ACP Test Server
Test Report

CONFIDENTIAL

edantech

Reference: 13004_ACPSRV_TR

Revision: 2.0

Date: 2009-07-13

Revisions

1.0	03/23/09	Initial Release.
2.0	07/13/09	Updated to include ACP Test Gateway test results.

Copyright Notice and Trade marks

ACP Test Server

Test Report

Document Version: 2.0

Document, Release Date, July 13th, 2009

EDANTECH (ILWICK S.A.),

Av. Libertador 1807,

11800, Montevideo, URUGUAY

Phone#: +598 2 929 0029

Fax#: +598 2 924 8013

e-mail: info@edantech.com

Contents

1 References

[ACP245]	ACP 245 – Application Communication Protocol v 1.2.1 http://www.denatran.gov.br/download/ACP_245_V1_2_1_19_05_09.pdf
[ACP_SPEC]	ACP Test Server Specification, Technical Specification, EDANTECH 13001_ACP_SPEC_2_1.pdf
[ACP_HD]	High level PDU Library Documentation, ACP Test Server, EDANTECH Suite 13005_ACPSRV_HD_1_1.pdf

2 Introduction

This document details the testing procedures a results for the ACP 245 Test Server.

2.1 Testing strategies and procedures

Edantech applies different types of tests to each of it's software releases, including the following:

- **Unit testing:** Code include unit tests of all low-level functions and libraries. Tools are used to check that code coverage is above minimum thresholds (70% minimum, depending on how critical is the library).

  Unit test is performed regularly, and 0 failed tests are enforced by the software release process.
- **Integration testing:** On high-level libraries and applications, integration testing is performed to simulate different system execution conditions.

  Integration testing is performed regularly, and 0 failed tests are enforced by the software release process.
- **Thrash testing:** Critical function calls of low-level libraries (in this case, ACP245 PDU parsing libraries) are tested millions of times by submitting random data on user provided parameters (ie. network data buffers) to find crashes and boundary conditions which are not being handled.
- **Dynamic checking:** Code is checked using tools to detect memory management errors, and on in case of multi-threaded code, possible race conditions.

  For this application, only memory management checks were performed.
- **Static checking:** We have a 0-warning policy on developed software, running under the most exigent conditions allowed by the used compiler[1]. Additionally, the code is checked with Splint[2], but under it's weak mode.
- **Profiling** is performed on performance critical code. For this application, profiling was not performed.

1 -Wall -Wextra -pedantic -std89 for GNU gcc compiler.

2 http://www.splint.org/ , weak mode enabled with --weak

2.2 Tools

The following are the tools that were used to check the developed software.

Test reports are provided on the following section.

2.2.1 check

URL: http://check.sourceforge.net

Check is a unit test framework for C applications.

Check was used to create unit tests for the ACP 245 and supporting libraries.

2.2.2 gcov & lcov

URL: http://gcc.gnu.org/onlinedocs/gcc/Gcov.html

URL: http://ltp.sourceforge.net/coverage/lcov.php

From ***gcov*** website:

> ***"gcov*** is a test coverage program. It is used to analyze programs to help create more efficient, faster running code and to discover untested parts of the program. ***gcov*** can be used as a profiling tool to help discover where optimization efforts will best affect the code. "

From ***lcov*** website:

> ***"lcov*** is a graphical front-end for GCC's coverage testing tool ***gcov***. It collects ***gcov*** data for multiple source files and creates HTML pages containing the source code annotated with coverage information. It also adds overview pages for easy navigation within the file structure."

Both gcov and lcov have been used to create coverage reports for the unit tests of the ACP 245 and supporting libraries.

2.2.3 valgrind

URL: http://valgrind.org

From the ***Valgrind*** site:

> ***"Valgrind*** is an instrumentation framework for building dynamic analysis tools. There are ***Valgrind*** tools that can automatically detect many memory management and threading bugs, and profile programs in detail."

We apply ***Valgrind*** to check for memory management errors (with *memcheck*), for code profiling (with *callgrind* and *kcallgrind* for visualization), and as a hint to profile heap usage for libraries used on embedded applications (with *massif*).

For the ACP Test Server, ***Valgrind*** has been used to run all the unit tests and check for memory management errors on the ACP 245 and supporting libraries.

No checks have been performed directly on the running application.

