Basic GTS Enterprise
Device Communication Server

Enfora

projects@geotelematic.com
http://www.geotelematic.com

Manual Revision HIstory			
Rev	**Date**	**Changed**	**Author**
0.1.0	2010/06/06	Initial Release	MDF
0.1.1	2011/01/28	Added "$MSGSND" parsing information.	MDF

Device Communication Server - Enfora

Contents:

1) Introduction

This manual describes how to configure and run the GTS Enterprise device communication server (DCS) for the **Enfora** hardware GPS tracking/telematic devices. The following features are supported for the **Enfora** DCS:

– Receive incoming data packets via UDP or TCP (UDP recommended).
– Support for Enfora packet "Optional Headers".
– Support for the Garmin Personal-Navigation-Device (PND) FMI interface.
– Support for sending commands to the device through the login web-interface.
– Support for the Comtrack J1708, J1939, OBDII hardware interface.
– Estimated GPS-based Odometer.
– Simulated Geozone Arrival/Departure.

The following **Enfora** hardware devices have been tested with the **Enfora** Device Communication Server:

– MT-Gu
– MT-uL
– MT-G
– MT-GL
– SA-G+
– Spider AT
– Mini-MT

Enfora provides ample documentation on all of their devices and configuration options. You can find additional information regarding **Enfora** device configuration at the following links:

– http://www.enfora.com
– http://www.enfora.com/index.cgi?CONTENT_ID=11&User:LANGUAGE=en

2) Configuring the Server

The following section refers to the runtime configuration file for the **Enfora** device communication server, which can be found in the GTS Installation directory at "`dcservers/dcserver_enfora.xml`" (if the Garmin FMI interface is in use, the runtime **Enfora** DCS configuration file may be "`dcservers/dcserver_enforaGarmin.xml`").

2.1) Changing the Server "Listen" Ports.

The ports on which the **Enfora** DCS listens for incoming data packets is specified on the "`ListenPorts`" tag:

```
<ListenPorts
    tcpPort="37400"
    udpPort="37400"
    />
```

If required, the "listen" port can be changed to fit the requirements of your runtime environment. The script programmed into the **Enfora** device will also need to be configured to transmit data to the same port as the server used to listen for incoming data packets.

The "listen" ports must be open through the firewall in order for the remote device to send data to the **Enfora** server.

If packet acknowledgment is required, any acknowledgments sent by the server back to the remote device must be sent from the same IP address to which the remote device sent it's data packet. If your server responds to more than one IP address, then the **Enfora** server listener must be bound to the same IP address/interface used by the remote tracking devices. This is set in the top-level "`dcservers.xml`" file, on the "`DCServerConfig`" tag, "`bindAddress`" attribute.

2.2) Setting the "Unique-ID" Prefix Characters.

The Unique-ID prefix characters can be set in the "`UniqueIDPrefix`" tag section:

```
<UniqueIDPrefix><![CDATA[
    en_
    enfora_
    imei_
    *
]]></UniqueIDPrefix>
```

These prefix characters are used to 'prepend' to the IMEI number reported by the device to look up the owning Account/Device record for this device. For instance, if the IMEI number is "`123456789012345`", then the system will search for the owning Device using the following Unique-ID keys, in the order specified:

```
en_123456789012345
enfora_123456789012345
imei_123456789012345
123456789012345
```

Note that the '`*`' character by itself indicates that the system should look up the IMEI number without any prefixing characters.

