FRANSON

GpsGate Server Protocol Specification

v1.0, v1.1 and v2.0

(last update January 26, 2009)

FRANSON

Contents

FRANSON

Introduction

If you plan to integrate a “tracker”, please take a look at “GpsGate TrackerOne” described in http://franson.com/gpsgateserver/GpsGateTrackerOne.pdf

This document describes communication to GpsGate Server using GpsGate Protocol. The main purpose is position updates, status messaging, track reports and configuration. Software and hardware developers are encouraged to use this protocol to integrate with GpsGate.com and GpsGate Server.

Communication can be made over TCP/IP, UDP, HTTP, SMS and XML/HTTP. Full functionality is achieved over TCP/IP.

The protocol can be used with GpsGate Server and GpsGate.com. The address to GpsGate.com is online.gpsgate.com port 30175, port 80 for HTTP and port 8008 for XML/HTTP

All sentences should always be followed by a NMEA checksum and carriage return + line feed “`\r\n`”. See Appendix 1 for an example algorithm to calculate the checksum.

GpsGate Server also supports JSON & XML/SOAP based web services which aren't covered in this document. Please read more here:
http://gpsgate.com/index.php?id=47

For help please visit our forum – http://franson.com/forum – or contact support@franson.com

Login commands

This section contains information on login commands. This includes both commands sent from client to server and possible server replies.

Login with username and password

Syntax:

`$FRLIN,domain,username,password*XX`

`domain`	Always empty. Reserved for future use.
`username`	The GpsGate.com / Server username
`Password`	Password with a very simple encryption. Algorithm is described in Appendix 2
`XX`	NMEA checksum. Algorithm to calculate shown in Appendix 1

Sample sentence:

```
$FRLIN,,user1,8IVHF*7A
```

Login with hardware IMEI number

Syntax:

`$FRLIN,domain,username,password*XX`

`domain`	should be the string "IMEI"
`username`	the device IMEI number e.g. 1234123412341324
`password`	always empty
`XX`	NMEA checksum. Algorithm to calculate shown in Appendix 1

Sample sentence:

```
$FRLIN,IMEI,1234123412341234,*7A
```

You need to create a device in the GpsGate Web interface for a specific user. This will connect a user to an IMEI number.

Successful login message

This message is returned from server on a successful login.

Syntax:

`$FRSES,sessionid*XX`

`sessionid`	Server side session id.
`XX`	NMEA checksum. Algorithm to calculate shown in Appendix 1

Sample sentence:

```
$FRSES,1221568*46
```

Error message

Error message. Will be returned from server on failed login. The server can also simply close the connection if some kind of error or time-out condition occur.

Syntax:

```
$FRERR,err_code,err_message*XX
```

err_code	Currently only value is "AuthError"
err_message	Human readable error message
XX	NMEA checksum. Algorithm to calculate shown in Appendix 1

Sample sentence:

```
$FRERR,AuthError,You have provided wrong username or password*18
```

Position data messages

User position request

Used to request position data from another user in requesting users buddy list. Position data is sent as GPRMC sentences from server.

Syntax:

```
$FRRDT,username,interval*XX
```

username	The username in the buddy list to receive position data from. Or "_buddies" to get data from all the users buddies.
Interval	in seconds. Can have decimals. 0 = receive all data unfiltered.
XX	NMEA checksum. Algorithm to calculate shown in Appendix 1

Username can be either one user or the buddy list. Depending on the value of “username” the following actions are taken:

One user: An GPRMC sentence is sent right away from server with latest known position, and the UTC time when that position was reported.

"_buddies": An FRPOS sentence is sent right away from server for each user with known position with latest known position, and the UTC time when that position was reported.

Notes:

Data is not sent back to the client faster than the interval specified.

Data is not sent if the position hasn't been updated.

If there is any error, the socket is closed.

Sample sentence:

```
$FRRDT,_buddies,10.0*7E
```

User position update request

Sent from client that wants to write position data to the server. This is the same thing as adding a "GpsGate.com (Send)" plugin to GpsGate Outputs.

