

**Users Guide:
GSM0308UG001**

Enfora GSM/GPRS Family API Reference

Version: 1.05

Enfora, Inc.
www.enfora.com

| | |
|-----------------------------|--------------------------------------|
| Document Title: | Enfora GSM/GPRS Family API Reference |
| Version: | 1.05 |
| Date: | 1/21/2011 |
| Status: | Released |
| Document Control ID: | GSM0308UG001 |

General

All efforts have been made to ensure the accuracy of material provided in this document at the time of release. However, the items described in this document are subject to continuous development and improvement. All specifications are subject to change without notice and do not represent a commitment on the part of Enfora, Inc. Enfora, Inc. will not be responsible for any loss or damages incurred related to the use of information contained in this document.

This product is not intended for use in life support appliances, devices or systems where a malfunction of the product can reasonably be expected to result in personal injury. Enfora, Inc. customers using, integrating, and/or selling this product for use in such applications do so at their own risk and agree to fully indemnify Enfora, Inc. for any damages resulting from illegal use or resale.

Copyright

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording or otherwise), or for any purpose, without the express written permission of Enfora, Inc.

Enfora may have patents, patent applications, trademarks, copyrights or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Enfora, the furnishing of this document does not give you any license to these patents, trademarks, copyrights or other intellectual property.

©2002 – 2008 Enfora, Inc. All rights reserved.

Enabler and Spider are either registered trademarks or trademarks of Enfora, Inc. in the United States.

| Date | Rev | Author | Description |
|----------------------------------|-------|----------------|---|
| December 11 th , 2006 | Draft | C. Patel | Draft |
| June 26, 2007 | 1.00 | Tom Cone | Initial Release |
| November 9, 2007 | 1.01 | Diane O'Neil | Edited Table 6 – API Table |
| October 24, 2008 | 1.02 | Scott Wittrock | Added API Optional Header |
| May 12, 2010 | 1.03 | Robert Keating | Corrected paragraph 4.4.7.4 (Event Sequence Number information) |
| June 2, 2010 | 1.04 | Jim Korns | Added new Information Element message |
| January 21 2011 | 1.05 | Jim Fairhurst | Corrected paragraph 4.4.4 (Added UDP and TCP sections) |

Table Of Contents

| | | |
|---------|---|----|
| 1 | Overview | 1 |
| 2 | References | 2 |
| 3 | Modem Communication | 3 |
| 3.1 | Serial Communication | 3 |
| 3.2 | Over The Air Communication | 4 |
| 4 | AT Commands Over DUN or OTA | 5 |
| 4.1 | IP Header | 8 |
| 4.2 | UDP Header | 10 |
| 4.3 | TCP Header | 11 |
| 4.4 | API Header | 12 |
| 4.4.1 | Wakeup Messages | 14 |
| 4.4.2 | Send Password to Modem | 15 |
| 4.4.3 | Read/Write AT Command Request | 16 |
| 4.4.4 | Unsolicited Messages | 17 |
| 4.4.5 | ACK Messages | 18 |
| 4.4.6 | ERROR Messages | 19 |
| 4.4.7 | API Optional Header | 19 |
| 4.4.7.1 | End of Option Sequence – Type 0 | 22 |
| 4.4.7.2 | MDMID – Type 1 | 23 |
| 4.4.7.3 | Output Message Event Format- Type 2 | 24 |
| 4.4.7.4 | Event Sequence Number – Type 3 | 24 |
| 4.4.7.5 | Firmware Version - Type 4 | 25 |
| 4.4.7.6 | Cause Code – Type 5 | 25 |
| 4.4.8 | FOTA Complete Message | 25 |
| 5 | Terms Explained | 27 |
| 5.1 | Big Endian | 27 |
| 5.2 | Byte | 27 |
| 5.3 | Little Endian | 27 |
| 6 | Example: | 28 |

1 Overview

This document provides a description of the Application Program Interface (API) for the Enfora's GSM/GPRS modems.

With this API, programmers can access information and control modem functions in real-time. A wide range of information is available via the API and includes modem management and status functions. This environment would best be utilized where a customized software application is being considered and real-time performance parameters are mandatory. A good example of the necessity for the API is in a real-time monitoring application that includes a status window to report performance and indicate when network conditions begin to degrade. Data is consistently being updated during the established session.

Many host computers, which use this API, will contain a TCP/IP stack, which includes UDP and Point-to-Point Protocol (PPP). However, this is not a requirement. This API includes documentation and details to create your own UDP and PPP formatting for a minimal implementation.

This document contains proprietary information and must not be reproduced without the prior written consent of Enfora, Inc. (Enfora).

2 References

1. GSM0308AT001 - Enabler III AT Command Set
2. GSM 07.05: “Digital cellular telecommunications systems (Phase 2+); Use of Data Terminal Equipment – Data Circuit terminating Equipment (DTE – DCE) interface for Short Message Service (SMS) and Cell Broadcast Service (CBS)”.
3. GSM 07.07: “Digital cellular telecommunications systems (Phase 2+); AT command set for GSM Mobile Equipment (ME)”.
4. ITU-T Draft new Recommendation V.25ter: “Serial asynchronous automatic dialing and control”.

3 Modem Communication

Most AT commands may be read and/or written via UDP/IP or TCP/IP. The implemented AT command set is a compliant subset of GSM Ref. 07.07, GSM Ref. 07.05, and ITU-T Ref. V.25ter.

The AT command state is entered upon power-up/reset. The modem always starts in AT command mode. Use Windows HyperTerminal or similar application to send AT commands to the modem. Connect directly to the COM port used by the modem.

If real-time status/control is not required, the AT Command set may offer an easier integration alternative. Refer to AT Command Set Reference document (Enfora Document GSM0107PB001MAN) for a list of AT commands.

The user may choose to use AT Commands to configure the modem and/or start GSM/GPRS registration before switching to the UDP/IP or TCP/IP messaging for real-time status during data transfers.

