OptConnect ema™

HTTP Using Socket Dials

Application Note: 001

V1.3 Updated May 2020

OptConnect
MANAGED WIRELESS SOLUTIONS

Table of Contents

1. Introduction

1.1 Scope

This document serves as a guide to performing raw socket dials with an OptConnect ema™ cellular modem. For demonstration purposes, OptConnect's ema:Play Evaluation Kit will be used as the host system throughout this document and is assumed to be configured for S2USB. Socket dialing is a technique that is used to download or upload information to and from a remote host server using a cellular connection. The process involves connecting to the host/server and opening a TCP/IP socket. This socket is then used by ema to send or receive data.

The remainder of this application note explains how to perform an HTTP session using socket dials with ema and offers several samples of HTTP exchanges.

1.2 Contact Information

For more information regarding OptConnect ema™ contact OptConnect Sales at 1.877.678.3343 ext. 2020 during normal business hours. For technical support contact OptConnect Customer Care Center at 1.877.678-3343 ext. 2021 from 8 am till 9 pm MST Monday through Saturday.

1.3 Orderable Part Numbers

Orderable Device	Primary Module Firmware Revision	Operating Temperature	LTE Bands	3G UMTS	Network	Region
EMA-L4-1-XX-A-A	20.00.505	-40 to +85˚C	FDD B2, B4, B5, B12, B13	B2, B5	AT&T, Verizon	North America
EMA-L4-1-US-B-A	20.00.005	-40 to +85˚C	FDD B2, B4, B5, B12, B13	B2, B5	AT&T, Verizon	United States
EMA-L4-1-XX-A-A-000	20.00.506	-40 to +85˚C	FDD B2, B4, B5, B12, B13	B2, B5	AT&T, Verizon	North America
EMA-L4-1-US-B-A-000	20.00.006	-40 to +85˚C	FDD B2, B4, B5, B12, B13	B2, B5	AT&T, Verizon	United States

Unless instructed otherwise EMA-L4-1-XX… will utilize AT&T as the primary carrier and Verizon as the secondary carrier. Unless instructed otherwise, EMA-L4-1-US… will utilize Verizon as the primary carrier and AT&T as the secondary carrier.

Orderable Device	Description	Operating Temperature	Region
EMA-ZZ-1-XX-Z-B	ema:Play Evaluation Kit, OptConnect ema™ evaluation platform	-40 to +85˚C	North America
EMA-L4-1-XX-A-B	ema:Play Evaluation Kit, OptConnect ema™ evaluation platform, EMA-L4-1-XX ema modem included	-40 to +85˚C	North America
EMA-L4-1-US-B-B	ema:Play Evaluation Kit, OptConnect ema™ evaluation platform, ema EMA-L4-1-US ema modem included	-40 to +85˚C	United States

1.4 Additional Resources

OptConnect ema™ is supported by a full range of documentation in addition to this document; including User Guides and Application Notes as well as an ema:Play User Guide and related code samples. The latest versions of these resources can be found at **http://optconnect.com/ema** . Suggested prerequisites for this document are the following:

- OptConnect ema™ Hardware Guide
- OptConnect ema™ emaLink AT Command Manual
- OptConnect ema™ Getting Started with ema
- ema:Play User Guide

2. HTTP Overview

2.1 Background

Before using the HTTP socket dial procedure with ema, it is important to first explain the communication protocol that enables this procedure.

HTTP or Hypertext Transfer Protocol is a networking protocol that allows for communication between a client and a host over a network connection. HTTP is a request-response protocol in which a client issues requests to the host, which the host then processes and returns an appropriate response to the client.

There are several different types of HTTP requests that a client can issue to a host, however, for the purposes of this application note only two will be covered. Namely, HTTP GET and HTTP POST requests.

