DEVELOPMENT OF THE SOFTWARE TOOLS USING PYTHON FOR EPICS-BASED CONTROL SYSTEM

T. T. Nakamura, K. Furukawa, J-I. Odagiri, N. Yamamoto, KEK, Tsukuba, Ibaraki, Japan

Abstract

In the commissioning phase of accelerators, many application programs are built and modified frequently by nonexpert programmers. Scripting language such as Python is suitable for such quick development. Since EPICS Channel Access interface library in Python was developed in KEKB accelerator control system, many programs has been written in Python. We have been developing, providing some tools and libraries for Python programming. Some of the recent developments in KEK are reported, and possible applications are also discussed.

INTRODUCTION

KEKB is an asymmetric electron-positron collider at 8 × 3.5 GeV/c, which is dedicated to B-meson physics. Its operation was started in December 1998. The KEKB accelerators control system has been constructed based on EPICS (Experimental Physics and Industrial Control System) tool kit [1]. EPICS provides core mechanism for the distributed control system. EPICS runtime database is running on a local control computer called IOC (Input/Output Controller). More than 100 VME/VxWorks computers are installed as IOC in the KEKB accelerators control system. Several workstations of 4 kinds of platform (PA-RISC/HP-UX, Alpha/OSF1, PC-AT/Linux and Macintosh/OSX) are also installed. Most of the higher level application programs run in these workstations. EPICS provides a network protocol called CA (Channel Access) to exchange data between computers. The atomic component of the data access is called "channel".

In KEKB accelerators control system, we have been using SAD and Python extensively as a programming language to build application program on the workstations [2]. Python is an easy to learn, interpretive programming language. It is not only simple but also has powerful features: efficient high level data structures, object-oriented programming and rich libraries, which cover wide range of areas. With these feature Python is a good tool for rapid application development.

We have developed various kinds of tools and libraries in Python for these years. Among them CA interface libraries, CA-Widgets and casave library are reported in following sections. Further applications of Python are also discussed.

CA INTERFACE LIBRARIES

In KEKB accelerators control system, we have developed Python-CA module, which is a Python interface module to EPICS CA. Python-CA provides basic functions of CA. All of Python programs which perform CA are built based on this module.

Basic functions of CA are get, put, and monitor operations. List 1 shows an example of get operation of Python-CA. Module name of Python-CA is "ca", which appears in import statement.

List 1: Example of Python-CA (get operation)

```
import ca
chan = ca.channel("channel_name")
chan.wait_conn()
chan.get()
ca.pend_event(1.0)
value = chan.val
```

Simple-CA library

While Python-CA provides powerful functions of CA client library, most people still feels complexity for CA programming. List 1 show that even for single get operation it needs 4 function (or method) calls.

We have also developed Simple-CA library, which is a kind of wrapper library of Python-CA. Simple-CA provides simple way, which perform single get or put operation in single function call. List 3 shows an example. Module name of Simple-CA is "cas". Programmer does not need to call pend_event function explicitly. caget function performs a get operation synchronously. It implicitly waits until the request is completed. Python-CA module itself also has similar functions. List 2 shows an example.

List 2: Example of synchronous get by Python-CA

```
import ca
value = ca.Get("channel_name")
```

List 3: Example of synchronous get by Simple-CA

```
import cas
value = cas.caget("channel_name")
```

While synchronous get function is easy to use, its efficiency is not good. Especially it is not suitable to handle a large number of channels. For example, loop over large number of channels takes much time to perform, because every single get operation need to wait completion of the request.

To overcome this problem, Simple-CA is designed to have ability to handle multiple channels at single function call. The caget function accepts list of channel names. It returns list of values. List 4 shows the example.

List 4: Example of multiple gets by Simple-CA

```
import cas
v1,v2,v3 = cas.caget(["name1","name2","name3"])
```

List is the powerful data structure in Python. Using list is simple but efficient solution.

CA-WIDGWTS

Most application programs for KEKB accelerator operation have GUI (Graphical User Interface). Python has some GUI extension modules. Among them "Tkinter" is most popular. Tkinter is an interface module to Tcl/Tk, with which Python gains ability to build GUI using Tk widgets.

Although the combination of Tkinter and Python-CA modules has powerful capability to build any kind of control application programs, most people still feel complexity about GUI programming. Thus we have developed a ready-made widget library "CA-Widgets" which is convenient parts set to build GUI of control application programs.

CA-Widgets are easily extensible using object-oriented feature of Python/Tkinter. See reference [3] for more details.