A user can communicate with the modem via one of the two possible methods:

- RS232 Serial communication
- Over The Air (OTA) via SMS or GPRS

3.1 Serial Communication

Modem's default serial communication is set at 115200 baud, no parity, 8 data bits, 1 stop bit, and hardware flow control enabled. A user can send/receive AT commands, data, or response to/from the modem via one of the two methods:

- Straight serial communication via HyperTerminal or similar application
- Serial communication via Dial-Up Network (DUN)

Straight serial communication provides the user with the following capabilities:

- Send AT commands and receive response
- Receive SMS notification
- Make a voice, data or fax call
- Receive any unsolicited message

Serial communication via DUN provides the user with the following capabilities:

- Send and receive UDP/IP and/or TCP/IP data
- Surf the internet (provided data service is provided by the service provider) via internet explorer or similar application software
- Send AT commands via UDP/IP (described in section 4 of this document). The destination IP address and port number for AT commands sent via UDP/IP can be configured via AT\$UDPAPI command.
- Make a voice or fax call by sending AT commands via UDP/IP
- Receive SMS notification via UDP/IP
- Receive any unsolicited message via UDP/IP

3.2 Over The Air Communication

A user can send/receive AT commands to configure the modem OTA via UDP/IP or TCP/IP. Configuration command request has to be sent at modem's IP address and port number configured by AT\$UDPAPI command for UDP/IP or AT\$TCPSRC command for TCP/IP. A user must make sure that they don't send non-configuration IP data at this port for the modem will ignore it. See section 7 for details on how AT commands can be sent OTA.

4 AT Commands Over DUN or OTA

Figure 1 describes the communication between a modem and a PC (or any RS232 compatible device) via a Dial-Up Network (DUN) connection. To send and receive AT commands over UDP/IP, a user must create the message first and then encapsulate it within a UDP/IP header. DUN connection must be established before the message is sent to the modem. The modem will process the request and respond depending on the type of request.

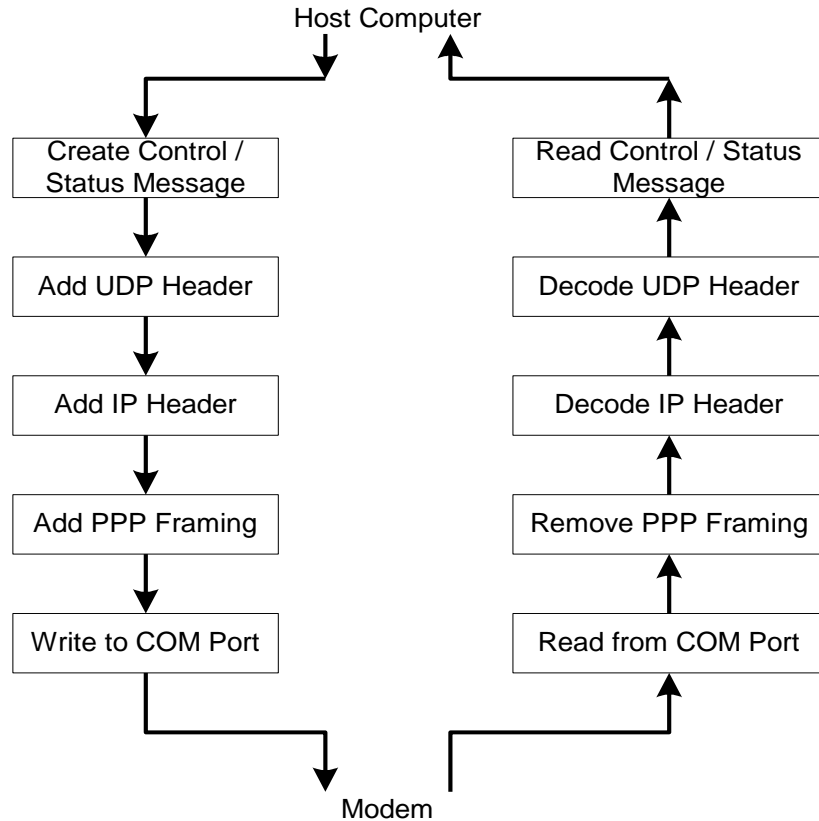


Figure 1. Communication Flow Chart.

Please note that API communication using the TCP protocol is **ONLY** available over the air (OTA). The TCP protocol is not operable over a PPP link. The UDP version of the API should be used for serial communication.

Table 1 describes the details of an AT command sent to the modem via UDP/IP over a DUN session. All messages are commanded from the host computer or remote host, and then responded to by the modem. The modem may generate unsolicited UDP/IP messages if subscribed.

Table 2 describes the details of an AT command sent to the modem via TCP/IP over the air. All messages are commanded from the remote host, and then responded to by the modem. The modem may generate unsolicited TCP/IP messages if subscribed.

The data order for all fields is big endian (most significant byte first).

As described below, most of the fields within the UDP and IP headers will remain constant for all messages. The fields that will have to be modified are: Length of IP Packet, Packet Number, IP header checksum, Source IP, Destination IP, Source Port number, Destination port numbers, UDP packet length, and UDP checksum

| Bytes | Bits 0 – 7 | Bits 8 – 15 | Bits 16 – 23 | Bits 24 – 31 | Header Information |
|-------------------|-------------------------------|--------------------|-------------------------|---|-----------------------|
| 0 – 3 | Version length | Type Of Service | Length of the IP Packet | | IP Header |
| 4 – 7 | Packet Number | | Fragmentation Offset | | |
| 8 – 11 | Time To Live | Protocol | IP header checksum | | |
| 12 – 15 | Source IP | | | | |
| 16 – 19 | Destination IP | | | | |
| 20 – 23 | Source Port Number | | Destination Port Number | | UDP Header |
| 24 – 27 | Length of UDP Packet | | UDP Checksum | | |
| 28 – 31 | API Number | | Command Type | API Optional Header Size (bytes) | API Message Header |
| 32 thru (32+m) | API Optional Header (m bytes) | | | | |
| (32+m) thru n | Data/AT command | | | | Data |