2.2 HTTP POST and HTTP GET Methods

The HTTP POST method involves a client sending a request for the host to accept a sequence of data with the intent that the host will store that data. For instance, an HTTP POST request may be sent by a client wishing to post a message to a message board or upload a file to the host server. The general formatting of an HTTP POST request is as follows:

POST /test/demo_form.asp HTTP/1.1

Where **POST** is the HTTP request being issued, **/text/demo_form.asp** is the endpoint on the server, and **HTTP/1.1** is the HTTP version being used.

The HTTP GET method involves the client sending a request for access to data or information stored on the host server. For instance, a client may issue an HTTP GET request with information for a database query, or a client may issue an HTTP GET request to obtain a representation of some entity stored on the host server. The general formatting of an HTTP POST request is as follows:

GET /test/demo_form.asp HTTP/1.1

Where **GET** is the HTTP request being issued, **/text/demo_form.asp** is the endpoint on the server, and **HTTP/1.1** is the HTTP version being used.

To terminate an HTTP request, the host expects two pairs of carriage return and newline characters after an HTTP request. This pairing of a carriage return character and a newline character is called a line break.

With regards to ema and ema:Play, the required line breaks can be entered into the serial console by pressing the combination of CTRL+M and CTRL+J sequentially, and once for each required line break.

2.3 HTTP Responses

After issuing an HTTP request to a host server, it will respond with something similar to:

HTTP/1.1 2xx OK
Access-Control-Allow-Origin: *
Content-Type: application/json
Content-Length: 203
Date: Fri, 11 Jan 2019 23:03:10 GMT
Connection: keep-alive
[text response from server]

Where **2xx** is the response code of the HTTP host server. **200** is the general HTTP/1.1 response for a successful request, and means that the host server was able to interpret the request and respond accordingly. However, if the request was not valid, the server may respond with some other response code. Please see the link below for more information on the other potential response codes:

http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

3. General Socket Dial Procedure

3.1 Overview

Section 3 outlines two different techniques for completing a socket dial using ema. Note that these are not working examples, but general overviews. Section 4 provides two working examples that make use of the different techniques.

3.2 Socket Connection Modes

ema features two different types of connection modes for socket dials. The first of these modes is called "Online" mode and the other is called "Command" mode.

In "Online" mode, anything that is typed into the serial console during a socket connection is relayed via the cellular connection directly to the online host server as opposed to ema itself. However, in "Command" mode, anything typed into the serial console is interpreted as a command for ema. As such, the user must issue AT commands to send and receive data during the socket transaction when using "Command" mode.

Each of these modes accomplishes the same end goal of opening a socket, however the process of issuing the HTTP request varies based on the chosen mode.

3.2.1 Online Mode

In "Online" mode, ema will immediately expect an HTTP request after the socket open command is issued. Furthermore, the user is not able to see the data being sent over the serial line during the socket transmission. This is because the data from the serial line is being relayed directly to the online resource by ema without any further action by ema. Accordingly, it is often helpful to paste the HTTP request into the serial console as opposed to typing it in manually.

3.2.2 Command Mode

In "Command" mode, the socket open command simply opens a socket between ema and the online host server and does not expect an immediate HTTP request. All HTTP requests are sent using an AT command, and all responses from the host are read using another AT command. Also, unlike "Online" mode, the user is able to see the input to the serial line during a socket send command, which can make entering the HTTP request easier and less error prone.

3.2.3 Choosing a Connection Mode

"Online" mode and "Command" mode each involve a different technique for completing a socket dial. However, each of the two modes ultimately achieves the same goal of opening a socket and completing a data transaction. As such, choose whichever mode best suits the intended use case of the device and allows for the most efficient execution of the socket dial procedure.

3.3 APN and PDP Context Considerations

It is not necessary to set the APN on ema. The proper APN for the current cellular network will be chosen automatically by ema. Furthermore, the PDP context will automatically be updated with the proper APN. Please reference the *OptConnect ema™ Getting Started With ema* document for further information relative to PDP and activating the PDP.