3 Test results

Test	Condition	Expected	Result	Notes
Unit Test - ACP libraries	All test pass.	All test pass.	√	All functions are being tested.
memcheck - ACP libraries	0 Errors	0 Errors	√	All functions are being tested.
Code Coverage - ACP libraries	90.00%	80.00%	√	
Integration testing ACP Server Test Scripts	All test pass.	All test pass.	√	
Browser Support IE >= 7	Firefox 3.5 and IE 7.0 supported	IE supported.	√	Mozilla Firefox 3.5 is the recommended browser.
Browser Support Firefox >= 3	Firefox supported	Firefox supported	√	
ACP Gateway Commands Testing	80.00%	All test pass.	√	Basic command coverage tested. Additional tests are required for border cases. **Customer acceptance test need to be performed.**

3.1 Unit test Report

Supporting libraries

```
Running suite(s): e_util
100%: Checks: 4, Failures: 0, Errors: 0
PASS: e_util_test
Running suite(s): e_log
100%: Checks: 4, Failures: 0, Errors: 0
PASS: e_log_test
Running suite(s): e_conf
100%: Checks: 7, Failures: 0, Errors: 0
PASS: e_conf_test
Running suite(s): e_buff
```

```
100%: Checks: 7, Failures: 0, Errors: 0
PASS: e_buff_test
Running suite(s): e_crc
100%: Checks: 3, Failures: 0, Errors: 0
PASS: e_crc_test
Running suite(s): e_queue
100%: Checks: 1, Failures: 0, Errors: 0
PASS: e_queue_test
Running suite(s): e_timer
100%: Checks: 1, Failures: 0, Errors: 0
PASS: e_timer_test
Running suite(s): e_port
100%: Checks: 2, Failures: 0, Errors: 0
PASS: e_port_test
Running suite(s): e_syslog
100%: Checks: 1, Failures: 0, Errors: 0
PASS: e_syslog_test
==================
All 9 tests passed
==================
```

ACP245 libraries

```
** RANDOM SEED: 1247572385 **
Running suite(s): acp_el
100%: Checks: 21, Failures: 0, Errors: 0
PASS: acp_el_test
** RANDOM SEED: 1247572385 **
Running suite(s): acp_msg
100%: Checks: 28, Failures: 0, Errors: 0
PASS: acp_msg_test
** RANDOM SEED: 1249325263 **
Running suite(s): acp_key
100%: Checks: 3, Failures: 0, Errors: 0
PASS: acp_key_test
==================
All 3 tests passed
==================
```

High-level Python Library

```
acp245.test.test_pdu
  PduTest
    test_acp_msg_cfg_upd_245_py ...                                               [OK]
    test_acp_msg_read_write_alarm_ka ...                                          [OK]
    test_acp_msg_read_write_alarm_ka_no_vehicle_desc ...                          [OK]
    test_acp_msg_read_write_alarm_ka_reply ...                                    [OK]
    test_acp_msg_read_write_alarm_ka_reply_no_vehicle_desc ...                    [OK]
    test_acp_msg_read_write_alarm_reply ...                                       [OK]
    test_acp_msg_read_write_cfg_activation ...                                    [OK]
    test_acp_msg_read_write_cfg_reply ...                                         [OK]
    test_acp_msg_read_write_cfg_reply_245 ...                                     [OK]
    test_acp_msg_read_write_cfg_upd_245 ...                                       [OK]
    test_acp_msg_read_write_func_cmd_1_31_1 ...                                   [OK]
    test_acp_msg_read_write_func_status_1_31_2 ...                                [OK]
    test_acp_msg_read_write_track_cmd ...                                         [OK]
    test_acp_msg_read_write_track_pos ...                                         [OK]
    test_acp_msg_read_write_track_pos_1_32_1 ...                                  [OK]
    test_acp_msg_read_write_track_pos_info_type ...                               [OK]
    test_acp_msg_read_write_track_reply ...                                       [OK]
    test_as_dict ...                                                              [OK]
    test_is_reply ...                                                             [OK]
    test_msg_thrash ...                                                           [OK]
    test_read_write ...                                                           [OK]
    test_read_write_msg_prov_reply ...                                            [OK]
    test_read_write_msg_prov_upd ...                                              [OK]
    test_type ...                                                                 [OK]
    test_vehicle_descriptor_serial_formats ...                                    [OK]


---------------------------------------------------------------------------------------
Ran 25 tests in 0.820s


PASSED (successes=25)
```

