Трекер обменивается с сервером пакетами следующей структуры (бинарные контейнеры):

Заголовок пакета тип t_binary_container
Содержимое пакета - произвольные бинарные данные

В ответ сервер должен прислать трекеру пакет такой же структуры и произвольним содержимом. В случае если трекер
получает корректный пакет с данной структурой в ответ на свою посылку пакет считается переданным успешно и трекер
переходит к отправке нового пакета. В случае если ответного пакета не получено или получен некорректный пакет (не
совпадает контрольная сумма или преамбула), исходный пакет считется отправленным с ошибкой и его отправка
повторяется.

Описание типов данных:
uint8_t - 8ми битный целый беззнаковый.
int8_t - 8ми битный целый знаковый.

uint16_t - 16ти битный целый беззнаковый. Порядок записи байтов - обратный (от младшего к старшему, интеловский, как в x86)
int16_t - 16ти битный целый знаковый.

uint32_t - 32х битный целый беззнаковый.
int32_t - 32х битный целый знаковый.

Все пакеты упакованы. Т.е. выравнивание элементов страктуры - 1 байт.

Алгоритм расчета контрольной суммы:

```
 1 /*
 2   Name  : CRC-16 CCITT
 3   Poly  : 0x1021    x^16 + x^12 + x^5 + 1
 4   Init  : 0xFFFF
 5   Revert: false
 6   XorOut: 0x0000
 7   Check : 0x29B1 ("123456789")
 8   MaxLen: 4095 (32767 bit)
 9 */
10
11 uint16_t crc16(unsigned char *pcBlock, uint16_t len)
12 {
13     unsigned short crc = 0xFFFF;
14     unsigned char i;
```

```
15
16     while(len--)
17     {
18         crc ^= *pcBlock++ << 8;
19
20         for(i = 0; i < 8; i++)
21             crc = crc & 0x8000 ? (crc << 1) ^ 0x1021 : crc << 1;
22     }
23
24     return crc;
25 }
```

Описание структуры **t_binary_container**:

```
1 typedef struct {
2   uint16_t crc;          // crc: sizeof (t_binary_container +
pay_load)
3   uint16_t preamble;
4   uint32_t tracker_id;
5   uint16_t data_len;
6 } t_binary_container;
```

crc - контрольная сумма пакета целиком начиная с поля preamble и заканчивая последним байтом содердимого пакета (кроме поля crc);
preamble - преамбула. Содердимое всегда 0x8A2C;
tracker_id - идентификатор трекера;
data_len - длина поля с данными;

Максимальная длина бинарного контейнера - 1400байт, включая заголовок.

В качестве полезного содержимого бинарного контейнера выступают данные со структурой **t_common_data_header** за которой следуют полезные данные пакета:

Заголовок - t_common_data_header
Полезные данные пакета

Таких пакетов в контейнере может быть несколько (а может не быть вовсе).

Описание структуры **t_common_data_header**:

```
1 typedef struct {
```

```
 2   uint16_t crc;

 3   uint16_t serial_id;

 4   uint32_t timestamp;

 5   uint16_t packet_type;

 6   uint16_t packet_len;

 7   uint32_t x_coord;

 8   uint32_t y_coord;

 9   uint8_t sf;

10 } t_common_data_header;
```

crc - контрольная сумма пакета, включая блок с данными (кроме поля crc);
serial_id - серийный номер пакета
timestamp - unixtime пакета (UTC+0)
packet_type - тип структуры, которая является полезными данными текущего пакета;
packet_len - длина полезных данных;
x_coord - широта - в соответсвии с правилами ЕГТС. Если равно 0xffffffff - значит нет фиксации валидных координат;
y_coord - долгота - в соответствии с правилапи ЕГТС. Если равно 0xffffffff - значит нет фиксации валидных координат;
sf - Дополнительные флаги пакета - битовая маска:

бит 0 - признак отправки данных из черного ящика (0 - свежесобранные данные, 1 - данные отправлены из черного ящика);
бит 6 - поле LAHS записи EGTS_SR_POS_DATA; 0 - северная широта; 1 - южная широта;
бит 7 - поле LOHS записи EGTS_SR_POS_DATA; 0 - восточная долгота; 1 - западная долгота;

