Wialon IPS v.2.0
Communication protocol

Wialon IPS communication protocol was developed by Gurtam for personal and vehicle GPS/GLONASS trackers, which send data to satellite monitoring system server over TCP or UDP.

Changes

Version	date	Changes
2.0	10/2014	Field «protocol_version» was added in packet L (login packet) and in packets for UDP protocol. Packets QT, IT, T, AIT, AT were added for tachograph files (ddd-files) transmission. Field «crc16» was added in packets L, SD, D, M, I, IT, T, US, UC. Error verifying checksum code was added for packets AL, ASD, AD, AM, AI, AIT, AT.

Incoming data (TCP)

All data are sent and received in plain text format over TCP protocol and has the following format:

#TP#msg\r\n

#	Start byte
TP	Type of packet, all possible types are listed in table 1
#	Separator
msg	Message
\r\n	End of message, <CR><LF> symbols (0D0A in HEX)

Packet types

Type	Description	Sender
L	Login	Tracker
AL	Answer to login	Server
D	Data packet	Tracker
AD	Answer to data packet	Server
P	Ping(heartbeat) packet	Tracker
AP	Answer to ping (heartbeat packet)	Server
SD	Short data packet	Tracker
ASD	Answer to short data packet	Server
B	Blackbox packet	Tracker
AB	Answer to blackbox packet	Server
M	Message to driver	Tracker/Server
AM	Reply to message from driver	Server
QI	Query photo command	Server
I	Packet with photo	Tracker
AI	Reply to packet with photo	Server
QT	Command to request a tachograph file (ddd-file)	Server
IT	Packet containing information about a ddd-file	Tracker
AIT	Reply to AT packet	Server
T	Packet containing a ddd-file part	Tracker
AT	Reply to T packet	Server
US	Packet with new firmware	Server
UC	Packet with configuration file	Server

Login packet

#L#protocol_version;imei;password;crc16\r\n

password	Protocol version. This field must contain value 2.0
imei	Controller unique ID, IMEI, or serial number
;	Separator
password	Password for access to device, if no password, then **NA** is sent
crc16	Checksum (Appendix 1)

Server sends answer to login packet, AL:
“1” – authorization successful
“0” – connection rejected by server
“01” – error checking password
“10” – error verifying checksum

Example:
#AL#1\r\n
#AL#0\r\n

Short data packet

#SD#date;time;lat1;lat2;lon1;lon2;speed;course;height;sats;crc16\r\n

date	Date in UTC format, DDMMYY, if no data, **NA** is sent
time	Time in UTC format, HHMMSS, if no data, **NA** is sent
lat1;lat2	Latitude (5544.6025;N), if no data, **NA;NA** is sent
lon1;lon2	Longitude (03739.6834;E), if no data, **NA;NA** is sent
speed	Speed, integer, km/h, if no data, **NA** is sent
course	Course, integer, degrees, if no data, **NA** is sent
height	Height, , integer, m, if no data, **NA** is sent
sats	Number of satellites, integer, if no data, **NA** is sent
crc16	Checksum (Appendix 1)

If date and time fields is **NA**, server will set current time for packet.

Server sends ASD packet as an answer to SD packet:
“-1” – packet structure error
“0” – incorrect time
“1” – packet successfully registered
“10” – error getting coordinates
“11” – error getting height, speed or course
“12” – error getting amount of satellites
“13” – error verifying checksum

Example:
#ASD#1\r\n
#ASD#0\r\n
#ASD#10\r\n

Data packet

#D#date;time;lat1;lat2;lon1;lon2;speed;course;height;sats;hdop;inputs;outputs;adc;ibutton;params;crc16\r\n

date	Date in UTC format, DDMMYY, if no data, **NA** is sent
time	Time in UTC format, HHMMSS, if no data, **NA** is sent
lat1;lat2	Latitude (5544.6025;N), if no data, **NA;NA** is sent
lon1;lon2	Longitude (03739.6834;E), if no data, **NA;NA** is sent
speed	Speed, integer, km/h, if no data, **NA** is sent
course	Course, integer, degrees, if no data, **NA** is sent
height	Height, , integer, m, if no data, **NA** is sent
sats	Number of satellites, integer, if no data, **NA** is sent
hdop	Horizontal Dilution of Precision, double, if no data, **NA** is sent
inputs	Digital inputs, each bit corresponds to one digital input beginning from the LSB, integer, if no data, **NA** is sent
outputs	Digital outputs, each bit corresponds to one digital output beginning from the LSB, integer, if no data, **NA** is sent
adc	Analog inputs, fractional numbers separated by comma, if no data, empty string is send. Input numbering begins from 1 (adc1..adcN). Example 14.77,0.02,3.6
ibutton	Driver key code, custom length string. If no data, **NA** is