Syntax:

```
$FRWDT,datatype*XX
```

datatype	Can be "ALL" or "NMEA". If ALL, then all data will be passed through unfiltered. If NMEA only $GPRMC data is accepted.
XX	NMEA checksum. Algorithm to calculate shown in Appendix 1

Sample sentence:

```
$FRWDT,NMEA*78
```

Position update - FRPOS

Received from server if positions from "_buddies" are requested.

Syntax:

```
$FRPOS,DDMM.mmmm,N,DDMM.mmmm,E,AA.a,SSS.ss,HHH.h,DDMMYY,hhmmss.dd,buddy*XX
```

01	DDMM.mmmm	Latitude (NMEA format)
02	<N|S>	Hemisphere N or S
03	DDMM.mmmm	Longitude DDMM.mmmm (NMEA format)
04	<E|W>	Hemisphere E or W
05	AA.a	Altitude in meters above sea level
06	SSS.ss	Speed over ground in knots
07	HHH.h	Heading over ground in degrees
08	DDMMYY	date
09	hhmmss.dd	Time (UTC)
10	buddy	name of buddy this position info belongs to.
11	XX	NMEA checksum. Algorithm to calculate shown in Appendix 1

Sample sentence:

```
$FRPOS,6311.64120,N,01438.02740,E,0.0,0.000,0.0,270707,154403.000,Johan*18
```

Position update - GPRMC

The position information sentence is used both to send and receive updates.It is received from server when requesting updates for a single user. It is sent to server when updating position for a single user.

Syntax:

```
$GPRMC,hhmmss.dd,A,DDMM.mmmm,N,DDMM.mmmm,E,SSS.ss,HHH.h,DDMMYY,,*0A
```

01	hhmmss.dd	UTC time hh = hours, mm = minutes, ss = seconds, dd = decimal part of seconds
02	<A|V>	Status indicator, A = valid, V = invalid
03	DDMM.mmmm	Latitude, DD = degrees, MM = minutes, mmmm = decimal part of minutes
04	<N|S>	Either character N or character S, (N = North, S = South)
05	DDMM.mmmm	Longitude, DD = degrees, MM = minutes, mmmm = decimal part of minutes
06	<E|W>	Either character E or character W, E = East, W = West
07	SSS.ss	Speed over ground, knots
08	HHH.h	Track made good, degrees true. Heading (1-360 degrees)
09	DDMMYY	Date
10	RMC_MagVarDir	Magnetic Variation, degrees
11	XX	NMEA checksum. Algorithm to calculate shown in Appendix 1

Sample sentence:

```
$GPRMC,154403.000,A,6311.64120,N,01438.02740,E,0.000,0.0,270707,,*0A
```

Sample scenarios with TCP/IP

Connect to online.gpsgate.com port 30175, or to your own GpsGate Server installation to test the examples.

Scenario: Failed login

The following scenario shows a user that tries to login using the wrong user name or password.

Client	Server
FRLIN	
	FRERR

Same scenario with real sample data:

Client	Server
$FRLIN,,user1,HHVMOLLX*5F	
	$FRERR,AuthError,You have provided wrong username or password*18

Note: If the checksum is wrong, there will be no response at all from the server.

Scenario: Receive data from from the server for one user

The following scenario shows the flow when a user connects to the server and starts receiving position updates for one user.

Client	Server
FRLIN	
	FRSES
FRRDT	
	GPRMC
	GPRMC
	...

Same scenario with real sample data

Client	Server
$FRLIN,,user1,8IVHF*7A	
	$FRSES,1221568*46
$FRRDT,johan,10.0*2B	
	$GPRMC,154403.000,A,6311.64120,N,01438.02740,E,0.000,0.0,270707,,*0A

New GPRMC sentences will be sent from server when user "johan" updates his position, but not more often than every 10 seconds as specified in the second field of FRRDT. In this example user “user1” receives data from user “johan”. You can also read data from the same user that logs in.

Scenario: Get data from all users in buddy list

Client	Server
FRLIN	
	FRSES
FRRDT	
	FRPOS
	FRPOS
	...

Same scenario with real sample data

Client	Server
$FRLIN,,user1,8IVHF*7A	
	$FRSES,1221640*4F
$FRRDT,_buddies,10.0*7E	
	$FRPOS,6311.64120,N,01438.02740,E,0.0,0.000,0.0,270707,154403.000,Johan*18
	$FRPOS,5920.71451,N,01803.28481,E,0.0,0.000,0.0,020707,222402.000,user1*50

New FRPOS sentences will be sent from server as users in your buddy list update their positions, but not more often than every 10 seconds as specified in the second field of FRRDT.

