Hardware Simulation - VENUS

VENUS = VENdor Unified Simulation

This page deals with industrial controls simulation of power supplies from the vendors CAEN, ISEG and Wiener - the VENUS project. VENUS consists of shared libraries (modules for each vendor) and a simulation engine which can replace the real hardware access interface and provide a software-only simulation of the power supplies. Installation and switching between real and simulated mode is straightforward for the developer and fully transparent for the OPC server.

Essentially VENUS consists of 3 modules:

- the glue code - a library replacing the real hardware access API for each vendor, running loaded inside the OPC-UA server, which communicates with the simulation engine
- the simulation engine - a standalone program which exhibits the behavior and interfaces of the simulated power supplies, including ramps, trips, etc
- the pilot UI - a standalone program which connects directly to the simulation engine in order to manage and fine-tune behavior like i.e. setting channel loads and provoking trips
- a fourth stand-alone module to run an independent CAEN discovery and dump it as a configuration (xml) , cc7 binary.

A full introduction (The "Basics Slides") is [here](#) and there is also the first [JCOP project proposal](#).

Presently (Nov 2017) the CAEN module is available, the ISEG is in progress and the Wiener is not yet started.

The two schematics below show a normal production environment, and the replacement of the vendor electronics with a simulation engine.

### Production running of OPC-UA servers

```
| OPC-UA binaries: vendor specific servers made in CERN, ICS-CIC |
| Software libs: vendor specific interfaces & abstraction ©vendors, collaboration |
| “vendor separation line”: C/C++ dynamically linked libraries |
| Distributed communication, complex vendor specific behaviour |
| vendor specific electronics: complex large scale systems in the experiments |
```

| Clients to OPC-UA: WinCCOA, uaexpert, .... (all upper layers) |
| OPC-UA (CAEN) |
| CAEN API/lib |
| (….) |
| OPC-UA (ISEG) |
| ISEG API/lib |
| (….) |
| OPC-UA (Wiener) |
| Wiener API/lib |
| (….) |