	sent
params	Set of additional parameters separated by comma. Each parameter has the following format: NAME:TYPE:VALUE NAME – custom string TYPE – parameter type, 1 – int/long long, 2 – double, 3 – string VALUE – parameter value, depends on type To send panic button use parameter with 1 type named “SOS”, 1 mean panic button was pressed. To send text message use parameter with 3 type named “text”. This parameter can be used to send driver message with position and other parameters. Example: count1:1:564,fuel:2:45.8,hw:3:V4.5
crc16	Checksum (Appendix 1)

If date and time fields is **NA**, server will set current time for packet.

Server sends AD packet as an answer to D packet:

“-1” – packet structure error
“0” – incorrect time

“1” – packet successfully registered
“10” – error getting coordinates
“11” – error getting height, speed or course
“12” – error getting amount of satellites or HDOP
“13” – error getting inputs or outputs
“14” – error getting adc
“15” – error getting additional parameters
“16” – error verifying checksum

Example:
#AD#1\r\n
#AD#0\r\n
#AD#10\r\n
#AD#11\r\n
...
#AD#15\r\n

Ping (heartbeat) packet

#P#\r\n

This packet is used for keeping active TCP-connection to server and checking
channel availability.

Server sends AP packet as an answer to P packet:
Example:
#AP#\r\n

Blackbox packet

#B#msg\r\n

Blackbox packet is used for transmission of several messages at one time.
In this case “msg” contains several SD or D packets bodies (without type signature), separated by “ | “symbol.

Example:
#B#date;time;lat1;lat2;lon1;lon2;speed;course;height;sats|
date;time;lat1;lat2;lon1;lon2;speed;course;height;sats|
date;time;lat1;lat2;lon1;lon2;speed;course;height;sats|crc16\r\n

Server sends AB packet as an answer to B packet, where number of registered messages is specified:

Example:
#AB#3\r\n
#AB#0\r\n

or an empty string which means error verifying checksum.

Example:
#AB#\r\n

Message to driver

#M#msg;crc16\r\n

Serves for sending a text message to driver. “msg” is the actual text. Message can be sent from either server or tracker.

Server sends AM command as a reply to message from driver.

“1” – message received
“0” – error receiving message
“01” – error verifying checksum

Example:
#AM#1\r\n
#AM#0\r\n

Query photo packet

#QI#\r\n

This type of packet is used to query photoes from the tracker.

Packet with photo

This type of packet is used for sending photos to Wialon's server. The whole photo is separated into blocks of bytes and then each of them is sent to the server. Recommended size of one single block is up to 50 kb. If server fails to receive any image block, it breaks the connection. In this case we recommend to reduce the size of blocks.

#I#sz;ind;count;date;time;name;crc16\r\nBIN

sz	size of packet's binary data (i.e., 512 bytes)
ind	index number of transmitting block (numbering starts from zero)
count	index number of last transmitting block
date	date in DDMMYY format, in UTC

time	time in HHMMSS format, in UTC
name	name of transmitting photo
crc16	checksum (Appendix 1)
BIN	photo's binary block

Server sends AI command as a reply to packet with photo.

#AI#NA;0\r\n means incorrect packet structure

#AI#ind;result\r\n
ind – the number of the transmitted block
result – packet processing result:
“1” – packet with block of photo is received
“0” – error receiving packet
“01” – error verifying checksum

#AI#1\r\n – photo is completely received and saved in Wialon

Example:
Tracker: #I#51200;0;1;070512;124010;sample.jpg;crc16\r\nBIN
Server: #AI#0;1\r\n
Tracker: #I#28923;1;1;070512;124010;sample.jpg;crc16\r\nBIN
Server: #AI#1;1\r\n
Server: #AI#1\r\n

Packet to request a tachograph file

#QT#driverid\r\n

This packet type is used to request a ddd-file from a tachograph.

driverid	driver identifier

Packet containing information about a ddd-file

This packet type must be sent to the server before sending a ddd-file.

