edantech

ACP 245
Test Server and Tools

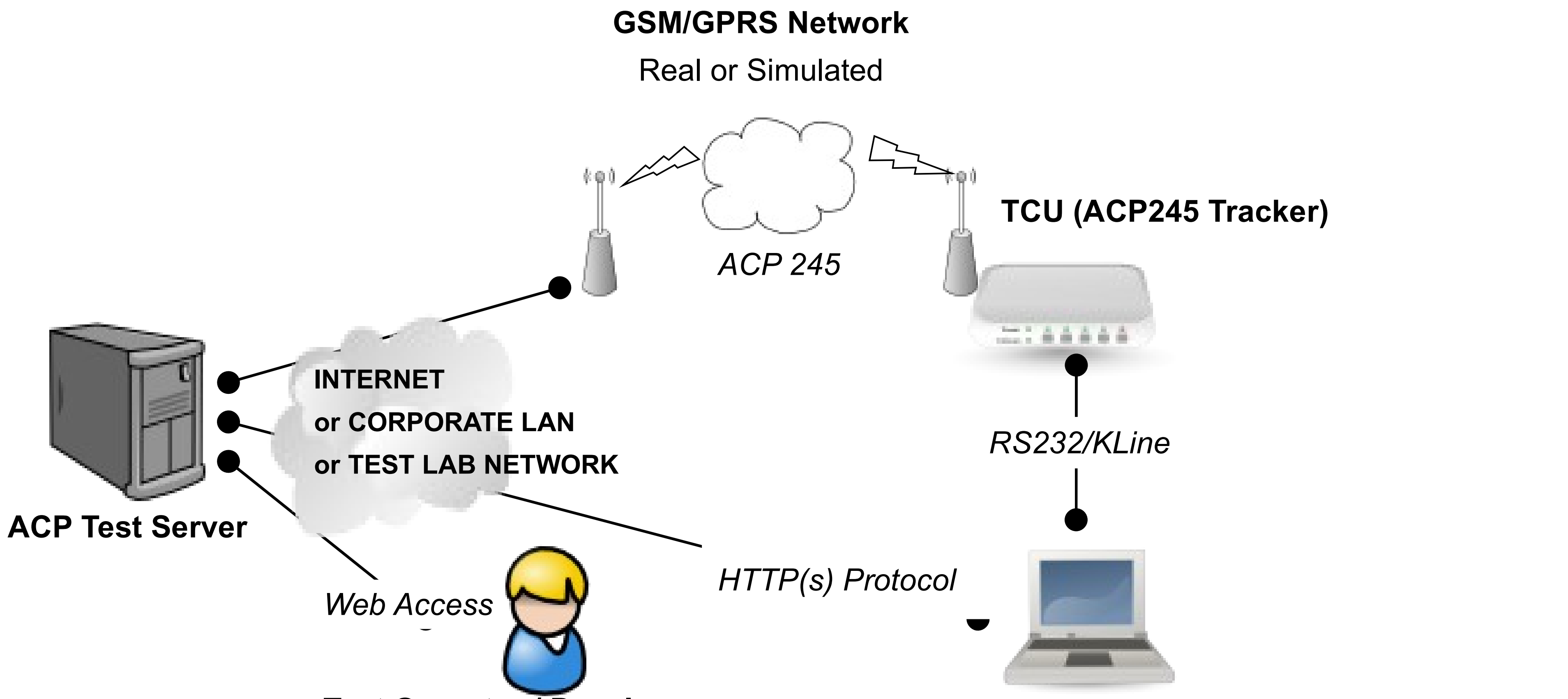
October, 2009

130010_ACP_SERVER_PRES_EXT

Rev. 1.1

What's the ACP 245 Test Server?

A main component in the ACP 245 *validation* and *system testing* infrastructure



Test Operator / Developer

•*Runs and writes custom tests.*

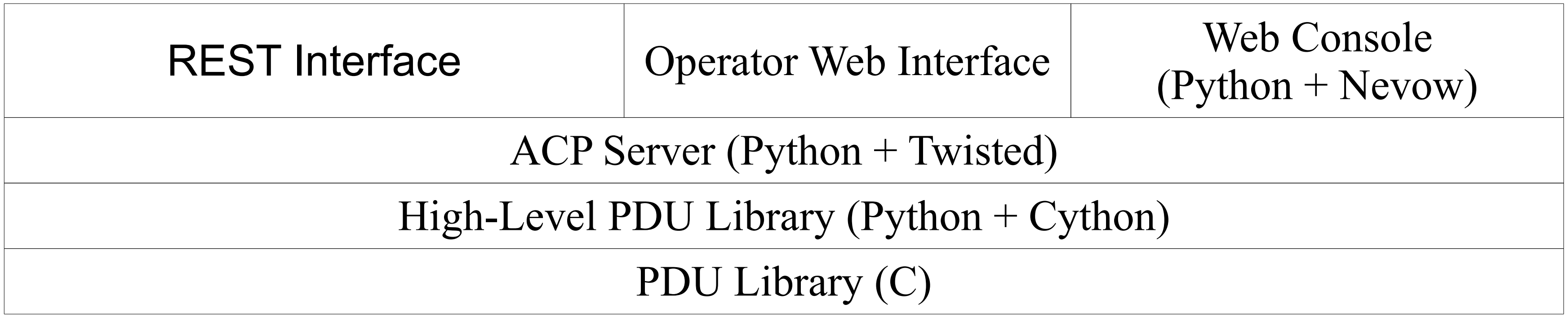
•*Understand the ACP 245 Protocol by analyzing message frames and flows.*

