CREATING EPICS SOFT CHANNELS THE EASY WAY WITH SDDSPCAS: FEATURES AND APPLICATIONS*

R. Soliday, M. Borland, ANL, Argonne, IL 60439, U.S.A.

Abstract

Using **sddspcas**, a portable channel access server that is configured by SDDS input files, it is relatively simple to create process variables (PVs). It can be run in a standalone mode or it can be run so that the PVs are checked to ensure that they don't conflict with other IOCs or portable channel access servers. It can also be run using the Run Control facility to prevent additional instances of the same **sddspcas** from being run. The SDDS configuration file provides the PV names, upper and lower limits, units, element counts if the PVs are waveforms, and the types of PVs. Valid types include various precision floats and integers as well as strings. One simple application of this program is that software developers can quickly test their code without requiring the coordination needed to update an IOC database to create PVs. Further details of the features, configuration, and applications of **sddspcas** will be discussed.

FEATURES

Based off of the portable Experimental Physics and Industrial Control System (EPICS) channel-access (CA) server [1], **sddspcas** adds the ability to be easily configured from SDDS (Self Describing Data Sets) [2] input files to create custom PVs. Basically a portable channel access server is an applicaton that can be built to run on a variety of operating systems and is not limited to running only in a VxWorks EPICS input/output controller (IOC). It is a standalone application that mimics some of the behavior of an IOC, such as hosting process variables, with which standard channel-access clients can interact [3]. One of the problems we have had with the standard portable channel access server is that it is slightly complicated to add new PVs to the server. The source code would have to be edited and the executable would have to be rebuilt. Also if you were going to run more than one portable channel access server you would need a different executable for each instance. The **sddspcas** application solves these problems.

Another advantage of **sddspcas** is that the input file is an SDDS file. This file can be generated using the SDDS toolkit from, for example, the output of an SDDS-compliant simulation or script. The file can be manipulated from the commandline or in scripts using the database-like features of the SDDS toolkit. Examples of manipulations would be combining multiple configurations, filtering out certain variables from a configuration, and changing the names or other properties of variables. This is easier and less error prone than editing source code or a non-SDDS configuration file. The configuration file for **sddspcas** can easily serve as a configuration file for any of the SDDS-based data collection programs, which provide PV time-series, alarm, and change logging. This means that data collection for **sddspcas**-hosted variables is nearly effortless.

At the Advanced Photon Source (APS) we have an SDDS file containing all the official PVs. This file is checked by **sddspcas** to ensure that no duplicate PVs are created. Duplicate PVs on the same subnet should always be avoided. There is also an SDDS file containing all the PVs created by other **sddspcas** processes and the name of the workstations on which they are running. This is also used to avoid duplication in PV names. Assuming no duplication is detected, it adds the PVs associated with this new **sddspcas** process into the SDDS file. When the **sddspcas** program is terminated it removes the PVs associated with this process from the SDDS file. Under certain conditions it may be useful to run without checking for PV name duplication, such as when running on a separate subnet that does not have access to the hardware that the original PVs control. In this case it is possible to run **sddspcas** in a standalone mode. It can also be run using the Run Control facility [4], which will ensure that two identical **sddspcas** processes will not run simultaneously.

The PVs created by this application are simple PVs that have no intelligence behind them. One exception to this is the floating point PVs, which have the option to add random noise. This one exception was carried over from the original portable EPICS CA server. Basically the PVs created are a blank canvas where the intelligence behind them can be added using an external script or a program to modify the PV values.

CONFIGURATION

Input File Description

Multiple SDDS input files can be used to configure **sddspcas**. The files must contain a ControlName column, which contains the names of the process variables that will be created. The Hopr and Lopr columns are optional. These are used to set the upper and lower limits for the related PV. If these columns are nonexistent the limits will be removed. Another optional column is ReadbackUnits. This is used to set the units of the related PV. If this column does not exist the units will all be empty strings. The ElementCount column is also optional. The values in this column are used to create waveform PVs with the number of elements equal to the values in this column. The default number of elements is one, thus resulting in a scalar PV instead of a vector PV. The last important optional column is the Type column. This is used to specify the type of PV to create. The valid values are: char, uchar, short, ushort, int, uint, float, double, and string. The default type is double.

