`acp245`

Version 1.1.0

EDANTECH

11 Sep 2009

Contents

Chapter 1

ACP245 message library.

1.1 Introduction

This library provides a portable implementation of the ACP245 protocol messages.

To use the library, you should include the acp245.h header file and link agains the provided library binaries.

The library does not include any network related code, only functions to read, write and validate ACP245 messages. An ACP245 server or client can be built by using this library to process the binary messages.

The main functions are:

- acp_msg_read_data : reads an ACP message from a byte array.
- acp_msg_write_data : writes an ACP message to a byte array.

Both functions operate on an acp_msg structure which includes a the application ID and message type of the ACP message. Based on that application ID and type, different fields are available on the data field of the acp_msg structure.

The following code illustrates a simple use of the API to read and write an empty Alarm Keepalive:

```
#include <stdio.h>
#include <stdlib.h>
#include "acp245.h"
int main(int argc, char** argv) {
     u8 buf[256];
     u32 readed;
     u32 written;
     acp_msg msg;
     acp_msg msg_read;
     e_ret rc;

     acp_msg_init(&msg, ACP_APP_ID_ALARM, ACP_MSG_TYPE_ALARM_KA);

     rc = acp_msg_write_data(buf, 256, &written, &msg);
     if (ACP_MSG_OK == rc) {
         printf("Written OK.\n");
     }

     rc = acp_msg_read_data(buf, written, &readed, &msg_read);
     if(ACP_MSG_OK == rc) {
         printf("ACP Message Application Id is: %x\n", msg_read.hdr.app_id);
         printf("ACP Message Type is: %x\n", msg_read.hdr.type);
     }
```

```
        getchar();
 }
```

Structures and functions make reference to the following documents:

- [ACP245]: ACP 245 v1.2.2, Protocol Specification, 14/08/09.

  `http://www.denatran.gov.br/download/ACP%20245%20V%201.2.2%2014_-08_09%2013_46.pdf.` Also included in project documentation.

- [ACP] ACP v. 3.0.1, March 2000. Included in project documentation.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 acp_el_apn_cfg Struct Reference

APN Configuration Element.

Data Fields

- acp_el_presence **present**
- ascii ∗ **address**
- ascii ∗ **login**
- ascii ∗ **password**

4.1.1 Detailed Description

APN Configuration Element.

See also:

Section 6.5.1.2 of [ACP245]

Definition at line 380 of file acp_el.h.

4.2 acp_el_breakdown_status Struct Reference

Breakdown Status Element.

Data Fields

- acp_el_presence **present**
- u8 **source** [ACP_EL_BREAKDOWN_STATUS_MAX_SOURCE]
- u8 **source_cnt**
- u8 **sensor**
- u32 **data_len**
- u8 $*$ **data**

4.2.1 Detailed Description

Breakdown Status Element.

See also:

Section 3.9 of [ACP245]

Definition at line 315 of file acp_el.h.

4.3 acp_el_ctrl_func Struct Reference

Control Function Element.

Data Fields

- acp_el_ctrl_entity **entity_id**
- bool **transmit_present**
- acp_el_transmit_unit **transmit_unit**
- u8 **transmit_interval**

4.3.1 Detailed Description

Control Function Element.

See also:

Section 3.6 of [ACP245]

Definition at line 193 of file acp_el.h.

4.4 acp_el_dead_reck Struct Reference

Dead Reckoning Element.

Data Fields

- acp_el_presence **present**
- s32 **lat**
- s32 **lon**

4.4.1 Detailed Description

Dead Reckoning Element.

See also:

Section 3.8.16 of [ACP245]

Definition at line 272 of file acp_el.h.

4.5 acp_el_error Struct Reference

Error Element.

Data Fields

- u8 **code**

4.5.1 Detailed Description

Error Element.

See also:

Section 3.5 of [ACP245]

Definition at line 185 of file acp_el.h.

4.6 acp_el_func_cmd Struct Reference

Function Command Element.

Data Fields

- u8 **cmd**
- acp_el_raw_data **raw_data**

4.6.1 Detailed Description

Function Command Element.

See also:

Section 3.7 of [ACP245]

Definition at line 216 of file acp_el.h.

4.7 acp_el_gps_raw_data Struct Reference

GPS Raw Data Element.

Data Fields

- acp_el_presence **present**
- u8 **flg1**
- u8 **flg2**
- u8 **area_type**
- u8 **location_type**
- u32 **time_diff**
- s32 lon

  longitude in milliarcsecond (1/3600000 degrees)

- s32 lat

  latitude in milliarcsecond (1/3600000 degrees)

- u16 alt

  altitude in meters

- u8 **pos_uncert**
- bool hdop

  3.8.11.1, 0 = use K, 1 = use DOP

- u8 **head_uncert**
- u8 heading

  heading in multiples of 15 degrees

- u8 **dist_unit**
- u8 **time_unit**
- u8 velocity

  velocity, unit given by dist_unit flag

- u8 **satellite_cnt**
- bool **satellites_id_present**
- u8 **satellites_id** [ACP_EL_GPS_RAW_DATA_SAT_MAX]

4.7.1 Detailed Description

GPS Raw Data Element.

See also:

Section 3.8.1 of [ACP245]

Definition at line 231 of file acp_el.h.

4.8 acp_el_info_type Struct Reference

Information Type Element.

Data Fields

- acp_el_presence **present**
- u8 **type**
- u32 **data_len**
- u8 ∗ **data**

4.8.1 Detailed Description

Information Type Element.

See also:

Section 3.10 of [ACP245]

Definition at line 328 of file acp_el.h.

4.9 acp_el_loc_delta Struct Reference

Location Delta Coding Element.

Data Fields

- acp_el_presence **present**
- u8 **delta_cnt**
- struct acp_el_loc_delta::latlon **delta** [ACP_EL_LOC_DELTA_MAX]

4.9.1 Detailed Description

Location Delta Coding Element.

See also:

Section 3.8.17 of [ACP245]

Definition at line 282 of file acp_el.h.

4.10 acp_el_location Struct Reference

Location Element.

Data Fields

- acp_el_gps_raw_data **curr_gps**
- acp_el_gps_raw_data **prev_gps**
- acp_el_dead_reck **dead_reck**
- acp_el_loc_delta **loc_delta**

4.10.1 Detailed Description

Location Element.

See also:

Section 3.8 of [ACP245]

Definition at line 295 of file acp_el.h.

4.11 acp_el_raw_data Struct Reference

Raw Data Element.

Data Fields

- acp_el_presence **present**
- u32 **data_len**
- u8 ∗ **data**

4.11.1 Detailed Description

Raw Data Element.

See also:

Section 3.7.2 of [ACP245]

Definition at line 206 of file acp_el.h.

4.12 acp_el_server_cfg Struct Reference

Server Configuration Element.

Data Fields

- acp_el_presence **present**
- u32 **server_1**
- u16 **port_1**
- u32 **server_2**
- u16 **port_2**
- u8 **proto_id**

4.12.1 Detailed Description

Server Configuration Element.

See also:

Section 6.5.1.3 of [ACP245]

Definition at line 391 of file acp_el.h.

4.13 acp_el_tcu_data Struct Reference

TCU Data Element.

Data Fields

- u8 **cnt**
- acp_el_tcu_data_item ∗ **items**

4.13.1 Detailed Description

TCU Data Element.

See also:

Section 3.11 of [ACP245]

Definition at line 350 of file acp_el.h.

4.14 acp_el_tcu_data_error Struct Reference

TCU Data Error Element.

Data Fields

- u8 **cnt**
- acp_el_tcu_data_error_item ∗ **items**

4.14.1 Detailed Description

TCU Data Error Element.

See also:

Section 3.12 of [ACP245]

Definition at line 371 of file acp_el.h.

4.15 acp_el_tcu_data_error_item Struct Reference

TCU Data Error Element Item.

Data Fields

- u16 **type**
- u8 **data_len**
- u8 ∗ **data**
- acp_el_error **error**

4.15.1 Detailed Description

TCU Data Error Element Item.

See also:

Section 3.12 of [ACP245]

Definition at line 359 of file acp_el.h.

4.16 acp_el_tcu_data_item Struct Reference

TCU Data Element Item.

Data Fields

- u16 **type**
- u8 **data_len**
- u8 $*$ **data**

4.16.1 Detailed Description

TCU Data Element Item.

See also:

Section 3.11 of [ACP245]

Definition at line 339 of file acp_el.h.

4.17 acp_el_tcu_desc Struct Reference

TCU Descriptor Element.