3.4 Preliminary ema Setup

Ensure that ema has been properly configured and is ready to be used on the cellular network. For more information on the preliminary setup of ema, please refer to the following document; *OptConnect ema™ Getting Started with ema*.

3.5 Configure the Socket Connection

The socket must be configured for the connection. To do this, issue some form of the following command:

AT#SCFG=<connId>,<cid>,<pktSz>,<maxTo>,<connTo>,<txTo>

The list below contains more information on the arguments of the "AT#SCFG" command:

- **<connId>** is the socket connection identifier which ranges from 1 to 10.

  NOTE: ema uses connection identifiers 9 and 10 natively, which leaves 1 to 8 available for end user use.

- **<cid>** is the PDP context identifier used by the socket and is either 1 (AT&T) or 6 (Verizon). ema automatically determines this identifier value based on the inserted SIM card(s), and sets this during PDP configuration. It should not be changed by the user.
- **<pktSz>** is the packet size used by the TCP/UDP/IP stack during data transmission. This value is set to 300 bytes by default, but can be set to any value from 1 to 1500.
- **<maxTo>** is the data exchange timeout. That is, a socket will remain open until **<maxTo>** seconds of inactivity in the socket have passed. At that time, the socket will close. The default value is 90 seconds, but the range of allowed values is 1 to 65535 seconds.
- **<connTo>** is the connection timeout. This value controls how long ema will wait for a socket connection to be established before aborting the attempt. This value has a unit of deci seconds, a range of 10 to 1200, and a default value of 600. Ex. A timeout of 10 seconds would be entered as 100.
- **<txTo>** is the data transmission timeout. This controls how long ema will wait before sending a packet of data that is less than the max packet size. That is, if the user passes a sequence of data to ema that is smaller than the max packet size, ema will wait the duration of the data transmission timeout before sending the data to the host. This value has a unit of deci seconds, a range of 1 to 255, and a default value of 50. Ex. A timeout of 5 seconds would be entered as 50.

Example usage AT#SCFG

AT#SCFG=1,1,300,90,600,50

This example configures the first socket to use the first PDP context, and uses the default values for each of the other arguments to the "AT#SCFG" command.

After configuring the socket, ema is now ready to open the socket and begin data transmission.

3.6 Open the Socket

To open the socket, issue some form of the command below:

AT#SD=<connId>,<txProt>,<rProt>,<IPaddr>,<closureType>,<IPort>,<connMode>

The list below contains more information on the arguments of the "AT#SD" command:

- **<connId>** is the socket connection identifier which ranges from 1 to 8.

NOTE: ema uses connection identifiers 9 and 10, which leaves 1 to 8 available for end user use.

- **< txProt >** sets the transmission protocol. 0 sets TCP, and 1 sets UDP.
- **< rProt >** sets the remote host port that the socket will connect to.
- **< IPaddr >** is the IP address of the remote host. This argument can be provided as a raw IP address in the format "xxx.xxx.xxx.xxx", or any hostname that can be resolved by a DNS query.
- **< closureType >** sets the socket closure behavior for TCP connections when the remote host has closed the connection. This argument can either be 0 or 255, with 0 being the default value. A value of 0 specifies that the local host should close the socket connection immediately after the remote host closes its end of the connection, whereas a value of 255 indicates that the local host should close the connection only after the socket close command is issued.
- **< IPort >** sets the local port of the UDP connection. This value can range from 1 to 65535.
- **<connMode>** sets the socket connection mode. A value of 0 sets the socket connection mode to "Online" mode. A value of 1 sets the socket connection mode to "Command" mode. The default value is 0.

Section 3.7 outlines "Online" mode, while section 3.8 outlines "Command" mode.

3.7 Socket Dial Using "Online" Mode

1. **Open the Socket:** Following the syntax in Section 3.6, issue the following command to open the socket using "Online" mode:

   AT#SD=1,0,80,"www.example.com",0,0,0

   In this case, the above command opens the first socket with identifier '1' in "Online" mode, using TCP protocol, with a destination of "www.example.com", using port 80.

   ema will respond with:

   CONNECT

   if the socket was opened successfully. If it was unsuccessful, ema will respond with an error code describing the error. Please refer to the module AT command manual for more information.

