Custom Test Scripts
Documentation

CONFIDENTIAL

edantech

Revision: 1.0

Date: 2010-01-15

Revisions

1	01/15/10	Initial Release.

1 Copyright Notice and Trade marks

Custom Test Scripts

Documentation

Document Version: 1.0

Document, Release Date, January 15th, 2010

EDANTECH has prepared the information contained herein and are for the use of its employees, consultants and the recipient employees.

The information contained in this document is confidential and is submitted by EDANTECH on the understanding that it will be used only by the employees or consultants to Customer, and that where consultants are employed, the use of this information is restricted to use in relation to the business of Customer.

Dissemination of the information and/or concepts contained herein to other parties is prohibited without the prior written consent of EDANTECH.

Trademarks

All names, products, and trademarks of other companies used in this document are the property of their respective owners.

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

2

3 References

[ACP245]	ACP 245 – Application Communication Protocol v 1.2.2 http://www.denatran.gov.br/download/ACP_245_V_1_2_%2009_09_09.pdf

4 Purpose

The following document provides SDL diagrams that describe the behavior of some of the custom test scripts.

5 Scope

The provided diagrams focus on the message flow. In general, to know which elements are included on each message you will need to consult the scripts directly.

6 Description

The diagrams were written using SDL (Specification and Description Language). Each script also has a set of parameters which are defined before their entry function, properly documented on the scripts. They are also provided in this document for reference.

6.1 SDL

The selected modeling language to write this diagrams is a simplified SDL (ITU Specification and Description Language).

SDL was originally defined on 1980, and later updated and standarized during following years, and currently managed by the SDL Forum Society[1]. It's used mainly by the telecommunications industry, but it's also being applied to other areas, like real-time embedded devices. Is often used by Edantech and other companies to model complex event-based state machines, for protocol and firmware development.

The [ACP245] document provides some sequence diagrams to describe possible interactions between a TCU and a SO (TIV). Sequence diagrams are useful to show independent message exchanges, when each exchange can be modeled as a sequential process. However, when model asynchronous events (like a timer expiration, or the reception of a message that can be received under multiple conditions, such as a keep alive), sequence diagrams fail to capture the behavior of the protocol.

For this reason, we have chosen to represent the scripts using a simplified form of the SDL language. We have not been strict on the use of the language, preferring clarity and simplicity over formality.

[1] http://www.sdl-forum.org/

7 Scripts

The following diagram shows the main SDL symbols used to model the scripts:

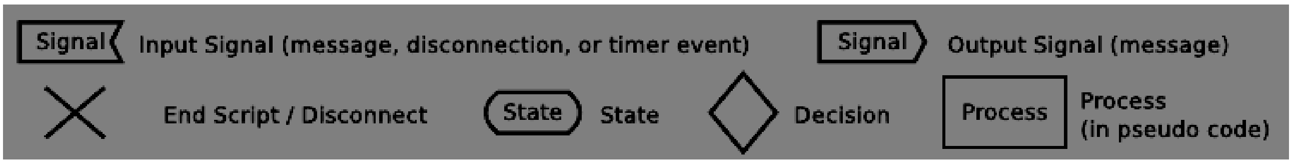

Figure 1: SDL main symbols

For each script, we provide a copy of their parameter documentation, as given on the script itself. To modify a parameter, you will need to edit the script and change the value assigned to the parameter on the script header.

7.1 client: emulator

7.1.1 Parameters

```
vehicle_desc = IEVehicleDesc(iccid='123456')
```

```
version = IEVersion()
```

- Define the vehicle descriptor and version to use on every message sent to the TIV

```
send_version = True
```

- If send_version == True, messages will contain the version element, otherwise the version element will not be included (or length will be 0)

```
send_vehicle_desc_in_ka = False
```

- If send_vehicle_desc_in_ka == True, keepalive messages will contain a vehicle descriptor element

```
start_in_event = False
```

- If start_in_event == True, TCU starts on event mode

```
start_coords = [-23.55,-46.63]
```

- Starting coordinates for TCU

```
time_to_wait_before_first_msg = 4
```

- Number of seconds to wait before sending the first message . Used to verify that the TIV does not close the connection for at least this number of seconds, in case the TCU is taking some time to send the first message.

```
time_to_run = 300
```

- Time to run the script (in seconds)

```
expect_reply = True
```

- If True, the TCU will mark it's TrackPos, AlarmKA and AlarmNotif as expecting replies