3.2 *Integration test Report*

```
Running 42 tests.
```

```
test_console
  ACPConsoleTest
    test_scripts_activate_immo ...                                    [OK]
    test_scripts_basic ...                                            [OK]
    test_scripts_configure ...                                        [OK]
    test_scripts_deactivate_immo ...                                  [OK]
    test_scripts_keep_alive ...                                       [OK]
    test_scripts_track ...                                            [OK]
  ACPConsoleUiTest
    test_ui ...                                                       [OK]
test_args
  ArgsTest
    test_BinHex ...                                                   [OK]
    test_Int8 ...                                                     [OK]
    test_Int8_hex ...                                                 [OK]
    test_Int8_required ...                                            [OK]
    test_Int8_with_opts ...                                           [OK]
    test_List ...                                                     [OK]
    test_List_int ...                                                 [OK]
    test_String ...                                                   [OK]
    test_Time ...                                                     [OK]
test_gateway
  ACPGwTest
    test_disconnect_tcu ...                                           [OK]
    test_disconnect_tcu_before_connect ...                            [OK]
    test_disconnect_tcu_timeout ...                                   [OK]
    test_pop_alarm_ka ...                                             [OK]
    test_pop_alarm_notif ...                                          [OK]
    test_pop_track_pos ...                                            [OK]
    test_send_alarm_reply ...                                         [OK]
    test_send_cfg_upd_245 ...                                         [OK]
    test_send_cfg_upd_245_reply_245 ...                               [OK]
    test_send_func_cmd ...                                            [OK]
    test_send_func_cmd_abort ...                                      [OK]
    test_send_func_cmd_abort_before_starting ...                      [OK]
    test_send_func_cmd_twice ...                                      [OK]
    test_send_prov_upd ...                                            [OK]
    test_send_prov_upd_twice ...                                        [OK]
    test_send_prov_upd_with_hdr ...                                     [OK]
    test_send_track_cmd ...                                             [OK]
    test_send_track_reply ...                                           [OK]
    test_start_test ...                                                 [OK]
    test_start_test_twice ...                                           [OK]
    test_timeout ...                                                    [OK]
    test_wait_for_connection ...                                        [OK]
    test_wait_for_disconnection ...                                     [OK]
  ACPGwTestSec
    test_ui ...                                                         [OK]
  ACPGwUiTest
    test_ui ...                                                         [OK]
test_testdb
  TestDbTest
    test_testdb ...                                                     [OK]

-------------------------------------------------------------------------------
Ran 42 tests in 19.093s

PASSED (successes=42)
```

3.3 *Code Coverage Report*

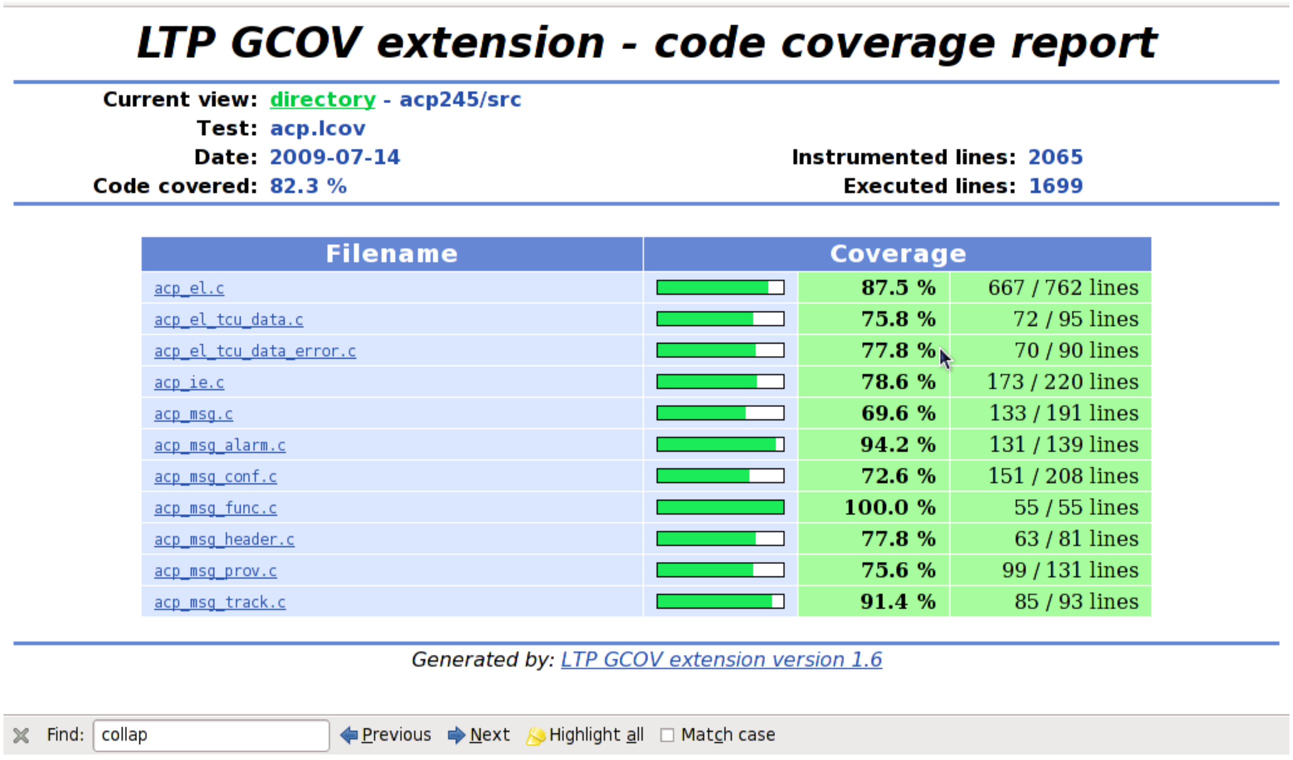
The Code Coverage Test is executed calling at the project root directory:

```
$ make coverage
```