To bind an **Enfora** device to a specified Account/Device record, set the "Unique ID:" field on the Device Admin page to the appropriate prefixed unique-id value. For example:

```
Unique ID:  en_123456789012345
```

2.3) Setting the Enfora Properties

Properties which effect the behavior of the **Enfora** server are set in the "`Properties`" tag section. The following properties may be set:

```
<Property key="payloadMask">9043967</Property>
```
This property defines the default data-mask configured in to the **Enfora** device which indicates which of the field values will be contained in the data packet. This value may be specified in decimal, or hexidecimal notation. Here is a list of the most common ("Table 0" - non-Garmin) "Binary" fields (Bit #0 = "1") that are available for events:

Bit	Description	Mask	Include	Notes
00	ASCII/Binary indicator	0x00000001	yes	Must be set for Binary
01	Include UserNumber	0x00000002	yes	Status Code
02	Include Modem ID	0x00000004	yes	IMEI #
03	Include GPIO	0x00000008	yes	Digital Inputs
04	Include Analog #1	0x00000010	yes	
05	Include Analog #2	0x00000020	yes	
06	Store Last GPS	0x00000040	yes	
07	Include Input Event #	0x00000080	yes	
08	Include Date	0x00000100	yes	GPS Date
09	Include Status	0x00000200	yes	
10	Include Latitude	0x00000400	yes	
11	Include Longitude	0x00000800	yes	
12	Include Speed	0x00001000	yes	
13	Include Heading	0x00002000	yes	
14	Include Time	0x00004000	yes	GPS Time
15	Include Altitude	0x00008000	yes	
16	Include # Satellites	0x00010000	yes	
17	No OTA during low power	0x00020000	no	Not Used
18	Send via SMS	0x00040000	no	Not Used
19	Use Last Valid GPS Fix	0x00080000	yes	
20	Include GPS based Odometer	0x00100000	optional	
21	Include Real-Time Clock	0x00200000	optional	Clock Date/Time
22	Use Short Modem-ID	0x00400000	no	Not Used
23	Include Battery Level	0x00800000	optional	
24	Overspeed	0x01000000	optional	
25	PCell Information	0x02000000	optional	
26	Alternate Overspeed	0x04000000	optional	MT3000
27	OBDII Info	0x08000000	optional	MT3000 (VIN, FW, RSSI)
28	OBDII MIL data	0x10000000	optional	MT3000
29	OBDII Odometer	0x20000000	optional	MT3000

The example decimal value "9043967" is "`0x89FFFF`" when specified in hexidecimal. The payload mask defined here must also be used within the **Enfora** script to define data fields included for events sent to the server. Alternatively, the **Enfora** script may include the command "`AT$APIOPT=1,1,0`" to specify that the **Enfora** packet should also include the actual payload mask in the data packet sent to the server (verify that the version of **Enfora** device you are using supports the "`AT$APIOPT`" command).

```
<Property key="minimumSpeedKPH">3.0</Property>
```
This is the minimum acceptable speed value, below which the device will considered not moving, and the speed will be explicityly set to "`0.0`".

```
<Property key="statusLocationInMotion">true</Property>
```
If "`true`", the DCS will replace an event which otherwise is defined to be a general STATUS_LOCATION status code instead with a STATUS_MOTION_IN_MOTION status code, if the indicated speed of the vehicle is greater

than zero.

```
<Property key="simulateDigitalInputs">0x7F</Property>
```

If specified, this mask value indicates which of the **Enfora** digital inputs should be checked for state changes, and if a state change is detected, an event with the appropriate digital input state change status code will be generated. The special value "false" is the same as entering a mask value of "0", which indicates that no digital input state changes should be detected. Note that the **Enfora** numbers their GPIO from left to right, thus the mask 0x80 indicates GPIO #0, and 0x01 indicates GPIO #7.

```
<Property key="estimateOdometer">true</Property>
```
If "`true`", the DCS will calculate the current event odometer based on the distance traveled since the last valid GPS location.

```
<Property key="simulateGeozones">true</Property>
```
If "`true`", the DCS will check for geozone arrivals/departures and insert the appropriate arrive/depart events.

```
<Property key="useRtcTimestamp">false</Property>
```
If "`true`", the DCS will use the RTC time provided in the data packet as the timestamp of the event.

```
<Property key="saveRawDataPackets">false</Property>
```
If "`true`", the DCS will save the packet data (in hex format) in the EventData record itself. This packet information will be stored in the EventData "`rawData`" column. (typically only used for additional auditing of events).

```
<Property key="printPacketsToStdout">false</Property>
```
If "`true`", the DCS will print the raw packet information (in hex format) to stdout. This packet information will appear in the log file "`logs/enfora.out`". (typically only used for additional auditing of events).

```
<Property key="MSGSNDParserClass"></Property>
```
This property can specify a custom $MSGSND packet handler which be called to parse custom data that may be available in general $MSGSND data packets sent by the remote device. An example custom $MSGSND data parser class can be found at "`src/org/opengts/custom/gts/enfora/CustomParser_MSGSND.java`" (class name "`org.opengts.custom.gts.enfora.CustomParser_MSGSND`"). See the Appendix C below for more information.

