DEVICE CONTROL DATABASE TOOL (DCDB)

Pavel Maslov, Matej Komel, Miroslav Pavleski, Klemen Zagar, Cosylab, Ljubljana, Slovenia

Abstract

We have developed a control system 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, and more. 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 [1], which automatically loads and links device support libraries. The DCDB tool is compliant with CODAC (used at ITER and ELI-NP), but can also be used in any other EPICS environment (e.g. it has been customized to work at ESS).

INTRODUCTION

In a physics facility containing numerous instruments, it is advantageous to reduce the effort and repetitiveness needed for changing the control system (CS) configuration: adding new devices, moving instruments from beamline to beamline, etc. We have developed the Device Control Database (DCDB) tool [2], which provides an easy-to-use interface for quick configuration of the control system for the entire facility. The DCDB-tool allows the user to quickly generate and deploy configuration for input/output controllers (IOCs) (EPICS start-up scripts, alarms and archive configuration) onto IOCs; start, stop and restart IOCs, alarm servers and archive engines, and more.

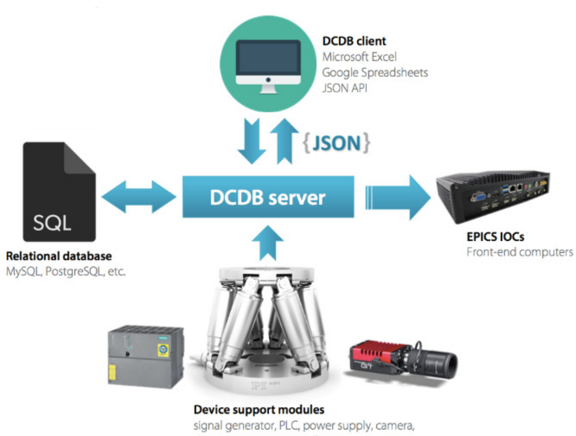

Figure 1: DCDB architecture.

DCDB ARCHITECTURE

The DCDB-tool uses a MySQL relational database. The backend is a typical web-server (Fig. 1), 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 [3]. The client-server communication is based on the exchange of JSON objects (strings).

DEVICE SUPPORT MODULES

To start using the DCDB-tool, simply prepare EPICS device support modules and register them with the DCDB server. In general, device support modules are created in the standard EPICS way. The DCDB-tool introduces the idea of using dynamic macros in the startup scripts. Three additional files, namely: init.cmd, init-pre.cmd and init-post.cmd (Fig. 2) contain macro definitions to be stored in MySQL, and IOC shell commands that register and setup the support module before and 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 2: Files to deploy.

The procedure of creating device support modules for ITER using special extensions to the *mvn iter* plugin was described in [4]. In the latest version we have tailored DCDB to work in the new EPICS environment developed at ESS [5], since they have a similar approach to using dynamic library loading, and in addition to our setup support dependency resolution and have a simpler unit development workflow. The latter is achieved by creating a Makefile, adding source files and running make. If you want to locally install the module, run the shell command: ***sudo make install***. In order to register the module with the DCDB tool, run ***make dcdb.import***. To delete the module from DCDB, run ***make dcdb.delete***.

The last two commands require two environmental variables *$(DCDB_SERVER)* and *$(DCDB_PORT)* to correspond with the parameters of a running DCDB server.

ISBN 978-3-95450-148-9

PLC SUPPORT

While doing EPICS integration of Siemens S7 PLCs, the developer spends a lot of time grouping PVs and calculating offsets in order to match EPICS PV types with STEP7 variable types. We have decided to simplify and automate this process using Excel and introduce this feature into the DCDB-tool. There are now sheets in the Excel file where you configure the hardware, generate UDTs and assemble DB blocks. As a result, the DCDB-tool generates an EPICS start-up script and two PLC configuration files to be deployed on the PLC.