Data Fields

- acp_el_presence **present**
- u8 **device_id**
- bool **is_str**
- union {
  u8 **id**
  ascii * **str**
  } **version**

4.17.1 Detailed Description

TCU Descriptor Element.

See also:

Section 3.3 of [ACP245]

Definition at line 148 of file acp_el.h.

4.18 acp_el_timestamp Struct Reference

Timestamp Element.

Data Fields

- u16 **year**
- u8 **month**
- u8 **day**
- u8 **hour**
- u8 **minute**
- u8 **second**

4.18.1 Detailed Description

Timestamp Element.

See also:

Section 3.2 of [ACP245]

Definition at line 135 of file acp_el.h.

4.19 acp_el_vehicle_desc Struct Reference

Vehicle Descriptor Element.

Data Fields

- acp_el_presence **present**
- u8 **flg1**
- u8 **flg2**
- u8 **lang**
- u8 **model_year**
- ascii * **vin**
- u8 **tcu_serial_len**
- acp_ie_any **tcu_serial**
- ascii * **license_plate**
- ascii * **vehicle_color**
- ascii * **vehicle_model**
- ascii * **imei**
- ascii * **iccid**
- u8 **auth_key_len**
- u8 * **auth_key**

4.19.1 Detailed Description

Vehicle Descriptor Element.

See also:

Section 3.4 of [ACP245]

Definition at line 162 of file acp_el.h.

4.20 acp_el_version Struct Reference

Version Element.

Data Fields

- acp_el_presence **present**
- u8 **car_manufacturer**
- u8 **tcu_manufacturer**
- u8 **major_hard_rel**
- u8 **major_soft_rel**

4.20.1 Detailed Description

Version Element.

See also:

Section 3.1 of [ACP245]

Definition at line 123 of file acp_el.h.

4.21 acp_hdr Struct Reference

Message Header.

Data Fields

- acp_msg_app_id **app_id**
- bool **test**
- acp_msg_type **type**
- u8 **version**
- u8 **msg_ctrl**
- acp_msg_hdr_prio **msg_prio**

4.21.1 Detailed Description

Message Header.

See also:

Section 4 of [ACP245]

Definition at line 150 of file acp_msg.h.

4.22 acp_ie_any Struct Reference

An information element of undetermined type.

Data Fields

- bool present

  If TRUE, the element has been included on the message, if FALSE, the element is not included because the message was truncated or was explicitely excluded with a control flag.

- u8 **id**
- u16 **len**
- union {
  u8 ∗ **bin**
  ascii ∗ **str**
  } data

  Data of the information element.

4.22.1 Detailed Description

An information element of undetermined type.

This structure is used to represent an element whose type has not be constrained by the ACP245 specification, and therefore can be represented with different data types.

Definition at line 83 of file acp_ie.h.

4.22.2 Field Documentation

4.22.2.1 union { ... } acp_ie_any::data

Data of the information element.

str will be valid only if id == ACP_IE_ISO_8859_1 or id == ACP_IE_PACKED_DEC, otherwise bin will have the byte array representing the information element data.

4.22.2.2 bool acp_ie_any::present

If TRUE, the element has been included on the message, if FALSE, the element is not included because the message was truncated or was explicitely excluded with a control flag.

Definition at line 87 of file acp_ie.h.

4.23 acp_msg Struct Reference

ACP245 Message Structure.

Data Fields

- acp_hdr hdr

  Message header.

- union {
  acp_msg_prov_upd **prov_upd**
  acp_msg_prov_reply **prov_reply**
  acp_msg_cfg_activation **cfg_activation**
  acp_msg_cfg_upd_245 **cfg_upd_245**
  acp_msg_cfg_reply **cfg_reply**
  acp_msg_cfg_reply_245 **cfg_reply_245**
  acp_msg_func_cmd **func_cmd**
  acp_msg_func_status **func_status**
  acp_msg_track_cmd **track_cmd**
  acp_msg_track_pos **track_pos**
  acp_msg_track_reply **track_reply**
  acp_msg_alarm_notif **alarm_notif**
  acp_msg_alarm_reply **alarm_reply**
  acp_msg_alarm_pos **alarm_pos**
  acp_msg_alarm_ka **alarm_ka**
  acp_msg_alarm_ka_reply **alarm_ka_reply**
  } data

  Message payload.

4.23.1 Detailed Description

ACP245 Message Structure.

Definition at line 372 of file acp_msg.h.

4.23.2 Field Documentation

4.23.2.1 union { ... } acp_msg::data

Message payload.

Before accessing this field, check that the message application ID and message type on the header match the type of the payload field that you will access, otherwise it's value is undefined.

4.24 acp_msg_alarm_ka Struct Reference

Message Keep Alive (TCU to SO).

Data Fields

- acp_el_vehicle_desc **vehicle_desc**

4.24.1 Detailed Description

Message Keep Alive (TCU to SO).

See also:

Section 9.5 of [ACP245]

Definition at line 359 of file acp_msg.h.

4.25 acp_msg_alarm_ka_reply Struct Reference

Message Keep Alive Reply (SO to TCU).

Data Fields

- acp_el_vehicle_desc **vehicle_desc**

4.25.1 Detailed Description

Message Keep Alive Reply (SO to TCU).

See also:

Section 9.6 of [ACP245]

Definition at line 367 of file acp_msg.h.

4.26 acp_msg_alarm_notif Struct Reference

Theft Alarm Notification (From TCU to SO).

Data Fields

- acp_el_version **version**
- acp_el_timestamp **timestamp**
- acp_el_location **location**
- acp_el_vehicle_desc **vehicle_desc**
- acp_el_breakdown_status **breakdown_status**
- acp_el_info_type **info_type**

4.26.1 Detailed Description

Theft Alarm Notification (From TCU to SO).

See also:

Section 9.2 of [ACP245]

Definition at line 321 of file acp_msg.h.

4.27 acp_msg_alarm_pos Struct Reference

Vehicle Position Message (TCU to SO).

Data Fields

- acp_el_version **version**
- acp_el_timestamp **timestamp**
- acp_el_location **location**
- acp_el_vehicle_desc **vehicle_desc**
- acp_el_breakdown_status **breakdown_status**
- acp_el_info_type **info_type**

4.27.1 Detailed Description

Vehicle Position Message (TCU to SO).

See also:

Section 9.4 of [ACP245]

Definition at line 346 of file acp_msg.h.

4.28 acp_msg_alarm_reply Struct Reference

Theft Alarm Reply (From SO to TCU).

Data Fields

- acp_el_version **version**
- u8 **confirmation**
- u8 **transmit_unit**
- u8 **ctrl_flg**
- acp_el_error **error**

4.28.1 Detailed Description

Theft Alarm Reply (From SO to TCU).

See also:

Section 9.3 of [ACP245]

Definition at line 334 of file acp_msg.h.

4.29 acp_msg_cfg_activation Struct Reference

Configuration TCU Service Activation/Deactivation Message ACP 245 (From SO to TCU).

Data Fields

- acp_el_apn_cfg **apn_cfg**
- acp_el_server_cfg **server_cfg**
- u8 **ctrl_byte**
- acp_el_vehicle_desc **vehicle_desc**

4.29.1 Detailed Description

Configuration TCU Service Activation/Deactivation Message ACP 245 (From SO to TCU).

See also:

Section 6.5 of [ACP245]

Definition at line 249 of file acp_msg.h.

4.30 acp_msg_cfg_reply Struct Reference

Configuration Reply (From TCU to SO).

Data Fields

- acp_el_version **version**
- u8 **target_app_id**
- u8 **appl_flg**
- u8 **ctrl_flg**
- u8 **status**
- u8 **tcu_resp**
- acp_el_error **error**
- acp_el_vehicle_desc **vehicle_desc**

4.30.1 Detailed Description

Configuration Reply (From TCU to SO).

See also:

Section 6.3 of [ACP245]

Definition at line 216 of file acp_msg.h.

4.31 acp_msg_cfg_reply_245 Struct Reference

Configuration Reply #2 ACP 245 (From TCU to SO).

Data Fields

- acp_el_version **version**
- u8 **target_app_id**
- u8 **appl_flg**
- u8 **ctrl_flg**
- u8 **status**
- u8 **tcu_resp**
- acp_el_tcu_data_error **error**
- acp_el_vehicle_desc **vehicle_desc**

4.31.1 Detailed Description

Configuration Reply #2 ACP 245 (From TCU to SO).

See also:

Section 6.4 of [ACP245]

Definition at line 232 of file acp_msg.h.

4.32 acp_msg_cfg_upd_245 Struct Reference

Configuration Update Message #2 ACP 245 (From SO to TCU).