Table 1 - ATI command sent via UDP/IP

| Bytes | Bits 0 – 7 | Bits 8 – 15 | Bits 16 – 23 | Bits 24 – 31 | Header Information |
|-------------------|---|---|---|-----------------|--------------------------------|
| 0 – 3 | Version length | Type Of Service | Length of the IP Packet | | IP Header |
| 4 – 7 | Packet Number | | Fragmentation Offset | | |
| 8 – 11 | Time To Live | Protocol | IP header checksum | | |
| 12 – 15 | Source IP | | | | |
| 16 – 19 | Destination IP | | | | |
| 20 – 23 | Source Port Number | | Destination Port Number | | TCP Header |
| 24 – 27 | Sequence Number | | | | |
| 28 – 31 | Acknowledge Number | | | | |
| 32 – 35 | Data Offset + Control Bits | | Window | | |
| 36 – 39 | TCP header checksum | | Urgent Pointer | | API Message Header /Data |
| 40 – 43 | Data Length* | | API Number | | |
| 44 – 47 | Command Type | API Optional Header Size (bytes) | API Optional Header (m bytes) or Data/AT command | | |
| 48 thru (46+m) | API Optional Header or Data/AT command (cont) | | | | |
| (46+m) thru n | Data/AT command | | | | Data |

* Only used for API commands sent via TCP/IP

Table 2 - ATI command sent via TCP/IP

4.1 IP Header

The Internet Protocol (IP) header consists of 20 bytes. The definition and minimal implementation consists of the following (see RFC 791 for further details):

| Bytes | Bits 0 – 7 | Bits 8 – 15 | Bits 16 – 23 | Bits 24 – 31 |
|---------|-------------------|--------------------|-------------------------|-----------------|
| 0 – 3 | Version length | Type Of Service | Length of the IP Packet | |
| 4 – 7 | Packet Number | | Fragmentation Offset | |
| 8 – 11 | Time To Live | Protocol | IP header checksum | |
| 12 – 15 | Source IP | | | |
| 16 – 19 | Destination IP | | | |

Table 3 - IP Header (RFC 791)

- Byte 0: 8-bit bit field for version and length. This API only supports version 4 with IP header length of $5 \times 4 = 20$ bytes. This field must be set to 0x45.
- Byte 1: 8-bit type of service. The API ignores this field. This field must be set to 0x00.
- Bytes 2 - 3: 16-bit total length of packet. This field must be changed for each API message. It includes the message data, IP header, and UDP header. This field equals data Length + 28 (size of UDP and IP headers).
- Bytes 4 - 5: 16-bit identification. This field may be incremented for each packet. It is not required and may be left 0x0000.
- Bytes 6 - 7: 16-bit Fragmentation offset. The API ignores this field. This field must be set to 0x0000.
- Bytes 8: 8-bit Time to live. The API ignores this field. This field must be set to 0x00.
- Bytes 9: 8-bit Protocol. The API only supports UDP. This field must be set to 17 (0x11).
- Bytes 10 - 11: 16-bit IP header checksum.
- Bytes 12 - 15: 32-bit source IP address. For messages from the host to the modem, this is the IP address of host's UDP port. This IP address may be any valid IP address desired by the user's application, but will need to match the host's TCP/IP stack. This IP address will be used as the destination IP address for all response messages from the modem.

Bytes 16 - 19: 32-bit destination IP address. For messages from the host to the modem, this is the IP address of modem's UDP API port. This IP address may be configured using AT commands if desired. This IP address will be used as the source IP address for all response messages from the modem. The default IP address for the modem's UDP API is 199.245.180.13. Unless changed via AT commands, byte 16 will be 199 (0xC7), byte 17 will be 245 (0xF5), byte 18 will be 180 (0xB4), and byte 19 will be 13 (0x0D).

4.2 UDP Header

The User Datagram Protocol (UDP) header consists of 8 bytes. The definition and minimal implementation consists of the following (see RFC 768 for further details):

| Bytes | Bits 0 – 7 | Bits 8 – 15 | Bits 16 – 23 | Bits 24 – 31 |
|-------|----------------------|----------------|-------------------------|-----------------|
| 0 – 3 | Source Port Number | | Destination Port Number | |
| 4 – 7 | Length of UDP Packet | | UDP Checksum | |

Table 4 - UDP Header (RFC 768)

- Bytes 0 - 1: 16-bit source port number. For messages from the host to the modem, this is the source port number of the host's UDP port. This port may be any number desired by the user's application. This number will be used as the destination port for all response messages from the modem.
- Bytes 2 - 3: 16-bit destination port number. For messages from the host to the modem, this is the port number of the modem's UDP API port. The modem's UDP API port number may be changed using AT commands (AT\$UDPAPI) if desired. The modem's default UDP API port number is 1720 (0x06B8).
- Bytes 4 - 5: 16-bit length of UDP packet. This is the data length only; it does not include the IP header length. This length must be filled in for each message depending upon the amount of data in the packet.
- Bytes 6 - 7: 16-bit UDP checksum. This checksum may be used to validate the UDP packet. If the value is 0, then the checksum is ignored.

4.3 TCP Header

The Transmission Control Protocol (TCP) header consists of 20 bytes. The definition and minimal implementation consists of the following (see RFC 793 for further details):

| Bytes | Bits 0 – 7 | | Bits 8 – 15 | | | | | | Bits 16 – 23 | Bits 24 – 31 |
|---------|------------------------|----------|----------------|-----|-----|-----|------------------|-----|-----------------|-----------------|
| 0 – 3 | Source Port | | | | | | Destination Port | | | |
| 4 – 7 | Sequence Number | | | | | | | | | |
| 8 – 11 | Acknowledgement Number | | | | | | | | | |
| 12 – 15 | Data Offset | Reserved | URG | ACK | PSH | RST | SYN | FIN | Window | |
| 16 – 19 | Checksum | | | | | | Urgent Pointer | | | |

Table 5 - TCP Header (RFC 793)

Bytes 0 - 1: 16-bit source port number

Bytes 2 - 3: 16-bit destination port number. For messages from the host to the modem, this is the port number of the modem's TCP API port. The modem's TCP API port number may be changed using AT commands (AT\$TCPSRC) if desired. The modem's default TCP API port number is 1024 (0x0400).