Automated Test Bench
(ie. third-party product or custom software).

• Enables automatic regression testing.

ACP 245 Suite Overview

The suite has a layered architecture, each layer using the layer that is directly underneath it, but without access to the lower layers.


REST Interface
Operator Web Interface
Web Console (Python + Nevow)
ACP Server (Python + Twisted)
High-Level PDU Library (Python + Cython)
PDU Library (C)

PDU Library
Defines the different type of messages supported by ACP245. This layer is responsible for parsing and writing frames into the corresponding data structures and providing these data structures to upper layers.

High-Level PDU Library
Provides an API for creating new ACP messages using Python, allowing for fast prototyping and simplifying the use of the protocol.

ACP Server
This is a general purpose server that handles transport of messages between the server, ACP clients, and provides gateways from other protocols to the ACP clients.

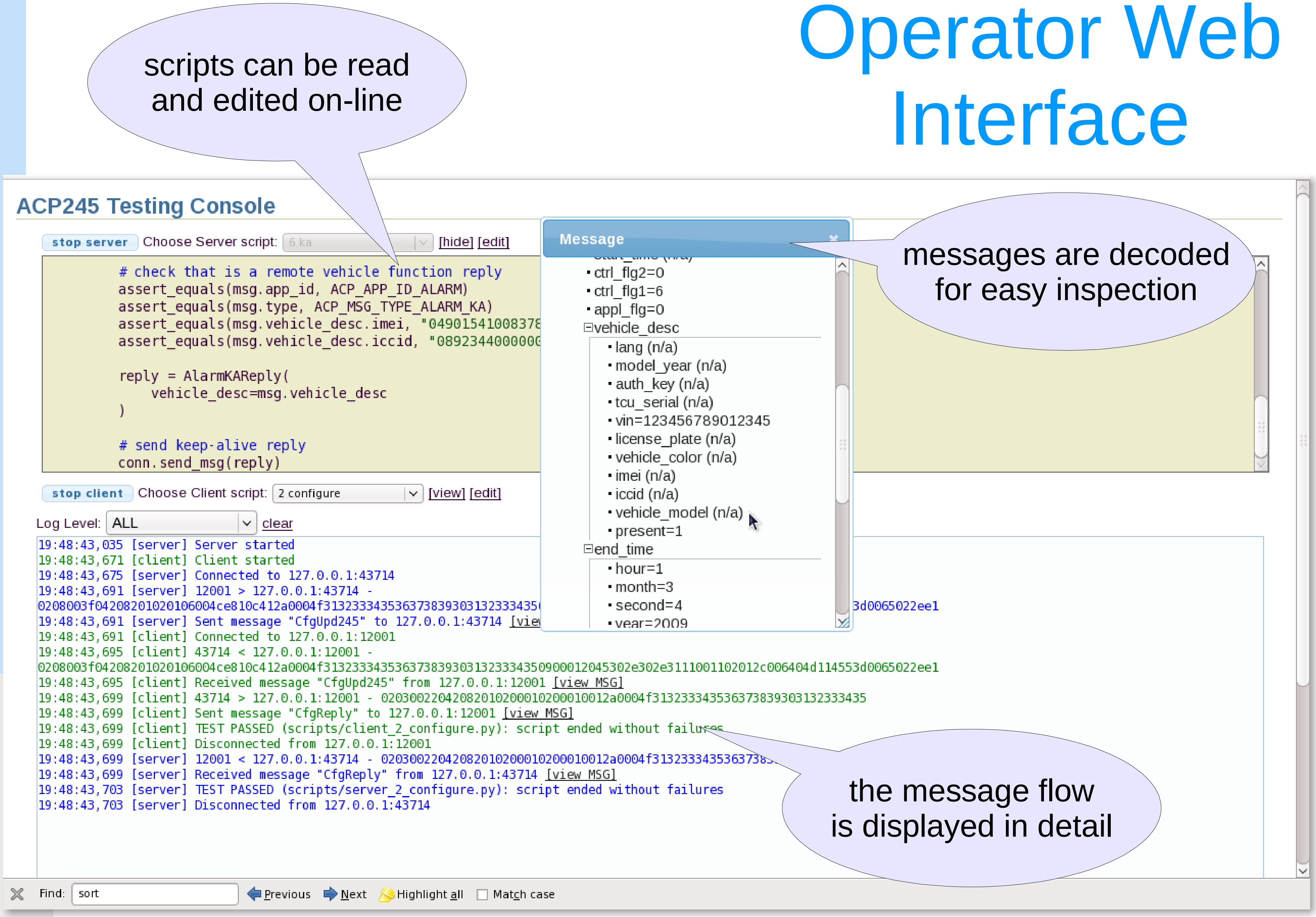
Web Console
The web console provides a testbench to execute predefined testing scripts on the server.