2.4) Changing the Default UserNumber to StatusCode Mapping.

The UserNumber to StatusCode mapping is specified in the "`EventCodeMap`" and "`Code`" tag sections:

```
<EventCodeMap enabled="true">
    <Code key="10">default</Code>
    <Code key="13">ignore</Code>
    <Code key="14">0xF420</Code>
</EventCodeMap>
```

This section is used to map a defined **Enfora** UserNumber to a GTS Enterprise status-code. It can also be used to ignore an event for a specific UserNumber value.

The value that should be specified on the Code "`key`" attribute is the defined UserNumber which is sent by the **Enfora** device on the data packet.   This value may be specified in either decimal, or hexidecimal notation.

The text value of the "`Code`" tag should be the status-code to which the UserNumber should be mapped. The special value "`ignore`" can be used to cause events which specify this UserNumber to be ignored. The special value "`default`" indicates that the status code on the generated event will be **STATUS_LOCATION** if the vehicle is not moving, and **STATUS_MOTION_IN_MOTION** if the vehicle is moving. The numeric values, specified as either decimal or hexidecimal will be used as the status code on the generated event.

If an event arrives with a UserNumber which is not specified in the "`EventCodeMap`" tag section, then it will be used unchanged as the status-code for the generated event.

The "`EventCodeMap`" attribute "`enabled`" can be set to "`false`" to disable UserNumber to StatusCode mapping. In which case **Enfora** defined UserNumbers will be used unchanged as the status-code for the generated event.

Refer to the "Status Codes and Description" documentation for a list of currently defined status codes.

2.5) Setting the Available Server-To-Device Commands.

Commands which are to be sent to the **Enfora** device based on user requests are defined in the "Commands" tag section. The following are some sample command definitions:

```xml
<Commands dispatchPort="31400">

    <AclName>acl.dcs.enfora</AclName>

    <!-- Request current location from the device (via UDP) -->
    <Command name="LocateNow" enabled="true">
        <Type>map,admin</Type>
        <Description>Locate Now</Description>
        <String protocol="udp"><![CDATA[AT$MDMID?;$GPSRD=10]]></String>
        <StatusCode></StatusCode>
    </Command>

    <!-- Request current location from the device (via SMS) -->
    <Command name="LocateSMS" enabled="true">
        <Type>map,admin</Type>
        <Description>Locate Now</Description>
        <String protocol="sms:body"><![CDATA[AT$MDMID?;$GPSRD=10]]></String>
        <StatusCode></StatusCode>
    </Command>

    <!-- Set Output #1 On -->
    <Command name="Output1_on" enabled="true">
        <Type>admin</Type>
        <Description>Set Ouput-1 ON</Description>
        <String protocol="sms:body"><![CDATA[AT$IOGP1=1]]></String>
        <StatusCode></StatusCode>
    </Command>

    <!-- Set Output #1 Off -->
    <Command name="Output1_off" enabled="true">
        <Type>admin</Type>
        <Description>Set Ouput-1 OFF</Description>
        <String protocol="sms:body"><![CDATA[AT$IOGP1=0]]></String>
        <StatusCode></StatusCode>
    </Command>

    <!-- Send a user entered "AT" command string to the device -->
    <Command name="ATCommandSMS" enabled="true">
        <Type>admin</Type>
        <Description>Send AT Command (SMS)</Description>
        <String protocol="sms:body"><![CDATA[${arg}]]></String>
        <StatusCode></StatusCode>
    </Command>

    ...

</Commands>
```

◆ "`Commands`" attribute "`dispatchPort`":
This is used to define the local port that the web-interface will use to connect to the **Enfora** device communication server to indicate that a given command should be sent.   The web-interface sends a command request to the **Enfora** DCS, which then forwards the command to the **Enfora** device based on the specified transport media (ie. UDP or SMS(email)). This port should **not** be made accessible through the firewall.