Bytes 4 - 7: 32-bit sequence number of the first data octet in this segment

Bytes 8 - 11: 32-bit acknowledge number

Bytes 12 - 15:

4 – bits: Data Offset, number of 32 bit words in the TCP header indicating where data begins

6 – bits: Reserved.

URG: Urgent Pointer field significant

ACK: Acknowledgment field significant

PSH: Push function

RST: Reset the connection

SYN: Synchronize sequence numbers

FIN: No more data from sender

16 bits: Number of data octets beginning with the one indicated in the acknowledgement field, which the sender of this segment is willing to accept

Bytes 16 - 17: 16-bit checksum

Bytes 18 - 19: Urgent Pointer

4.4 API Header

Configuration commands/messages can be sent to the modem either OTA or when a DUN connection is present. Commands sent over a DUN connection can only be sent via UDP/IP. However, commands sent OTA can be sent via UDP/IP or TCP/IP. Table 1 describes a high level overview for data sent over UDP/IP while table 2 describes a high level overview for data sent over TCP/IP. The API commands provide easy integration and message parsing for the embedded developer and application developer. The Registration of Unsolicited message registers the requestor's IP address and Port number and sends unsolicited messages to the requestor's IP address and Port number.

The base API message header is 4 bytes long for commands sent via UDP/IP or 6 bytes long for commands sent via TCP/IP. The API message header, for both UDP/IP and TCP/IP, can be extended up to an additional 255 bytes by the inclusion of an API Optional Header. The Supported API number and commands are described in table 5 below. The definition and minimal implementation consists of the following:

Messages sent via UDP/IP has the following API header

- Bytes 0 - 1: 16-bit API number
- Bytes 2: 8-bit Command Type information. This value determines the type of message being sent or received by the host
- Bytes 3: API Optional Header Size. This field defines the size of the API Optional Header in Bytes. This field is set to zero (0) if an API Optional Header is not included.
- Bytes 4 thru (4+m): API Optional Header. See section 4.4.7 for details

Messages sent via TCP/IP has the following API header

- Bytes 0 - 1: 16-bit Data length (data length should include the 6 byte of API header as part of its length – ex: **00 1a** in below given example)
- Bytes 2 - 3: 16-bit API number (ex: **00 0a** in below given example)
- Bytes 4: 8-bit Command Type information. This value determines the type of message being sent or received by the host (ex: **08** in below given example)
- Bytes 5: API Optional Header Size. This field defines the size of the API Optional Header in Bytes. This field is set to zero (0) if an API Optional Header is not included.(ex: **00** in below given example)
- Bytes 6 thru (6+m): API Optional Header. See section 4.4.7 for details

Example 26 byte TCP connect ID message:

```
00 1a 00 0a 08 00 31 32 33 34 35 36 37 38 39 30 | .'. . . . 1234567890 |
31 32 33 34 35 36 37 38 39 30 | 1234567890 |
```

| API number (values given below are decimal) | | Command Type | API Optional Header Size | | |
|---|----------|-----------------------------------|-------------------------------|-------------|------------------------|
| Byte – 0 | Byte – 1 | Byte – 2 | Byte – 3 | Direction | |
| 0 – 4 Reserved 5 – GPS Binary Read* 6 – 65535 Reserved | | 0 (Read Request) | 0 | Modem ← OTA | |
| 0 – Unsolicited Msg Request 1 – 9 Reserved 10 – ACK 11 – Password 12 – 65535 Reserved | | 1 (Write Request) | 0 | Modem ← OTA | |
| API number (values given below are decimal) | | Command Type | API Optional Header Size | | |
| Byte – 0 | Byte – 1 | Byte – 2 | Byte – 3 | Direction | |
| 0 – Unsolicited Msg 1 – 3 Reserved 4 – ASCII Event Data (Param2>=256)* 5 – Binary Event Data* 6 – Reserved 7 – ASCII TAI Data* 8 – \$MSGSND Data 9 – Reserved 10 – ASCII Event Data (Param2<256) 12 – Information Element Data | | 2 (General Status Information) | (Size of API Optional Header) | Modem → OTA | |
| Echo first 2 bytes of an incoming data | | 3 (Error) | (Size of API Optional Header) | Modem → OTA | |
| 0 – 65535 | | 4 (AT Command) | 0 | Modem ← OTA | |
| Echo first 2 bytes of an incoming AT command request | | 5 (AT Command Response) | (Size of API Optional Header) | Modem → OTA | |
| First 2 bytes of AT\$UDPMSG command in ASCII format | | 6 | (Size of API Optional Header) | Modem ↔ OTA | |
| First 2 bytes of AT\$UDPMSG command in Binary format | | 7 | (Size of API Optional Header) | Modem ↔ OTA | |
| 10 | | 8 | 0 | Modem ID | Modem → OTA (TCP only) |

* GPS data only transmitted on select devices

Table 6 - API Table

4.4.1 Wakeup Messages

The wakeup/keep-alive message is sent to the local host as a “Status” command type message. Following data will be sent by the modem at the periodic interval configured by the AT\$WAKEUP command:

| Bytes | Data Description | Comments |
|-------|------------------|------------------------------------|
| 0 | 0x00 | Parameter Number |
| 1 | 0x0A | |
| 2 | 0x02 | Status |
| 3 | 0x00 | Reserved |
| 4 | 0x20 | Parameter Number (1) |
| 5 | 0x20 | |
| 6 | 0x20 | |
| 7 | 0x20 | |
| 8 | 0x20 | |
| 9 | 0x20 | |
| 10 | 0x20 | |
| 11 | 0x20 | |
| 12 | 0x20 | |
| 13 | 0x31 | |
| 14 | 0x20 | Modem ID (12345678901234567890) |
| 15 | 0x20 | |
| 16 | 0x31 | |
| 17 | 0x32 | |
| 18 | 0x33 | |
| 19 | 0x34 | |
| 20 | 0x35 | |
| 21 | 0x36 | |
| 22 | 0x37 | |
| 23 | 0x38 | |
| 24 | 0x39 | |
| 25 | 0x30 | |
| 26 | 0x31 | |
| 27 | 0x32 | |
| 28 | 0x33 | |
| 29 | 0x34 | |
| 30 | 0x35 | |
| 31 | 0x36 | |
| 32 | 0x37 | |
| 33 | 0x38 | |
| 34 | 0x39 | |
| 35 | 0x30 | |
| 36 | 0x20 | |