The coverage test results are presented in as HTML files accessed via:

```
tests/coverage/index.html
```

Coverage tests were applied to ACP 245 libraries.



LTP GCOV extension - code coverage report

Current view: directory - acp245/src
Test: acp.lcov
Date: 2009-07-14
Code covered: 82.3 %
Instrumented lines: 2065
Executed lines: 1699

Filename	Coverage		
acp_el.c	🟩	87.5 %	667 / 762 lines
acp_el_tcu_data.c	🟩	75.8 %	72 / 95 lines
acp_el_tcu_data_error.c	🟩	77.8 %	70 / 90 lines
acp_ie.c	🟩	78.6 %	173 / 220 lines
acp_msg.c	🟩	69.6 %	133 / 191 lines
acp_msg_alarm.c	🟩	94.2 %	131 / 139 lines
acp_msg_conf.c	🟩	72.6 %	151 / 208 lines
acp_msg_func.c	🟩	100.0 %	55 / 55 lines
acp_msg_header.c	🟩	77.8 %	63 / 81 lines
acp_msg_prov.c	🟩	75.6 %	99 / 131 lines
acp_msg_track.c	🟩	91.4 %	85 / 93 lines

Generated by: LTP GCOV extension version 1.6

3.4 valgrind - memcheck Report

Supporting Libraries

tests/e_buff_test.valgrind.out

```
ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 5 from 1)
malloc/free: in use at exit: 0 bytes in 0 blocks.
malloc/free: 12,659 allocs, 12,659 frees, 835,169 bytes allocated.
For counts of detected errors, rerun with: -v
All heap blocks were freed -- no leaks are possible.
```

tests/e_conf_test.valgrind.out

```
ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 5 from 1)
malloc/free: in use at exit: 0 bytes in 0 blocks.
malloc/free: 1,374 allocs, 1,374 frees, 471,781 bytes allocated.
For counts of detected errors, rerun with: -v
All heap blocks were freed -- no leaks are possible.
```

tests/e_crc_test.valgrind.out

```
ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 5 from 1)
malloc/free: in use at exit: 0 bytes in 0 blocks.
malloc/free: 123 allocs, 123 frees, 14,494 bytes allocated.
```

```
For counts of detected errors, rerun with: -v
All heap blocks were freed -- no leaks are possible.
```

tests/e_log_test.valgrind.out

```
ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 5 from 1)
malloc/free: in use at exit: 0 bytes in 0 blocks.
malloc/free: 147 allocs, 147 frees, 15,118 bytes allocated.
For counts of detected errors, rerun with: -v
All heap blocks were freed -- no leaks are possible.
```