Data Fields

- acp_el_version **version**
- u8 **target_app_id**
- u8 **appl_flg**
- u8 **ctrl_flg1**
- u8 **ctrl_flg2**
- acp_el_timestamp **start_time**
- acp_el_timestamp **end_time**
- acp_el_timestamp **grace_time**
- acp_el_vehicle_desc **vehicle_desc**
- acp_el_tcu_desc **tcu_desc**
- acp_el_tcu_data **tcu_data**

4.32.1 Detailed Description

Configuration Update Message #2 ACP 245 (From SO to TCU).

See also:

Section 6.2 of [ACP245]

Definition at line 196 of file acp_msg.h.

4.33 acp_msg_func_cmd Struct Reference

Vehicle Function Command (From SO to TCU).

Data Fields

- acp_el_version **version**
- acp_el_ctrl_func **ctrl_func**
- acp_el_func_cmd **func_cmd**
- acp_el_vehicle_desc **vehicle_desc**

4.33.1 Detailed Description

Vehicle Function Command (From SO to TCU).

See also:

Section 7.2 of [ACP245]

Definition at line 260 of file acp_msg.h.

4.34 acp_msg_func_status Struct Reference

Vehicle Function Status (From TCU to SO).

Data Fields

- acp_el_version **version**
- acp_el_ctrl_func **ctrl_func**
- acp_el_func_cmd **func_status**
- acp_el_error **error**
- acp_el_vehicle_desc **vehicle_desc**

4.34.1 Detailed Description

Vehicle Function Status (From TCU to SO).

See also:

Section 7.3 of [ACP245]

Definition at line 271 of file acp_msg.h.

4.35 acp_msg_prov_reply Struct Reference

Provision Reply Message #1 (From TCU to SO).

Data Fields

- acp_el_version **version**
- u8 **target_app_id**
- u8 **appl_flg**
- u8 **ctrl_flg1**
- u8 **status**
- u8 **tcu_resp**
- acp_el_error **error**
- acp_el_vehicle_desc **vehicle_desc**

4.35.1 Detailed Description

Provision Reply Message #1 (From TCU to SO).

See also:

Section 5.3 of [ACP245]

Definition at line 181 of file acp_msg.h.

4.36 acp_msg_prov_upd Struct Reference

Provision Update Message #1 (From SO to TCU).

Data Fields

- acp_el_version **version**
- u8 **target_app_id**
- u8 **appl_flg**
- u8 **ctrl_flg1**
- u8 **ctrl_flg2**
- acp_el_timestamp **start_time**
- acp_el_timestamp **end_time**
- acp_el_timestamp **grace_time**
- acp_el_tcu_desc **tcu_desc**
- acp_el_vehicle_desc **vehicle_desc**

4.36.1 Detailed Description

Provision Update Message #1 (From SO to TCU).

See also:

Section 5.2 of [ACP245]

Definition at line 164 of file acp_msg.h.

4.37 acp_msg_track_cmd Struct Reference

Vehicle Tracking Command (From SO to TCU).

Data Fields

- acp_el_version **version**
- acp_el_ctrl_func **ctrl_func**
- acp_el_func_cmd **func_cmd**
- acp_el_vehicle_desc **vehicle_desc**

4.37.1 Detailed Description

Vehicle Tracking Command (From SO to TCU).

See also:

Section 8.2 of [ACP245]

Definition at line 285 of file acp_msg.h.

4.38 acp_msg_track_pos Struct Reference

Vehicle Position Message (From TCU to SO).

Data Fields

- acp_el_version **version**
- acp_el_timestamp **timestamp**
- acp_el_location **location**
- acp_el_vehicle_desc **vehicle_desc**
- acp_el_breakdown_status **breakdown_status**
- acp_el_info_type **info_type**

4.38.1 Detailed Description

Vehicle Position Message (From TCU to SO).

See also:

Section 8.3 of [ACP245]

Definition at line 296 of file acp_msg.h.

4.39 acp_msg_track_reply Struct Reference

Vehicle Reply Message (From SO to TCU).

Data Fields

- acp_el_version **version**
- u8 **confirmation**
- u8 **transmit_unit**
- u8 **ctrl_flg**
- acp_el_error **error**

4.39.1 Detailed Description

Vehicle Reply Message (From SO to TCU).

See also:

Section 8.4 of [ACP245]

Definition at line 309 of file acp_msg.h.

4.40 SHA256Context Struct Reference

SHA 256 context.

Data Fields

- u32 **totalLength**
- u32 **hash** [SHA256_HASH_WORDS]
- u32 **bufferLength**
- union {
    u32 **words** [16]
    u8 **bytes** [64]
  } **buffer**

4.40.1 Detailed Description

SHA 256 context.

Internal representation, API users should not access these fields.

Definition at line 58 of file sha256.h.

Chapter 5

File Documentation

5.1 acp245.h File Reference

ACP 245 message library main header.

Defines

- #define ACP245_VERSION 1.1.0

  ACP245 library version number.

5.1.1 Detailed Description

ACP 245 message library main header.

Include this header to use the library in your own code.

Date:

03/13/2009 01:42:35 PM

Author:

Edantech

See also:

acp_msg.h
acp_el.h

Definition in file acp245.h.

5.2 acp_el.h File Reference

ACP 245 information element description and processing functions.

Data Structures

- struct acp_el_version

  Version Element.

- struct acp_el_timestamp

  Timestamp Element.

- struct acp_el_tcu_desc

  TCU Descriptor Element.

- struct acp_el_vehicle_desc

  Vehicle Descriptor Element.

- struct acp_el_error

  Error Element.

- struct acp_el_ctrl_func

  Control Function Element.

- struct acp_el_raw_data

  Raw Data Element.

- struct acp_el_func_cmd

  Function Command Element.

- struct acp_el_gps_raw_data

  GPS Raw Data Element.

- struct acp_el_dead_reck

  Dead Reckoning Element.

- struct acp_el_loc_delta

  Location Delta Coding Element.

- struct acp_el_location

  Location Element.

- struct acp_el_breakdown_status

  Breakdown Status Element.

- struct acp_el_info_type

  Information Type Element.

- struct acp_el_tcu_data_item

TCU Data Element Item.

- struct acp_el_tcu_data

  TCU Data Element.

- struct acp_el_tcu_data_error_item

  TCU Data Error Element Item.

- struct acp_el_tcu_data_error

  TCU Data Error Element.

- struct acp_el_apn_cfg

  APN Configuration Element.

- struct acp_el_server_cfg

  Server Configuration Element.

Defines

- #define ACP_EL_GPS_RAW_DATA_SAT_MAX (16)

  Maximum number of satellite IDs stored by the library.

- #define ACP_EL_LOC_DELTA_MAX (10)

  Maximum number of location delta items stored by the library.

- #define ACP_MORE_FLG 0x80

  More flag.

- #define ACP_LOCATION_WGS_84 0

  Location Type Coding.

- #define ACP_MSG_CFG_PROTO_ID_ACP245 (0)

  ACP 245 Protocol ID.

Number of allowed breakdown status source fields

- #define ACP_EL_BREAKDOWN_STATUS_MAX_SOURCE (5)

  Maximum number of breakdown status elements.

- #define ACP_EL_BREAKDOWN_STATUS_MIN_SOURCE (1)

  Minimum number of breakdown status elements.

Vehicle Descriptor Flags

See also:

Section 3.4.1 of [ACP245]

- #define **ACP_VEHICLE_DESC_FLG_ADDL_FLG** 0x80

- #define **ACP_VEHICLE_DESC_FLG1_LANG** 0x40
- #define **ACP_VEHICLE_DESC_FLG1_VIN** 0x20
- #define **ACP_VEHICLE_DESC_FLG1_TCU_SERIAL** 0x10
- #define **ACP_VEHICLE_DESC_FLG1_VEHICLE_COLOR** 0x08
- #define **ACP_VEHICLE_DESC_FLG1_VEHICLE_MODEL** 0x04
- #define **ACP_VEHICLE_DESC_FLG1_LICENSE_PLATE** 0x02
- #define **ACP_VEHICLE_DESC_FLG1_IMEI** 0x01
- #define **ACP_VEHICLE_DESC_FLG2_MODEL_YEAR** 0x40
- #define **ACP_VEHICLE_DESC_FLG2_SIM_CARD_ID** 0x20
- #define **ACP_VEHICLE_DESC_FLG2_AUTH_KEY** 0x10

Area Location Status Flag 1

See also:

Section 3.8.3 of [ACP245]