2. **Send the HTTP Request:** At this point, the modem is now routing any data entered into the serial line directly to the remote host. As such, either type in or paste an HTTP request using the format detailed in Section 2.2. Finally, enter the termination sequence of CTRL+M, CTRL+J, CTRL+M, CTRL+J to complete the HTTP request.
3. **View the HTTP Response:** After a few moments, ema will output the response from the host server directly to the serial line and should be viewable on the serial console. The response will be similar in form to that of the response detailed in Section 2.3, with an HTTP response code indicating the success or failure of the request.
4. **Close the Socket:** To manually close the socket in "Online" mode, an escape sequence must be entered. This sequence is:
   - Wait greater than one second of no data transfer
   - Enter **+++**
   - Wait greater than one second of no data transfer

This time buffer on each side of the escape sequence is to minimize the chance that your application or your server will send +++ as valid data so ema doesn't misinterpret it.

3.8 Socket Dial Using "Command" Mode

1. **Open the Socket:** Following the syntax in Section 3.6, issue the following command to open the socket using "Command" mode:

   AT#SD=1,0,80,"www.example.com",0,0,1

   In this case, the above command opens the first socket with identifier '1' in "Command" mode, using TCP protocol with a destination of "www.example.com", using port 80.

   ema will respond with:

   OK

   After the socket has been opened, ema is now ready to issue HTTP requests to the host.
2. **Send the HTTP Request:** To send an HTTP request, issue the following command:

   AT#SSEND=<connId>

   where **<connId>** is replaced with the identifier for the currently opened socket.

   After issuing the above command, ema will respond with:

   >

   and will wait for an HTTP request. At this point, either type in or paste an HTTP request using the format detailed in Section 2.2. Finally, enter the termination sequence of CTRL+M, CTRL+J, CTRL+M, CTRL+J to complete the HTTP request.

   After the HTTP request has been properly entered, press CTRL+Z to instruct the modem to send the request.

   After sending the request, ema should respond with the following URC:

   SRING: <connId>

   where **<connId>** is replaced with the connection identifier for the socket in use. This URC indicates that the host has received the request, and has returned a response to ema.
3. **Read the HTTP Response:** To read the response, issue the following command:

   AT#SRECV=<connId>,<maxByte>

   where **<connId>** is the socket connection identifier and **<maxByte>** is the number of bytes to read from the response from the host server. The range of allowable values for this parameter is 1 to 1500. If the response from the host is greater than the specified **<maxByte>** value, then the **SRING: <connId>** will be returned, indicating that there is more of the response left to read. After issuing the above command, ema will output the response from the remote host to the serial line for viewing in the serial console.

4. **Close the Socket:** After the HTTP response has been read, close the socket by issuing the following command:

   AT#SH=<connId>

   where **<connId>** is replaced with the socket identifier for the currently active socket.

4. Working Examples

4.1 Overview

Section 4 outlines four working examples of socket dials using ema.

Two of the examples utilize *dweet.io*, which is a free publish-subscribe (pub-sub) platform for Internet of Things (IoT) devices. The first of the *dweet.io* examples uses "Online" mode to complete the socket dial, while the second of the *dweet.io* examples uses "Command" mode.

The next two examples demonstrate a more complex HTTP request using *httpbin.org*, which is an open-source HTTP request-response service. Again, the first of the *httpbin.org* examples uses "Online" mode to complete the socket dial, while the second of the *httpbin.org* examples uses "Command" mode.

4.2 *dweet.io* Examples

4.2.1 Overview

dweet.io is a lightweight messaging service specifically designed for IoT devices. In addition to being lightweight, *dweet.io* does not require the creation of an account before using the service. The only thing needed for the use of *dweet.io* is a unique device identifier, which is used to identify the "thing" that is created when data is sent to and from a device.