### Simulation running of unmodified OPC-UA servers

```
functionality

- runtime environment is CERN CentOS 7.2
- CAEN power supplies and EASY crates can be simulated. One simulation engine is needed for one controller, connecting to one OPC-UA server. The set/get of parameters, ramping up/down, event mode and tripping behavior are simulated. When the pilotUI is used additionally, realistic tripping and load scenarios can be configured.
- ISEG power supply integration is in progress (Nov 2017).
- Wiener has not yet started

prepare the simulation

1. Make up your mind WHAT you want to simulate, the s.engine/OPC server need an xml-configuration which represents the hardware (power supplies CAEN, ISEG...) to be simulated. We can simulate configurations which do not yet exists (synthetic), which are unusual (stress test, debugging) or which are more or less precise copies of existing infrastructures (clones). Contact me if in doubt.
2. Decide HOW you want to simulate
   2.1. as a service which I provide: I'll set up a machine for you, and you just get an opc.tcp://<ipnumber>:4841 URL to speak to which fakes your electronics. This is ideal for client development and larger-scale simulations. This will usually be set up as a running service as a docker-swarm.
   2.2. run-it-yourself as a package: You download and setup a s.engine and an OPC server yourself step-by-step and care for it yourself (please report bugs ;-) 
   2.3. run-it-yourself as an OPC-UA server developer: You just run a "naked" s.engine, and are developing your own OPC server/server layer on top of it (please report bugs ;-) 
3. decide if you need fine tuning or not. If you want to provoke i.e. trips, your channels will have to produce a certain current, therefore you need to set up channel loads. Maybe you need to (re-)play a very specific scenario or a sequence of operation, for testing purposes. For these features you need to run the pilot-UI which connects directly "under the hood" to the s.engine to set up the according parameters. If you are happy without these features you don't need to worry about the pilot-UI.

run the simulation

knowing the bins

- simulationEngine: runs as a stand-alone process . It mimicks the behaviour of CAEN power supplies following a general model of a tree of power supplies (the CAEN API behaviour)
- the glue-code library libgluecaen.so (loader part1) which replaces the CAEN hardware lib and loads the simulation glue code
- the simulation adaptor libsimcaen.so (loader part2) which connects to the s.engine via google-protocol zmq messaging both libs are loaded by the OPC-UA caen server, instead of the CAEN hardware access libs from the vendor.
- simulationPilotUI: runs as a standalone process, interfacing directly with the simulation engine. Just call it with -h, it will tell you what parameters it needs. This can be obviously scripted for many calls to provide a specific simulation scenario which you need to repeat, i.e. setting up loads and provoking trips for many channels, which can normally NOT be achieved by the SCADA clients.

- a CAEN OPC-UA Unified Automation server snapshot (binary, licensed) can be taken from jenkins (i.e. build 939). For full compatibility to clients and slightly better speed you should run the .ua server-flavour, for a fully open-source you can try the .open6. The s.engine and glue code does not make any difference between them.

**directories:**

opc/bin: binary of the OPC server, from a stable jenkins build. Run the OpcUaCaenServer.ua (Unified Architecture for WinCCOA 3.11 compatibility)

opc/bin/CAEN/hwAccess/lib: glue code lib loader (part1)

lib: glue code functional (part2) and dependent libs, essentially xml2, protocol, zeroMQ

sim: binary of the simulation engine plus a few example configurations

tools: the caen standalone disvovery tool, the simulationPilotUI

---

**run-it-yourself-as-a-package**

**CC7 ICS-VM or workstation or openstack VM: CAEN s.engine & CAEN .ua server (a VENUS Combo)**

- import the ccc7 cernbox into oracle's virtualBox, running under your host system, boot the VM
- no need to log into the VM, just open a terminal, make an empty dir <mydir> and cd into it

1. download and deflate the latest version (Dec2017: 0.9.1) of the VENUSCombo-tarball from here (nexus) into a new directory (this is your base directory `pwd`):
   a. wget <tgz>; tar -xvzf <tgz>
   b. this creates sub-directories ./lib, ./opc, ./sim, ./tools

2. run the simulation engine, and give it a hardware configuration. Use i.e. an existing synthetic setup (1 controller, 16boards, each 20channels):
   a. add the library path if needed, so that the bins can find their specific libs:
   b. export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/lib
   c. cd ./sim; ./simulationHAL -cfg ./sim_config_1c_16b_20h.xml (or use your own config)

3. run the OPC server, open an extra terminal. Take the production flavour based on the UA toolkit, and let the server discover the hardware of the simulation engine:
   a. add the library path also in the new terminal/shell:
   b. export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/lib
   c. cd ./opc/bin; ./OpcUaCaenServer.ua --hardwareDiscovery

4. connect some clients
   a. you might need to install some missing libs (sudo yum install) depending on your system, but all specific libs are shipped
   b. now you can connect an OPC client, i.e. uaexpert, to opc.
   c. you can also use the pilotUI to i.e. ramp up all channels or provoke a trip (add the lib path if needed)
      i. add the library path also in the new terminal/shell:
      ii. export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/lib
      iii. cd ./tools/; simulationPilotUI -h (more on the pilot here)

---

Other VenusCombos for ISEG, Wiener are in development.

**run-it-yourself with your own OPC server (as OPC developer)**

follow steps 1. and 2. from above. Use the simulationPilotUI to check the simulation engine if needed.

3. run your own OPC server, on the same host or on your own development station.

You must use the glue code libs instead of the CAEN hardware access. Replace ./opc/bin/CAEN/hwAccess/lib/libcaenhwwrapper.so.5.82 by a symlink and point to the 1st stage glue code lib in ./lib/libgluecaen.so:

```bash
lnwxwxwx 1 scada scada 14 Nov 22 17:05 libcaenhwwrapper.so.5.82 -> libgluecaen.so
lnwxwxwx 1 scada scada 14 Nov 22 17:05 libcaenhwwrapper.so.5.83 -> libgluecaen.so
lnwxwxwx 1 scada scada 30 Nov 22 17:05 libgluecaen.so -> ../lib/libgluecaen.so
```

If you run on a different host, set up also the glue code to find the simulation engine. Modify ./opc/bin/caenGlueConfig.xml attribute HALIP to point to the host where the simulation engine is running:

```xml
<caenGlue HALIP="127.0.0.1" HALPORT="8880" DEBUGLEVEL="0"></caenGlue>
```

---

**further documentation**

egroup: caen-hw-simulation@cern.ch
JIRA issues: jira  project = OPCUA AND component = "Industrial CAEN Power Supply Simulator"

doxygen code
- caen simulation engine
- caen glue code
- caen pilot UI

comments, questions: michael.ludwig@cern.ch 163095