Тип пакета 0x0040 - информация о GPS-координатах

```
 1 typedef struct {

 2   uint16_t v1224; - напряжение на аккумуляторе автомобиля, сотые доли
вольта

 3   uint16_t v_bat; - напряжение на аккумуляторе трекера - сотые доли
вольта;

 4   uint16_t fs_data; - данные топливного датчка 1 на канале RS232 или
RS485

 5   uint8_t  stop_state; - 1 если автомобиль стоит на месте; 0 - если
едет;

 6   uint8_t  ign_state; - 1 если зажигание включено; 0 - если зажигание
выключено;

 7   uint8_t  d_state; - состояние дискретных входов - битовая маска;

 8   uint16_t freq2; - частота на дискретном входе 2;

 9   uint16_t c_counter2; - счетчик импульсов на дискретном входе 2;

10   uint16_t freq1; - частота на дискретом входе 1;

11   uint16_t c_counter1; - счетчик импульсов на дискретном входе 1;
```

```
12   uint8_t  ant_state; - состояние GPS-антенны;

13   uint8_t  fix_type; - тип фиксации спутников;

14   uint8_t  sat_count; - количество отслеживаемых спутников;

15   uint16_t altitude; - высота;

16   uint16_t geoid_height; - высота геоида;

17   uint32_t x_coord; - широта в системе шифрования ЕГТС;

18   uint32_t y_coord; - долгота в системе шифрования ЕГТС;

19   uint16_t speed; - сотые км-час;

20   uint16_t course; - азимут, градусы;

21   uint16_t adc1; - данные АЦП1 - слтые доли вольта;

22   uint16_t c_counter3; - счетчик дискретного входа 3;

23   int16_t  ow_data; - данные датчика 1-wire N1, сотые доли градуса;

24   uint8_t  egts_flags; - флаги ЕГТС;

25   uint8_t  egts_src; - источник данных ЕГТС;

26 } t_l2b_gps_info;
```

Тип пакета 0x0041 - информация о GSM-сети

```
 1 typedef struct {

 2   char operator_name{8}; название оператора. ASCIIZ строка. Максимум
7 символов + 1 терминатор

 3   uint8_t active_sim_number; - номер активной СИМ-карты (1 или 2)

 4   uint8_t active_server_number; - номер активного сервера (1 или 2)

 5   uint8_t network_register_state; - состояние регистрации в GSM сети
(0,2 - нет регистрации 1 - регистрция в домашней сети; 5 - роуминг)

 6   uint8_t gsm_strength; - мощность GSM сигнала - от 0 до 31 ед

 7   uint8_t module_temperature; -  температура GSM модуля в градусах

 8   uint16_t operator_id; - ID сотового оператора

 9   char lac{5}; LAC - ASCIIZ строка

10   uint16_t mnc; - MNC

11   char cell_id{5}; - CELL_ID - ASCIIZ строка;

12   uint16_t mcc; - MCC

13   uint16_t bsic;- BSIC

14   uint16_t rxl; -  RXL
```

```
15   uint16_t rxq; - RXQ

16   uint16_t ta; - TA

17 } t_l2b_gsm_info;
```

Тип пакета 0x0042 - информация о датчиках

```
1 typedef struct {

2   uint16_t acc_max_x; - данные акселерометра по оси X

3   uint16_t acc_max_y; - данные акселерометра по оси Y

4   uint16_t acc_max_z; - данные акселерометра по оси Z

5   uint16_t fuel_data{4}; - данные 4х топливных датчиков RS485

6   uint32_t uptime_cnt; - время с момента запуска трекера, сек

7 } t_l2b_sd_line;
```