◆ "`AclName`" Tag:
This is used to specify the ACL name used to allow/deny access to this device command feature for specific users. This ACL name should also be specified in the "`private.xml`" or "`private_common.xml`" file. To control specific commands, the ACL key defined in the "`private.xml`" or "`private_common.xml`" file, shold be this ACL name, followed by a colon (":"), followed by the specific command name.

◆ "`Commands`" Tag:
This section includes zero or more "`Command`" sub-tags which define the specific available commands.

◆ "`Command`" Tag:
This tag defines a specific command which is used to send instructions to the remote device. The "`name`" attribute specifies the command name. The "`enabled`" attribute specifies whether this command is enabled or disabled. Disabled commands will not be displayed in the web-interface.

◆ "`Type`" Tag:
This specifies where this command should be made available for user selection. Valid values may be spearated by commas, and may be "`map`" to specify that the command should be available on the Device Track Map page in a command pull-down selection menu, or "`admin`" to indicate that the command should be available on the "`Properties`" page which can be displayed from the Device Admin list page.

◆ "`Description`" Tag:
Theis specifies the Description/Title that will be used to describe the command in the pull-down selection menu, or when displayed on the Device Admin Properties page.

◆ "`String`" Tag:
This specifies the actual command string which is to be sent to the remote device.

The "`protocol`" attribute specified the transport media used to send the command to the device. Acceptable values are "`udp`" or "`sms:type`". The "`:type`" indication on the "`sms`" protocol is a directive indicating which type of outbound gateway to use (current valid values for "`:type`" are "`:body`" and "`:subject`", which indicate that an email should be sent to the sms email address and the command should be placed in either the body of the email, or the subject line. If the "`:type`" directive is not specified, "`:body`" is assumed).

The value of this tag is the actual command string which will be sent to the remote device. This command must begin with the "`AT`" prefix. If the command value contains the value string "`${arg}`", this indicates that input should be received from the user, which should replace this variable string indicator.

◆ "`StatusCode`" Tag:
This specifies the status-code of an audit event that should be generated when this command is sent to the device at the request of a user. If no status-code is specified, then no event will be generated.

3) Running the Server

To begin listening for incoming events the server must be started. This section describes the process for manually starting the **Enfora** server, and how to set up the system to have is automatically start the **Enfora** server on system reboot.

3.1) Manually Starting the Server

The command for manually starting the **Enfora** server is as follows:

```
>   cd $GTS_HOME
>   bin/runserver.pl -s enfora
```

To start the **Enfora** server with debug logging (useful when testing or debugging), the option "-debug" may be added to the command line.

The server will start, and logging information will be sent to the file "$GTS_HOME/logs/enfora.log".

For debug purposes, it is sometimes useful to have the logging output sent directly to the console, instead of the log file. In this case the option "-i" can also be added to the command-line.   When logging to the console, hit control-C to stop the server.

To view the running server, you can use the "bin/psjava" command:

```
>   $GTS_HOME/bin/psjava

  PID   Parent  L User      Java class/jar
 ------ ------  - --------  -------------------------------------------------
 54639(     1) 1 opengts   org.apache.catalina.startup.Bootstrap
 68936(     1) 1 opengts   /usr/local/GTS_2.2.4-B23/build/lib/enfora.jar
```

To stop the running **Enfora** server, enter the following command:

```
>   cd $GTS_HOME
>   bin/runserver.pl -s enfora -kill
```

This will stop the running **Enfora** server.

3.2) Automatically Starting the Server on System Reboot

The auto-start script for Fedora is located at "$GTS_HOME/bin/onboot/fedora/opengts", and should have been installed into the system directory "/etc/init.d" when the GTS was installed.