Table 7 - Wakeup/keep alive message

4.4.2 Send Password to Modem

The modem requires that the remote request must come from a “friendly IP” (set by AT\$FRIEND command) or be the most recent IP to correctly provide the password. All remote requests will be accepted if the password is set to NULL. The password has to be 8 bytes alphanumeric (A-Z, 0 –9) upper case characters only. Password should be filled with NULL characters if the password is less than 8 characters long. Send the following data to send the password:

| Bytes | Data Description | Comments |
|-------|------------------|---|
| 0 | 0x00 | Parameter Number |
| 1 | 0x0B | |
| 2 | 0x01 | Write Request |
| 3 | 0x00 | Reserved |
| 4 | 0x41 | Password (ABCDEF) <8 – Alpha-Numeric upper case characters (0 – 9 or A – Z)> |
| 5 | 0x42 | |
| 6 | 0x43 | |
| 7 | 0x44 | |
| 8 | 0x45 | |
| 9 | 0x46 | |
| 10 | 0x00 | |
| 11 | 0x00 | |

Table 8 - Sending of password to the modem OTA

4.4.3 Read/Write AT Command Request

AT commands mentioned in the *Enabler-IIG AT Command Set GSM0108PB001MAN* document can be sent to the modem when a DUN session is present or OTA. The host-to-modem message structure for reading/writing an AT command is as follows:

| Bytes | Data Description | Comments |
|-------|------------------|-----------------------|
| 0 | 0x00 | Sequence Number |
| 1 | 0x01 | |
| 2 | 0x04 | AT Command Read/Write |
| 3 | 0x00 | Reserved |
| 4 | 0x41 | AT Command (ATI) |
| 5 | 0x54 | |
| 6 | 0x49 | |

Table 9 - AT command to the modem

The modem will respond with the following message:

| Bytes | Data Description | Comments |
|-------|------------------|------------------------------------|
| 0 | 0x00 | Sequence Number |
| 1 | 0x01 | |
| 2 | 0x05 | AT Command Response |
| 3 | 0x00 | Reserved |
| 4 | 0x0D | AT Command Response (Enfora, Inc.) |
| 5 | 0x0A | |
| 6 | 0x45 | |
| 7 | 0x6E | |
| 8 | 0x66 | |
| 9 | 0x6F | |
| 10 | 0x72 | |
| 11 | 0x61 | |
| 12 | 0x2C | |
| 13 | 0x20 | |
| 14 | 0x49 | |
| 15 | 0x6E | |
| 16 | 0x63 | |
| 17 | 0x2E | |
| 18 | 0x0D | |
| 19 | 0x0A | |

Table 10 - AT command response

4.4.4 Unsolicited Messages

4.4.4.1 UDP API

A host has to register, with the modem, to receive any unsolicited messages. The modem saves the host's IP address and Port number. Unsolicited messages will then be sent to the IP and Port number that the user sends its request from. The host should send the following message structure to register the reception of unsolicited messages:

| Bytes | Data Description | Comments |
|-------|------------------|---|
| 0 | 0x00 | Parameter Number (unsolicited message registration request) |
| 1 | 0x00 | |
| 2 | 0x01 | Write Request |
| 3 | 0x00 | Response status |

Table 11 - Message structure for registration of unsolicited messages

The modem will send the following message structure for the registered unsolicited messages:

| Bytes | Data Description | Comments |
|-------|------------------|---|
| 0 | 0x00 | Parameter Number (unsolicited message registration request) |
| 1 | 0x00 | |
| 2 | 0x02 | Write Request |
| 3 | 0x00 | Response status |
| 4 – n | ... | Unsolicited message |

Table 12 - Message structure of an incoming unsolicited message

4.4.4.2 TCP API

The AT command \$TCPURC is used to enable or disable the sending of URCs over TCP API.

AT\$TCPURC= will enable URCs to be sent to the TCPAPI server

AT\$TCPURC= will disable URCs to be sent to the TCPAPI server

The default is the sending of URCs is disabled.

The format of the URCs sent over TCP is shown in Table 12a.

| Bytes | Data Description | Comments |
|-------|---------------------------|----------------------------|
| 0-1 | 16-bit data length | |
| 2 | 0x00 | 16-bit parameter number |
| 3 | 0x00 | |
| 4 | 0x02 | Status message |
| 5 | Optional header length | See 4.4.7 |
| 6 | ... | Unsolicited message |

Table 12a - Message structure of URC over TCP

4.4.5 ACK Messages

To disable sending of a message that requires acknowledgement, the server should send the following data, indicating an ACK, to stop sending of the messages.