REST Interface to High-Level Library
Provides a REST interface to some functions exported by the library to be used by an automated test bench or third party product.

What are it's main features?

- **Web Interface** for operators
  - Can be accessed from anywhere, with a browser.
- **High-Level** Test language
  - Creating, sending and receiving ACP messages is as simple as it can be.
- **Predefined Tests** for common ACP flows
  - Defined by Edantech, run by the customer.
  - Can be **very easily** tailored to customer needs, if necessary.

Operator Web Interface


scripts can be read and edited on-line
messages are decoded for easy inspection
the message flow is displayed in detail

ACP245 Testing Console

stop server Choose Server script: 6 ka [hide] [edit]

```
        # check that is a remote vehicle function reply
        assert_equals(msg.app_id, ACP_APP_ID_ALARM)
        assert_equals(msg.type, ACP_MSG_TYPE_ALARM_KA)
        assert_equals(msg.vehicle_desc.imei, "04901541008378
        assert_equals(msg.vehicle_desc.iccid, "0892344000000

        reply = AlarmKAReply(
            vehicle_desc=msg.vehicle_desc
        )

        # send keep-alive reply
        conn.send_msg(reply)
```

Message

- ctrl_flg2=0
- ctrl_flg1=6
- appl_flg=0
- vehicle_desc
  - lang (n/a)
  - model_year (n/a)
  - auth_key (n/a)
  - tcu_serial (n/a)
  - vin=123456789012345
  - license_plate (n/a)
  - vehicle_color (n/a)
  - imei (n/a)
  - iccid (n/a)
  - vehicle_model (n/a)
  - present=1
- end_time
  - hour=1
  - month=3
  - second=4
  - year=2009

stop client Choose Client script: 2 configure [view] [edit]

Log Level: ALL clear

```
19:48:43,035 [server] Server started
19:48:43,671 [client] Client started
19:48:43,675 [server] Connected to 127.0.0.1:43714
19:48:43,691 [server] 12001 > 127.0.0.1:43714 -
0208003f04208201020106004ce810c412a0004f3132333435363738393031323334350                      3d0065022ee1
19:48:43,691 [server] Sent message "CfgUpd245" to 127.0.0.1:43714 [vie
19:48:43,691 [client] Connected to 127.0.0.1:12001
19:48:43,695 [client] 43714 < 127.0.0.1:12001 -
0208003f04208201020106004ce810c412a0004f3132333435363738393031323334350900012045302e302e3111001102012c006404d114553d0065022ee1
19:48:43,695 [client] Received message "CfgUpd245" from 127.0.0.1:12001 [view MSG]
19:48:43,699 [client] 43714 > 127.0.0.1:12001 - 02030022042082010200010200010012a0004f3132333435363738393031323334 35
19:48:43,699 [client] Sent message "CfgReply" to 127.0.0.1:12001 [view MSG]
19:48:43,699 [client] TEST PASSED (scripts/client_2_configure.py): script ended without failures
19:48:43,699 [client] Disconnected from 127.0.0.1:12001
19:48:43,699 [server] 12001 < 127.0.0.1:43714 - 020300220420820102000102000100 12a0004f31323334353637383
19:48:43,699 [server] Received message "CfgReply" from 127.0.0.1:43714 [view MSG]
19:48:43,703 [server] TEST PASSED (scripts/server_2_configure.py): script ended without failures
19:48:43,703 [server] Disconnected from 127.0.0.1:43714
```