This startup script uses the configuration specified in the file "$GTS_HOME/bin/serverList" to determine which device communication servers to start up when the system is rebooted. The line in this file that refers to the **Enfora** DCS should appear similar to the following:

```
    execServer  "Enfora"  "enfora"  "${option}"  ""
```

The first quoted string contains the DCS description. The second quoted string contains the ID of the device communication server and must match a library jar file with the same name at "$GTS_HOME/build/lib/enfora.jar". The third quoted string must contain the exact value "${option}", which is used within the startup script to pass command-line arguments to the DCS startup code. The forth quoted string is used to pass other optional default or constant arguments to the DCS startup code.

3.3) Monitoring the Log Files

When started, the **Enfora** DCS will create the following output log files:

`$GTS_HOME/logs/enfora.pid`
This file contains the process-id (PID) of the **Enfora** DCS execution process. This PID is used by the "`-kill`" option to terminate the running **Enfora** DCS.

`$GTS_HOME/logs/enfora.out`
This file is used to output the raw packet data from the various remote **Enfora** tracking devices, in the format:
` Packet: now=<EpochTime> id=<IMEI#> time=<EpochTime> data=<Hexidecimal_Packet_Data>`
This information is typically only used for debug purposes.

`$GTS_HOME/logs/enfora.log`
This log file is where all other logging information is placed regarding the receipt and parsing of data from the remote **Enfora** tracking devices. As this file grows, it will be "rotated" into other log files timestamped as follows:
` enfora.log.YYYYMMDDHHMMSS.log`
Where "*YYYYMMDDHHMMSS*" represents the Year/Month/Day/Hour/Minutes/Seconds time that the file was trimmed and rotated.

The "`enfora.log`" file will reflect any current connection attempts from remote **Enfora** tracking devices. As devices send their data to the server, the receipt of the incoming data packets, along with parsing results, will be displayed in this log file.

4) Adding Devices to an Account

When data is received from a remote **Enfora** tracking device, the **Enfora** server looks up the IMEI number in the Device table to determine which Account/Device owns this device. This section describes how to create a Device record and associate an **Enfora** tracking device with the Device record.

4.1) Creating a New Device Record

Using the web-interface, log in to the appropriate Account which should own the **Enfora** tracking device, then traverse to the "Device Admin" page (or "Vehicle Admin", etc, if so named). Create a new Device as indicated in the GTS Enterprise Tutorial documentation, then "Edit" the newly created Device record.

On the Edit page, there will be a field described as follows:

    Unique ID: [   ]

In this field enter the value "**en_***<IMEI_Number>*", replacing "*<IMEI_Number>*" with the device IMEI number. For instance, if the IMEI number is "123456789012345", then enter the value "en_123456789012345" in this "Unique ID:" field.

After making changes to the Device record, click the "Change" button.

4.2) The "Server ID" Field

The "Server ID" field displayed as a column title on the Device list page, and as a read-only field on the Device Edit page, is assigned a value when the **Enfora** device sends its first data packet to the server. Until then, this value will remain blank.

When viewing a list of created Device records with assigned **Enfora** devices, records which still have blank "Server ID" fields indicate that no incoming data packet has been received for this particular Device.

5) Testing a New Configured Device

This section describes the process for monitoring newly configured **Enfora** devices that have been assigned to an Account/Device record.

5.1) Monitoring for Incoming Connections

The Account report "Last Known Device Location" can be used to display the last know location of a given device, which can also be used to determine whether any events have been received from a specific **Enfora** device.

The "Server ID" field on the Device record will also indicate if a data packet has arrived from a specific **Enfora** device and successfully assigned to the Device record.

If no indication on the Device reports, or "Server ID" field is evident, then the log file itself can be consulted for indications of incoming data packets from the **Enfora** device. The information in the log file can indicate whether an IMEI number may not have been properly assigned, so the **Enfora** DCS is unable to determine which Account/Device the incoming data packet belongs to.