- #define **ACP_LOCATION_FLG1_NO_3D_FIX** 0x20
- #define **ACP_LOCATION_FLG1_NO_2D_FIX** 0x10
- #define **ACP_LOCATION_FLG1_INVALID_POS** 0x08
- #define **ACP_LOCATION_FLG1_DIFF_GPS** 0x04
- #define **ACP_LOCATION_FLG1_INVALID_HEAD** 0x02
- #define **ACP_LOCATION_FLG1_ALMANAC_BAD** 0x01

Area Location Status Flag 2

See also:

Section 3.8.4 of [ACP245]

- #define **ACP_LOCATION_FLG2_NEW_GPS_DATA** 0x20
- #define **ACP_LOCATION_FLG2_HEAD_MASK** 0x07
- #define **ACP_LOCATION_NORTH** 0
- #define **ACP_LOCATION_NORTH_EAST** 1
- #define **ACP_LOCATION_EAST** 2
- #define **ACP_LOCATION_SOUTH_EAST** 3
- #define **ACP_LOCATION_SOUTH** 4
- #define **ACP_LOCATION_SOUTH_WEST** 5
- #define **ACP_LOCATION_WEST** 6
- #define **ACP_LOCATION_NORTH_WEST** 7

Area Type

See also:

Section 3.8.5 of [ACP245]

- #define **ACP_LOCATION_POINT_1_MILLIARC** 0
- #define **ACP_LOCATION_POINT_100_MILLIARC** 1

Distance Flag

See also:

Section 3.8.14 of [ACP245]

- #define **ACP_LOCATION_DIST_UNIT_ND** 0
- #define **ACP_LOCATION_DIST_UNIT_KM** 1
- #define **ACP_LOCATION_DIST_UNIT_MI** 2

Time Flag

See also:

Section 3.8.15 of [ACP245]

- #define **ACP_LOCATION_TIME_UNIT_SECONDS** 0
- #define **ACP_LOCATION_TIME_UNIT_MINUTES** 1
- #define **ACP_LOCATION_TIME_UNIT_HOURS** 2

Device ID

See also:

Section 3.3.1 of [ACP245]

- #define **ACP_EL_TCU_DEVICE_ID_TCU_HARD_VER** 1
- #define **ACP_EL_TCU_DEVICE_ID_TCU_MANUFACT** 2
- #define **ACP_EL_TCU_DEVICE_ID_TCU_SOFT_VER** 3
- #define **ACP_EL_TCU_DEVICE_ID_TCU_CAN_VER** 4
- #define **ACP_EL_TCU_DEVICE_ID_ACP_TRANP_VER** 5
- #define **ACP_EL_TCU_DEVICE_ID_ACP_APP_VER** 6

Valid Error Codes

See also:

Section 3.5.1 of [ACP245]

- #define **ACP_ERR_OK** 0
- #define **ACP_ERR_SERVICE_UNAVAILABLE** 1
- #define **ACP_ERR_INCORRECT_APP** 2
- #define **ACP_ERR_UNKNOWN_VERSION** 3
- #define **ACP_ERR_UNKNOWN_MSG_TYPE** 4
- #define **ACP_ERR_UNKNOWN_DATA_IN_MSG** 5
- #define **ACP_ERR_UNKNOWN_TRANSPORT_VER** 6
- #define **ACP_ERR_DATA_ERROR** 7
- #define **ACP_ERR_SEC_VIOLATION** 8
- #define **ACP_ERR_NO_ACC_NO_CUSTOMER** 9
- #define **ACP_ERR_NO_ACC_NO_SERVICE** 10
- #define **ACP_ERR_NO_ACC_AUTH_FAIL** 11
- #define **ACP_ERR_NO_ACC_OTHER** 12
- #define **ACP_ERR_INVALID_SES_ID** 13
- #define **ACP_ERR_UNSUPPORTED_LANG** 15
- #define **ACP_ERR_PROV_UPDATE_MISMATCH** 16
- #define **ACP_ERR_PROV_SIM_ID_MISMATCH** 17
- #define **ACP_ERR_PROV_UNABLE_TO_PROC** 18
- #define **ACP_ERR_GENERAL** 19
- #define **ACP_ERR_NO_ACC_SIM** 20
- #define **ACP_ERR_EEPROM** 21
- #define **ACP_ERR_INVALID_PHONE** 22
- #define **ACP_ERR_VIN_MISMATCH** 23
- #define **ACP_ERR_VEHICLE_MISMATCH** 24
- #define **ACP_ERR_PROV_TOO_MANY_TARGETS** 25
- #define **ACP_ERR_MISSING_PHONE** 26
- #define **ACP_ERR_INVALID_ACT** 27
- #define **ACP_ERR_INVALID_DEACT** 28
- #define **ACP_ERR_BUFF_OVERFLOW** 29

Function Command

See also:

Section 3.7.1 of [ACP245]

- #define **ACP_FUNC_CMD_PERMIT** 0
- #define **ACP_FUNC_CMD_REJECT** 1
- #define **ACP_FUNC_CMD_ENABLE** 2
- #define **ACP_FUNC_CMD_DISABLE** 3
- #define **ACP_FUNC_CMD_REQUEST** 4

Function Status

See also:

Section 3.7.1 of [ACP245]

- #define **ACP_FUNC_STATE_PERMITTED** 0
- #define **ACP_FUNC_STATE_REJECTED** 1
- #define **ACP_FUNC_STATE_ENABLED** 2
- #define **ACP_FUNC_STATE_DISABLED** 3
- #define **ACP_FUNC_STATE_COMPLETED** 4

Breakdown Source 1

See also:

Section 3.9.1 of [ACP245]

- #define **ACP_BKD_MANUALLY_ACTIVATED** 0x40
- #define **ACP_BKD_VEHICLE_ROLLED** 0x20
- #define **ACP_BKD_AIR_BAG_ACTIVATED** 0x10
- #define **ACP_BKD_CRASH_SENSOR_ACTIVATED** 0x08
- #define **ACP_BKD_FLOATING_CAR_DATA_INPUT** 0x04
- #define **ACP_BKD_TOW_TRUCK_NEEDED** 0x02
- #define **ACP_BKD_THEFT_ALARM** 0x01

Breakdown Source 2

See also:

Section 3.9.1 of [ACP245]

- #define **ACP_BKD_VEHICLE_ON** 0x40
- #define **ACP_BKD_VEHICLE_OFF** 0x20
- #define **ACP_BKD_VEHICLE_MOVED** 0x10
- #define **ACP_BKD_OTHER_SENSOR_ACT** 0x08
- #define **ACP_BKD_RE_SEND_POS_TCU** 0x04
- #define **ACP_BKD_RE_SEND_POS_SO** 0x02
- #define **ACP_BKD_UNAUTH_VEHICLE_MOVE** 0x01

Breakdown Source 3

See also:

Section 3.9.1 of [ACP245]

- #define **ACP_BKD_SIREN_ON** 0x40
- #define **ACP_BKD_SIREN_OFF** 0x20
- #define **ACP_BKD_MAIN_BATT_RECONN** 0x10
- #define **ACP_BKD_MAIN_BARR_DISCONN** 0x08
- #define **ACP_BKD_PANIC_ON** 0x04

- #define **ACP_BKD_BLOCKING_ON** 0x02
- #define **ACP_BKD_BLOCKING_OFF** 0x01

Breakdown Sensor

See also:

Section 3.9.2 of [ACP245]

- #define **ACP_BKD_SENSOR_ADDL_FLG** 0x80
- #define **ACP_BKD_SENSOR_ROLLOVER** 0x40
- #define **ACP_BKD_SENSOR_FRONT** 0x20
- #define **ACP_BKD_SENSOR_REAR** 0x10
- #define **ACP_BKD_SENSOR_SIDE** 0x08
- #define **ACP_BKD_SENSOR_ALARM** 0x04
- #define **ACP_BKD_STATUS** 0x01

Information Type

See also:

Section 3.10.1 of [ACP245]

- #define **ACP_IT_VERBAL_INFO** 1
- #define **ACP_IT_STOCK_INFO** 2
- #define **ACP_IT_TRAVEL_ROUTE_INFO** 3
- #define **ACP_IT_HOTEL_INFO** 4
- #define **ACP_IT_TRAFFIC_INFO_VERBAL** 5
- #define **ACP_IT_TRAFFIC_INFO_AUTOMATED** 6
- #define **ACP_IT_ASCII_STRING** 7
- #define **ACP_IT_POI** 8
- #define **ACP_IT_CARGO** 9
- #define **ACP_IT_PRIVATE** 10
- #define **ACP_IT_ENVIRONMENTAL** 11
- #define **ACP_IT_TIMESTAMP** 12
- #define **ACP_IT_COUNTRY_CODE** 13
- #define **ACP_IT_MENU_BUTTON** 14

