# COSYLAB
CONTROL SYSTEM LABORATORY

Device Control Database Tool (DCDB)

Pavel Maslov (oPAC* fellow at Cosylab, Ljubljana, Slovenia),
Matej Komel, Klemen Žagar (Cosylab)

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
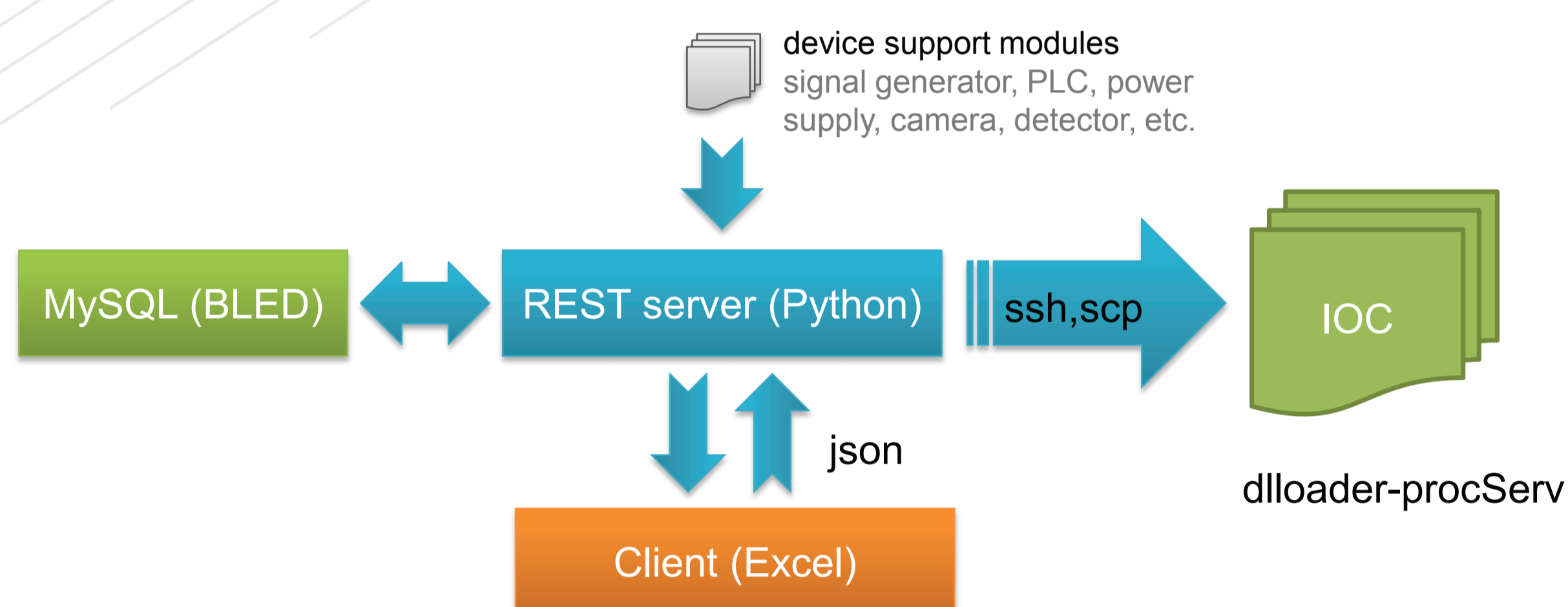
Software

- MySQL database (BLED)
- Python back-end (flask-restful, sqlalchemy, paramiko)
- Microsoft Excel front-end (C# .NET)
- ESS CODAC v.4.1
- procServ (developed by Ralph Lange)
- dlloader (Dirk Zimoch, PSI)

Device support modules

- Create a support module (using dlloader epics template):

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

- Register support module with BLED database using bled import tool:

```
bled@bled:~$ bled
Usage: bled [--pom=] [--pre=] [--db=] [-v] [--help] [--version] [[--delete]]
```

- Files to deploy:

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

Dynamic library loader

- is an EPICS-based tool (in the form of IOC or shared library)
- load device support libraries "on the fly" (no need to recompile IOCs)
- just issue *require <lib_name>* in the EPICS iocshell
- uses procServ for attaching the terminal to the IOC shell
- integrated in CODAC v.4