NOTE: Acknowledge message should only be sent for messages configured to be sent via UDP/IP

| Bytes | Data Description | Comments |
|-------|------------------|------------------|
| 0 | 0x00 | Parameter Number |
| 1 | 0x0A | |
| 2 | 0x01 | Write Request |
| 3 | 0x00 | Response status |

Table 13 - ACK message

4.4.6 ERROR Messages

If there is an error in processing an API request by the modem, the modem will respond with the following message structure

| Bytes | Data Description | Comments |
|-------|------------------|--|
| 0 | xx | First two bytes of an incoming request will be echoed here |
| 1 | xx | |
| 2 | 0x03 | ERROR Response |
| 3 | 0x00 | Response status |

Table 14 - Error message

4.4.7 API Optional Header

The API Optional Header can be appended to the end of the UDPAPI or TCPAPI Header. It is comprised of a sequence of Optional Header Fields as shown below.

| Optional Header Field | Bytes | Bits 0-7 |
|-----------------------|------------------------|----------|
| 1 | 0 | Size (m) |
| | 1 | Type |
| | 2 thru (m-1) | Data |
| 2 | m | Size (n) |
| | (m+1) | Type |
| | (m+2) thru (m+n-1) | Data |
| 3 | (m+n) | Size (p) |
| | (m+n+1) | Type |
| | (m+n+1) thru (m+n+p-1) | Data |
| ... | (m+n+p) | ... |

Table 15 - API Optional Header Format

Each Optional Header Field has the following generic format:

- Byte 0: 8 bit field indicating the size of the Optional Header Field including Size and Type Fields
- Byte 1: 8 bit field indicating the type of data contained in Data Field. Currently defined types are shown in the table below.
- Bytes 2+: Optional Header Data Field. The content is defined by the Optional Header Type

Currently there are four defined Optional Header Types. The Complete list of defined types is as follows:

| Optional Header Type | Definition | Reference |
|----------------------|------------------------------|-----------|
| 0 | End of Options Sequence. | 4.4.7.1 |
| 1 | MDMID. | 0 |
| 2 | Output Message Event Format. | 0 |
| 3 | Event Sequence Number. | 4.4.7.4 |
| 4 | Firmware Version | 4.4.7.5 |
| 5 | Cause Code | 4.4.7.6 |

Table 16 - Optional Header Type

The inclusion of the Optional Headers is selected by the AT\$APIOPT and is only sent for applicable API message types. The following table defines when the Optional Headers are applicable based on the API Number and Command Types. For complete syntax of the AT\$APIOPT Command, see *GSM0308AT001 - Enabler III AT Command Set*.

| API Message | | Optional Fields | | |
|--|-----------------------------------|--------------------------------|-----------------------|-----------------------------|
| UDP-API Number | Command Type | MDMID | Event Sequence Number | Output Message Event Format |
| 0 – 4 Reserved | 0 (Read Request) | N/A | N/A | N/A |
| 5 – GPS Binary Read | | N/A | N/A | N/A |
| 6– 65535 Reserved | | N/A | N/A | N/A |
| 0 – Unsolicited Msg Request | 1 (Write Request) | N/A | N/A | N/A |
| 1 – 9 Reserved | | N/A | N/A | N/A |
| 10 – ACK | | N/A | N/A | N/A |
| 11 – Password | | N/A | N/A | N/A |
| 12 – 65535 Reserved | | N/A | N/A | N/A |
| 0 – Unsolicited Msg | 2 (General Status Information) | TCP/UDP | N/A | N/A |
| 1 – 3 Reserved | | TCP/UDP | N/A | N/A |
| 4 – ASCII Event Data (Param2>=256) | | TCP/UDP | TCP/UDP | TCP/UDP |
| 5 – Binary Event Data | | TCP/UDP | TCP/UDP | TCP/UDP |
| 6 – Reserved | | TCP/UDP | N/A | N/A |
| 7 – ASCII TAIP Data | | TCP/UDP | N/A | N/A |
| 8 – \$MSGSDN Data | | TCP/UDP | N/A | N/A |
| 9 – Reserved | | TCP/UDP | N/A | N/A |
| 10 – ASCII Event Data (Param2<256) | | TCP/UDP | TCP/UDP | TCP/UDP |
| 12 – Information Elements | | TCP/UDP | N/A | N/A |
| Echo first 2 bytes of an incoming data | 3 (Error) | TCP/UDP | N/A | N/A |
| 0 - 65535 | 4 (AT Command Request) | N/A | N/A | N/A |
| Echo first 2 bytes of an incoming AT command request | 5 (AT Command Response) | TCP/UDP | N/A | N/A |
| First 2 bytes of AT\$UDPMSG command in ASCII format | 6 | TCP/UDP (Modem to OTA only) | N/A | N/A |
| First 2 bytes of AT\$UDPMSG command in Binary format | 7 | TCP/UDP (Modem to OTA only) | N/A | N/A |
| 10 | 8 | N/A | N/A | N/A |

Table 17 - Conditions for Including Optional Fields into API Optional Header

A detailed description for each of the Optional Field types is discussed in the following paragraphs.

4.4.7.1 End of Option Sequence – Type 0

This API Optional Header Field Type indicates it is the last of the Optional Header Fields. It is only used if the API Optional Header Size is defined to be larger than the combined Option Header sizes. The size field for this type shall include the remaining bytes of the API Optional Header. The Data Field shall contain all zeros. The primary purpose of this type is to allow padding to re-align the Data contents of the API Message to a word boundary. To accommodate single byte padding of the API Optional Header a size field of one (1) shall be allowed as the last byte of the API Optional Header. If one (1) is seen in the size field it shall be interpreted as the last byte of the API Optional Header with the Optional Header Field Type assumed as End of Option Sequence. The End of Option Sequence formats are shown below:

| Description | Size | Type | Data | Data |
|-----------------|----------|----------|----------|------------|
| | (Byte 0) | (Byte 1) | (Byte 2) | (Byte 3-n) |
| Single Byte Pad | 0x01 | - | - | - |
| Two Byte Pad | 0x02 | 0x00 | - | - |
| Three Byte Pad | 0x03 | 0x00 | 0x00 | - |
| Multi-Byte Fill | (n) | 0x00 | 0x00 | 0x00(s) |