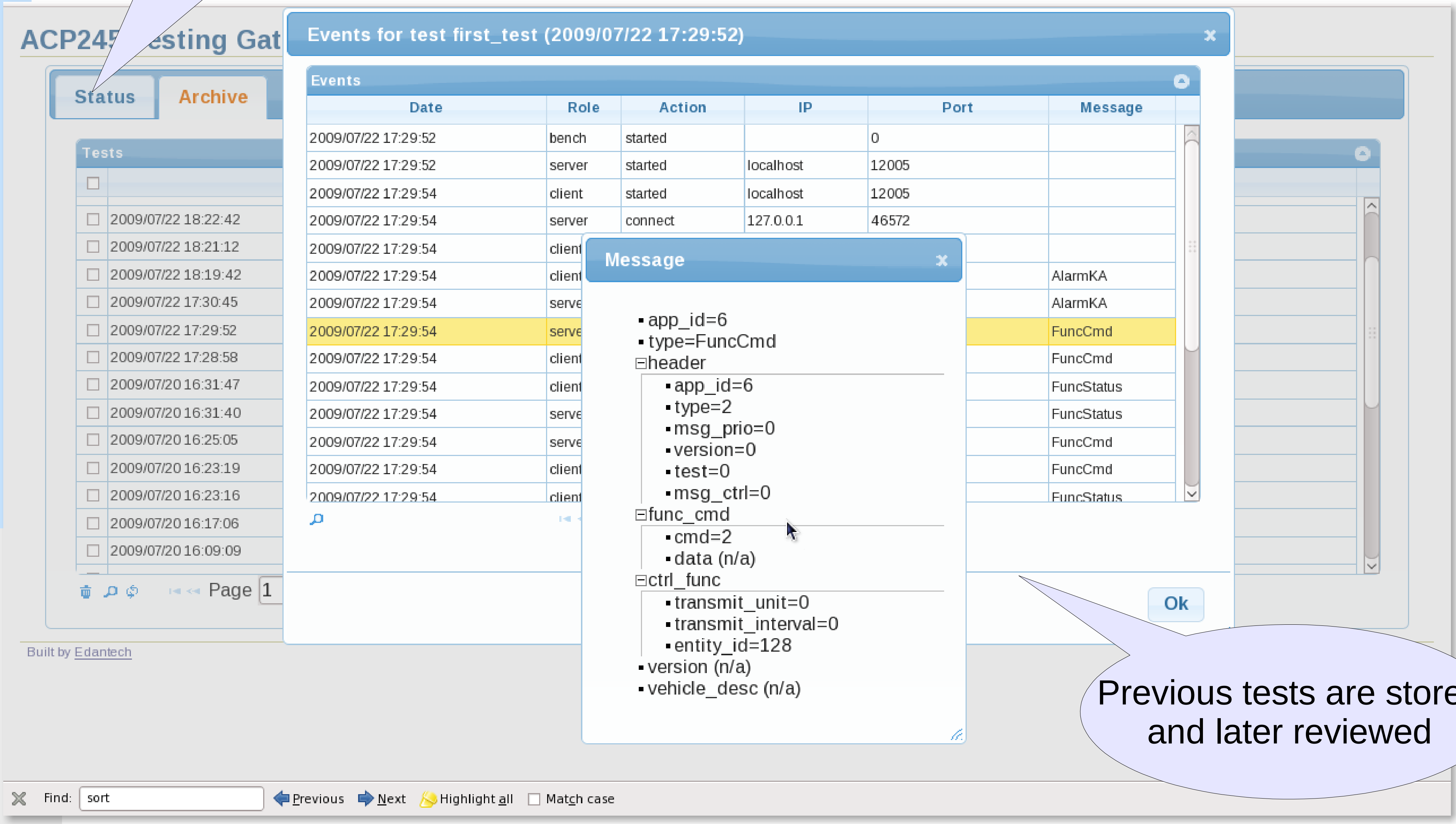
Find: sort Previous Next Highlight all Match case

More features

- **Integrates** with third party tools through the HTTP(s) protocol.
  - An external test bench can be used to write automated tests without having to understand the details of the ACP protocol.
  - Works as an ACP245-to-HTTP gateway.
- Third party interactions with the server are stored and can be later reviewed.
- In process of integrating it with **GPRS/GSM simulators** for end-to-end testing.

Gateway Management Console

Current server status can be supervised



2009/07/22 18:22:42
2009/07/22 18:21:12
2009/07/22 18:19:42
2009/07/22 17:30:45
2009/07/22 17:29:52
2009/07/22 17:28:58
2009/07/20 16:31:47
2009/07/20 16:31:40
2009/07/20 16:25:05
2009/07/20 16:23:19
2009/07/20 16:23:16
2009/07/20 16:17:06
2009/07/20 16:09:09

Events for test first_test (2009/07/22 17:29:52)

Events

Date	Role	Action	IP	Port	Message
2009/07/22 17:29:52	bench	started		0	
2009/07/22 17:29:52	server	started	localhost	12005	
2009/07/22 17:29:54	client	started	localhost	12005	
2009/07/22 17:29:54	server	connect	127.0.0.1	46572	
2009/07/22 17:29:54	client				
2009/07/22 17:29:54	client				AlarmKA
2009/07/22 17:29:54	serve				AlarmKA
2009/07/22 17:29:54	serve				FuncCmd
2009/07/22 17:29:54	client				FuncCmd
2009/07/22 17:29:54	client				FuncStatus
2009/07/22 17:29:54	serve				FuncStatus
2009/07/22 17:29:54	serve				FuncCmd
2009/07/22 17:29:54	client				FuncCmd
2009/07/22 17:29:54	client				FuncStatus

