DEVICE CONTROL DATABASE TOOL (DCDB)

Pavel Maslov, Matej Komel, Klemen Žagar, Cosylab, Ljubljana, Slovenia

Abstract

In a physics facility containing numerous instruments, it is advantageous to reduce the amount of effort and repetitive work needed for changing the control system (CS) configuration: adding new devices, moving instruments from beamline to beamline, etc. We have developed a CS configuration tool, which provides an easy-to-use interface for quick configuration of the entire facility. It uses Microsoft Excel as the front-end application and allows the user to quickly generate and deploy IOC configuration (EPICS start-up scripts, alarms and archive configuration) onto IOCs; start, stop and restart IOCs, alarm servers and archive engines, etc. The DCDB tool utilizes a relational database, which stores information about all the elements of the accelerator. The communication between the client, database and IOCs is realized by a REST server written in Python. The key feature of the DCDB tool is that the user does not need to recompile the source code. It is achieved by using a dynamic library loader, which automatically loads and links device support libraries. The DCDB tool is compliant with CODAC (used at ITER and ESS), but can also be used in any other EPICS environment.

DCDB ARCHITECTURE

The DCDB-tool uses MySQL relational database together with the BLED [1] schema (a set of tables representing the whole facility).

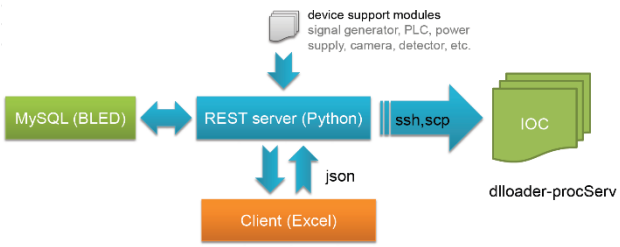


Figure 1: DCDB architecture.

The backend is a typical web-server, which is realized with a combination of the following python modules: flask-restful (REST server), sqlalchemy and pymysql (database communication layer), and paramiko (ssh). The front-end is a Microsoft Excel plugin written in C# using .NET technology. IOCs are Linux machines running EPICS and procServ [2]. The client-server communication is based on the exchange of JSON objects (strings).

DEVICE SUPPORT MODULES

Device support modules are created using standard commands that come with the Maven ITER plugin and Java API for managing CODAC development unit life-cycle (packages m-iter-plugin, m-codac-unit-api), supplemented by dlloader support (Fig.2).

```
bled@bled:~$ mvn newunit -Dunit=m-BeamPositionMonitor
bled@bled:~$ cd m-BeamPositionMonitor
bled@bled:~$ mvn newdlloader
bled@bled:~$ mvn clean compile test package
```

Figure 2: The procedure of creating EPICS device support libraries.

Device support modules should contain a library (lib/*.so), an EPICS database definition file (dbd/*.dbd files), an EPICS database template/substitution file (db/*.db) and three scripts: init.cmd (contains the string *require <module>, <version>*), init-pre.cmd (a set of configuration fields or macros to be extracted into the BLED database, plus epics shell commands to be executed before IOC initialization) and init-post.cmd (epics shell commands to be executed after IOC initialization).

```
.
├── db
│   └── BeamPositionMonitor.db
├── dbd
│   └── BeamPositionMonitor.dbd
├── init.cmd
├── init-post.cmd
├── init-pre.cmd
└── lib
    └── linux-x86_64
        ├── libBeamPositionMonitor.a
        └── libBeamPositionMonitor.so
```

Figure 3: Files to deploy.

In order to load the information about the device support module into the BLED database, use the *bled* helper script that you should invoke from the module's $(TOP) directory. It will extract the information about the module stored in the pom.xml and init-pre.cmd files and send it to the REST server over the network.

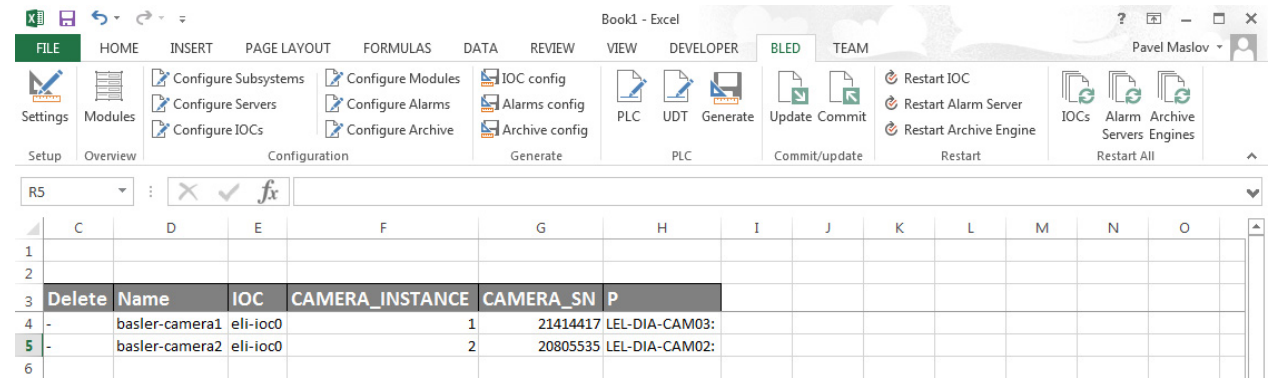
DYNAMIC LIBRARY LOADER

Dynamic library loader (or dlloader)[1] is an EPICS-based tool that allows you to load EPICS device support libraries by just adding its' definitions in the startup script (st.cmd). Hence, the integrator is freed from the necessity of compiling (recompiling) their EPICS applications/IOCs.