Version Field

See also:

Section 4.1.6 of [ACP245]

- #define **ACP_VER_1_2** 0
- #define **ACP_VER_1_2_1** 1
- #define **ACP_VER_1_2_2** 2

ApplFlg1 (appl_flg)

See also:

Section 5.2.1.3.2 of [ACP245]

- #define **ACP_MSG_PROV_NO_CHANGE** 0
- #define **ACP_MSG_PROV_ACTIVATE** 1
- #define **ACP_MSG_PROV_DEACTIVATE** 2
- #define **ACP_MSG_PROV_CHANGE** 3

ControlFlag1 (ctrl_flg1)

See also:

Section 5.2.1.3.3 of [ACP245]

- #define **ACP_MSG_PROV_ADDL_FLG_MASK** 0x20
- #define **ACP_MSG_PROV_GRACE_TIME_MASK** 0x10
- #define **ACP_MSG_PROV_START_TIME_MASK** 0x08
- #define **ACP_MSG_PROV_END_TIME_MASK** 0x04
- #define **ACP_MSG_PROV_VEHICLE_DESC_MASK** 0x02
- #define **ACP_MSG_PROV_USE_COMMIT_MASK** 0x01

ControlFlag2 (ctrl_flg1)

See also:

Section 5.2.1.3.4 of [ACP245]

- #define **ACP_MSG_PROV_USE_PROFILE**(b) (0==((b&0x70)>>4))
- #define **ACP_MSG_PROV_USE_SAMPLE**(b) (1==((b&0x70)>>4))
- #define **ACP_MSG_PROV_NUM_SAMPLES_MASK** 0x08
- #define **ACP_MSG_PROV_NO_SAMPLE_UNIT**(b) (0==((b&0x07)))
- #define **ACP_MSG_PROV_SAMPLES_IN_MIN**(b) (1==((b&0x07)))
- #define **ACP_MSG_PROV_SAMPLES_IN_KM**(b) (2==((b&0x07)))

StatusFlag1 (status)

See also:

Section 5.3.1.3.1 of [ACP245]

- #define **ACP_MSG_PROV_STATUS_ALREADY_PROV** 0
- #define **ACP_MSG_PROV_STATUS_NOT_PROV** 1
- #define **ACP_MSG_PROV_STATUS_SEE_ERROR** 2

Control Value.

See also:

Section 6.5.1.4.1 of [ACP245]

- #define **ACP_MSG_CFG_CTRL_VALUE_DEACTIVATE** (0)
- #define **ACP_MSG_CFG_CTRL_VALUE_ACTIVATE** (1)

TCU Response Flag.

See also:

Section 5.3.1.3.2 of [ACP245]

- #define **ACP_MSG_PROV_TCU_INIT_MODE** 1
- #define **ACP_MSG_PROV_TCU_RESP_TO_UPD** 2
- #define **ACP_MSG_PROV_TCU_RESP_TO_COMMIT** 3

Confirmation.

See also:

Section 8.4.1.3.1 of [ACP245]

- #define **ACP_MSG_TRACK_CONFIRM_ADDL_FLG** 0x08
- #define **ACP_MSG_TRACK_CONFIRM_ACCEPTED** 0x04
- #define **ACP_MSG_TRACK_CONFIRM_TURN_SPEAKER** 0x02
- #define **ACP_MSG_TRACK_CONFIRM_PROCESS_START** 0x01

EcallControlFlag2

See also:

Section 8.4.1.3.3 of [ACP245]

- #define **ACP_MSG_TRACK_CTRL_ADDL_FLG** 0x80
- #define **ACP_MSG_TRACK_CTRL_CANCEL_ALARM** 0x40
- #define **ACP_MSG_TRACK_CTRL_RE_SEND_REQ** 0x20
- #define **ACP_MSG_TRACK_CTRL_DISABLE_VOICE** 0x10

Enumerations

- enum acp_el_presence {

  ACP_EL_NOT_PRESENT = 0,

  ACP_EL_EMPTY = 1,

  ACP_EL_PRESENT = 2 }

  Indicates the presence state of an element.

- enum acp_el_ctrl_entity {

  ACP_ENT_ID_DOOR_LOCKS = 0,

  ACP_ENT_ID_VEHICLE_TRACK = 1,

  ACP_ENT_ID_COVERT_MODE = 2,

  ACP_ENT_ID_MICROPHONE = 3,

  ACP_ENT_ID_TRANSMIT_INT = 5,

  ACP_ENT_ID_VEHICLE_TRACK_WITH_COMMIT = 7,

  ACP_ENT_ID_VEHICLE_TRACK_COMMIT = 8,

  ACP_ENT_ID_ALARMS = 9,

  ACP_ENT_ID_IMMOBILIZE = 10,

  ACP_ENT_ID_REMOTE_DOOR_LOCK = 11,

  ACP_ENT_ID_PRIMARY_ANTENNA = 12,

  ACP_ENT_ID_CALL_SO = 13,

  ACP_ENT_ID_CALL_SO_DATA = 14,

  ACP_ENT_ID_FUEL_PUMP_BLOCK = 15,

  ACP_ENT_ID_SIREN = 16,

  ACP_ENT_ID_POS_HISTORY = 17,

  ACP_ENT_ID_VEHICLE_BLOCK = 128 }

  Controlled Entity ID.

- enum acp_el_transmit_unit {

  ACP_EL_TIME_UNIT_SECOND = 0,

  ACP_EL_TIME_UNIT_MINUTE = 1,

  ACP_EL_TIME_UNIT_HOUR = 2,

  ACP_EL_TIME_UNIT_ONEMORE = 3,

  ACP_EL_TIME_UNIT_ONLYONE = 4 }

  Transmit Unit.

5.2.1 Detailed Description

ACP 245 information element description and processing functions.

This file defines the structure of ACP information elements as handled by this library and provides functions to read and write information elements from byte buffers.

The functions exported by this file are not generally useful to external applications. Users of the ACP 245 library should use the functions exported on acp_msg.h instead of this one.

Date:

03/13/2009 02:01:17 PM

Author:

Edantech

Definition in file acp_el.h.

5.2.2 Define Documentation

5.2.2.1 #define ACP_EL_BREAKDOWN_STATUS_MAX_SOURCE (5)

Maximum number of breakdown status elements.

Definition at line 307 of file acp_el.h.

5.2.2.2 #define ACP_EL_BREAKDOWN_STATUS_MIN_SOURCE (1)

Minimum number of breakdown status elements.

Definition at line 309 of file acp_el.h.

5.2.2.3 #define ACP_EL_GPS_RAW_DATA_SAT_MAX (16)

Maximum number of satellite IDs stored by the library.

Definition at line 222 of file acp_el.h.

5.2.2.4 #define ACP_LOCATION_WGS_84 0

Location Type Coding.

See also:

Section 3.8.6 of [ACP245]

Definition at line 579 of file acp_el.h.

5.2.2.5 #define ACP_MORE_FLG 0x80

More flag.

See also:

Section 2.3 of [ACP245]

Definition at line 511 of file acp_el.h.

5.2.2.6 #define ACP_MSG_CFG_PROTO_ID_ACP245 (0)

ACP 245 Protocol ID.

See also:

Section 6.5.1.3.1 of [ACP245]

Definition at line 827 of file acp_el.h.

5.2.3 Enumeration Type Documentation

5.2.3.1 enum acp_el_ctrl_entity

Controlled Entity ID.

See also:

Section 3.6.1 of [ACP245]

Definition at line 83 of file acp_el.h.

5.2.3.2 enum acp_el_presence

Indicates the presence state of an element.

This flag is included on all the elements that support being empty (length = 0) or not included because the message was truncated or they were explicitely not included by using a control flag.

Enumerator:

ACP_EL_NOT_PRESENT The element was not included on the message (truncated or explicitely not included by using a control flag).

ACP_EL_EMPTY The element was included with length 0.

ACP_EL_PRESENT The elment was included with length > 0.

Definition at line 69 of file acp_el.h.

5.2.3.3 enum acp_el_transmit_unit

Transmit Unit.

See also:

Section 3.6.2 of [ACP245]

Definition at line 111 of file acp_el.h.

5.3 acp_err.h File Reference

ACP 245 error codes.