Message

- app_id=6
- type=FuncCmd
- header
  - app_id=6
  - type=2
  - msg_prio=0
  - version=0
  - test=0
  - msg_ctrl=0
- func_cmd
  - cmd=2
  - data (n/a)
- ctrl_func
  - transmit_unit=0
  - transmit_interval=0
  - entity_id=128
- version (n/a)
- vehicle_desc (n/a)

Previous tests are stored and later reviewed

Examples of Use

- **Customer** *outsources* development of ACP protocol to **Company A**
- **Company A** uses the ACP 245 Test Server to *diagnose and improve* it's protocol implementation
- **Customer** uses the ACP 245 Test Server HTTP interface to *integrate* his current test bench
- **Customer** uses his test bench to *perform automated regression tests* of **Company A** implementation and *TCU validation*

Examples of Use (cont.)

- TCU or Service Operator Emulator
  - Simple script is executed on ACP Test Server.
  - Test Console is used to diagnose communication problems
  - Test Console is used to check for errors on the implementation of the ACP Protocol

High-Level Library

Write ACP interactions easily

Service Operator (Server) sends a Message:

```
msg = FuncCmd(
    version=IEVersion(
        car_manufacturer=0x08,
        tcu_manufacturer=0x82,
        major_hard_rel=1,
        major_soft_rel=3
    ),
    ctrl_func=IECtrlFunc(
        entity_id=ACP_ENT_ID_IMMOBILIZE,        # 0x0A
    ),
    func_cmd=IEFuncCmd(
        cmd=ACP_FUNC_CMD_ENABLE,                # 0x02
    ),
    vehicle_desc=IEVehicleDesc(
        iccid="08923440000000000003",
    ),
)
conn.send_msg(msg)
```

High-Level Library (cont.)

TCU (Client) receives message and replies:

```
# Get received messsage
msg = conn.pop_msg()

# Create Reply
reply = FuncStatus(
    # Copy msg elements to reply
    version = msg.version,
    ctrl_func = msg.ctrl_func,
    vehicle_desc = msg.vehicle_desc,
    func_status = IEFuncCmd(
        cmd=ACP_FUNC_STATE_ENABLED # 0x02
    ),
)

# Send reply
conn.send_msg(reply)
```

Communication Monitoring

- Previous interaction can be monitored using the Test Console

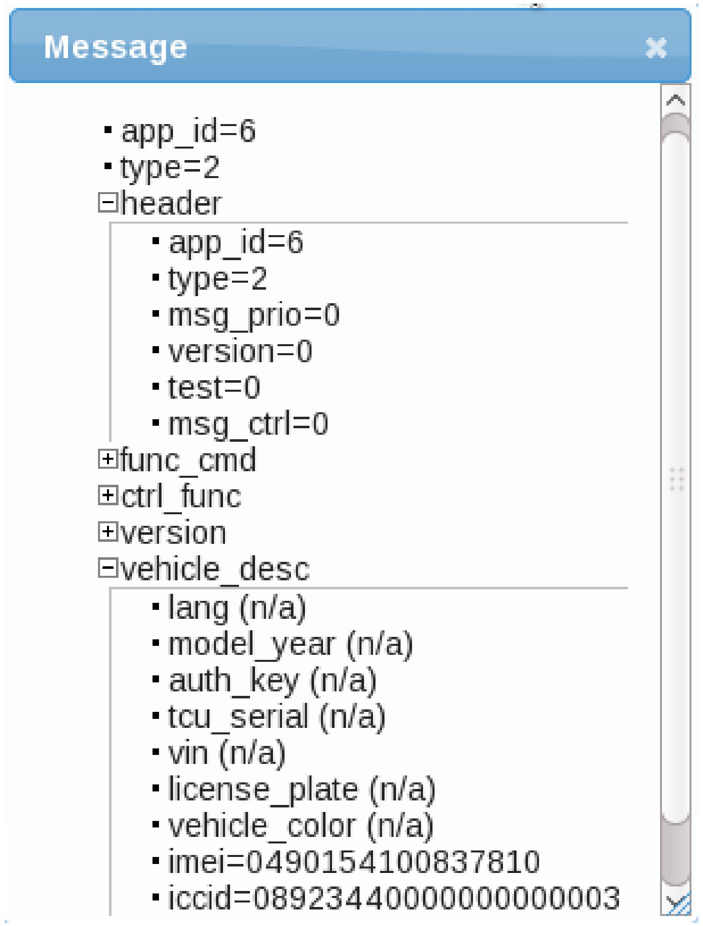
Example:
Server listens at port 20001
Client connects from port 50797

```
19:20:52,975 [server] Server started
19:20:53,422 [client] Client started
19:20:53,423 [server] Connected to 127.0.0.1:50797
19:20:53,425 [server] 20001 > 127.0.0.1:50797 -
060200270408820103030a003c0202001681208804901541008378108a08923440000000000003
19:20:53,426 [server] Sent message "FuncCmd" to 127.0.0.1:50797 [view MSG]
19:20:53,426 [client] Connected to 127.0.0.1:20001
19:20:53,428 [client] 50797 < 127.0.0.1:20001 -
060200270408820103030a003c0202001681208804901541008378108a08923440000000000003
19:20:53,429 [client] Received message "FuncCmd" from 127.0.0.1:20001 [view MSG]
19:20:53,430 [client] 50797 > 127.0.0.1:20001 – 0
6030029040882010303 0a003c02020001001681208804901541008378108a08923440000000000003
19:20:53,431 [client] Sent message "FuncStatus" to 127.0.0.1:20001 [view MSG]
19:20:53,431 [client] TEST PASSED (activate_immo): script ended without failures
19:20:53,432 [client] Disconnected from 127.0.0.1:20001
19:20:53,432 [server] 20001 < 127.0.0.1:50797 -
0603002904088201030 30a003c02020001001681208804901541008378108a08923440000000000003
19:20:53,434 [server] Received message "FuncStatus" from 127.0.0.1:50797 [view MSG]
19:20:53,447 [server] TEST PASSED (activate_immo): script ended without failures
19:20:53,447 [server] Disconnected from 127.0.0.1:50797
```