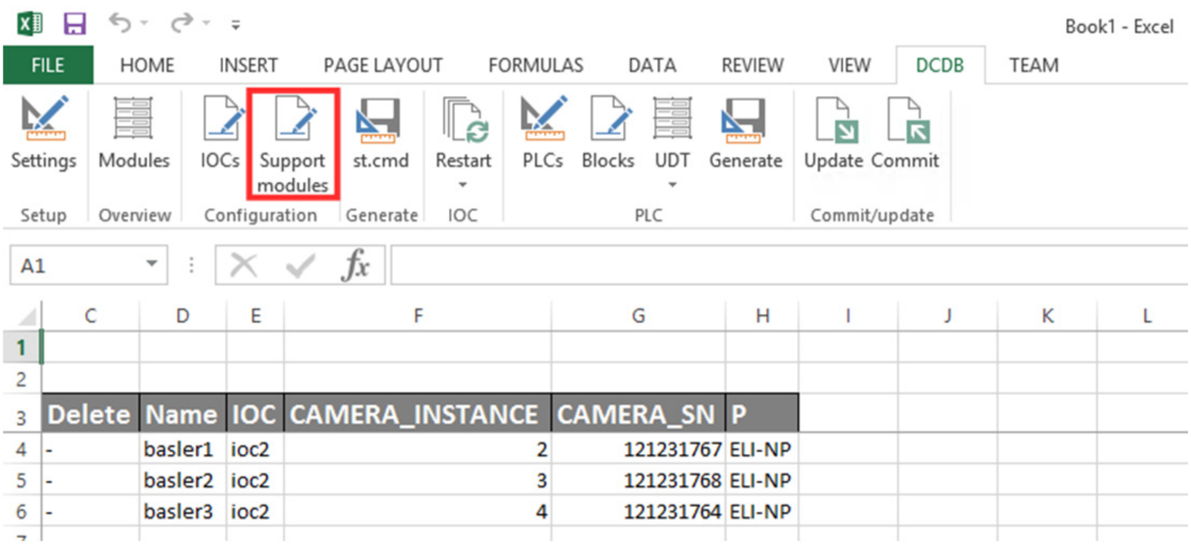

Figure 3: Microsoft Excel client. Configuration of support modules instances.

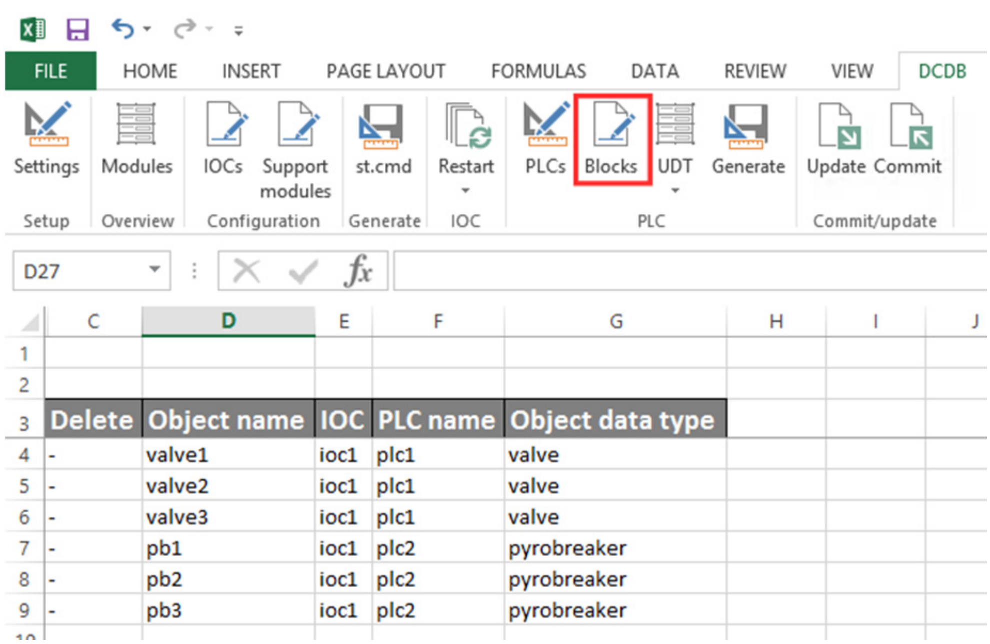

Figure 4: Microsoft Excel client. PLC blocks configuration.

ISBN 978-3-95450-148-9

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 with which you can easily edit your support modules' configuration (Fig. 3), configure IOCs (Fig. 5), PLCs (Fig. 4); start/stop/restart IOCs, and more.

As a free alternative to Microsoft Excel, the DCDB-tool also has a Google Spreadsheets client (Fig. 5), which has been published to the Chrome webstore (Fig. 6). The client is written in HTML/Javascript and fully supports the functionality provided by the JSON API.