This section will detail how to perform a socket dial using *dweet.io* as the host server, and will demonstrate the "Online" and "Command" mode techniques in Section 4.2.3 and Section 4.2.4, respectively.

4.2.2 *dweet.io* Unique Device Identifier

It is easiest to use the IMEI of ema as the unique device identifier. However, feel free to choose any arbitrary unique device identifier, provided that it is at least several characters long.

To find the IMEI of an ema, issue the following AT command:

AT+CGSN

Record this identifier for future use, if using for this example.

Since the IMEI of an ema is sensitive information and should not be shared, the following working examples will use:

optconnect-test

as the unique device identifier.

4.2.3 “Online” Mode AT Command Log

The AT command log below contains an example of the "Online" mode socket dial technique with two HTTP requests sent in tandem during the same socket connection. The first request is an HTTP POST request, and the second is an HTTP GET request.

In the case of *dweet.io*, the HTTP POST request creates a "thing" on *dweet.io's* servers, and the HTTP GET request pulls down the information for the newly created "thing".

AT Commands entered by the user are in bold font, responses from the modem are in regular font, explanatory comments are in red font.

Note: If ema outputs the following:

NO CARRIER

then the socket has been closed and the open socket (**AT#SD**) command needs to be sent again.

```
// Configure the socket with default argument values. connId = 1, PDP context = 1 (AT&T).
AT#SCFG=1,1,300,90,600,50
OK

// Activate the PDP context, PDP context = 1 (AT&T).
AT#SGACT=1,1
#SGACT: XX.XXX.XX.XXX
OK

// Open the socket in "Online" mode.
AT#SD=1,0,80,"dweet.io",0,0,0
CONNECT

// Paste in the HTTP POST request, and press the CTRL key sequences.
// NOTE: this text is not visible to the user when it is entered in “Online” mode.
POST /dweet/for/optconnect-test?hello=world HTTP/1.1
CTRL+M, CTRL+J, CTRL+M, CTRL+J

// Here is an example of the response from the remote host. A response code of '200' indicates a
// successful request.
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
Content-Length: 203
Date: Mon, 14 Jan 2019 17:17:26 GMT
Connection: keep-alive
```

{"this":"succeeded","by":"dweeting","the":"dweet","with":{"thing":"optconnect-test","created":"2019-01-14T17:17:26.080Z","content":{"hello":"world"},"transaction":"52ca3d84-2856-4694-9fdd-94614d69dfad"}}

// Try an HTTP GET request in the same socket connection. Simply paste in the request as before, and **press the CTRL key sequences**.
GET /get/latest/dweet/for/optconnect-test HTTP/1.1
CTRL+M, CTRL+J, CTRL+M, CTRL+J

// Here is an example of the response from the remote host.
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
Content-Length: 152
Date: Mon, 14 Jan 2019 17:18:45 GMT
Connection: keep-alive

{"this":"succeeded","by":"getting","the":"dweets","with":[{"thing":"optconnect-test","created":"2019-01-14T17:17:26.080Z","content":{"hello":"world"}}]}

// Close the socket by issuing the escape sequence.
+++

OK
// The socket has been closed.
NO CARRIER

// Deactivate the PDP context, PDP context = 1 (AT&T).
AT#SGACT=1,0
OK

4.2.4 "Command" Mode AT Command Log

The AT command log below contains an example of the "Command" mode socket dial technique with two HTTP requests sent in tandem during the same socket connection. The first request is an HTTP POST request, and the second is an HTTP GET request.

In the case of *dweet.io*, the HTTP POST request creates a "thing" on *dweet.io's* servers, and the HTTP GET request pulls down the information for the newly created "thing".

Commands entered by the user are in bold font, responses from the modem are in regular font, and explanatory comments are in red font.

Note: If ema outputs the following:

NO CARRIER

then the socket has been closed and the open socket (**AT#SD**) command needs to be sent again.