CASAVE LIBRARY

In KEKB control system we have developed "casave", which is a library module to save channel values to files. casave can save files with several formats. Table 1 shows currently available formats. IOC can read "db" format file using dbLoadRecord function at start up time. "dbpf" is also IOC-readable format. "simple" and "valstat" formats are general purpose.

Table 1: Save file formats available in casave

Format	Description
db	Compatible to EPICS database file
dbpf	List of dbpf commands
simple	Each line contains channel name and value.
valstat	Each line contains channel name, value, severity, status, and timestamp.

List 5: Example of configuration file of casave

```
from casave import defSaveFile
defSaveFile(
    iocname = "ioc_name",
    filename = "save_file_name.db",
    filedir = "dir_path/",
    backup = "flat",
    format = "db",
    chanlist = [("channel_name1","rec_name1","INP","ai"),
                ("channel_name2","rec_name2","DOL","ao"),
                ("cahnnel_name3","rec_name3","DOL","bo")]
)
```

Example of configuration file is shown in List 5. Some parameters can be omitted under some conditions. The configuration file itself is written in python. The configuration is defined by calling defSaveFile function. Each channel entry is defined by 4 parameters (4-tuple). They are a channel name to be read, a record name to be written to the save file, a field name to be written, and a record type to be written. In most case, except "db" format, the latter 3 parameters can be omitted. User can call multiple defSaveFile in a configuration file.

When defSaveFile function is called, the definition of the configuration is recorded internally in casave module. All of the configuration can be accessed through global variable SaveFileDict. To perform save operation, user needs a few lines of coding. List 6 shows an example of save operation. In this example 3 configuration files are imported, and then getting values and saving them to files are processed for all of the configurations.

List 6: Example of save operation using casave

```
from casave import SaveFileDict
import configuration1, configuration2, configuration3
for v in SaveFileDict.values():
    v.get()
    v.dbsave()
```

AutoSaver

One of the interesting applications of casave is "AutoSaver" program. AutoSaver runs as background process and monitors channels defined in the configuration files. Whenever the connection of the monitored channels becomes disconnected, saving process is triggered. Usually IOC-readable "db" or "dbpf" format is used for AutoSaver. Figure 1 shows how AutoSaver acts.

When IOC is rebooted, monitored channels suddenly become disconnected. Then AutoSaver saves last values of monitored channels into files. During start up process of the IOC the saved files can be read by IOC. Thus content values of monitored channels are kept over reboot. Because rebooting and saving proceed asynchronously, this scenario is not always guaranteed. But even such rough method is practically useful.

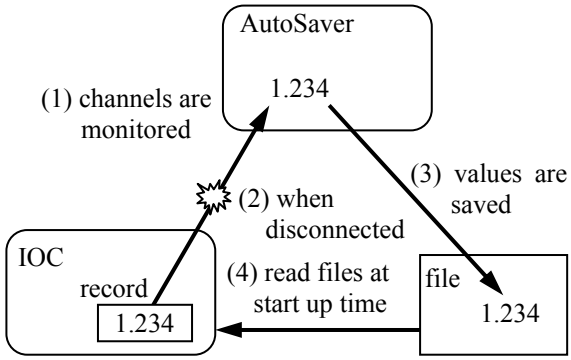

Figure 1: Behaviour of AutoSaver.

POTENTIAL OF PYTHON

In KEKB we have used Python only on workstations. Most IOC's in KEKB run with EPICS R3.13 on VxWorks. Nowadays Linux IOC's with EPICS R3.14 are gradually introduced. We will consider the possibility to apply Python on IOC level programming in Linux environment.

Python can be powerful tool for writing Device Support, especially for the devices which require complicated

control logics. Calculation record driven by Python script can be powerful tool for quick development of runtime database. Sequence program written in Python can be a substitution of SNL/sequencer. In Python 2.3 or later generator is introduced. Generator can be suitable to describe sequence program in another manner than state machine.

CONCLUSION

We have developed several Python libraries for accelerator control systems. Although each tool has tiny power, accumulation of such small developments makes programming environment more efficient and comfortable. Python is suitable language not only for quick development but also for cumulative development in long term.

REFERENCES

[1] N. Yamamoto et al., “KEKB Control System: The Present and the Future”, PAC-99, New York, 29 Mar.-2 Apr. 1999, p. 343.

[2] N. Yamamoto et al., “Use of Object Oriented Interpretive Languages in an Accelerator Control System”, ICALEPCS'99, Trieste, Oct. 1999, p. 600.

[3] T. T. Nakamura, T. Katoh and N. Yamamoto, “Development of the Python/Tk Widgets for the Control System based on EPICS”, EPAC 2000, Vienna, June 2000, p. 1865.