FRANSON

Scenario: Write data to server

Client	Server
FRLIN	
	FRSES
FRWDT	
GPRMC	
GPRMC	
...	

Same scenario with real sample data

Client	Server
$FRLIN,,user1,8IVHF*7A	
	$FRSES,1221640*4F
$FRWDT,NMEA*78	
$GPRMC,154403.000,A,6311.64120,N,01438.02740,E,0.000,0.0,270707,,*0A	
$GPRMC,154423.000,A,6311.64143,N,01438.02743,E,0.000,0.0,270707,,*0C	
...	

And then keep sending GPRMC data to the server.

If you need altitude information send GGA sentences as well.

Same scenario using IMEI number

Client	Server
$FRLIN,IMEI,1234123412341234,*7B	
	$FRSES,1221640*4F
$FRWDT,NMEA*78	
$GPRMC,154403.000,A,6311.64120,N,01438.02740,E,0.000,0.0,270707,,*0A	
$GPRMC,154423.000,A,6311.64143,N,01438.02743,E,0.000,0.0,270707,,*0C	
...	

And then keep sending GPRMC data to the server.

Scenario to write data to server using UDP

You can also send data to over UDP. Send to online.gpsgate.com port 30175, or to your own GpsGate Server installation.

Client
FRLIN
GPRMC

Note: The server will not respond to UDP datagrams.

Same scenario with real sample data-

Client
$FRLIN,,user1,8IVHF*7A
$GPRMC,154403.000,A,6311.64120,N,01438.02740,E,0.000,0.0,270707,,*0A

Same scenario but login in using IMEI number

Client
$FRLIN,IMEI,1234123412341234,*7B
$GPRMC,154403.000,A,6311.64120,N,01438.02740,E,0.000,0.0,270707,,*0A

HTTP position updates

GpsGate Server can also receive data in HTTP format. The URL posted will have the following parameters:

```
longitude=34.2333&latitude=23.2222&altitude=34.0&speed=30.3&heading=234.5&date=20070529&time=123445.234&username=myuser&pw=encryptedpassword
```

or using IMEI

```
longitude=34.2333&latitude=23.2222&altitude=34.0&speed=30.3&heading=234.5&date=20070529&time=123445.234&imei=123456789012345
```

longitude	Decimal degrees (WGS84)
latitude	Decimal degrees (WGS84)
altitude	Above sea level in meters.
Speed	Speed over ground in knots
heading	Heading in degrees. 0 is north, 90 is east, 180 is south, 270 is west.
date	UTC data in the format: YYYYMMDD
time	UTC time in the format: HHMMSS.sss
username	The username specified
pw	The password specified with a simple encryption. The encryption/decryption algorithm can be found in Appendix 2
imei	Device IMEI number. Replaces username + pw. Available in GpsGate Server 1.2 build 415 and later.

For GpsGate.com the URL is http://online.gpsgate.com/GpsGate.aspx

For a GpsGate Server installation the URL is http://serveraddress/GpsGate.aspx
(where server address is the DNS to the server where GpsGate Server is installed).

FRANSON

GpsGate Server Protocol v1.1 Framework

v1.1 of the protocol adds a framework to send commands from client to server, and from server to client.

Note: The client must send a FRVER sentence after it has received FRSES to indicate it supports v1.1

GpsGate Protocol over TCP/IP

Client version notification

Sent by client to server to indicate which version of GpsGate Server protocol it supports.

This sentence is always answered by a similar from the server which contains the server protocol version, the server name, and the server version.

```
$FRVER,major,minor,name_and_version*XX
```

`major`	Major version of the protocol used. Should be 1
`minor`	Minor version of protocol used. Should be 1
`name_and_version`	Name of peer sending this message. E.g. "TestClient 1.0". This should always be a name followed by a version number. **Note!** Here you should have the real name of your client plus the version number.
`XX`	NMEA checksum. Algorithm to calculate shown in Appendix 1

Example:

`Client`	`Server`
`$FRLIN,,user1,8IVHF*7A`	
	`$FRSES,1221640*4F`
`$FRVER,1,1,TestClient 1.0*79`	
	`$FRVER,1,1,GpsGate Server 1.1.0.360*04`

Command structure

This structure is used when sending messages to execute commands. Messages can be sent from server to client and client to server. Several commands can be executed between client and server in one session.