// Configure the socket with default argument values. connId = 1, PDP context = 1 (AT&T).
AT#SCFG=1,1,300,90,600,50
OK

// Activate the PDP context, PDP context = 1 (AT&T).
AT#SGACT=1,1
#SGACT: XX.XXX.XX.XXX
OK

// Open the socket in "Command" mode.
AT#SD=1,0,80,"dweet.io",0,0,1
OK

// Paste in the HTTP POST request, and **press the CTRL key sequences.** Note the ">" character is // output by ema and should not be pasted/entered.
AT#SSEND=1
> **POST /dweet/for/optconnect-test?hello=world HTTP/1.1**
CTRL+M, CTRL+J, CTRL+M, CTRL+J, CTRL+Z

OK
// ema has received a response from the host.
SRING: 1

// Read up to 1500 bytes of the response from the first socket.
AT#SRECV=1,1500
// 368 bytes have been received in total.
#SRECV: 1,368

// Here is an example of the response from the remote host. A response code of '200' indicates a // successful request.
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
Content-Length: 203
Date: Mon, 14 Jan 2019 17:38:00 GMT
Connection: keep-alive

{"this":"succeeded","by":"dweeting","the":"dweet","with":{"thing":"optconnect-test","created":"2019-01-14T17:38:00.139Z","content":{"hello":"world"},"transaction":"3f3aad1c-4096-43f8-9164-28fef11eea73"}}

OK

// Try an HTTP GET request in the same socket connection. Paste in the HTTP GET request, and **press** // **the CTRL key sequences.** Note the ">" character is output by ema and should not be

```
// pasted/entered.
AT#SSEND=1
> GET /get/latest/dweet/for/optconnect-test HTTP/1.1
CTRL+M, CTRL+J, CTRL+M, CTRL+J, CTRL+Z

OK
// The modem has received a response from the host.
SRING: 1

// Read 1500 bytes of the response from the first socket.
AT#SRECV=1,1500
// 317 bytes have been received in total.
#SRECV: 1,317

// Here is an example of the response from the remote host.
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
Content-Length: 152
Date: Mon, 14 Jan 2019 17:38:17 GMT
Connection: keep-alive

{"this":"succeeded","by":"getting","the":"dweets","with":[{"thing":"optconnect-test","created":"2019-
01-14T17:38:00.139Z","content":{"hello":"world"}}]}

OK
// Close the socket.
AT#SH=1
OK

// Deactivate the PDP context, PDP context = 1 (AT&T).
AT#SGACT=1,0
OK
```

4.3 *httpbin.org* Examples

4.3.1 Overview

httpbin.org is an open-source HTTP request-response service that allows a user to test out several types of HTTP requests. The website has many different endpoints that the user can send requests to that will each return a unique response.

Section 4.3 contains two examples of a GET and a POST request for "Online" mode and "Command" mode.

The HTTP POST example demonstrates sending an HTTP POST request to the endpoint that contains some keys and associated data. The server then responds with the JSON formatting of the sent data, as well as the headers that were sent in the request.

The HTTP GET example involves sending a request to *httpbin.org* for a randomly generated Universally Unique Identifier (UUID). The host/server then responds with the UUID in JSON format.

This section will detail how to perform a socket dial using *httpbin.org* as the host/server, and will demonstrate the "Online" and "Command" mode techniques in Section 4.3.2 and Section 4.3.3, respectively.

4.3.2 HTTP Header Information

The examples in Section 4.3 involve the use of HTTP headers. These headers allow the client and the host/server to pass additional information during the transaction, and allow the client to specify attributes for the transaction such as permissible data formats, additional security measures, content length, and many other types of attributes.

4.3.3 "Online" Mode AT Command Log

The AT command log below contains an example of the "Online" mode socket dial technique using *httpbin.org*. First, an HTTP POST request is shown in which several keys with associated data are sent to the host/server. Second, an HTTP GET request is demonstrated in which a UUID is returned in the response from the host/server.