Defines

- #define **ACP_MSG_OK** 0x0000
- #define **ACP_MSG_ERR_TOO_SHORT** 0x8001
- #define **ACP_MSG_ERR_TOO_LONG** 0x8002
- #define **ACP_MSG_ERR_INCOMPLETE** 0x8003
- #define **ACP_MSG_ERR_BAD_FORMAT** 0x8004
- #define **ACP_MSG_ERR_BAD_LENGTH** 0x8005
- #define **ACP_MSG_ERR_INVALID_DEFAULT** 0x8006
- #define **ACP_MSG_ERR_UNKNOWN_MSG_TYPE** 0x8007
- #define **ACP_MSG_ERR_UNKNOWN_APP_ID** 0x8008
- #define **ACP_MSG_ERR_UNSUPPORTED** 0x80A0
- #define **ACP_MSG_ERR_UNSUP_MSG_TYPE** 0x80A1
- #define **ACP_MSG_ERR_NO_MEM** 0x80FE
- #define **ACP_MSG_ERR_FATAL** 0x80FF

5.3.1 Detailed Description

ACP 245 error codes.

Date:

03/13/2009 09:12:03 PM

Author:

Edantech

Definition in file acp_err.h.

5.4 acp_ie.h File Reference

ACP 245 generic information element description and processing functions.

Data Structures

- struct acp_ie_any

  An information element of undetermined type.

Defines

- #define ACP_IE_MAX_LEN ((u16)(0xFFFF))

  Maximum IE length supported.

Information Element IDs

See also:

Section 1.5 of [ACP245]

- #define **ACP_IE_BINARY** 0
- #define **ACP_IE_ISO_8859_1** 1
- #define **ACP_IE_PACKED_DEC** 2
- #define **ACP_IE_EXTENDED** 3
- #define **ACP_IE_EXT_BINARY** 0
- #define **ACP_IE_EXT_ISO_8859_1** 1
- #define **ACP_IE_EXT_PACKED_DEC** 2
- #define **ACP_IE_EXT_RESERVED** 3
- #define **ACP_IE_EXT_UNICODE** 4
- #define **ACP_IE_EXT_UTF8** 5
- #define **ACP_IE_EXT_SHIFT_JIS** 6
- #define **ACP_IE_EXT_PRIVATE** 31

5.4.1 Detailed Description

ACP 245 generic information element description and processing functions.

This file defines the structure of generic ACP information elements, and provides functions to read and write generic information elements from byte buffers.

A generic information element is used when the library does not know the exact type of information elements that must be processed. Otherwise, a function from acp_el.h should be used instead.

The functions exported by this file are not generally useful to external applications. Users of the ACP 245 library should use the functions exported on acp_msg.h instead of this one.

Date:

03/13/2009 02:12:18 PM

Author:

Edantech

Definition in file acp_ie.h.

5.5 acp_init.h File Reference

Library initilization functions.

Defines

- #define ACP_INIT_OK (0)

  Library initialized OK.

- #define ACP_INIT_INVALID_LICENSE (-1)

  Invalid license.

- #define ACP_INIT_ERROR (-2)

  Generic initialization error.

- #define ACP_INIT_DEFAULT_LICENSE "license.sig"

  Default license filename.

- #define ACP_INIT_ENV_LICENSE_FILE "E_ACP245_LICENSE"

  Name of the environment variable that may hold the license file name.

Enumerations

- enum acp_init_opt { ACP_INIT_END }

  Initialization options.

Functions

- E_EXPORT e_ret acp_init (void)

  Initialize library with default arguments.

- E_EXPORT e_ret acp_init_opts (const ascii ∗license_filename,...)

  Initializes the library with the given initialization options.

5.5.1 Detailed Description

Library initilization functions.

Date:

09/03/2009 04:28:31 PM

Author:

Edantech

Definition in file acp_init.h.

5.5.2 Define Documentation

5.5.2.1 #define ACP_INIT_DEFAULT_LICENSE "license.sig"

Default license filename.

Path is relative to working directory.

Definition at line 57 of file acp_init.h.

5.5.2.2 #define ACP_INIT_ERROR (-2)

Generic initialization error.

Definition at line 54 of file acp_init.h.

5.5.2.3 #define ACP_INIT_INVALID_LICENSE (-1)

Invalid license.

Library can not be initialized.

Definition at line 52 of file acp_init.h.

5.5.3 Enumeration Type Documentation

5.5.3.1 enum acp_init_opt

Initialization options.

Enumerator:

ACP_INIT_END This value must be the last parameter when using initialization options.

Definition at line 63 of file acp_init.h.

5.5.4 Function Documentation

5.5.4.1 E_EXPORT e_ret acp_init (void)

Initialize library with default arguments.

The function will first check for a valid license on ACP_INIT_DEFAULT_LICENSE and if it that file doesn't exists, it will then check on the value of the environment variable referenced by ACP_INIT_-ENV_LICENSE_FILE.

Returns:

ACP_INIT_OK if library was successfully initialized. ACP_INIT_INVALID_LICENSE if the library license is invalid. ACP_INIT_ERROR if there's another error initializing the library.

5.5.4.2 E_EXPORT e_ret acp_init_opts (const ascii ∗ *license_filename*, ...)

Initializes the library with the given initialization options.

The options must be sent in the format: acp_init_opt code, <value>

At the current time, no options are supported.

Parameters:

license_filename the file name of the license file to use for license verification.

Returns:

ACP_INIT_OK if library was successfully initialized. ACP_INIT_INVALID_LICENSE if the library license is invalid. ACP_INIT_ERROR if there's another error initializing the library.

5.6 acp_key.h File Reference

ACP 245 activation key verifier functions.

Defines

- #define **ACP_KEY_MAX_KT_LEN** (20)
- #define **ACP_KEY_MAX_KS_LEN** (32)
- #define **ACP_KEY_MAX_MSG_LEN** (32)
- #define **ACP_KEY_AUTH_KEY_LEN** (8)
- #define **ACP_KEY_ERR_INVALID_AUTH_KEY_LEN** ((e_ret)0x8001)
- #define **ACP_KEY_ERR_INVALID_PARAM_LEN** ((e_ret)0x8002)
- #define **ACP_KEY_ERR_BAD_KEY** ((e_ret)0x8003)
- #define **ACP_KEY_ERR_NO_KEY** ((e_ret)0x8004)
- #define **ACP_KEY_ERR_INVALID_MSG** ((e_ret)0x8005)

Functions

- E_EXPORT e_ret acp_key_verify (u8 ∗kt, u8 kt_len, u8 ∗ks, u8 ks_len, u8 ∗iccid, u8 iccid_len, u8 ∗date, u8 date_len, u8 ∗msg, u16 msg_len)

  Verifies if a key is valid for a byte array.

- E_EXPORT e_ret acp_key_verify_msg (u8 ∗kt, u8 kt_len, u8 ∗iccid, u8 iccid_len, u8 ∗date, u8 date_len, acp_msg ∗msg)

  Verifies if the given activation message contains a valid authentication key.

- E_EXPORT e_ret acp_key_get (u8 ∗kt, u8 kt_len, u8 ∗iccid, u8 iccid_len, u8 ∗date, u8 date_len, u8 ∗msg, u16 msg_len, u8 ∗ks, u8 ks_len)

  Computes the Ks to be included as the authentication key of the message.

- E_EXPORT e_ret acp_key_get_msg (u8 ∗kt, u8 kt_len, u8 ∗iccid, u8 iccid_len, u8 ∗date, u8 date_-len, acp_msg ∗msg)

  Computes the Ks to be included as the authentication key of the message and sets it.

5.6.1 Detailed Description

ACP 245 activation key verifier functions.

This file provides a set of functions to generate and verify an ACP 245 activation key.

Date:

03/13/2009 01:51:28 PM

Author:

Edantech

Definition in file acp_key.h.

5.6.2 Function Documentation

5.6.2.1 E_EXPORT e_ret acp_key_get (u8 ∗ *kt*, u8 *kt_len*, u8 ∗ *iccid*, u8 *iccid_len*, u8 ∗ *date*, u8 *date_len*, u8 ∗ *msg*, u16 *msg_len*, u8 ∗ *ks*, u8 *ks_len*)

Computes the Ks to be included as the authentication key of the message.

The key is stored in the space pointed by ks. The space must be allocated before calling this function.

Parameters:

kt the TCU secret key (Kt)

kt_len the length of the TCU key

iccid the ICCID of the TCU

iccid_len the size of the ICCID.

date the current date.

date_len the size of the date.

msg the byte content of the message to sign.

msg_len the size of the message.

ks a pointer where to store the computed Ks.

ks_len the size of the allocated memory space pointed by ks.