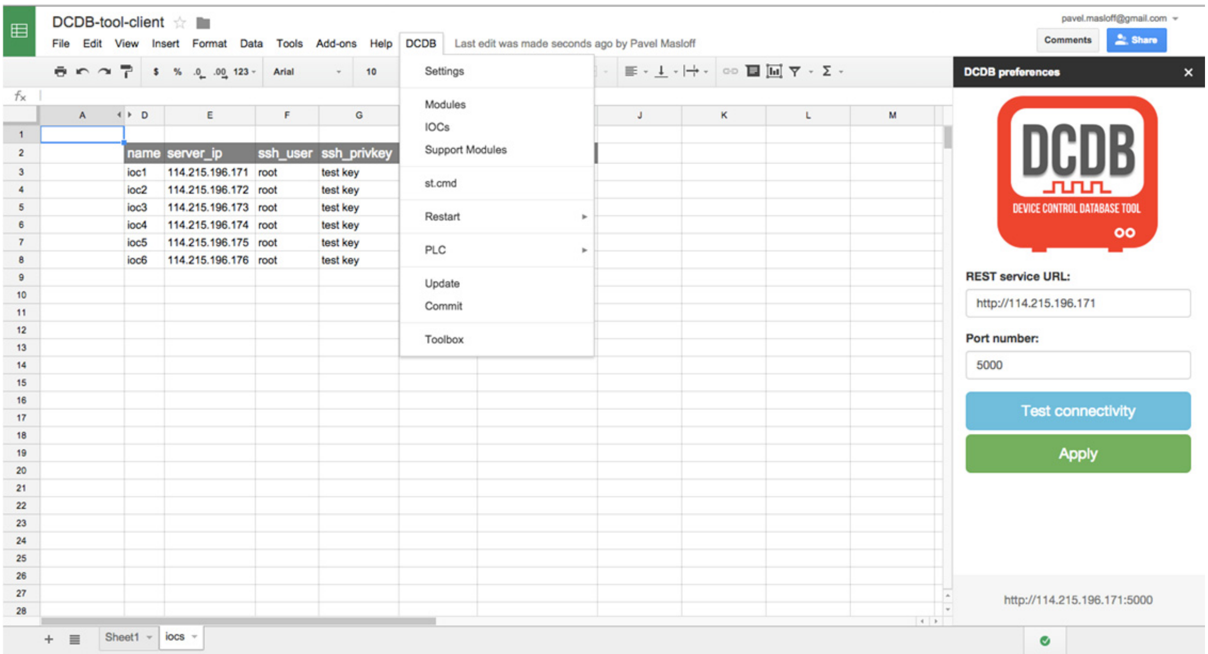

Figure 5: Google Spreadsheets client. IOC configuration sheet.

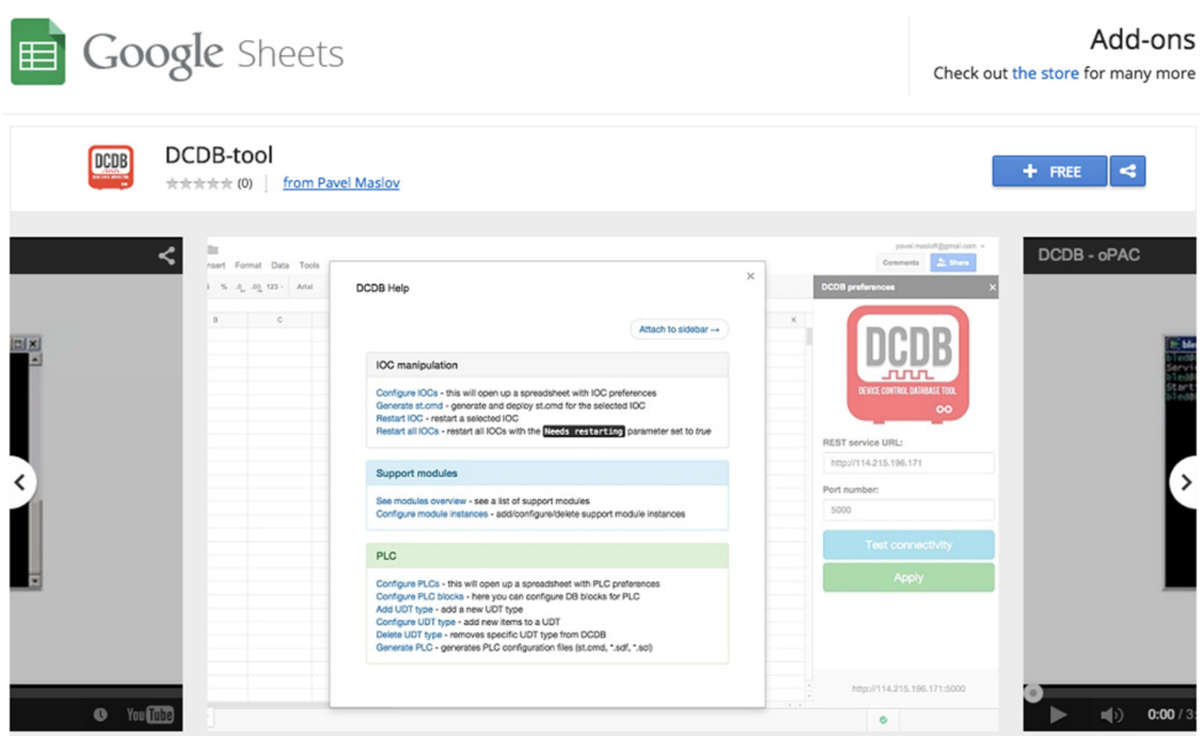

Figure 6: Google Spreadsheets client in the Chrome webstore.

ISBN 978-3-95450-148-9

SUMMARY

The DCDB-tool is a powerful control system configuration tool that reduces integration effort and thus, time. It was tested on different platforms including RHEL, Ubuntu, Scientific Linux, Mac OS X and Windows. It is developed with the best practices from the EPICS community, compliant with the CODAC Core System (used at ITER, ELI-NP), but can also be used in any other EPICS environment (i.e. ESS).

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

[2] DCDB-tool official web-page: http://users.cosylab.com/~pmaslov/dcdb/

[3] procServ (written by Ralph Lange): http://sourceforge.net/projects/procserv/

[4] DCDB tool, PCaPAC 2014: http://accelconf.web.cern.ch/AccelConf/PCaPAC2014/papers/fpo015.pdf

[5] ESS EPICS Environment: https://ess-ics.atlassian.net/wiki/display/HAR/EPICS+Environment