Table - 18. End of Optional Header Sequence Formats

4.4.7.2 MDMID – Type 1

The MDMID Optional Header Field allows servers to easily identify the source of the packet for easier handling. The Optional Header Data Field would include the value set by the AT\$MDMID command. No additional padding will be added to the data. An example for a MDMID of '12345678901234567890' is as follows:

| Bytes | Data Description | Comments |
|-------|------------------|---------------------------------|
| 0 | 0x16 (22) | Size |
| 1 | 0x01 | Type = MDMID |
| 2 | 0x31 | MDMID (12345678901234567890) |
| 3 | 0x32 | |
| 4 | 0x33 | |
| 5 | 0x34 | |
| 6 | 0x35 | |
| 7 | 0x36 | |
| 8 | 0x37 | |
| 9 | 0x38 | |
| 10 | 0x39 | |
| 11 | 0x30 | |
| 12 | 0x31 | |
| 13 | 0x32 | |
| 14 | 0x33 | |
| 15 | 0x34 | |
| 16 | 0x35 | |
| 17 | 0x36 | |
| 18 | 0x37 | |
| 19 | 0x38 | |
| 20 | 0x39 | |
| 21 | 0x30 | |

Table 19 - MDMID Optional Header Field Format

4.4.7.3 Output Message Event Format- Type 2

The Output Message Event Format-Optional Header Field is available to messages generated by the event engine. It contains the 32-bit <param2> value of the Output Message Event Format. The inclusion of this option field is selected by the AT\$APIOPT Command, see **GSM0308AT001 - Enabler III AT Command Set**. The format for this optional header field is as follows:

| Bytes | Data Description | Comments |
|-------|------------------|--|
| 0 | 0x06 | Size |
| 1 | 0x02 | Type = Output Message Event Format |
| 2 | 0x12 | Output Message Event Format (0x12345678) |
| 3 | 0x34 | |
| 4 | 0x56 | |
| 5 | 0x78 | |

Table 20 - Output Message Event Format Header Field Format

4.4.7.4 Event Sequence Number – Type 3

The Event Sequence Number-Optional Header Field is available to messages generated by the event engine. The optional data field would include the value of the Event Sequence Number. The Event Sequence Number increments with each Output Event. The Event Sequence Number is not reset by power off or the AT&F command. The inclusion of this option field is selected by the AT\$APIOPT Command. When the Event Sequence Number reaches its maximum value it will roll over to zero and restart counting. The size of the Event Sequence Number can be set to 8, 16, 24, 32 bits by the AT\$APIOPT Command, see **GSM0308AT001 - Enabler III AT Command Set**. The Event Sequence Number Optional Header Field formats are shown below for all available sizes:

| Description | Size | Type | Data | Data | Data | Data |
|---------------|----------|----------|----------|----------|----------|----------|
| | (Byte 0) | (Byte 1) | (Byte 2) | (Byte 3) | (Byte 4) | (Byte 5) |
| 8-bit Format | 0x03 | 0x03 | 0x78 | - | - | - |
| 16-bit Format | 0x04 | 0x03 | 0x56 | 0x78 | - | - |
| 24-bit Format | 0x05 | 0x03 | 0x34 | 0x56 | 0x78 | - |
| 32-bit Format | 0x06 | 0x03 | 0x12 | 0x34 | 0x56 | 0x78 |

Table 21 - Example of Event Sequence Number Header - Seq Num = 0x12345678

4.4.7.5 Firmware Version - Type 4

The Firmware Version Optional Header Element is used to specify the Enfora Version Number of the current firmware.

| Bytes | Data Description | Comments |
|-------|------------------|-------------------|
| 0 | 0x06 | Size |
| 1 | 0x04 | Type = FW Version |
| 2 | | Data |
| 3 | | |
| 4 | | |
| 5 | | |

4.4.7.6 Cause Code – Type 5

The Cause Code Optional Header Element is used to specify a status code for an operation.

| Bytes | Data Description | Comments |
|-------|------------------|-------------------|
| 0 | 0x04 | Size |
| 1 | 0x05 | Type = Cause Code |
| 2 | | Cause Code Value |
| 3 | | |

Table 22: Cause Code Optional Header Element

| Code | Description |
|------|--------------------------------------|
| 0 | Unknown |
| 1 | Restart due to system initialization |
| 2 | Restart due to FOTA completion |

Table 23: Cause Code Value Definitions

4.4.8 FOTA Complete Message

The FOTA Complete Message is composed of the elements:

- Modem ID Header Element
- Cause Code Header Element
- Firmware Version Header Element

| | | |
|---------------------|---------------------|--|
| ModemID Size | ModemID Type (1) | Modem ID Data (Variable Length) |
| Cause Code Size (4) | Cause Code Type (4) | Cause Code Data (16 bits) = 2 for FOTA Restart |
| FW Version Size (6) | Firmware Type (5) | Firmware Version Data (32 bits) |

Table 24: FOTA Complete Message

The FOTA process will send a FOTA Complete Message to the first destination in the FRIENDS list upon completion of the FOTA operation. The FOTA operation is normally completed on the next Modem startup following the new Firmware load. This message is composed the Cause Code indicating that a FOTA operation caused the system to restart and the new Firmware Version.

5 Terms Explained

5.1 Big Endian

Big endian format means that the most significant byte is sent first. For example, a decimal value of 1234567 will be displayed as hex 0x0012D687. While sending this data over a communication link, the most significant byte – 0x00 is sent first followed by 0x12, followed by the third byte 0xD6, followed by the least significant byte 0x87.

1234567 (decimal) = 0x0012D687 (hex)

| Byte-0 | Byte-1 | Byte-2 | Byte-3 |
|--------|--------|--------|--------|
| 0x00 | 0x12 | 0xD6 | 0x87 |

5.2 Byte

In this document, One Byte = 8 bits. Bit-0 is the right most bit and is also referred to as pin-1 while Bit-7 is the left most bit and is referred to as Pin-8.

| Byte | | | | | | | |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| Upper Nibble | | | | Lower Nibble | | | |
| Bit-7 | Bit-6 | Bit-5 | Bit-4 | Bit-3 | Bit-2 | Bit-1 | Bit-0 |
| Pin-8 | Pin-7 | Pin-6 | Pin-5 | Pin-4 | Pin-3 | Pin-2 | Pin-1 |
| 2 ⁷ | 2 ⁶ | 2 ⁵ | 2 ⁴ | 2 ³ | 2 ² | 2 ¹ | 2 ⁰ |

5.3 Little Endian

Little endian format means that the least significant byte is sent first. For example, a decimal value of 1234567 will be displayed as hex 0x0012D687. While sending this data over a communication link, the least significant byte – 0x87 is sent first followed by 0xD6, followed by the third byte 0x12, followed by the most significant byte 0x00.