Command Sentences

Syntax:

```
$FRCMD,username,command,Nmea,size*XX
```

`username`	The logged in user. This field can be empty.
`command`	the command to be executed. E.g. "_getupdaterules"
`Nmea`	"Nmea" if one or more NMEA sentences follows as arguments to command.
`size`	Number of NMEA sentences following.

XX	NMEA checksum. Algorithm to calculate shown in Appendix 1

or

Syntax:

```
$FRCMD,username,command,Inline,param1,param2,...,paramN*XX
```

username	The logged in user. This field can be empty.
command	the command to be executed. E.g. "_getupdaterules"
Inline	indicates that the rest of the fields in the sentence are arguments to the command.
param1 - N	Parameters to the command
XX	NMEA checksum. Algorithm to calculate shown in Appendix 1

Example:

```
$FRCMD,,_getupdaterules,Inline*1E
```

Return values

A return value is sent as a reply to an executed command.

Syntax:

```
$FRRET,username,command,Nmea,size*XX
```

username	User which executed the original FRCMD to which this is an answer.
command	The command to which this is an answer.
Nmea	"Nmea" if one or more NMEA sentences follows as arguments to command.
size	Number of NMEA sentences following.
XX	NMEA checksum. Algorithm to calculate shown in Appendix 1

Example:

```
$FRRET,Johan,_getupdaterules,Nmea,4*43
$FRVAL,DistanceFilter,500.0*67
$FRVAL,TimeFilter,60.0*42
$FRVAL,DirectionFilter,40.0*30
$FRVAL,DirectionThreshold,10.0*42
```

or

Syntax:

```
$FRCMD,username,command,Inline,param1,param2,...,paramN*XX
```

username	The logged in user. This field can be empty.
command	the command to be executed. E.g. "_getupdaterules"
Inline	indicates that the rest of the fields in the sentence are arguments to the command.
param1 - N	Parameters to the command
XX	NMEA checksum. Algorithm to calculate shown in Appendix 1

Command Variables

Sentence which holds a variable. Is typically used in combination with FRCMD and FRRET

Syntax:

```
$FRVAL,name,value*XX
```

name	Name of variable
value	Value of variable.
XX	NMEA checksum. Algorithm to calculate shown in Appendix 1

Example:

```
$FRVAL,DistanceFilter,500.0*67
```

Error handling

If a command for some reason cannot be executed FRERR is returned and not FRRET. The reason can be that the there is some kind of execution error, the command is not supported, or there is a protocol error. When FRERR is returned the client should assume no data has been affected on the server. The protocol is transaction based.

In all cases expect when there is a protocol or connection error the exchange of commands can continue between the server and the client.

Interrupted commands

If several FRCMD is sent without the previous one being finished, the peer executing the command should roll back the current command and start executing the new command. Example: If two FRCMD is received in a row, the last FRCMD should be executed.

Failed to execute command

If a FRCMD fails to execute a FRERR sentence with code “CannotExecute” is returned. The client is free to execute a new command after receiving FRERR.

Sample communication:

If the client tries to save data to a track recorder which doesn't belong to the logged in user.

Client	Server
$FRCMD,,_SaveTrackData,Nmea,2,2*54	
	$FRERR,CannotExecute,TrackRecorder not

	found for user*2B

Command not supported

A client or server that implements v1.1 of the GpsGate protocol does not need to implement any of the commands (see blow for a list of supported commands) described in this document. For commands that aren't supported a FRERR with code “NotSupported”. The client is free to execute a new command after receiving this error message.

Sample communication:

The command “_dummy” is not supported by the peer (in this case the server).

Client	Server
$FRCMD,,_dummy,Inline*6C	
	$FRERR,NotSupported,_dummy not supported*66

Invalid use of protocol

If the server thinks the client uses the protocol in an invalid way it returns FRERR with the code set to “Invalid”. The client must close the connection, and login again after this happens-

Sample communication:

If the client doesn't send FRVER before it starts to execute a command. In the case below the client has logged in using FRLIN, but forgot to send FRVER. And then it tries to execute a command. This scenario is not a valid v1.1 scenario, and the server responds with a FRERR Invalid.