Back-end

- REST server written in Python
- Uses JSON as the data exchange format
- Uses SSH to deploy configuration onto IOCs
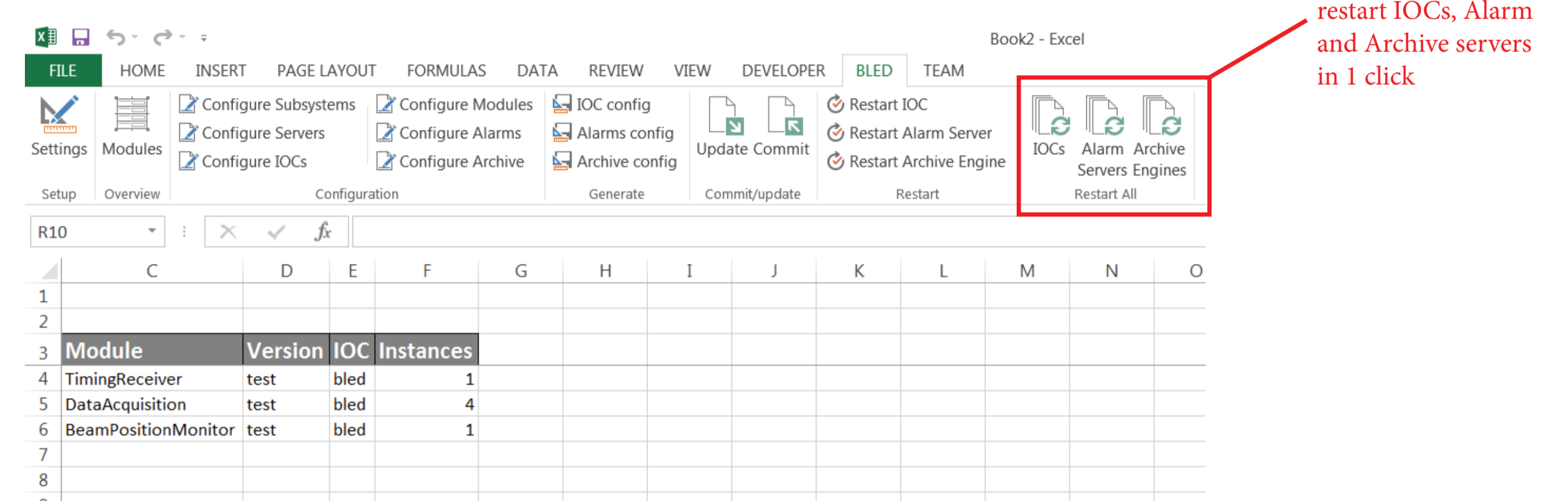- Deployed as CODAC-service:

```
bled@bled:~$ bled-server
Usage: bled-server {start|stop|status|restart|fg} [--port=]
```
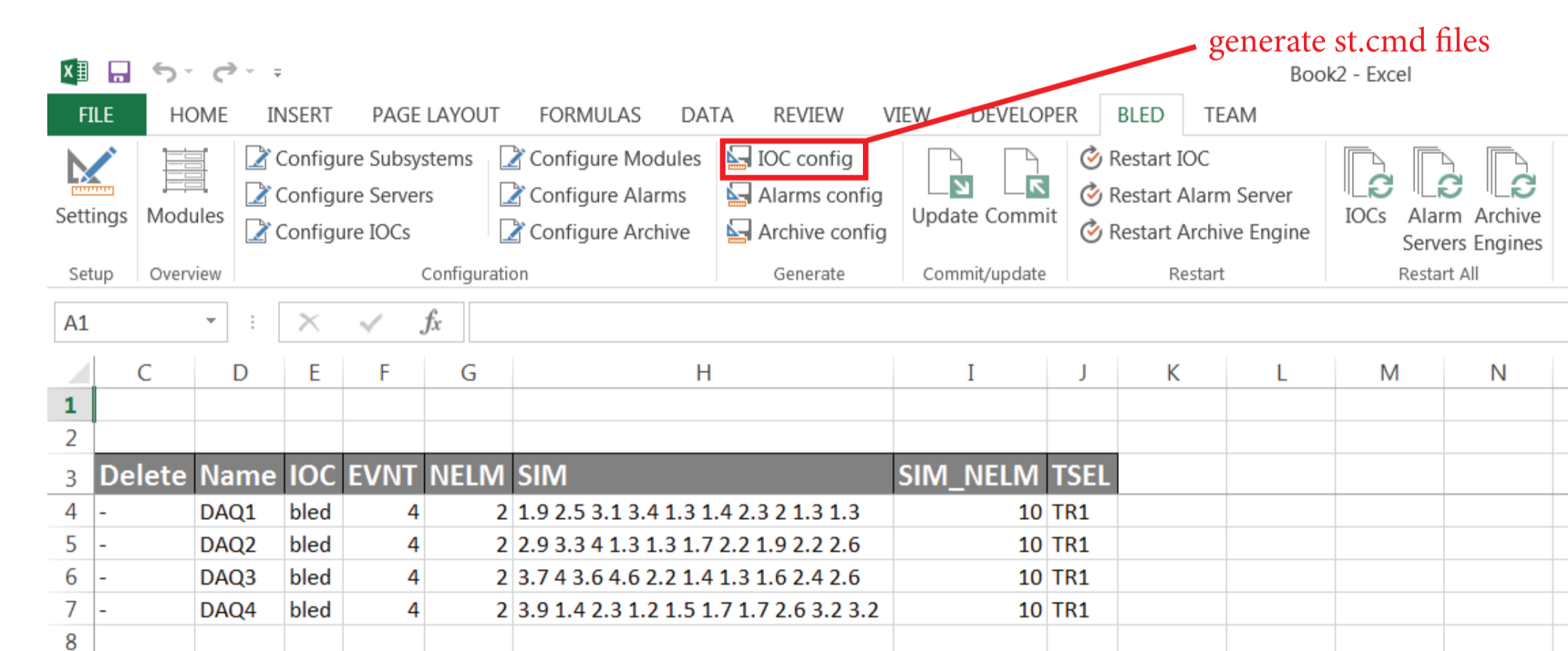
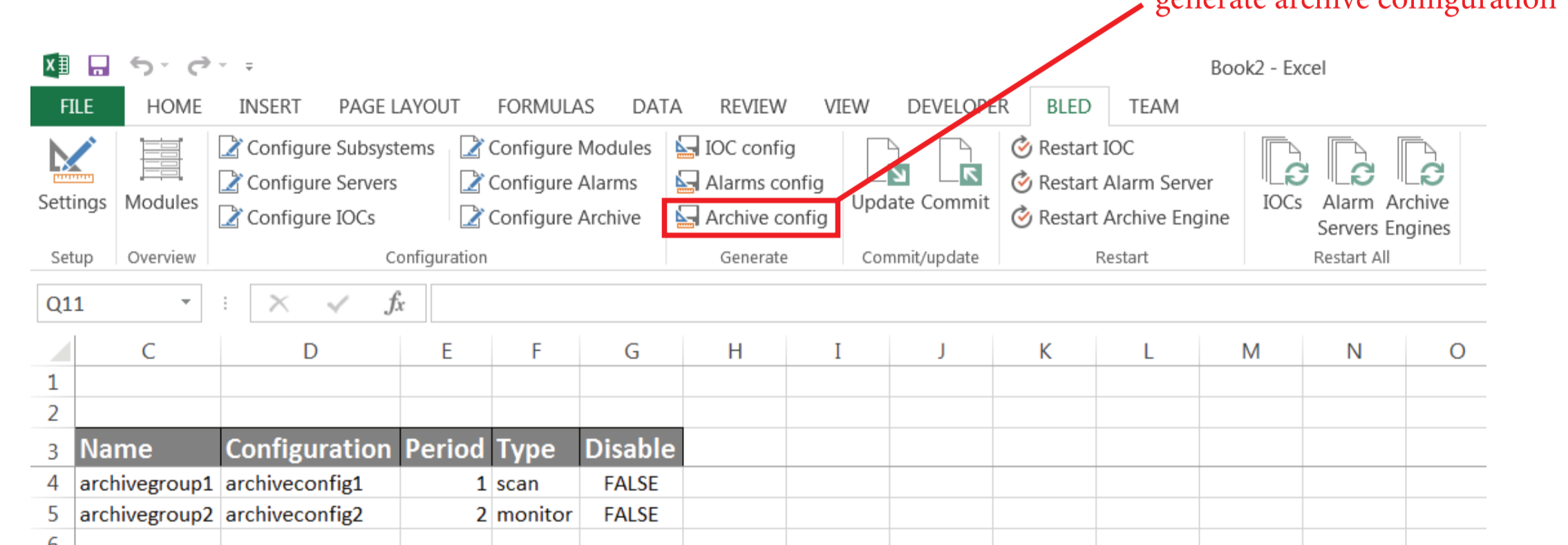- Supports logging

Front-end

- Modules list:

- Module instances (fields correspond to the ones configured in init-pre.cmd):

- Archive configuration:

- Servers configuration:

RPM packages

- m-common (complemented with dlloader support)
- m-maven-iter-plugin (complemented with dlloader support)
- m-codac-unit-api (complemented with dlloader support)
- m-epics-dlloader (IOC, library, EPICS templates, stcmdsaver service)
- m-python-modules (vendor python packages)
- m-python-bled-rest (REST server, import tool)
- m-dotnet-bled-ribbon (client in the form of a MS Excel add-on)

CAoPAC, GSI, Darmstadt, Germany