The DCDB-tool uses dlloader to dynamically load device support modules (described in the previous chapter). As it was already mentioned there is no need to recompile the source code, since all we have to provide is a st.cmd script, which in turn is generated by the REST server.

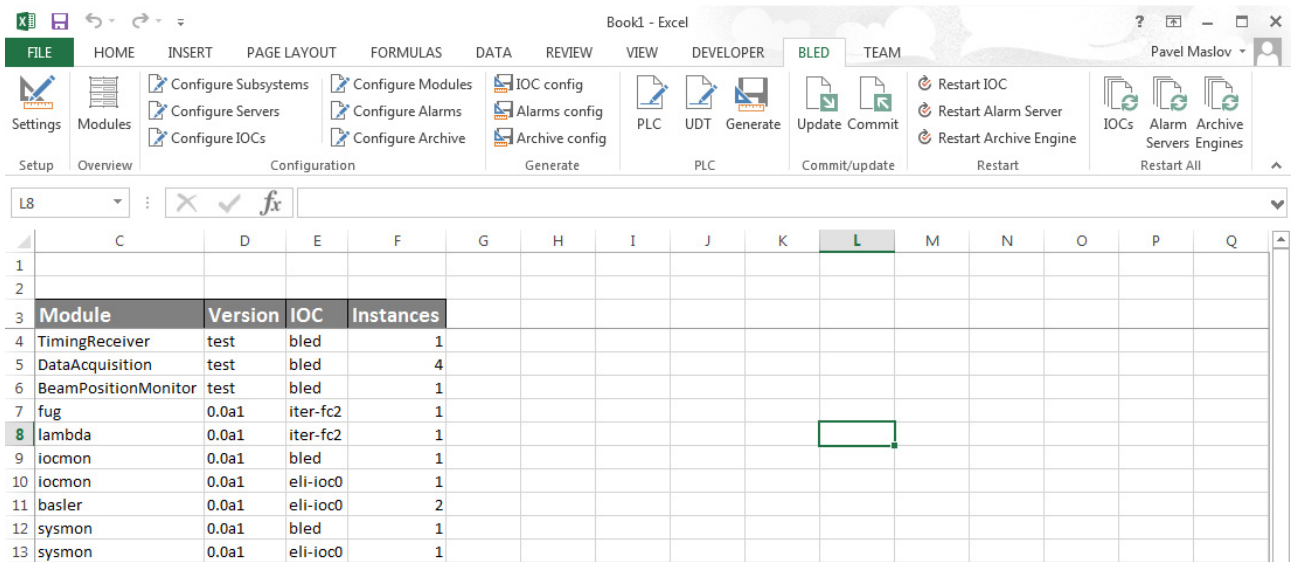
1. The concept of dynamically loadable device support modules, including the require function is developed by Dirk Zimoch (PSI).

ISBN 978-3-95450-146-5



Delete	Name	IOC	CAMERA_INSTANCE	CAMERA_SN	P
-	basler-camera1	eli-ioc0	1	21414417	LEL-DIA-CAM03:
-	basler-camera2	eli-ioc0	2	20805535	LEL-DIA-CAM02:

Figure 4: Front-end application (adding device instances).



Module	Version	IOC	Instances
TimingReceiver	test	bled	1
DataAcquisition	test	bled	4
BeamPositionMonitor	test	bled	1
fug	0.0a1	iter-fc2	1
lambda	0.0a1	iter-fc2	1
iocmon	0.0a1	bled	1
iocmon	0.0a1	eli-ioc0	1
basler	0.0a1	eli-ioc0	2
sysmon	0.0a1	bled	1
sysmon	0.0a1	eli-ioc0	1

Figure 5: Front-end application (showing all modules registered in the database).

FRONT-END

The user communicates with the REST server via an HMI, which is realized as an Excel add-in (ribbon). It provides a set of buttons, using which you can easily edit your support modules' configuration (Fig. 4), configure IOCs, alarm servers and archive engines; start/stop/restart IOCs, and more.

Figure 4 shows an Excel configuration sheet for the Basler camera device support module with two camera instances running on the *eli-ioc0* IOC.

Another interesting feature of the DCDB-tool is when you click on the Modules button (Fig.5, top left corner), you can see the information about all the device support modules registered in the BLED database: which versions and which IOCs they are activated on, plus the number of instances. This can be handy when you want to see an overview of what is deployed on which IOC and in which quantities.

Along with the hardware modules, you can also have "soft modules" deployed with dlloader support (notice *iocmonitor* and *sysmonitor* in Fig.5).

The front-end application was tested with Microsoft Excel 2007 and 2013.

SUMMARY

The DCDB tool is a powerful control system configuration tool that saves a lot of integration effort (and thus, time). It is developed with best practices from the EPICS community, compliant with CODAC core system (used at ITER, ESS, ELI-NP), but can also be used in any other EPICS environment.

ACKNOWLEDGEMENT

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 289485.

REFERENCES

[1] BLED – system for central management of the accelerator data (ESS, Cosylab d.d.)
[2] procServ – Process Server (written by Ralph Lange); http://sourceforge.net/projects/procserv/

ISBN 978-3-95450-146-5