5.2) Viewing the Unassigned Device Report

In the case where an **Enfora** device is put into service without having been assigned to an Account/Device record, or where the IMEI number was incorrectly entered in to the Device record, the **Enfora** DCS may not know to which Account/Device the incoming data packet belongs.

When the **Enfora** DCS cannot determine the ownership of an incoming data packet, it will place the IMEI and currently GPS location into the "UnassignedDevices" table. The "Unassigned Devices" report can be selected from the System Administrator login panel ("System Admin" tab, "SysAdmin Reports" menu option, "Unassigned Devices" report). This report will show the "Server ID" (**Enfora**), "Unique ID" (IMEI number), and the last time data was received from this device. This information can be used to determine whether an IMEI number was ever assigned to an Account/Device record, or if an IMEI number was incorrection assigned to an Account/Device (ie. transposed digits, etc).

Appendix)

A) Troubleshooting Enfora Device Connection Issues

The following are fequently-asked-questions regarding commonly occurring connection issues.

Q: I've configured an **Enfora** device to send data to the server, but have not received any data.
A: Monitor the "`enfora.log`" file for possible incoming connections from the device. If there is no indication that the server is receiving any communication from the remote device, the most common reasons to check include:
- Make sure device has a valid/active SIM card.
- Make sure the device has been programmed with the proper APN ("Access Point Name") configuration as specified by your wireless service provider.
- Make sure the device has been programmed with the proper host and port of your server.
- Make sure the server firewall allows incoming UDP/TCP connections on the specified port.   If the server itself provides its own firewall, then check the firewall settings. On Linux, this is usually controlled by "`iptables`". The command to display the current iptables configuration is "`iptables-save`" (must be run as "`root`"). See "http://www.faqs.org/docs/iptables/iptables-save.html" for more information.

Q: I see data arriving for my device in the "enfora.log" file, but it is always the same event that is being sent over and over.
A: If this occurs for all configured/connected **Enfora** devices, the problem is likely that returned UDP acknowledgments are not being returned properly to the device. The most likely reason for this is that your computer responds to more than one IP address, and the returned UDP packets are being sent from a different IP address than the one that the device is configured to send data to. This can be fixed by setting the "`bindAddress`" attribute in the "`dcservers.xml`" file in the GTS installation directory (then restart the **Enfora** DCS). In some cases, the SIM card wireless service provider does not allow returned UDP packets to be sent from the server back to a device. In this case, it may be necessary to program the **Enfora** devices to not require a return acknowledgment.
A: If this occurs for only one device (ie. other devices are reporting as expected), this this is likely due to the GPS receiver's inability to obtain a new GPS fix, and the previous GPS fix is being resent. This usually means that the device is simply in an area where the GPS satellites cannot be seen (ie. Indoors, etc). On rare occasions, this can mean that the GPS antenna has become unplugged, or has been damaged.

Q: Data is arriving for my device (ie. events are being generated), however not all of the valid data fields are being populated, or are being populated with invalid data (ie. missing/invalid latitude/longitude, speed, heading, etc).
A: This usually means that the programmed **Enfora** script may not be including the appropriate fields in the data packet. This can also mean that the datamask specified in the **Enfora** script does not match the datamask configured in the **Enfora** DCS runtime configuration file ("`dcserver_enfora.xml`"). Check the **Enfora** script to make sure that the specified datamask matches the mask configured in the **Enfora** DCS.

Q: The received events have a valid latitude/longitude, but do not have an odometer value.
A: The **Enfora** DCS property "`estimateOdometer`" allows enabling a calculated odometer value, based on the distance traveled between successive GPS points. To enable a calculated estimated odometer value, make sure this property is set to "`true`". Alternatively, the **Enfora** can be programmed to include an odometer value in the data packet, calculated internally based on successive GPS locations.

B) Enfora Scripting Tool and Example Script

B.1) Scripting Tool

An **Enfora** scripting tool is provided in the GTS Enterprise installation directory at the location
"`$GTS_HOME/bin/enfora/Enfora.exe`". This tool can be used to program **Enfora** devices with custom
scripts.