Client	Server
$FRCMD,,_getupdaterules,Inline*1E	
	$FRERR,Invalid,FRVER must be sent to server first*25

GpsGate Protocol over XML/HTTP

The HTTP/XML implementation is very similar to the TCP/IP protocol and follow this pattern

Client sends a **HTTP POST** on **port 8008** path "**/GpsGate/**" to server with the following content

```
<cmd name="CommandName" a1="arg" ...>
      <nmea><d>$FRPOS</d><d>...</d></nmea>
      <nmea><d>...</d></nmea>
      ...
</cmd>
```

The server response is

```
<ret name="CommandName"/>
```

Or for an error

```
<ret name="CommandName">
      <nmea><d>$FRERR</d><d>Code</d><d>Free text</d></nmea>
</ret>
```

A post can include any number of commands. Most commands requires you to login, therefor you should start with a _Login command.

```
<cmd name="_Login">
      <nmea><d>$FRVAL</d><d>Username</d><d>user1</d></nmea>
      <nmea><d>$FRVAL</d><d>PW</d><d>8IVHF</d></nmea>
</cmd>
```

Note that the password uses a simple "encryption" algorithm described in Appendix 2

Most commonly you then execute the _SendMessage command. Use a "GpsGate Generic Device" device mapper. Send status messages to the "GpsGateDevice" namespace. But you can execute any command supported by your GpsGate Server installation.

A sample dump (including HTTP header). From client to server. In this sample chunked data is used, fixed size is also ok. Note the path "/GpsGate/"

```
POST /GpsGate/ HTTP/1.1
Host: localhost:7500
Transfer-Encoding: chunked
Expect: 100-continue
Connection: Keep-Alive

196
<gpsgate><cmd
name="_Login"><nmea><d>$FRVAL</d><d>Username</d><d>user1</d></nmea
><nmea><d>$FRVAL</d><d>PW</d><d>8IVHF</d></nmea></cmd><cmd
name="_SendMessage" a
0="0"
a1="1"><nmea><d>$FRPOS</d><d>0200.00000</d><d>N</d><d>00100.00000</d><d>E<
/d><d>0.0</d><d>5.832</d><d>4.0</d><d>121107</d><d>142600.000</d><d
/></nmea><nm
ea><d>$FRVAL</d><d>Switch1</d><d>1</d><d
/><d>GpsGateDevice</d></nmea></cmd></gp
sgate>
0

The response is

<ret name="_Login"/>
<ret name="_SendMessage"/>
```

Incoming commands

A client or server supporting GpsGate Server Protocol v1.1 shall support the framework of sending commands using FRCMD + FRRET as described above. But it does not need to support any specific commands.

If a peer tries to execute a command that isn't supported FRERR with err_code “NotSupported” should be returned.

Below are some commands supported by GpsGate Server listed.

_getupdaterules

GpsGate Server v1.1 build 360 and later.

The command “_getupdaterules” is used by a client to ask server for recommended intervals to send position

updates to the server.

The values typically include distance interval, time interval, speed and direction changes.

The rules should be OR:ed by the client. Rules not understood should simply be ignored.

Sample communication:

Client	Server
`$FRCMD,,_getupdaterules,Inline*1E`	

	`$FRRET,Johan,_getupdaterules,Nmea,4*43`
	`$FRVAL,DistanceFilter,500.0*67`
	`$FRVAL,TimeFilter,60.0*42`
	`$FRVAL,DirectionFilter,40.0*30`
	`$FRVAL,DirectionThreshold,10.0*42`

NOTE!

1. The number of sentences following FRRET can be 0 to many.
2. The 4th field "4" in FRRET determines that there are 4 sentences following FRRET.
3. The answer may contain zero or many FRVAL sentences, and zero or many other sentences.

The client should simply ignore sentences it doesn't understand.