tests/e_port_test.valgrind.out

```
ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 5 from 1)
malloc/free: in use at exit: 0 bytes in 0 blocks.
malloc/free: 109 allocs, 109 frees, 13,739 bytes allocated.
For counts of detected errors, rerun with: -v
All heap blocks were freed -- no leaks are possible.
```

tests/e_queue_test.valgrind.out

```
ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 5 from 1)
malloc/free: in use at exit: 0 bytes in 0 blocks.
malloc/free: 8,906 allocs, 8,906 frees, 701,798 bytes allocated.
For counts of detected errors, rerun with: -v
All heap blocks were freed -- no leaks are possible.
```

tests/e_timer_test.valgrind.out

```
ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 5 from 1)
malloc/free: in use at exit: 288 bytes in 1 blocks.
malloc/free: 89 allocs, 88 frees, 13,254 bytes allocated.
For counts of detected errors, rerun with: -v
searching for pointers to 1 not-freed blocks.
checked 10,612,272 bytes.
```

89 allocations. vs. 88 frees is explained by pthreads usage on test case. In any event, `e_timer` is not used on ACP245 server.

tests/e_util_test.valgrind.out

```
ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 5 from 1)
malloc/free: in use at exit: 0 bytes in 0 blocks.
malloc/free: 223 allocs, 223 frees, 21,307 bytes allocated.
For counts of detected errors, rerun with: -v
All heap blocks were freed -- no leaks are possible.
```

ACP245 libraries

acp_el_test.valgrind.out

```
ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 32 from 1)

malloc/free: in use at exit: 0 bytes in 0 blocks.

malloc/free: 2,603 allocs, 2,603 frees, 155,610 bytes allocated. For counts of
detected errors, rerun with: -v

All heap blocks were freed -- no leaks are possible.
```

acp_msg_test.valgrind.out

```
ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 28 from 1)

malloc/free: in use at exit: 0 bytes in 0 blocks.

malloc/free: 2,467 allocs, 2,467 frees, 145,962 bytes allocated.

For counts of detected errors, rerun with: -v

All heap blocks were freed -- no leaks are possible.
```

4 Test code example

This is an example of tests cases used on validation of the ACP 245 protocol library.

The example shows a 'configuration reply to ACP 245'. For more information about the message composition see page 45 of [ACP245].

The body of the test code consists in: *variables definition*, the *header data*, the *body data*, the *comparison-verification code* and the *write and read verification code*.

```
START_TEST (test_acp_msg_read_write_cfg_reply_245)
{
```

-- *variables definition*

```
   e_ret res;
   acp_msg msg;
   acp_hdr hdr;
   acp_msg_cfg_reply_245 body;
   e_buff buff_s;
   e_buff *buff = &buff_s;
   u8 buff_data[512];
   u8 r_buff_data[512];
```

<u>MESSAGE COMPOSITION</u>

HEADER ELEMENT
VERSION ELEMENT
MESSAGE FIELDS
TCU DATA ERROR ELEMENT
VEHICLE DESCRIPTOR ELEMENT

Header data: Contains APPLICATION ID, MESSAGE TYPE, APPLICATION VERSION, MESSAGE CONTROL FLAG , MESSAGE LENGHT AND OTHER FLAGS

See page 45 for more information

-- *header data*

```
    u8 hdr_data1[] = {
    0x02,                   /* APPLICATION ID = 2 */
    0x09,                   /* MESSAGE TYPE = 9 */
    0x01,                   /* MESSAGE CONTROL FLAG = 1 */
    0x4E                    /* LENGHT of body + header = 78 */
```

```
};
```

