Skywire® Nano NL-SW-LTE-NRF9160 Socket Dial Application Note

NimbeLink Corp
Updated: May 2020

Table of Contents

1. Introduction

1.1 Orderable Part Numbers

Orderable Device	Description	Carrier	Network Type
NL-SWNDK	Skywire Nano Development Kit	Any	Any
NL-SWN-LTE-NRF9160	LTE CAT M1	Any	LTE

1.2 Prerequisites

This document assumes that the initial setup of the requisite modem and development kit has been completed using the Skywire® Nano Development Kit User Manual.

If these steps are incomplete, please refer to the link above and complete the modem setup before proceeding.

2. Default Socket Dial Procedure

2.1 Overview

Socket dials are a useful process for uploading or downloading information from a website or database using HTTP, MQTT, other standards, or custom formatted data. This section describes the command mode method of sending a socket dial, and Section 3 displays a working example.

2.2 Socket Configuration

First, the socket must be created. The command to do so follows:

Command: `AT#XSOCKET=socket label,open|close,TCP|UDP`

Response: `#XSOCKET: socket label, TCP|UDP`
`OK`

2.3 Initiate Socket Dial

Next the socket needs to be connected to the host. The command to do so is as follows:

Command: `AT#XTCPCONN=socket label,hostname,port`

Response: `#XTCPCONN: 1`
`OK`

2.4 Send Data via HTTP

To send data, use an HTTP POST command. Before the POST command, the user should define the number of bytes to be sent.

Command: `AT#XTCPSEND=socket label[,length]`

The length parameter is optional and represents the number of bytes to send. The maximum value this can take on is 576, whether or not the length is specified.

Response: `>`

Now, type the correct command but do not hit the Enter key. The syntax of the `POST` and `GET` command are below:

Command: `POST` **`[ENDPOINT] HTTP/[HTTP VERSION]`**

Command: `GET` **`[ENDPOINT] HTTP/[HTTP VERSION]`**

Once the appropriate command has been entered, enter the sequence **CTRL+M, CTRL+J, CTRL+M, CTRL+J**. This will enter two sequences of new line/carriage return characters, which are necessary to signal to the server that data transmission is complete. Then enter **CTRL+Z, CTRL+Z**, which will signal the modem to send the data. The hex codes of the **CTRL+** combos and expected response are shown below:

CTRL+M: 0x0D **CTRL+J:** 0x0A **CTRL+Z:** 0x1A

Response: `#XTCPSEND: bytes sent`
`OK`

For more information on the method, please consult the documentation at the following URL:

https://tools.ietf.org/html/rfc1945#section-8

2.5 Receive Data

To view the data returned from the POST request, enter the following command:

Command: `AT#XTCPRECV=socket label,length,timeout`

Response: `#XTCPRECV: bytes received,data`

3. Working Example 1: Dweet.io

This section will provide a working example of sending and receiving data via an HTTP transfer using a Skywire NL-SWN-LTE-NRF9160 modem using a 4G Verizon LTE CAT M1 SIM.

dweet.io is a lightweight messaging service specifically designed for IoT (Internet of Things) devices. In addition to being lightweight, dweet.io does not require an account to get up and running. At www.dweet.io, they have an excellent “hello world” example that this example is based on.

3.1 Preliminary Setup Procedure

Before beginning the socket dial procedure, it is crucial to verify that cellular functionality is enabled. Type the following commands into the terminal:

Command: `AT+CFUN=1`

Response: `OK`

Command: `AT+CEREG?`

Response: `0,status`

The `status` value is the important one. If `status` is `1` or `5`, the modem is connected to the home network or roaming, respectively. If `status` is `2, 3, or 4,` more set up has to be done, as the device is not connected. Once network connectivity has been established, proceed to the next section.

3.2 Get IMEI of Modem

dweet.io requires a unique device name in order to send and receive data. It is recommended to use a device’s IMEI as this unique indicator. To display the IMEI, type the following command into the terminal, followed by the Enter key:

Command: `AT+CGSN`

Response: xxxxxxxxxxxxxxx
OK

This number should be identical to the IMEI printed on the modem label.

3.3 Socket Configuration

First, the socket must be opened. Issue this command into the terminal, followed by the Enter key to create a TCP socket labeled as 3:

Command: AT#XSOCKET=3,1,1

Response: #XSOCKET: 3,6
OK

3.4 Initiate Socket Dial

In the terminal program, type the following command to connect to dweet.io on that socket that was just opened**:**

Command: AT#XTCPCONN=3,"dweet.io",80

Response: OK

In this case, socket 3 is connected to “dweet.io” (the host), and 80 is the TCP port being used (TCP port 80 is used for HTTP).

3.5 Send Data via HTTP

To send data, use an HTTP POST command.

Command: AT#XTCPSEND=3

Response: >

Now, type the complete POST command but do not hit the Enter key. The syntax of the POST command is as follows:

Command: POST /dweet/for/xxxxxxxxxxxxxxx?hello=world HTTP/1.1

Where xxxxxxxxxxxxxxx is the IMEI found in Section 4.2. Once the appropriate command has been entered, press the sequence **CTRL+M, CTRL+J, CTRL+M, CTRL+J, CTRL+Z, CTRL+Z**. This will enter two sequences of new line/carriage return characters and send the request. The carriage returns/line feed characters are necessary to signal to the server that data transmission is complete.

Response: #XTCPSEND: 56
OK

3.6 Receive Data

To view the data returned from the POST request, enter the following command:

Command: AT#XTCPRECV=3, 500, 10

Response:

```
#XTCPRECV: 368,HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
Content-Length: 203
Date: <Today's date and time in GMT>
Connection: keep-alive


{"this":"succeeded","by":"dweeting","the":"dweet","with":{"thing":"xxx
xxxxxxxxxxx","created":"<time>","content":{"hello":"world"},"transact
ion":"<unique transaction id>"}}

OK
```

3.7 Close the Socket

Congratulations, you've successfully communicated with dweet.io. Finally, to follow good practice, we should close the socket.

Command: `AT#SOCKET?`

Response: `#XSOCKET=3,6`

This command shows that socket 3 is open. To close, enter the following:

Command: `AT#SOCKET=3,0`

Response: `#XSOCKET=0`

A response of `0` represents that the socket has been closed successfully.

4. Troubleshooting

4.1 Serial Clients

If the POST sequences are being inputted manually into a serial program such as PuTTY or TeraTerm, some serial clients may interpret the "Carriage Return" (CTRL+M) and "Line Feed" (CTRL+J) characters differently. For instance, if issues are encountered when sending information using an HTTP POST, try sending four "Line Feed" characters instead of the sequence of "Carriage Return" and "Line Feed".

For instance, instead of sending:

CTRL+M, CTRL+J, CTRL+M, CTRL+J, CTRL+Z, CTRL+Z

Try sending:

CTRL+J, CTRL+J, CTRL+J, CTRL+J, CTRL+Z, CTRL+Z

The issue could manifest itself as a malformed response, like the following:

```
>POST /dweet/for/xxxxxxxxxxxxxxx?hello=world HTTP/1.1#XTCPSEND: 52
OK
```

Notice the response starts immediately following `HTTP/1.1.` Trying a `AT#XTCPRECV=1` results with no data being read, as the request wasn't interpreted correctly by the host server. The response would look as follows:

```
#XTCPRECV: 0,
OK
```

After trying this `AT#XTCPRECV=1`, the socket will close itself, and issuing the `AT#SOCKET?` results in just `OK`, as the one socket we had open was closed.