The "`Enfora.exe`" tools must be run from a DOS command-line on a Windows PC. Copy this command to a
Windows PC using any convenient method (USD flash drive, network copy, etc), then 'cd' into the directory
containing the file 'Enfora.exe'. When the command is entered without any command-line options, the following
help information is displayed:

```
> Enfora.exe

Display program version and exit:
   Enfora.exe -version

Send configuration from file:
   Enfora.exe -com <comport> [-var <key=value>] -config <file>

Check network connectivity (Ctrl-C to exit):
   Enfora.exe -com <comport> -netip

Test I/O (Ctrl-C to exit):
   Enfora.exe -com <comport> -io

Reset device to factory settings:
   Enfora.exe -com <comport> -reset

Query device IMEI#:
   Enfora.exe -com <comport> -imei

Perform read test:
   Enfora.exe -com <comport> [-bps <bps>] -readTest
```

The value "`<comport>`" shown above should be replaced with the COM# value which represents the serial port to
which the Enfora device is attached.

Most present-day computers will require a USB to RS232 converter to provide the connection required to connect
the Enfora device to a PC. The Windows OS then assigns a COM port name to the RS232 connection. The
following command options will display a list of currently available COM ports:

```
> Enfora.exe -com list
  Checking for valid COM ports ...
    COM1
    COM3
    COM7
```

(Try running this command with and without the USB to RS323 device attached to see which COM Port name has changed).

> **B1.1) MT-Gu Programming Cable:**
> The programming cable for the MT-Gu (GSM2338) can be purchased from various locations on Internet. Here is one possible vendor:
>   – http://www.pwsstore.com/powerserialcableforusewithmt-gu.aspx
> The programming/scripting process will also require a 12 volt power supply (a wall transformer will suffice for this purpose).

> **B1.2) Mini-MT Programming Cable:**
> The programming cable for the Mini-MT (GSM2228) is a standard USB Type-A to USB 5-pin Mini-B, available at most locations that sell mobile-phones or digital cameras. A "serial driver" may also be needed to support the Mini-MT programming from your PC. This driver should be obtained from Enfora (http://www.enfora.com).

After plugging in the **Enfora** device to the PC, using the appropriate cable, and preparing a script for loading to the device, the following command will load the script onto the **Enfora** device ("COM7" is assumed to be the ComPort to which the **Enfora** device is connected in this example, and should be replaced with the actual ComPort used by your system).

```
> Enfora.exe -com COM7 -config Script.enfora
```

The tool will then attempt to communicate with the **Enfora** device through the specified ComPort, and send the specified script to the device. The tool output will indicate the progress of the programming process, and will display any connection errors, or scripting syntax errors, if they occur.

B.2) Example Script

A simple example **Enfora** script is also included at "`$GTS_HOME/bin/enfora/Script.enfora`" (please refer to the commented text within this script for additional information). When loaded, this script instructs the **Enfora** device to report a location event to the server every couple minutes.

A set of predefined replacement variable definitions are included in the script which can be used to assist in programming a specific device. At a minimum the following variables should be set in the script:

```
#! Host      = 192.168.0.1
#! Port      = 37400
#! APN.Name  = WAP.CINGULAR
#! APN.User  = WAP@CINGULARGPRS.COM
#! APN.Pass  = CINGULAR1
#! Interval  = 120
#! DataMask  = 9043967
```

Host
The "`Host`" value must match the IP address of your server. Some **Enfora** devices may allow this value to be a domain name (ie "`data.example.com`").

Port
The "`Port`" value must be set to the port used by the **Enfora** DCS to listen for incoming connections. Unless otherwise required, this port value should remain as "`37400`".

APN.Name
The "`APN.Name`" value must be set to the "Access Point Name" provided by your wireless service provider.

APN.User
The "`APN.User`" value must be set to the Access Point user name provided by your wireless service provider.

APN.Pass

The "`APN.Pass`" value must be set to the Access Point user password provided by your wireless service provider.