Тип пакета 0x0043 - отладочная информация

```
 1 typedef struct {

 2   uint32_t uptime_cnt; - время с момента запуска трекера, сек

 3   uint32_t wr_flash_address; - указатель адреса записи во flash
память

 4   uint32_t rd_flash_address; - указатель адреса чтения из flash-
памяти

 5   uint32_t rcc_csr; - флаги запуска трекера

 6   char imei{20}; - IMEI модема

 7   char iccid{21}; -  ICCID SIM карты

 8   uint16_t vcc; - напряжение питания по шине 3.3V, сотые доли вольта

 9 } t_l2b_debug_info_line;
```

Тип пакета 0x0044 - информация о датчиках 1-wire

```
1 typedef struct {

2   t_single_ow_data ow_data{4}; - данные 4х температурных датчоков

3   uint32_t i_button_fix_time; - время последнего считывания кнопки
iButton, unixtime

4   uint8_t i_button_id{8}; - ID кнопки - 8 байт

5 } t_l2b_ow_info_line;
```

Структура t_single_ow_data описана ниже:

```
1 typedef struct {

2   uint8_t ow_id; - идентификатор температурного датчика - последний
байт OW-ID датчика

3   int16_t ow_temp; - температура, сотые доли градуса

4 } t_single_ow_data;
```

Тип пакета 0x0045 - минимальная информация о маршруте

```
1 typedef struct {

2   uint8_t  ant_state;  - состояние GPS антенны

3   uint8_t  fix_type;   - тип фиксации спутников

4   uint8_t  sat_count;  - количество спутников в обзоре

5   uint16_t speed;      - текущая скорость - сотые км-ч

6   uint16_t course;     - текущий вектор перемещения

7   uint8_t egts_flags;  - флаги ЕГТС

8 } t_l2b_gps_info_min;
```

Тип пакета 0x004A - расширенная информация по GPS-модулю

```
1 typedef struct {

2   uint8_t sv_gps;       - количество видимых спутников GPS

3   uint8_t sv_glonass;   - количество видимых спутников GLONASS

4   uint8_t sc_gps;       - количество отслеживаемых спутников GPS

5   uint8_t sc_glonass;   - количество отслеживаемых спутников GLONASS

6   uint8_t snr_min;      - минимальное значение SNR отслеживаемых
спутников, dbHz

7   uint8_t snr_max;      - максимальное значение SNR отслеживаемых
спутников, dbHz

8 } t_l2b_gps_ex_info;
```