1234567 (decimal) = 0x0012D687 (hex)

| Byte-0 | Byte-1 | Byte-2 | Byte-3 |
|--------|--------|--------|--------|
| 0x87 | 0xD6 | 0x12 | 0x00 |

6 Example:

Sending AT Command:

Note: IP and UDP checksum are not calculated in this example. They are left for the user to calculate as an exercise.

| Bytes | Description | Hex Value | Notes |
|-------------|----------------------------------|--------------------|--|
| 0 | Version Length: | 45 | |
| 1 | Type of Service: | 00 | |
| 2 & 3 | Length of Packet: | 00 23 | IP + UDP Header + Message Data |
| 4 & 5 | Identification: | | 00 00 |
| 6 & 7 | Fragmentation Offset: | 00 00 | |
| 8 | Time to live: | 00 | |
| 9 | Protocol: | 11 | UDP |
| 10 & 11 | IP Header Checksum: | 00 00 | |
| 12,13,14,15 | Source IP: | A6 85 AB 13 | 166.133.171.19 |
| 16,17,18,19 | Destination IP: | C7 F5 B4 0D | 199.245.180.13 |
| 20 & 21 | Source Port: | 04 4C | 1100 |
| 22 & 23 | Destination Port: | 06 B8 | 1720 |
| 24 & 25 | Length of Packet: | 00 04 | |
| 26 & 27 | UDP Checksum: | 00 00 | |
| 28 & 29 | UDP API Command: | 00 01 | UDP API 00 01 |
| 30 | UDP API Read: | 04 | 'Sequence Number AT Command Request = 04 |
| 31 | UDP API Optional: Header Size | No Optional Header | |
| 32,33,34 | AT Command | 41 54 49 | AT Command = ATI |

Response from modem to host:

| Bytes | Description | Hex Value | Notes |
|-------------|----------------------------------|--------------------|---|
| 0 | Version Length: | 45 | |
| 1 | Type of Service: | 00 | |
| 2 & 3 | Length of Packet: | 00 30 | IP + UDP Header + Message Data |
| 4 & 5 | Identification: | 00 00 | |
| 6 & 7 | Fragmentation Offset: | 00 00 | |
| 8 | Time to live: | 00 | |
| 9 | Protocol: | 11 | UDP |
| 10 & 11 | IP Header Checksum: | 00 00 | |
| 12,13,14,15 | Source IP: | A6 85 AB 13 | 166.133.171.19 |
| 16,17,18,19 | Destination IP: | C7 F5 B4 0D | 199.245.180.13 |
| 20 & 21 | Destination Port: | 06 B8 | 1720 |
| 22 & 23 | Source Port: | 04 4C | 1100 |
| 24 & 25 | Length of Packet: | 00 04 | |
| 26 & 27 | UDP Checksum: | 00 00 | |
| 28 & 29 | UPD API Command: | 00 01 | UDP API 00 01 |
| 30 | UDP API Status: | 05 | 'Sequence Number AT Command Response = 05 |
| 31 | UDP API Optional: Header Size | No Optional Header | |
| 32,33,34,35 | AT Command Response | 0D 0A 45 6E | Enfora, Inc. |
| 36,37,38,39 | Response continued | 66 6F 72 61 | |
| 40,41,42,43 | Response continued | 2C 20 49 6E | |
| 44,45,46,47 | Response continued | 63 2E 0D 0A | |

Receiving UDP ASCII Event Data with Option Header MDMID and Sequence Number Enabled

| Bytes | Description | Hex Value | Notes |
|-------------|-----------------------|-------------|-----------------------------------|
| 0 | Version Length: | 45 | |
| 1 | Type of Service: | 00 | |
| 2 & 3 | Length of Packet: | 00 32 | IP + UDP Header + Message Data |
| 4 & 5 | Identification: | 00 00 | |
| 6 & 7 | Fragmentation Offset: | 00 00 | |
| 8 | Time to live: | 00 | |
| 9 | Protocol: | 11 | UDP |
| 10 & 11 | IP Header Checksum: | 00 00 | |
| 12,13,14,15 | Source IP: | A6 85 AB 13 | 166.133.171.19 |
| 16,17,18,19 | Destination IP: | C7 F5 B4 0D | 199.245.180.13 |
| 20 & 21 | Destination Port: | 06 B8 | 1720 |
| 22 & 23 | Source Port: | 04 4C | 1100 |
| 24 & 25 | Length of Packet: | 00 04 | |
| 26 & 27 | UDP Checksum: | 00 00 | |

| | | | |
|------------|-----------------------|----------------|---------------------|
| 28 & 29 | UDP API Number: | 00 0A | ASCII Event Data |
| 30 | UDP API Status: | 02 | General Status Info |
| 31 | UDP API Optional: | 0B | Combined Optional |
| | Header Size | | Header Size |
| 32 | Optional Header Size: | 07 | Size of MDMID |
| | | | Optional Header |
| 33 | Optional Header Type: | 01 | Type is MDMID |
| 34 thru 38 | Optional Header Data: | 31 32 33 34 35 | MDMID - '12345' |
| 39 | Optional Header Size: | 04 | Size of Sequence |
| | | | Number Optional |
| | | | Header |
| 40 | Optional Header Type: | 03 | Type is Sequence |
| | | | Number |
| 41 & 42 | Optional Header Data: | AB CD | Event Sequence |
| | | | Number 'ABCD' |
| 43 thru 50 | Output Message Event | xx xx xx ... | |