OSDP Conformance Testing

Setting up the Test Environment

1.1 set up for **calibration.** 3 stations on the rs-485 bus. See appendix for configuration of Pi. These run the open source conformance tool (on github, securityindustryassociation/libosdp-conformance.)

1 Pi as the PD
1 Pi as the CP
1 Pi as the Monitor

1.2. With **tracing** enabled at the monitor, start the PD, then run the CP conformance test on the other PI. Save the trace. Confirm the trace shows what you expect. Confirm the conformance results are as you expect.

1.3 If you've got **your own test equipment**, perhaps now is a good time to set that up with a Pi as a monitor and confirm you see the expected signal.

3. Remove the reference PD. Insert **your DUT** as the reader. Re-run the tests. If it doesn't pass, check the monitor log as well as the logs from the conformance tool running the test (as the CP.)

Appendix

A. Pi configuration

PI 2b, 16 gig sd card
SIA-provided image. Devuan-based linux with web and SSH (and console, via keyboard and HDMI monitor)

configuration:
10.0.0.201, 202, 203 (three images built for ISC West 2017.)

B. Software Configuration

It's a C program. It uses a stock RS-485 USB converter, which shows up in linux as a conventional "tty" (/dev/ttyUSB0.) It's built with clang or gcc. The configuration and control details are passed using JSON files. It assumes Jansson is present. If built for TLS it assumes gnutls is present. It uses a CGI/HTML UI, it assumes apache2 is installed. There are command line tools for everything so you can run it from a command line (JSON files are pushed into the running OSDP process.) It runs with Devuan 1.0. It's in active use on Pi's and as virtualbox virtual machines (amd64, 512meg.) It's meant to use mainstream Linux tools and should port reasonably to Debian or Ubuntu or other Linux environments. It uses Posix 'select' calls and file I/o. For TLS, it uses Gnutls (although experimental ports have been done to Bearssl and other environments.)