-- body data

```
u8 body_data1[] = {
/* VERSION ELEMENT */
/* IE Identifier = 0 - More flag - Length */
0x04,
/* Car manufacturer ID */
0x08,
/* TCU Manufacturer ID */
0x83,
/* Major hardware release */
0x01,
/* Major software release */
0x03,
/* END VERSION ELEMENT*/

/* MESSAGE FIELDS */
/*More flags*/
0x00,
/*Applflag1 - Control flag = 2*/
0x02,
/*Status flag1 - TCU Response Flag - Reserved*/
0x00,
/* END MESSAGE FIELDS */

/* TCU DATA ERROR ELEMENT */
/*IE Identifier - More Flag - Length*/
0x06,
/*Data Type MSB*/
0x00,
/*Data Type LSB*/
0x11,
/*Length Data Type*/
0x01,
/*Configuration Data*/
0x00,
/*Error Element*/
/*IE Identifier = 0 - More flag - Length*/
0x01,
    /*Error code*/
    0x00,
   /* END  TCU DATA ERROR ELEMENT*/

    /* VEHICLE DESCRIPTOR ELEMENT */
    0x20, 0x39,
       /* Length = 57 */
    0xB1, 0x30,
    /*VIN Number*/
    0x50,
                    /*IE Identifier = 1 - length = 16 */
    0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39,
    /*TCU Serial number*/
    0x48,
                    /*IE Identifier = 2 - length = 8*/
    0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39,
    0x41, 0x42, 0x43, 0x44, 0x45, 0x46,
    /*BCD IMEI Number*/
    0x88,
                    /*IE Identifier = 2 - length = 15/2 = 8 bytes*/
    0x23, 0x45, 0x67, 0x89, 0x01, 0x23, 0x45, 0x67,
    /*BCD SIM CARD ID*/
    0x8A,
                    /*IE = 2 - length = 19/2 = 10 bytes*/
    0x01, 0x23, 0x45, 0x67, 0x89, 0x01, 0x23, 0x45, 0x67, 0x89,
    /*Binary format (Auth Key)*/
    0x08,
                    /*IE = 0 - length = 8*/
    0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37
    /* END VEHICLE DESCRIPTOR ELEMENT */
    };
```

-- comparsion-verification code

```
    e_buff_wrap(buff, buff_data, sizeof(buff_data));
       /* Puts the data of the array 'buff' in the 'buff_data' array predefined */
    e_buff_write_buff(buff, hdr_data1, sizeof(hdr_data1));
    e_buff_write_buff(buff, body_data1, sizeof(body_data1));

    res = acp_msg_read(buff, &msg);
       /* Parses the data in the 'buff' array to the message 'msg' that
       only contains the relevan information of the message*/

    ARE_EQ_INT(OK, res);
```

```
   /* Compares the result of the  previous parsin, verifing that is Ok*/
hdr = msg.hdr;
ARE_EQ_INT(ACP_APP_ID_CONFIGURATION, hdr.app_id);
   /* Checks if the given Application ID  was the right one*/
IS_FALSE(hdr.test);
ARE_EQ_INT(ACP_MSG_TYPE_CFG_REPLY_245, hdr.type);
   /* Checks if the given Header Type was the right one*/
ARE_EQ_INT(0, hdr.version);
ARE_EQ_INT(0x1, hdr.msg_ctrl);

IS_TRUE(hdr.msg_ctrl & ACP_HDR_MSG_CTRL_RESP_EXP);
body = msg.data.cfg_reply_245;
ARE_EQ_INT(0x8, body.version.car_manufacturer);
ARE_EQ_INT(0x83, body.version.tcu_manufacturer);
ARE_EQ_INT(1, body.version.major_hard_rel);
ARE_EQ_INT(3,body.version.major_soft_rel);
ARE_EQ_STR("0123456789ABCDEF", body.vehicle_desc.vin);
ARE_EQ_STR("01234567", body.vehicle_desc.tcu_serial);
ARE_EQ_STR("2345678901234567", body.vehicle_desc.imei);
ARE_EQ_STR("01234567890123456789", body.vehicle_desc.iccid);
```

-- *write and read verification code*

```
/* Writes and read the message while compares if the data written is the same of the
   data readed */
e_buff_reset(buff);

ARE_EQ_INT(sizeof(buff_data), e_buff_write_remain(buff));
res = acp_msg_write(buff, &msg);
ARE_EQ_INT(OK, res);
ARE_EQ_INT(sizeof(hdr_data1) + sizeof(body_data1), e_buff_read_remain(buff));
e_buff_read_buff(buff, r_buff_data, e_buff_read_remain(buff));
ARE_EQ_BINC(body_data1, r_buff_data + sizeof(hdr_data1), sizeof(body_data1));
ARE_EQ_BINC(hdr_data1, r_buff_data, sizeof(hdr_data1));
acp_msg_free(&msg);
END_TEST
```

5 Validation Tests

The validation of the ACP245 Server suite is done by invocation of the predefined test scripts on the Web Console and Gateway and checking expected results.

The test performed on the Gateway are a kind of integration test, since the full server is excercised by starting all required services and performing HTTP requests against the Gateway. For the client, we implemented a basic client to perform these integration tests.

The same scripts are executed on the integration tests performed automatically upon software release.