Possible values returned by server

`DistanceFilter`	Distance in meters before a new position update should be sent to server.
`TimeFilter`	Interval in seconds before a new position should be sent to server
`SpeedFilter`	Change in speed in meters per second, before a new position update should sent to server by client.
`DirectionFilter`	Change of heading in degrees, before a new position update should be sent.
`DirectionThreshold`	Distance in meters travelled before "DirectionFilter" should be considered.

Notes:

The rules should be "OR:ed".

Rules that are inactivated will not be sent back to the client.

Rules that are not understood by the client should simply be ignored.

The rules are recommendations from the server, so the server will also be able to handle position updates not following the rules.

More rules will be added in coming versions.

Example that demonstrates how a client logs into a server, ask for recommended update rules and then starts sending position updates to server according to rules.

Client	Server
`$FRLIN,,user1,8IVHF*7A`	

	$FRSES,1221640*4F
$FRVER,1,1,TestClient 1.0*79	
	$FRVER,1,1,GpsGate Server 1.1.0.360*04
$FRCMD,,_getupdaterules,Inline*1E	
	$FRRET,User1,_getupdaterules,Nmea,2*07 $FRVAL,DistanceFilter,500.0*67 $FRVAL,TimeFilter,60.0*42
$FRWDT,NMEA*78	
$GPRMC,154403.000,A,6311.64120,N,01438.02740,E,0.000,0.0,270707,,*0A	
...	

_SendPosition

GpsGate Server 1.2 build 410 or later

The command "_SendPosition" is used to send a position update for the logged in user. You can control which track to save the position to, or if the position should not be saved to any track at all. The users current position will be updated, and the position information will be routed to other users that listens using FRRDT.

Syntax:

$FRCMD,*username*,_SendPosition,Nmea,1,*track_recorder_id**XX

$FRPOS...

username	The logged in user. This field can be empty.
_SendPosition	Name of command. Should be "_SendPosition"
Nmea	Indicates we have an NMEA sentence following the FRCMD sentence. Should be "Nmea"
1	One FRPOS sentence follows the FRCMD sentence. Should be "1"
track_recorder_id	Determines if and to which track the position should be saved. If left out or set to "0" the position is saved to the default "life track" If set to "-1" the position is only routed to listeners using FRRDT. Track data or current position is not saved. If set to "-2" the position is not saved to any track. The users current position is however updated. Or it can be set to a TrackRecorderID returned by the command "_CreateTrackRecorder". The TrackRecorderID is an integer.
XX	NMEA checksum

Sample communication:

In this sample the client has logged in as "User1". The current position is updated for the user, and the default track – the "life track" - is updated.

Client	Server

`$FRCMD,,_SendPosition,Nmea,1*12` `$FRPOS,0200.00000,N,00100.00000,E,0.0,5.832,4.0,121107,142600.000,*5D`	
	`$FRRET,User1,_SendPosition,Inline*08`

_CreateTrackRecorder

GpsGate Server 1.2 build 383 or later

The command "_CreateTrackRecorder" creates a track recorder on the server for the logged in user. A track recorder is necessary to record a track. Any number of track recorders can be created and used at the same time by one user. In this way parallel tracks can be recorded.

NOTE! You do not need to create a new track recorder each time the user logs in. Let the client remember the track recorder ID, and continue to use it. The server will determine when and how to create new tracks of the data, depending on the server side track recorder settings. Only create several track recorders if you want to record parallel tracks in time.

Syntax client command:

`$FRCMD,`*`username`*`,_CreateTrackRecorder,Inline,`*`track_name`*`*XX`

`username`	The logged in user. This field can be empty.
`_CreateTrackRecorder`	Name of command. Should be "_CreateTrackRecorder"
`Inline`	Should be "Inline"
`track_name`	Name of track recorder. Tracks created by this track recorder will get this name.

Syntax return value from server:

`$FRRET,`*`username`*`,_CreateTrackRecorder,Inline,`*`track_recorder_id`*`*42`

`username`	The logged in user.
`_CreateTrackRecorder`	Name of command. Always "_CreateTrackRecorder"
`Inline`	Should be "Inline"
`track_recorder_id`	ID of created track recorder. This is an unsigned 32-bit integer value.

Sample communication:

This sample creates a new track recorder named "TestTrack" for the logged in user. The server returns the track recorder id "2".