Interval

The "`Interval`" specifies the number of seconds between events generated by the Enfora device. The default value specified is "`120`", which indicates that the device should generate a location event every 2 minutes.

Datamask

The "`Datamask`" should be set to the message format (also called "data payload mask") used to specify which field are to be included in the data packet which the device sends to the server. This value should match the value specified in the "`dcserver_enfora.xml`" file, on the "`payloadMask`" property. This value may also be specified in hexidecimal using the following format "`${0x0089FFFF}`". This indicates to the scripting tool that the hex value specified within the curly brackets should be converted to its decimal equivalent.

The "`Script.enfora`" script is only an example of the types of programming that can be performed within the **Enfora** device. Please refer to the **Enfora** website (http://www.enfora.com) for more information on programming instructions and options.

C) Custom "$MSGSND" Data Parsing

The **Enfora** DCS provides a programming interface for parsing custom "$MSGSND" events which may be configured into the **Enfora** device.

To enable this feature, the "MSGSNDParserClass" property (in the "dcservers/dcserver_enfora.xml" file) can be set to the custom class which will handle the custom "$MSGSND" packets from the **Enfora** device. The example property specification below specifies the the example Enfora "$MSGSND" parser which is included with the Enfora DCS:

```
<Property key="MSGSNDParserClass">org.opengts.custom.gts.enfora.CustomParser_MSGSND</Property>
```

The source code to the above example custom "$MSGSND" data parser class can be found at the following directory:

```
src/org/opengts/custom/gts/enfora/CustomParser_MSGSND.java
```

This custom "$MSGSND" data parser must implement the interface "org.opengts.db.CustomParser".

The method used to parse incoming "$MSGSND" data is called "parseData(...)", and is listed below:

```
/**
*** Callback to parse raw data received from a remote tracking device
*** through its device communication server.
*** @param account  The assigned device Account instance
*** @param device   The addigned Device instance
*** @param data     The byte array containing the raw data
*** @param props    A map where parsed data should be placed
***                 (to be inserted into the EventData record)
*** @return The response which will be sent back to the device
**/
public byte[] parseData(Account account, Device device,
    byte data[], Map<String,Object> props)
{
    // Handle parsing of 'data' and fill "props" map as required
    // IE.
    //  props.put(EventData.FLD_timestamp , new Long   (timestamp ));
    //  props.put(EventData.FLD_statusCode, new Integer(statusCode));
    //  props.put(EventData.FLD_latitude  , new Double (latitude  ));
    //  props.put(EventData.FLD_longitude , new Double (longitude ));
    return null;
}
```

This payload to the "$MSGSND" data is passed to the above method in the "data" byte array. This can be parsed by this custom handler, then the various applicable EventData fields can be set according to the contents of the data. For instance, to set a "fuelLevel" percentage, based on data parsed from the payload, you can set the EventData field as follows:

```
double fuelLevel = <Parsed Fuel Level Value Here>;
props.put(EventData.FLD_fuelLevel, new Double(fuelLevel));
```

When this method returns, the EventData field values set will be applied to a new EventData record.

Note that if an EventData field is set, it must also be configured to be included in the EventData record itself. For instance, the above "fuelLevel" field is part of the "J1708FieldInfo" data, so the following property should also be included in one of the runtime ".conf" files (eg. "custom_gts.conf"):

```
startupInit.EventData.J1708FieldInfo=true
startupInit.EventData.CANBUSFieldInfo=true
```

(The "bin/dbAdmin.pl -tables=ca" command should then be executed to update the table columns.)

D) Comtrack J1708/J1939/OBDII Hardware Interface

The Comtrack Tacs (http://www.comtrack.ca) is a hardware option that connects to an **Enfora** modem to send J1708, J1939, OBDII information from the vehicle on-board computer to the server.

The Comtrack DCS is configured using a separate runtime configuration file, and runs as a separate process to listen for incoming Comtrack data packets (using a separate "listening" port).

For more information on the runtime configuration options, see the Comtrack DCS configuration file at "`dcservers/dcserver_comtrack.xml`".