#IT#date;time;driverid;code;count;crc16\r\n

date	date in DDMMYY format, in UTC
time	time in DDMMYY format, in UTC
driverid	driver identifier
code	error code. If there is no error, an empty string is sent
count	total number of ddd-file blocks
crc16	checksum for the following part: date;time;driverid;code;count;

Server responds to #IT#:

#AIT#state\r\n

state	“1” – packet received “0” – error receiving packet “01” – error verifying cheksum

Packet #IT# must be followed by #T# packets containing ddd-file blocks. Packet #T# format is described below.
DDD-file is saved under the following name on the server: driverid_yyyymmdd_hhmmss.ddd

Packet with a ddd-file block

#T#code;sz;ind;crc16\r\nBIN

code	error code. If there is no error, an empty string is sent
sz	size of packet's binary data (bytes)
ind	index number of the transmitted block (numbering starts from zero)

crc16	checksum for BIN part (Appendix 1)
BIN	file binary part of sz size

Server responds to every #T# packet:

#AT#ind;state\r\n

ind	index number of the transmitted block
state	“1” – packet received “0” – error receiving packet “01” – error verifying cheksum

When ddd-file is completely received and saved server responds:

#AT#1\r\n

All #T# packets with ddd-file blocks must be transmitted within the same TCP-connection as #IT# packet.

Packet with new firmware

Serves for sending new firmware to tracker.
#US#sz;crc16\r\nBIN

sz	Size of firmware's binary data
crc16	Checksum (Appendix 1)
BIN	Firmware in binary mode

Packet with configuration file

Serves for sending configuration file to tracker.
#UC#sz;crc16\r\nBIN

sz	Size of configuration file, bytes
crc16	Checksum (Appendix 1)
BIN	Content of configuration file

Incoming data (UDP)

All data are sent and received in plain text format and have the same structure as in TCP protocol, but with adding of protocol version and IMEI in the beginning of packet. Protocol version field must contain value “2.0”. For example, short data packet will look in the following way:

2.0;imei#SD#date;time;lat1;lat2;lon1;lon2;speed;course;height;sats;crc16\r\n

Data compression

All WIalon IPS data packets directed to server can be compressed before sending. It's useful for transferring large #B# packets.

For compression should be used the DEFLATE algorithm from **zlib** library (http://www.zlib.net/, RFC 1951).

Both of transport protocols are supported (TCP and UDP). Server always sends usual data packets (without compression) because its answers are small.

Structure of the container with compressed package:

Size	1 byte	2 bytes	Specified size
Content	Compression sign –0xFF byte	Size of compressed data (little-endian, 16-bit integer)	Compressed data block, specified size, as-is

Container should contain only one packet of text protocol.

When compression is used, there is no necessary to append chars **\r\n** to end of text protocol packet so they can be omitted before compression.

Compression is transparent for server that's why it can receive compressed and usual packets from one tracker at the same time.

Example of compressed packet:

Source packet	#L#imei;password
Full data of compressed packet (27 bytes, in HEX)	FF180078DA53F651CECC4DCDB42E482C2E2ECF2F4A01002D1C05E5 FF – compression sign 1800 – size of compressed data, means 24 (0x18) bytes 78DA... - compressed data

Appendix 1

Crc16 must be a hexadecimal number with a big-endian order of bytes without leading zeroes on the left, for example:
AA13BB which is 11146171 in a decimal format.

The part of a packet between #TP# and crc16 is used for checksum calculation in packets L, SD, D, B and M.

Packet sample:
#SD#date;time;lat1;lat2;lon1;lon2;speed;course;height;sats;crc16\r\n

In this case crc16 is calculated for the following part of the packet:
date;time;lat1;lat2;lon1;lon2;speed;course;height;sats;

Packet sample:
#B#date;time;lat1;lat2;lon1;lon2;speed;course;height;sats|
date;time;lat1;lat2;lon1;lon2;speed;course;height;sats|crc16\r\n

In this case crc16 is calculated for the following part of the packet:
date;time;lat1;lat2;lon1;lon2;speed;course;height;sats|
date;time;lat1;lat2;lon1;lon2;speed;course;height;sats|

Field BIN is used for crc16 calculation in packets I, US, UC, T.

