},"reported":{"attribute1":{"timestamp":1484004333},"attribute2":{"timestamp":1484004333}}},"version":11,"timestamp":1484004333}
OK
```

Close the socket connection

```
at^sisc=0
OK
```

8. Troubleshooting

There are many steps in the process of setting up a connection to AWS on the Skywire 4G LTE Cat 1 modem. Some of the common errors and possible solutions are listed below.

8.1 OpenSSL

8.1.1 “Unable to write random state”

If you get the error "unable to write random state" after issuing the command to create a .pfx file (Section 3.1), make sure you issued the "set RANDFILE=.rnd" command. OpenSSL needs to be pointed to the location where it can generate the random output for the credentials.

8.2 Skywire Commands

8.2.1 "The Certificate does not exist"

After issuing the open socket command "at^siso=0", the modem may return the response "^SIS: 0,0,77,"The certificate does not exist". In this case, try power-cycling the modem and go through the AT command sequence from the beginning as outlined in Section 5.

8.2.2 403 Forbidden

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

8.2.3 400 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 your "GET" or "POST" command is correct.

8.2.4 RXTX Mismatch Warning

When trying to upload the certificates, if you get the below error message:

```
WARNING: RXTX Version mismatch

Jar version - RXTX-2.2-pre1.2
native lib version - RXTX-2.2pre2
```

```
Connection error! Disable flow control on module.
```

verify the the modem is powered on. If you are using the Skywire Development Kit, press the ON_BTN for 1-2 seconds and verify that LED D1 is lit.

8.3 Verify Credentials

If you believe that your credentials are valid but are unable to establish a connection using the Skywire modem, you can verify your credentials using OpenSSL. There are two ways to check the credentials: verifying that they are in a valid format, and using them to establish a connection to AWS.

8.3.1 Verify file format

To verify the private key, run the following command (substitute in your private key filename):

```
openssl rsa -in 2a6d9b3215-private.der -inform der -check
```

The response should be:

```
RSA key ok
Writing RSA key
-----BEGIN RSA PRIVATE KEY-----
…
-----END RSA PRIVATE KEY-----
```

To verify the certificate, run the following command (substitute in your certificate filename):

```
openssl x509 -in 2a6d9b3215-certificate.der -inform der
-text -noout
```

The response should show your certificate including header information.

8.3.2 Establish a connection to AWS

If your credentials pass verification with OpenSSL, you can use them with OpenSSL to establish a connection to AWS. To do this, issue the following command (replace bold text with text specific to your credentials and connection):

```
openssl s_client -servername
ACVWOJPAQ6LEI.iot.us-west-2.amazonaws.com -connect
ACVWOJPAQ6LEI.iot.us-west-2.amazonaws.com:8443 -CAfile
"VeriSign-Class%203-Public-Primary-Certification-Authority-
```

```
G5.der" -cert 2a6d9b3215-certificate.der -key
2a6d9b3215-private.der -certform DER -keyform DER
```

The response should say "CONNECTED" followed by information about the connection including the server certificate. At this point you can issue a "GET" command to the server, such as the following:

```
GET /things/AWS_Test_Thing/shadow HTTP/1.1
```

followed by "Enter" twice. The server should respond with

```
HTTP/1.1 200 OK
```

followed by header information as well as the JSON data associated with your thing's shadow.