Тип пакета 0x004B - универсальный пакет GPS-lite

```
 1

 2 typedef struct {

 3   uint32_t flags1;       // Битовое поле определяющее присутствие
полей группы 0 в пакете

 4   uint8_t  ant_state;    // поля от сюда

 5   uint8_t  fix_type;
```

```
 6   uint8_t  sat_count;

 7   uint16_t speed;

 8   uint16_t course;

 9   uint8_t egts_flags;   // до сюда присутсвуют в пакете всегда

10   // Остальные поля присутствуют в пакете только если установлен
соответствущий бит в поле flags1 или flags2

11   uint32_t flags2;            // 0.0 - битовое поле определяющее
присутствие полей группы 1 в пакете

12   t_altitude_info altitude_info; // 0.1 - информация о высоте

13   uint16_t v1224;        // 0.2 - напряжение на входе питания
трекера, сотые вольта

14   uint16_t v_bat;        // 0.3 - напряжение на аккумуляторе трекера,
сотые вольта

15   uint16 t adc1;         // 0.4 - напряжение на аналоговом входе 1
трекера, сотые вольта

16

17   uint16_t freq1;        // 0.5 - частота, измеренная на частотном
входе 1, Гц

18   uint16_t freq2;        // 0.6 - частота, измеренная на частотном
входе 2, Гц

19   uint16_t c_counter1;   // 0.7 - количество импульсов, подсчитанных
на дискретном входе 1, шт

20   uint16_t c_counter2;   // 0.8 - количество импульсов, подсчитанных
на дискретном входе 2, шт

21   uint16_t c_counter3;   // 0.9 - количество импульсов, подсчитанных
на дискретном входе 3, шт

22

23   uint16_t fuel_data1;   // 0.10 - показания первого топливного
датчика на входе RS485/232

24   uint16_t fuel_data2;   // 0.11 - показания второго топливного
датчика на входе RS485

25   uint16_t fuel_data3;   // 0.12 - показания третьего топливного
датчика на входе RS485

26   uint16_t fuel_data4;   // 0.13 - показания четвертого топливного
датчика на входе RS485

27

28   uint8_t  stop_state;   // 0.14 - 1 - трекер находится в состоянии
покоя, 0 - трекер находится в движении
```

```
29   uint8_t  ign_state;    // 0.15 - 1 - зажигание включено, 0 -
зажигание выключено

30   uint8_t  d_state;      // 0.16 - состояние дискретных входов
(битовая маска)

31   uint8_t  egts_src;     // 0.17 - источник данных ЕГТС

32

33   uint16_t acc_data[3];  // 0.18, массив с показаниями акселерометра
по осям X,Y,Z

34

35   t_single_ow_data ow_data1; // 0.19 - номер и результаты замеров
температурного датчика 1 (см описание структуры t_single_ow_data ниже)

36   t_single_ow_data ow_data2; // 0.20 - номер и результаты замеров
температурного датчика 2

37   t single ow data ow data3; // 0.21 - номер и результаты замеров
температурного датчика 3

38   t_single_ow_data ow_data4; // 0.22 - номер и результаты замеров
температурного датчика 4

39

40   t_ibutton_data   ibut_data; // 0.23 - данные по кнопке iButton  (см
описание структуры t_ibutton_data ниже)

41

42   char operator_name[8];                    // 0.24 - ASCIIZ строка
- наименование текущего сотового оператора

43   uint8_t active_sim_number;                // 0.25 - номер текущей
сим-карты (0 - СИМ-карта номер 1, 1 - симкарта номер 2)

44   uint8_t active_server_number;             // 0.26 - номер текущего
сервера (0/1)

45   uint8_t network_register_state;           // 0.27 - состояние
регистрации в сети (0 - нет регистрации, 1 - регистрация в домашней
сети, 5 - регистрация в роуминге)

46   uint8_t gsm_strength;                     // 0.28 - мощность GSM
сигнала (0 - минимум, 31 - максимум)

47   uint8_t module_temperature;               // 0.29 - температура
GSM модуля

48   uint16_t operator_id;                     // 0.30 - operator_id

49   uint32_t uptime_cnt;                      // 0.31 - время работы
трекера от последнего включения / перезагрузки, сек

50

51   char lac[5];                              //1.0 - lac (ASCIIZ)
```

```
52   uint16_t mnc;                                //1.1 - mnc

53   char cell_id[5];                             //1.2 - cell_id
(ASCIIZ)

54   uint16_t mcc;                                //1.3 - mcc

55   uint16_t bsic;                               //1.4 - bsic

56   uint16_t rxl;                                //1.5 - rxl

57   uint16_t rxq;                                //1.6 - rzq

58   uint16_t ta;                                 //1.7 - ta

59   uint32_t distance_travalled;                 //1.8 - пробег ТС

60 } t_universal_data_full;

61

62 typedef struct {

63   int16_t altitude;

64   int16_t geoid_height;

65 } t_altitude_info;

66

67 typedef struct {

68   uint8_t ow_id;

69   int16_t ow_temp;

70 } t_single_ow_data;

71

72 typedef struct {

73   uint8_t  i_button_id[8];

74   uint32_t i_button_fix_time;

75 } t_ibutton_data;
```

Биты входящие в состав флагов flags1 и flags2 определяют наличие соответсвующих полей в структуре t_universal_data_full.
Например:
- бит 0 поля flags1 определяет присутствие поля 0.0 (flags2). Если flags2 отсутствует, то отсутствуют и все поля группы 1.* (lac, mnc, cell_id и т.д...).
- бит 1 поля flags1 определяет присутствие поля 0.1 (altitude_info);
- бит 2 поля flags1 определяет присутствие поля 0.2 (v1224);
- бит 3 поля flags1 определяет присутствие поля 0.3 (v_bat);

и т.д...