Returns:

OK if the key was successfully generated

Precondition:

kt != NULL
iccid != NULL
date != NULL
msg != NULL

Postcondition:

return != OK || authentication key stored on ks

5.6.2.2 E_EXPORT e_ret acp_key_get_msg (u8 ∗ *kt*, u8 *kt_len*, u8 ∗ *iccid*, u8 *iccid_len*, u8 ∗ *date*, u8 *date_len*, acp_msg ∗ *msg*)

Computes the Ks to be included as the authentication key of the message and sets it.

The key will be stored in the auth_key field of the msg.data.cfg_activation structure. If the auth_key field does not point to NULL, the pointer will first be freed.

Required memory will be allocated for the key and should be later freed by calling acp_msg_free or freeing the auth_key pointer.

Parameters:

kt the TCU secret key (Kt)

kt_len the length of the TCU key

iccid the ICCID of the TCU

iccid_len the size of the ICCID.

date the current date.

date_len the size of the date.

acp_msg the activation message to sign.

Returns:

OK if the key was successfully generated and stored.

Precondition:

kt != NULL
iccid != NULL
date != NULL
msg != NULL

Postcondition:

return != OK || msg->data.cfg_activation.vehicle_desc.auth_key != NULL

5.6.2.3 E_EXPORT e_ret acp_key_verify (u8 ∗ *kt*, u8 *kt_len*, u8 ∗ *ks*, u8 *ks_len*, u8 ∗ *iccid*, u8 *iccid_len*, u8 ∗ *date*, u8 *date_len*, u8 ∗ *msg*, u16 *msg_len*)

Verifies if a key is valid for a byte array.

Parameters:

kt the TCU secret key (Kt)

kt_len the length of the TCU key

ks the authentication key.

ks_len the length of the authentication key.

iccid the ICCID of the TCU

iccid_len the size of the ICCID.

date the current date.

date_len the size of the date.

msg the byte array to verify.

msg_len the length of the message.

Returns:

OK if the authentication key was valid. ACP_KEY_ERR_BAD_KEY if the authentication key was invalid. ACP_KEY_ERR_INVALID_PARAM_LEN if any of the given lengths are invalid.

Precondition:

kt != NULL
ks != NULL
iccid != NULL
date != NULL
msg != NULL

See also:

acp_msg_cfg_activation

5.6.2.4 E_EXPORT e_ret acp_key_verify_msg (u8 $*$ *kt*, u8 *kt_len*, u8 $*$ *iccid*, u8 *iccid_len*, u8 $*$ *date*, u8 *date_len*, acp_msg $*$ *msg*)

Verifies if the given activation message contains a valid authentication key.

The key is stored in the auth_key field of the activation message.

Parameters:

kt the TCU secret key (Kt)

kt_len the length of the TCU key

iccid the ICCID of the TCU

iccid_len the size of the ICCID.

date the current date.

date_len the size of the date.

msg the message to verify.

Returns:

OK if the authentication key was valid. ACP_KEY_ERR_BAD_KEY if the authentication key was invalid.

Precondition:

kt != NULL
iccid != NULL
date != NULL
msg != NULL

See also:

acp_msg_cfg_activation

5.7 acp_license.h File Reference

ACP License verification.

Defines

- #define **ACP_LICENSE_VALID** ((e_ret)0)
- #define **ACP_LICENSE_NO_LICENSE** ((e_ret)-1)
- #define **ACP_LICENSE_INVALID_FORMAT** ((e_ret)-2)
- #define **ACP_LICENSE_INVALID** ((e_ret)-3)

Functions

- E_EXPORT e_ret **acp_license_verify** (const char ∗license_filename)
- E_EXPORT bool **acp_license_verified** (void)

5.7.1 Detailed Description

ACP License verification.

Date:

09/01/2009 04:10:19 PM

Author:

Edantech

For internal use only.

Definition in file acp_license.h.

5.8 acp_msg.h File Reference

ACP 245 message description and processing functions.

Data Structures

- struct acp_hdr

  Message Header.

- struct acp_msg_prov_upd

  Provision Update Message #1 (From SO to TCU).

- struct acp_msg_prov_reply

  Provision Reply Message #1 (From TCU to SO).

- struct acp_msg_cfg_upd_245

  Configuration Update Message #2 ACP 245 (From SO to TCU).

- struct acp_msg_cfg_reply

  Configuration Reply (From TCU to SO).

- struct acp_msg_cfg_reply_245

  Configuration Reply #2 ACP 245 (From TCU to SO).

- struct acp_msg_cfg_activation

  Configuration TCU Service Activation/Deactivation Message ACP 245 (From SO to TCU).

- struct acp_msg_func_cmd

  Vehicle Function Command (From SO to TCU).

- struct acp_msg_func_status

  Vehicle Function Status (From TCU to SO).

- struct acp_msg_track_cmd

  Vehicle Tracking Command (From SO to TCU).

- struct acp_msg_track_pos

  Vehicle Position Message (From TCU to SO).

- struct acp_msg_track_reply

  Vehicle Reply Message (From SO to TCU).

- struct acp_msg_alarm_notif

  Theft Alarm Notification (From TCU to SO).

- struct acp_msg_alarm_reply

  Theft Alarm Reply (From SO to TCU).

- struct acp_msg_alarm_pos

Vehicle Position Message (TCU to SO).

- struct acp_msg_alarm_ka

  Message Keep Alive (TCU to SO).

- struct acp_msg_alarm_ka_reply

  Message Keep Alive Reply (SO to TCU).

- struct acp_msg

  ACP245 Message Structure.

Defines

- #define **ACP_MSG_HDR_MAX_LEN** 6

Message Control Flag.

See also:

Section 4.1.7 of [ACP245]

- #define ACP_HDR_MSG_CTRL_DONT_USE_TLV 0x4

  Dont Use TLV.

- #define ACP_HDR_MSG_CTRL_16BIT_LEN 0x2

  Set if the message length field is 2 bytes long.

- #define ACP_HDR_MSG_CTRL_RESP_EXP 0x1

  Set if a response is expected.

Enumerations

- enum acp_msg_app_id {
  ACP_APP_ID_PROVISIONING = 1,
  ACP_APP_ID_CONFIGURATION = 2,
  ACP_APP_ID_REMOTE_VEHICLE_FUNCTION = 6,
  ACP_APP_ID_VEHICLE_TRACKING = 10,
  ACP_APP_ID_ALARM = 11 }

  Application ID.

- enum acp_msg_type {
  ACP_MSG_TYPE_PROV_UPD = 1,
  ACP_MSG_TYPE_PROV_REPLY = 3,
  ACP_MSG_TYPE_PROV_UPD_COMMIT = 2,
  ACP_MSG_TYPE_PROV_REPLY_COMMIT = 4,
  ACP_MSG_TYPE_PROV_REQUEST = 5,
  ACP_MSG_TYPE_PROV_STATUS = 6,

ACP_MSG_TYPE_CFG_REPLY = 3,
ACP_MSG_TYPE_CFG_UPD_245 = 8,
ACP_MSG_TYPE_CFG_REPLY_245 = 9,
ACP_MSG_TYPE_CFG_ACT_245 = 10,
ACP_MSG_TYPE_CFG_UPD = 1,
ACP_MSG_TYPE_CFG_UPD_COMMIT = 2,
ACP_MSG_TYPE_CFG_REPLY_COMMIT = 4,
ACP_MSG_TYPE_CFG_REQUEST = 5,
ACP_MSG_TYPE_CFG_STATUS = 6,
ACP_MSG_TYPE_CFG_EDIT = 7,
ACP_MSG_TYPE_FUNC_CMD = 2,
ACP_MSG_TYPE_FUNC_STATUS = 3,
ACP_MSG_TYPE_FUNC_REQ = 1,
ACP_MSG_TYPE_TRACK_CMD = 1,
ACP_MSG_TYPE_TRACK_POS = 2,
ACP_MSG_TYPE_TRACK_REPLY = 3,
ACP_MSG_TYPE_TRACK_WITH_COMMIT = 4,
ACP_MSG_TYPE_TRACK_COMMIT = 5,
ACP_MSG_TYPE_ALARM_NOTIF = 1,
ACP_MSG_TYPE_ALARM_REPLY = 2,
ACP_MSG_TYPE_ALARM_POS = 3,
ACP_MSG_TYPE_ALARM_KA = 4,
ACP_MSG_TYPE_ALARM_KA_REPLY = 5 }

ACP Message types.

- enum acp_msg_hdr_prio {
  ACP_HDR_MSG_PRIO_RESERVED = 0,
  ACP_HDR_MSG_PRIO_ABORT = 1,
  ACP_HDR_MSG_PRIO_PAUSE = 2,
  ACP_HDR_MSG_PRIO_RESUME = 3 }

  Message Priority Flag.