It is recommended to copy and paste each command into the serial terminal since the user is unable to see what is being sent to the serial line when using "Online" mode to complete the socket dial.

AT Commands entered by the user are in bold font, responses from the modem are in regular font, explanatory comments are in red font.

Note: If ema outputs the following:

NO CARRIER

then the socket has been closed and the open socket (**AT#SD**) command needs to be sent again.

```
// Configure the socket with default argument values. connId = 1, PDP context = 1 (AT&T).
AT#SCFG=1,1,300,90,600,50
OK

// Activate the PDP context, PDP context = 1 (AT&T).
AT#SGACT=1,1
#SGACT: XX.XXX.XX.XXX
OK

// Open the socket in "Online" mode.
AT#SD=1,0,80,"httpbin.org",0,0,0
CONNECT

// Paste in the HTTP POST request by pasting in each line as shown below. Press the CTRL key
// sequences accordingly.
```

```
// NOTE: this text is not visible to the user when it is entered in "Online" mode.
POST /post HTTP/1.1
CTRL+M, CTRL+J

Host: httpbin.org
CTRL+M, CTRL+J

Accept: application/xhtml+xml
CTRL+M, CTRL+J

Connection: keep-alive
CTRL+M, CTRL+J

Content-Type: application/x-www-form-urlencoded; charset=utf-8
CTRL+M, CTRL+J

Content-Length: 36
CTRL+M, CTRL+J, CTRL+M, CTRL+J

Key1=abcd&Key2=123&Key3=Hello World!

// Here is an example of the response (responses may vary)  from the remote host. A response code
// of '200' indicates a successful request.
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: *
Content-Type: application/json
Date: Wed, 29 May 2019 15:07:47 GMT
Referrer-Policy: no-referrer-when-downgrade
Server: nginx
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Content-Length: 427
Connection: keep-alive

{
   "args": {},
    "data": "",
    "files": {},
    "form": {
      "Key1": "abcd",
      "Key2": "123",
      "Key3": "Hello World!"
     },
    "headers": {
```

```
      "Accept": "application/xhtml+xml",
      "Content-Length": "36",
      "Content-Type": "application/x-www-form-urlencoded; charset=utf-8",
      "Host": "httpbin.org"
    },
    "json": null,
    "origin": "204.156.181.113, 204.156.181.113",
    "url": "https://httpbin.org/post"
}

OK

// Try an HTTP GET request in which the host will respond with
// a randomly generated UUID.
// Paste in the HTTP GET request, and press the CTRL key sequences.
// NOTE: this text is not visible to the user when it is entered in "Online" mode.
GET /uuid HTTP/1.1
CTRL+M, CTRL+J

Host: httpbin.org
CTRL+M, CTRL+J

Accept: application/json
CTRL+M, CTRL+J

Connection: Close
CTRL+M, CTRL+J, CTRL+M, CTRL+J

// Here is an example of the response (responses may vary)  from the remote host
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: *
Content-Type: application/json
Date: Wed, 29 May 2019 16:13:24 GMT
Referrer-Policy: no-referrer-when-downgrade
Server: nginx
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Content-Length: 53
Connection: Close

{
  "uuid": "34cca9e0-4b71-44af-bd35-2698f7a5bb63"
}
```

```
// The socket has been closed.
NO CARRIER

// Deactivate the PDP context, PDP context = 1 (AT&T).
AT#SGACT=1,0
OK
```

4.3.4 "Command" Mode AT Command Log

The AT command log below contains an example of the "Command" mode socket dial technique using *httpbin.org*. First, an HTTP POST request is shown in which several keys with associated data are sent to the host/server. Second, an HTTP GET request is demonstrated in which a UUID is returned in the response from the host/server.

AT Commands entered by the user are in bold font, responses from the modem are in regular font, explanatory comments are in red font.