Simple Message Decoding

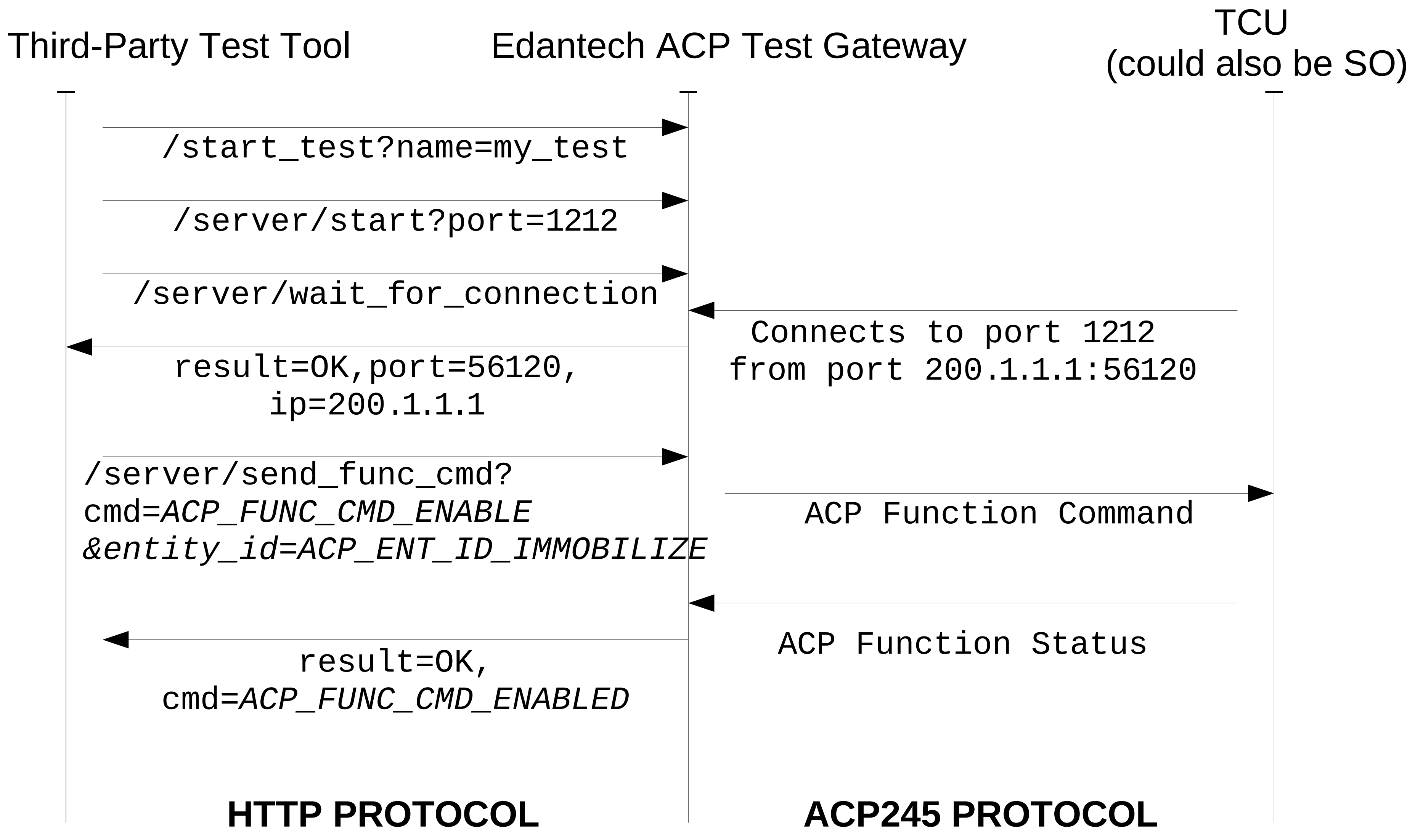
- Click on message to see it's structure