Functions

- E_EXPORT e_ret acp_msg_init (acp_msg ∗msg, acp_msg_app_id app_id, acp_msg_type type)

  Initializes an ACP message struct so it can be safely used.

- E_EXPORT e_ret acp_msg_read_data (u8 ∗data, u32 data_len, u32 ∗readed, acp_msg ∗msg)

  Reads an ACP message from the byte array.

- E_EXPORT e_ret acp_msg_write_data (u8 ∗data, u32 data_len, u32 ∗written, acp_msg ∗msg)

  Writes an ACP message to a byte array.

- E_EXPORT bool acp_msg_is_reply_codes (acp_msg_app_id id, acp_msg_type type, acp_msg_-app_id reply_id, acp_msg_type reply_type)

  Returns if a message with the given reply_id and reply_type identify a reply for a message with the given application id and message type.

- E_EXPORT void acp_msg_free (acp_msg ∗msg)

  Frees the internal structures of the ACP message.

5.8.1 Detailed Description

ACP 245 message description and processing functions.

This file defines the structure of ACP messages as handled by this library and provides functions to read and write messages from byte buffers.

Information Elements are described in "acp_el.h".

Date:

03/13/2009 01:51:28 PM

Author:

Edantech

See also:

acp_el.h

Definition in file acp_msg.h.

5.8.2 Define Documentation

5.8.2.1 #define ACP_HDR_MSG_CTRL_RESP_EXP 0x1

Set if a response is expected.

Definition at line 131 of file acp_msg.h.

5.8.3 Enumeration Type Documentation

5.8.3.1 enum acp_msg_app_id

Application ID.

See also:

Section 1.1, Telematics Applications of [ACP245].

Definition at line 63 of file acp_msg.h.

5.8.3.2 enum acp_msg_hdr_prio

Message Priority Flag.

See also:

Section 4.1.8 of [ACP245]

Definition at line 138 of file acp_msg.h.

5.8.4 Function Documentation

5.8.4.1 E_EXPORT void acp_msg_free (acp_msg ∗ *msg*)

Frees the internal structures of the ACP message.

When reading an ACP message with acp_msg_read or acp_msg_read_data, resources may be allocated for some fields (ie. string information elements with variable length). By calling this function, these resources will be deallocated.

You should call this function only with an acp_msg structure that was previously used on a cal to acp_-msg_read or acp_msg_read_data.

After calling this function, the acp_msg structure can be reused on a new acp_msg_read or acp_msg_-read_data.

Parameters:

msg a pointer to an ACP message structure.

Postcondition:

msg invalid

5.8.4.2 E_EXPORT e_ret acp_msg_init (acp_msg ∗ *msg*, acp_msg_app_id *app_id*, acp_msg_type *type*)

Initializes an ACP message struct so it can be safely used.

If you are creating the message (instead of reading it with acp_msg_read), you must call this function before calling on acp_msg_write and acp_msg_write_data.

You can pass an unknown application ID or message type. In that case, calls to functions that read, write or operate on the message will fail if they do not support that application ID or message type.

Parameters:

msg the ACP message structure to initialize

app_id the application ID.

type the message type.

Returns:

OK if the message was successfully initialized.

Precondition:

msg != NULL

Postcondition:

msg.hdr.app_id == app_id
msg.hdr.type == type

5.8.4.3 E_EXPORT bool acp_msg_is_reply_codes (acp_msg_app_id *id*, acp_msg_type *type*, acp_msg_app_id *reply_id*, acp_msg_type *reply_type*)

Returns if a message with the given reply_id and reply_type identify a reply for a message with the given application id and message type.

Returns:

TRUE if it's a valid reply, FALSE otherwise.

Parameters:

id the application ID of the message.

type the message type of the message.

reply_id the application ID of the reply message.

reply_type the message type of the reply message.

5.8.4.4 E_EXPORT e_ret acp_msg_read_data (u8 ∗ *data*, u32 *data_len*, u32 ∗ *readed*, acp_msg ∗ *msg*)

Reads an ACP message from the byte array.

If there's an error reading the message, an error code will be returned, and the value of the readed parameter is undefined. Otherwise, the readed parameter is not NULL, its value will be the number of bytes readed from the the byte array.

If this function returns OK, you must call acp_msg_free when you no longer need the message.

This function calls acp_msg_init on your behalf, you don't need to initialize the message before calling it.

Returns:

OK or an error code, as defined on acp_err.h

Parameters:

data the byte array.

data_len the length of the byte array. readed if return is OK, the number of bytes readed, undefined otherwise. If NULL, the parameter will be ignored.

msg a pointer to ACP message.

Precondition:

data != NULL
msg != NULL

5.8.4.5 E_EXPORT e_ret acp_msg_write_data (u8 ∗ *data*, u32 *data_len*, u32 ∗ *written*, acp_msg ∗ *msg*)

Writes an ACP message to a byte array.

If there's an error reading the message, an error code will be returned, and the value of the written parameter is undefined. Otherwise, if the written parameter is not NULL, its value will be the number of bytes written to the the byte array.

Returns:

OK or an error code, as defined on acp_err.h

Parameters:

data the byte array.

data_len the length of the byte array. written if return is OK, the number of bytes written, undefined otherwise. If NULL, the parameter will be ignored.

msg a pointer to a valid ACP message.

Precondition:

data != NULL
msg != NULL

5.9 acp_types.h File Reference

ACP 245 primitive type definitions.

Typedefs

- typedef s16 **e_ret**

5.9.1 Detailed Description

ACP 245 primitive type definitions.

This file will be included by the ACP 245 library if not packed with e_libs, otherwise the standard e_libs type definition file will be used instead.

Date:

05/02/2009 03:22:18 PM

Author:

Edantech

Definition in file acp_types.h.

5.10 hmac_sha256.h File Reference

HMAC calculation functions.

Functions

- void hmac_sha256 (const u8 ∗k, u16 lk, const u8 ∗d, u16 ld, u8 ∗out, u16 t)
  Calculates the HMAC-SHA256 function of the given byte array.

5.10.1 Detailed Description

HMAC calculation functions.

Date:

06/23/2009 10:01:28 AM

Definition in file hmac_sha256.h.

5.10.2 Function Documentation

5.10.2.1 void hmac_sha256 (const u8 ∗ *k*, u16 *lk*, const u8 ∗ *d*, u16 *ld*, u8 ∗ *out*, u16 *t*)

Calculates the HMAC-SHA256 function of the given byte array.

Calculates out = HMAC-SHA256(k, d). If out is too small to store the result, the output will be truncated.

Parameters:

k secret key

lk length of the key in bytes

d data

ld length of data in bytes

t output buffer, at least "t" bytes

Precondition:

k != NULL
d != NULL
out != NULL

Postcondition:

out = HMAC-SHA256(k, d)

5.11 sha256.h File Reference

SHA256 processing functions.

Data Structures

- struct SHA256Context

  SHA 256 context.

Defines

- #define SHA256_HASH_SIZE (32)

  Size of a SHA256 hash, in bytes.

- #define SHA256_HASH_WORDS (8)

  Size of a SHA256 hash, in 8 bit words.

Functions

- void SHA256Init (SHA256Context ∗sc)

  Initializes a SHA256 context.

- void SHA256Update (SHA256Context ∗sc, const void ∗data, u32 len)

  Update the current SHA256 state by adding the given data.

- void SHA256Final (SHA256Context ∗sc, u8 hash[SHA256_HASH_SIZE])

  Calculates the SHA256 digest of current SHA256 context.

5.11.1 Detailed Description

SHA256 processing functions.

Date:

06/21/2009 01:11:12 PM

Definition in file sha256.h.

5.11.2 Function Documentation

5.11.2.1 void SHA256Final (SHA256Context ∗ *sc*, u8 *hash*[SHA256_HASH_SIZE])

Calculates the SHA256 digest of current SHA256 context.

Parameters:

sc a context previously initialized with SHA256Init.

hash a byte array of length SHA256_HASH_SIZE.

Precondition:

sc != NULL
hash != NULL and hash is SHA256_HASH_SIZE bytes long.

5.11.2.2 void SHA256Init (SHA256Context ∗ *sc*)

Initializes a SHA256 context.

Parameters:

sc the context.

Precondition:

sc != NULL

5.11.2.3 void SHA256Update (SHA256Context ∗ *sc*, const void ∗ *data*, u32 *len*)

Update the current SHA256 state by adding the given data.

Parameters:

sc a context previously initialized with SHA256Init.

data the data

len the length of the data buffer.

Precondition:

sc != NULL
data != NULL

Index