7.1.2 SDL Diagram

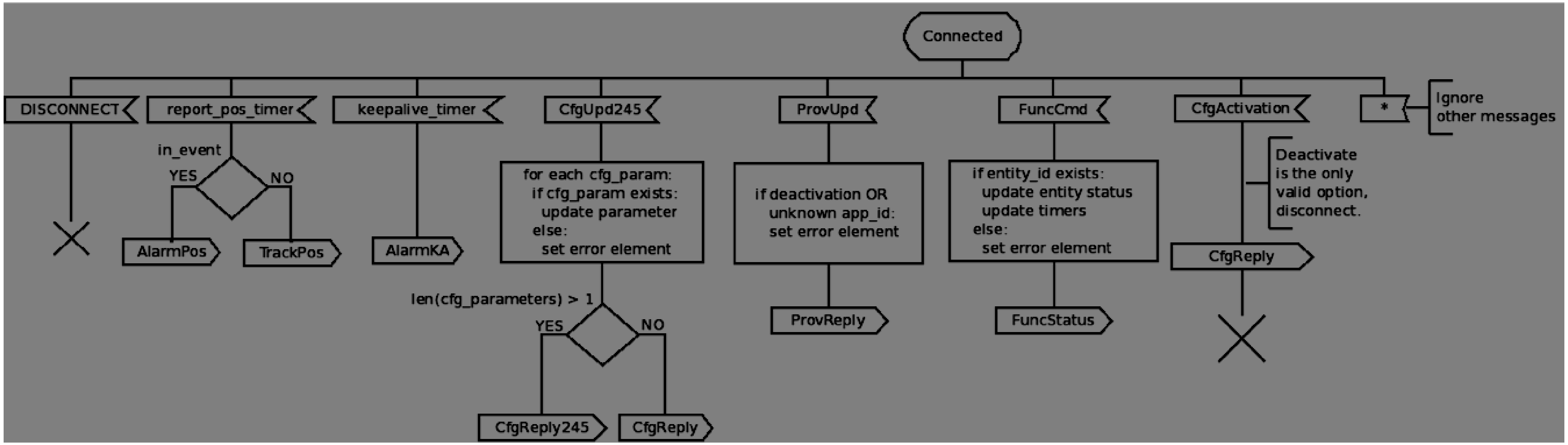

Figure 2: Client emulator SDL diagram

7.2 server: emulator

7.2.1 Parameters

```
send_version = True
```

- If send_version == True, messages will contain the version element, otherwise the version element will not be included (or length will be 0)

```
time_to_wait_for_first_message = 120
```

- Number of seconds to wait for the first message before failing the test because no message was received

```
send_vehicle_desc_in_ka_reply = False
```

- If send_vehicle_desc_in_ka_reply == True, keepalive reply messages will contain a vehicle descriptor element

```
send_track_pos_reply = True
send_alarm_notif_reply = True
send_alarm_pos_reply = True
send_alarm_ka_reply = True
```

- If send_<msg>_reply == True, a reply to keepalive messages will be always sent. Otherwise, it will be sent only if the reply expected flag is set on the message header.

7.2.2 SDL Diagram

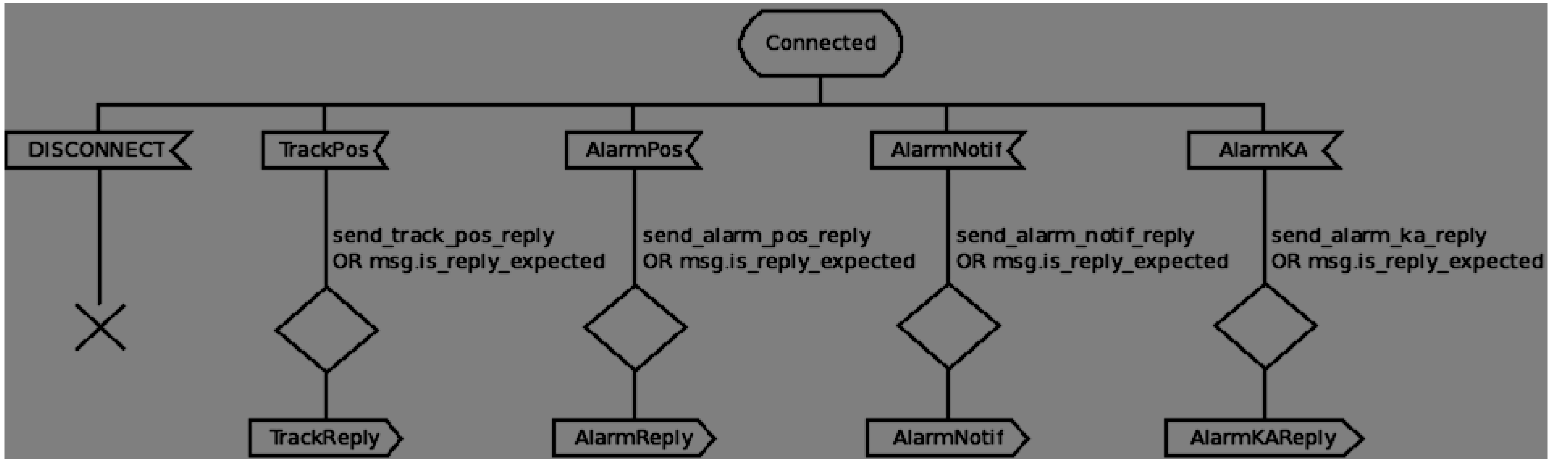

Figure 3: Server emulator SDL diagram