19:20:53,429 [client] Received message "FuncCmd" from 127.0.0.1:20001 [view MSG]



```
Message
  ▪ app_id=6
  ▪ type=2
  ⊟header
     ▪ app_id=6
     ▪ type=2
     ▪ msg_prio=0
     ▪ version=0
     ▪ test=0
     ▪ msg_ctrl=0
  ⊞func_cmd
  ⊞ctrl_func
  ⊞version
  ⊟vehicle_desc
     ▪ lang (n/a)
     ▪ model_year (n/a)
     ▪ auth_key (n/a)
     ▪ tcu_serial (n/a)
     ▪ vin (n/a)
     ▪ license_plate (n/a)
     ▪ vehicle_color (n/a)
     ▪ imei=0490154100837810
     ▪ iccid=08923440000000000003
```

HTTP Gateway Example



ACP245 Embedded Library

- The ACP 245 protocol implementation used by the ACP 245 Test Server.
- Written entirely in **portable** ANSI C.
- Designed for memory constrained devices.
- Running on Wavecom processors on real-time Open AT, and Intel processors running Linux and Windows.
- Provides **High-Level** bindings to other languages (*Python* supported at the time).

QA Process

The ACP 245 library is extensively tested.

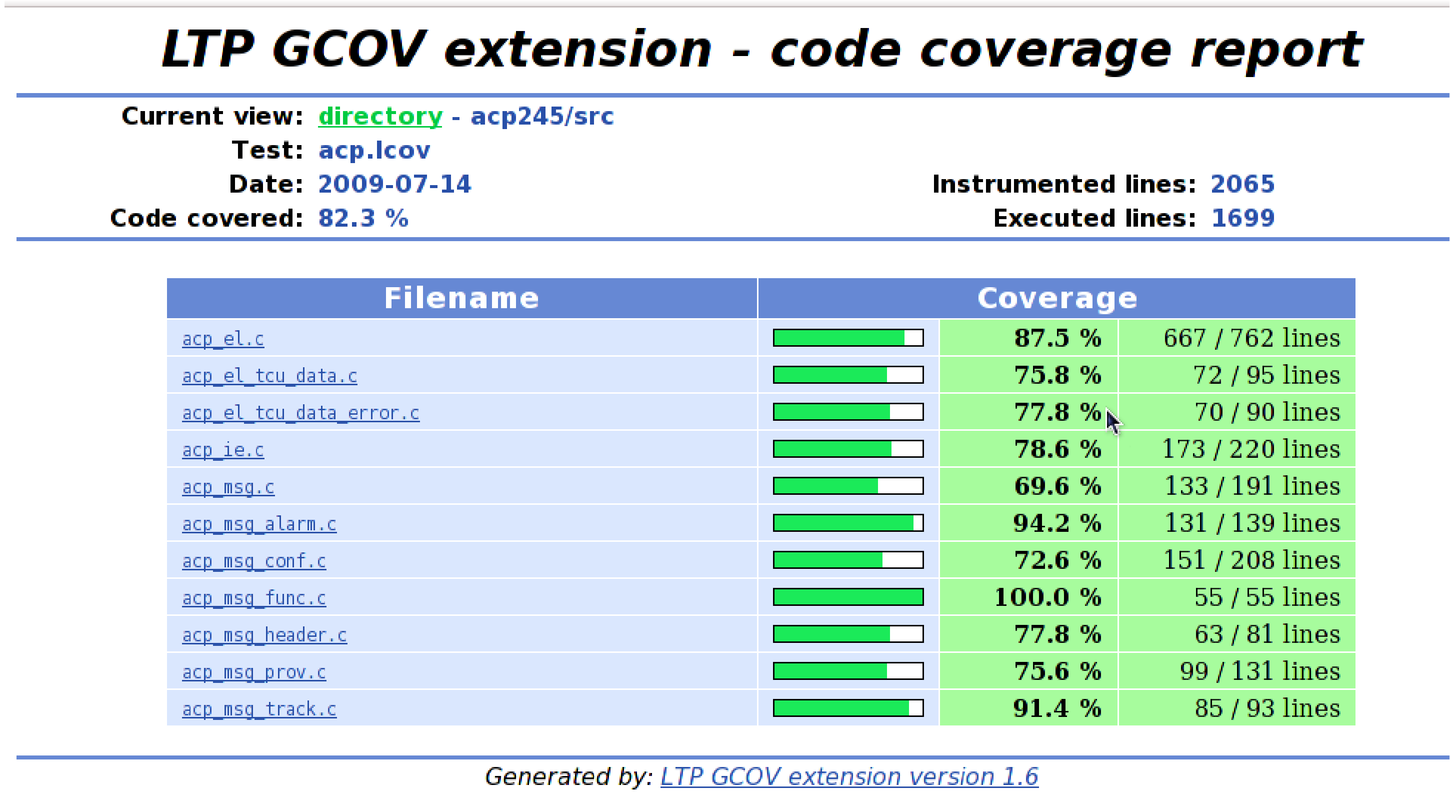
- Different test types:
  - **Unit testing:** each function tested independently
  - **Static and Dynamic code validation:** code analyzers to check for bugs, dynamic validation to check for memory handling errors
  - **Thrash testing:** input random data to try to crash the implementation
  - **Multiple test levels:** Server tests test high-level library, which tests low-level implementation.