Client	Server
`$FRCMD,,_CreateTrackRecorder,Inline,TestTrack*00`	
	`$FRRET,User1,_CreateTrackRecorder,Inline,2*42`

_SaveTrackData

GpsGate Server 1.2 build 383 or later

The command "_SaveTrackData" saves a range of positions to a track recorder created by "_CreateTrackRecorder". "_SaveTrackData" cannot be used to store data to the "life track".

Note that the server has track recorder settings that will limit the number of saved positions in the track depending on how the settings are made. See "_getupdaterules" for more information.

If the command fails no positions are stored into the track. A FRRET returned from the server indicates the positions has been successfully saved to the server track.

Syntax:

`$FRCMD,`*`username`*`,_SaveTrackData,Nmea,`*`position_count`*`,`*`track_recorder_id`*`*XX`

`$FRPOS...`

`$FRPOS...`

`...`

username	The logged in user. This field can be empty.
_SaveTrackData	Name of command. Should be "_SaveTrackData"
Nmea	Indicates we have an NMEA sentence following the FRCMD sentence. Should be "Nmea"
position_count	Number of FRPOS sentences following the FRCMD sentence. Each FRPOS sentence represents one position to be saved into the server track recorder. Do not send more than twenty (20) FRPOS sentences at a time. The server may or may not refuse larger transactions.
track_recorder_id	Determines if and to which track the position should be saved. Set to a track recorder returned by the command "_CreateTrackRecorder". The track recorder is an integer. Value cannot be left out.
XX	NMEA checksum

Sample communication:

This sample saves two positions for the logged in user to a track recorder created by "_CreateTrackRecorder" (see above). The user name in FRPOS can be left out to save bandwidth.

Client	Server
$FRCMD,,_SaveTrackData,Nmea,2,2*54 $FRPOS,0100.00000,N,00100.00000,E,0.0,1.944,0.0,011206,120000.000,*5C $FRPOS,0200.00000,N,00200.00000,E,0.0,3.888,0.0,011206,120002.000,User1*5D	
	$FRRET,User1,_SaveTrackData,Inline*53

_SendMessage

GpsGate Server v2.0 and later.

The command "_SendMessage" is used by a client send geo coded messages to the server.

The values typically include status information from a vehcile, like pressure, temperature, engine on/off, an alarm signal etc.

Sample communication to GpsGate Message Namespace:

Client	Server
$FRCMD,,_SendMessage,Nmea,3*62 $FRPOS,1558.80000,N,04748.00000,E,34.0, 19.438,89.0,140608,122341.562,*61 $FRVAL,Temperature,23.4,,*18 $FRVAL,Engine on,1,,*71	
	$FRRET,User1,_SendMessage,Inline* 7A

The number of FRVAL may be zero or as many is you like. The FRPOS is optional, if it is not included the messages will not be geo coded.

The message name (in the sample above "Temperature" and "Engine on") must be registered on the server. See Appendix 3 for more information.

Sample communication to GpsGateDevice Message Namespace:

Client	Server
$FRCMD,,_SendMessage,Nmea,3*62 $FRPOS,1558.80000,N,04748.00000,E,34.0, 19.438,89.0,140608,122341.562,*61 $FRVAL,Switch1,1,,GpsGateDevice*36 $FRVAL,Analog3,34.56,,GpsGateDevice*37	
	$FRRET,User1,_SendMessage,Inline* 7A

See Appendix 4 for complete overview of GpsGateDevice Message Namespace.

The GpsGateDevice Message values are mapped into GpsGate Message values using the Device Mapper. If you implement a tracker device or proxy using GpsGate Protocol, it is recommended that you use GpsGateDevice Message Namespace, and map the signals to more meaningful variables on the server using Device Mapper.

Either you must use IMEI number as login. Or if you use username+password as login for the device the device's name must be the same as the "name_and_version" filed in FRVER sent to the server.