Note: If ema outputs the following:

NO CARRIER

then the socket has been closed and the open socket (**AT#SD**) command needs to be sent again.

```
// Configure the socket with default argument values. connId = 1, PDP context = 1 (AT&T).
AT#SCFG=1,1,300,90,600,50
OK

// Activate the PDP context, PDP context = 1 (AT&T).
AT#SGACT=1,1
#SGACT: XX.XXX.XX.XXX
OK

// Open the socket in "Command" mode
AT#SD=1,0,80,"httpbin.org",0,0,1
OK

// Paste in the HTTP POST request by pasting in each line as shown below. Press the CTRL key
// sequences accordingly. Note the ">" character is output by ema and should not be
// pasted/entered.
AT#SSEND=1
> POST /post HTTP/1.1
CTRL+M, CTRL+J

Host: httpbin.org
CTRL+M, CTRL+J
```

Accept: application/xhtml+xml
CTRL+M, CTRL+J

Connection: keep-alive
CTRL+M, CTRL+J

Content-Type: application/x-www-form-urlencoded; charset=utf-8
CTRL+M, CTRL+J

Content-Length: 36
CTRL+M, CTRL+J, CTRL+M, CTRL+J

Key1=abcd&Key2=123&Key3=Hello World!
CTRL+Z

OK
// ema has received a response from the host.
SRING: 1

// Read up to 1500 bytes of the response from the first socket.
AT#SRECV=1,1500
// 781 bytes were received in total.
#SRECV: 1,781

// Here is an example of the response (responses may vary) from the remote host. A response code
// of '200' indicates a successful request.
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: *
Content-Type: application/json
Date: Wed, 29 May 2019 18:01:48 GMT
Referrer-Policy: no-referrer-when-downgrade
Server: nginx
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Content-Length: 427
Connection: keep-alive

{
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "Key1": "abcd",
    "Key2": "123",

```
    "Key3": "Hello World!"
  },
  "headers": {
    "Accept": "application/xhtml+xml",
    "Content-Length": "36",
    "Content-Type": "application/x-www-form-urlencoded; charset=utf-8",
    "Host": "httpbin.org"
  },
  "json": null,
  "origin": "204.156.181.113, 204.156.181.113",
  "url": "https://httpbin.org/post
}

OK

// Try an HTTP GET request by pasting in each line as shown below. Press the CTRL key
// sequences accordingly. Note the ">" character is output by ema and should not be
// pasted/entered.
AT#SSEND=1
> GET /uuid HTTP/1.1
CTRL+M, CTRL+J

Host: httpbin.org
CTRL+M, CTRL+J

Accept: application/json
CTRL+M, CTRL+J

Connection: Close
CTRL+M, CTRL+J, CTRL+M, CTRL+J, CTRL+Z

OK
// ema has received a response from the host.
SRING: 1

// Read up to 1500 bytes of the response from the first socket.
AT#SRECV=1,1500
// 401 bytes were received in total.
#SRECV: 1,401

// Here is an example of the response (responses may vary)  from the remote host. A response code
// of '200' indicates a successful request.
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: *
Content-Type: application/json
```

```
Date: Wed, 29 May 2019 18:08:03 GMT
Referrer-Policy: no-referrer-when-downgrade
Server: nginx
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Content-Length: 53
Connection: Close

{
   "uuid": "49e5baae-3d7d-4d2b-aabb-5e76dc0ef555"
}

OK
// The socket has been closed.
NO CARRIER

// Deactivate the PDP context, PDP context = 1 (AT&T).
AT#SGACT=1,0
OK
```


5. Revision History

Revision	Date	Description	Author
1.0	6/4/2019	Initial Release	MSV
1.1	9/10/2019	Added PDP activate/deactivate commands to samples in sections 4.2 and 4.3	MSV
1.2	1/15/2020	ema Management UART is now referred to as emaLink interface	MSV
1.3	5/18/2020	Updated ema p/n's	MSV