QA Process (cont).

- Controlled and packaged releases
  - Version Control System
  - Bug Tracking System
    - Could be made public to simplify implementation testing
  - Released as a Linux Package (RPM) and Windows shared Library (.dll)
- First **Stable** Release: 23/03/09.
  - Updated to latest ACP 245 1.2.2 specifications.

QA Process Tools

- Dynamic Checking with valgrind: 0 errors
- GCC: 0 warnings in strict mode (-Wall + extra)
- Unit Test Coverage Analysis



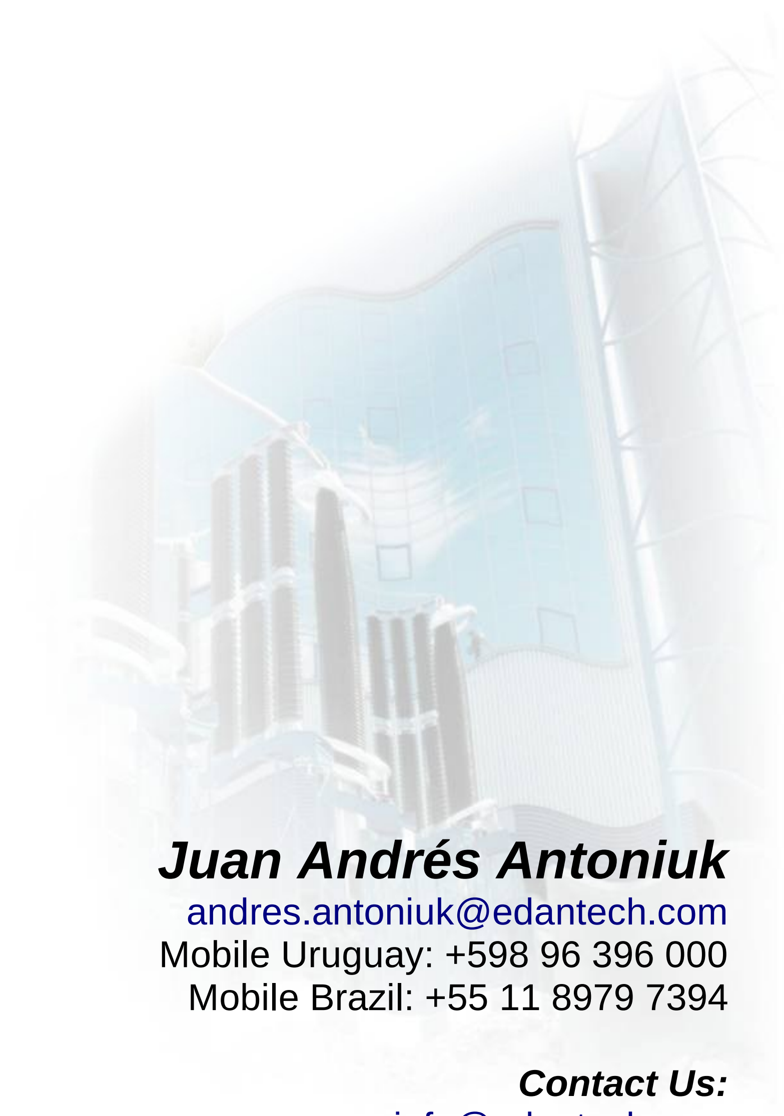
LTP GCOV extension - code coverage report

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

Thanks



Juan Andrés Antoniuk
andres.antoniuk@edantech.com
Mobile Uruguay: +598 96 396 000
Mobile Brazil: +55 11 8979 7394

Contact Us:
info@edantech.com
http://www.edantech.com

Av. Libertador 1807, 11.800
Montevideo, Uruguay
+598 2929 0029

edantech