_CmdQueueStatus

_CmdQueueReset

_DeviceReset

_Ping

_SetTrackRecorder

_SupportsProxyMessage

_EndTrack

_ProxyMessage

Custom incoming GpsGate Command plugins

Outgoing commands

_GprsSettings

_StartTracking

_StopTracking

_PollPosition

_PlainText

_SetDeviceTrackRecorder

Custom outgoing GpsGate Command plugins

More information - http://franson.com/forum/topic.asp?TOPIC_ID=8098

Appendix 1 - NMEA checksum calculation

C++

```
#define NIBBLE2HEX(c) ((c) > 9 ? (c) + 'A' - 10 : (c) + '0')

// buf is a char[] array which contains the NMEA sentence without trailing
checksum.
// E.g. "$FRLIN,,user1,8IVHF" and "*7A" will be added

// buf_inx is an index to the last free position in the buffer

int checksum = 0;
int inx;

for(inx = 1; inx < buf_inx; inx++)
{
      checksum ^= buf[inx];
}

buf[buf_inx++] = '*';
buf[buf_inx++] = NIBBLE2HEX((checksum >> 4) & 0xf);
buf[buf_inx++] = NIBBLE2HEX(checksum & 0xf);
```

Appendix 2 - Password algorithm in C#

```
private string m_InvertString(string strToInvert)
{
    StringBuilder builder = null;

    if (strToInvert != null)
    {
        builder = new StringBuilder();

        int iLength = strToInvert.Length;

        for (int iIndex = iLength - 1; iIndex >= 0; iIndex--)
        {
            char c = strToInvert[iIndex];

            if (c >= '0' && c <= '9')
            {
                builder.Append((char)(9 - (c - '0') + '0'));
            }
            else if (c >= 'a' && c <= 'z')
            {
                builder.Append((char)(('z' - 'a') - (c - 'a') + 'A'));
            }
            else if (c >= 'A' && c <= 'Z')
            {
                builder.Append((char)(('Z' - 'A') - (c - 'A') + 'a'));
            }
        }
    }

    return builder != null ? builder.ToString() : null;
}

Encryption sample: "coolness" -> "HHVMOLLX"
```

Appendix 3 – Messages GpsGate Namespace

Message variables *GpsGate* namespace. Messages in this namespace will be saved on the server without passing a Device Mapper. The values below are mapped into the GpsGate Message Namespace (Appendix 3) using the Device Mapper.

Data type	Name	Unit
System.Double	Pressure	Pascal
System.Double	Oil Pressure,	Pascal
System.Double	Tire Pressure	Pascal
System.Double	Temperature	Celsius
System.Double	Oil Temperature	Celsius
System.Double	Cargo Temperature	Celsius
System.Double	Outdoor Temperature	Celsius
System.Double	Indoor Temperature	Celsius
System.Double	Voltage	Volt
System.Double	Battery Voltage	Volt
System.Double	Volume	liter
System.Double	Fuel level	liter
System.Double	Distance	meter
System.Boolean	Door open	switch
System.Boolean	Engine on	switch
System.Boolean	Left GEO fence (switch)	switch
System.Boolean	Speed limit (switch)	switch
System.Boolean	Speed limit (push button)	button
System.Boolean	Left GEO fence (push button)	button
System.Boolean	SOS	button
System.Boolean	Alarm	button
System.Boolean	Clock	button
System.Boolean	Man down	button
System.Boolean	Low Battery	button
System.Boolean	Motion	button

Appendix 4 – Messages GpsGateDevice Namespace

Message variables *GpsGateDevice* namespace. Messages in this namespace will use the Device Mapper for "GpsGate Generic Device".

Data type	Name	Unit
System.Boolean	Switch1	switch
System.Boolean	Switch2	switch
System.Boolean	Switch3	switch
System.Boolean	Switch4	switch
System.Boolean	Switch5	switch
System.Boolean	Switch6	switch
System.Boolean	Switch7	switch
System.Boolean	Switch8	switch
System.Boolean	Switch9	switch
System.Boolean	Switch10	switch
System.Boolean	Switch11	switch
System.Boolean	Switch12	switch
System.Boolean	Switch13	switch
System.Boolean	Switch14	switch
System.Boolean	Switch15	switch
System.Boolean	Switch16	switch
System.Boolean	Button1	button
System.Boolean	Button2	button
System.Boolean	Button3	button
System.Boolean	Button4	button
System.Double	Analog1	none
System.Double	Analog2	none
System.Double	Analog3	none
System.Double	Analog4	none
System.Double	Analog5	none
System.Double	Analog6	none
System.Double	Analog7	none
System.Double	Analog8	none