Packet sample:
#I#51200;0;1;070512;124010;sample.jpg;crc16\r\nBIN

In this case crc16 is calculated for field BIN.

C language code sample for crc16 calculation:

```
static const unsigned short crc16_table[256] =
{
   0x0000,0xC0C1,0xC181,0x0140,0xC301,0x03C0,0x0280,0xC241,
   0xC601,0x06C0,0x0780,0xC741,0x0500,0xC5C1,0xC481,0x0440,
   0xCC01,0x0CC0,0x0D80,0xCD41,0x0F00,0xCFC1,0xCE81,0x0E40,
   0x0A00,0xCAC1,0xCB81,0x0B40,0xC901,0x09C0,0x0880,0xC841,
   0xD801,0x18C0,0x1980,0xD941,0x1B00,0xDBC1,0xDA81,0x1A40,
   0x1E00,0xDEC1,0xDF81,0x1F40,0xDD01,0x1DC0,0x1C80,0xDC41,
   0x1400,0xD4C1,0xD581,0x1540,0xD701,0x17C0,0x1680,0xD641,
   0xD201,0x12C0,0x1380,0xD341,0x1100,0xD1C1,0xD081,0x1040,
   0xF001,0x30C0,0x3180,0xF141,0x3300,0xF3C1,0xF281,0x3240,
   0x3600,0xF6C1,0xF781,0x3740,0xF501,0x35C0,0x3480,0xF441,
   0x3C00,0xFCC1,0xFD81,0x3D40,0xFF01,0x3FC0,0x3E80,0xFE41,
   0xFA01,0x3AC0,0x3B80,0xFB41,0x3900,0xF9C1,0xF881,0x3840,
   0x2800,0xE8C1,0xE981,0x2940,0xEB01,0x2BC0,0x2A80,0xEA41,
   0xEE01,0x2EC0,0x2F80,0xEF41,0x2D00,0xEDC1,0xEC81,0x2C40,
   0xE401,0x24C0,0x2580,0xE541,0x2700,0xE7C1,0xE681,0x2640,
   0x2200,0xE2C1,0xE381,0x2340,0xE101,0x21C0,0x2080,0xE041,
   0xA001,0x60C0,0x6180,0xA141,0x6300,0xA3C1,0xA281,0x6240,
   0x6600,0xA6C1,0xA781,0x6740,0xA501,0x65C0,0x6480,0xA441,
   0x6C00,0xACC1,0xAD81,0x6D40,0xAF01,0x6FC0,0x6E80,0xAE41,
   0xAA01,0x6AC0,0x6B80,0xAB41,0x6900,0xA9C1,0xA881,0x6840,
   0x7800,0xB8C1,0xB981,0x7940,0xBB01,0x7BC0,0x7A80,0xBA41,
   0xBE01,0x7EC0,0x7F80,0xBF41,0x7D00,0xBDC1,0xBC81,0x7C40,
   0xB401,0x74C0,0x7580,0xB541,0x7700,0xB7C1,0xB681,0x7640,
   0x7200,0xB2C1,0xB381,0x7340,0xB101,0x71C0,0x7080,0xB041,
   0x5000,0x90C1,0x9181,0x5140,0x9301,0x53C0,0x5280,0x9241,
   0x9601,0x56C0,0x5780,0x9741,0x5500,0x95C1,0x9481,0x5440,
   0x9C01,0x5CC0,0x5D80,0x9D41,0x5F00,0x9FC1,0x9E81,0x5E40,
   0x5A00,0x9AC1,0x9B81,0x5B40,0x9901,0x59C0,0x5880,0x9841,
   0x8801,0x48C0,0x4980,0x8941,0x4B00,0x8BC1,0x8A81,0x4A40,
   0x4E00,0x8EC1,0x8F81,0x4F40,0x8D01,0x4DC0,0x4C80,0x8C41,
   0x4400,0x84C1,0x8581,0x4540,0x8701,0x47C0,0x4680,0x8641,
   0x8201,0x42C0,0x4380,0x8341,0x4100,0x81C1,0x8081,0x4040
};

unsigned short crc16 (const void *data, unsigned data_size)
{
   if (!data || !data_size)
      return 0;

   unsigned short crc = 0;
   unsigned char* buf = (unsigned char*)data;

   while (data_size--)
      crc = (crc >> 8) ^ crc16_table[(unsigned char)crc ^ *buf++];

   return crc;
}
```