*Work is supported by U.S. Department of Energy, Office of Basic Energy Sciences, under Contract No. W-31-109-ENG-38.

Process Variable Name Database

The location of the master SDDS PV file containing all the IOC process variables is set by a commandline option as is the location of the **sddspcas** SDDS PV file containing all the process variables created by the **sddspcas** program. The default values for these are the locations where these files reside at the APS. The –standalone command line option can be used to avoid checking the two SDDS PV files mentioned above.

Miscellaneous Commandline Options

Other commandline options include –executionTime to specify the number of seconds the server will run. The default execution time is forever. A prefix can be added to all the process variable names by using the –pvPrefix option. This is useful in testing situations where there is a need to avoid PV name conflicts with other **sddspcas** processes or IOCs that are running on the same subnet. Random noise can be added and updated at a given rate by using the –noise option. The last commandline options are –runControlPV to specify a Run Control PV name and –runControlDescription to specify a Run Control PV description record. The Run Control feature is used on various types of EPICS applications at APS that are generally run in the background. It permits an application to register itself with an EPICS record, thereby preventing additional instances of the same application from being run. In addition, the executing application may be suspended or aborted via a Motif Editor and Display Manager (MEDM) control screen or other standard channel access client.

APPLICATIONS

Short-Term Applications

Occasionally the ability to create PVs rapidly can be useful when utilizing limited studies time to diagnose or solve various problems. It can also be useful when the PVs are expected to only be temporary and the time needed to coordinate with EPICS IOC managers to add and then remove PVs is excessive when compared with configuring an **sddspcas** process. Here at the APS an example of this is when we set up a process to monitor temperature differentials on the IR detectors. This was done using **sddspcas** to create some soft PVs. We then used **sddscontrollaw** [5] to compute and post the differences to the **sddspcas** PVs. Then **PVmonitor** was used to monitor the levels and issue alarms and/or dump beam.

Long-Term Applications

Long-term usage of **sddspcas** is possible as long as assurances are made that the **sddspcas** process will not be terminated without notification. This can be achieved through use of the Run Control facility and the alarm handler. If the **sddspcas** application were terminated for any reason such as a workstation reboot, the Run Control facility would set an alarm field when the **sddspcas** process fails to communicate with it after a specified timeout period. The alarm handler would then alert the operators as to the nature of the problem. Steps could then be taken to restart the **sddspcas** application. An alternative to this method is to run a cron job every few minutes that would attempt to connect to **sddspcas** PVs. If they fail to connect, it can be assumed that the process has been terminated or has yet to be started after a reboot. The cron job could then launch a new **sddspcas** process.

One example of a long-term application of **sddspcas** at the APS is a source parameter computation at all insertion device (ID) and bending magnet (BM) beamlines. A Tcl/Tk script reads 35BM data to provide emittance and coupling, and uses this data with a calibrated model to perform the computations. This script creates an HTML file that is posted to the web server. It also writes the results to a set of process variables hosted by an instance of **sddspcas**. Finally, a data logger archives the PVs. The script and the data logger are synchronized using the 128-second EPICS strobe.

Software Development Applications

Frequently in developing high-level software, we are faced with the lack of access to the systems on which the software will operate. For example, at APS and other operating facilities, the accelerator is dedicated to user operations most of the time, restricting the time available for software development and testing. In the past, we have used a simulator [3] based on the EPICS sequencer to meet such needs. However, this is relatively complicated to implement. We can use **sddspcas** to simplify creation of a simulator, while simultaneously providing more capabilities.

Two recent examples [6] of this are recounted here. The first example is support of development of software for emittance measurement using the quadrupole scanning technique. In this technique, one or more quadrupoles (electromagnets that focus electron beams) are scanned in current, while collecting beam size data in the form of x and y profiles from a fluorescent screen. We used **sddspcas** to host process variables for the quadrupole currents and the beam profiles. A Tcl/Tk script was written to watch for changes in the quadrupole currents (using the **cawait** commandline program). When a change was seen, the script converted the quadrupole currents to gradients and ran the accelerator simulation program **elegant** [7] to compute the beam profiles at the screen location. These profiles were written to the waveform variables hosted by **sddspcas**. The high-level application that scanned the currents collected these profiles and analyzed them using **sddsemitmeas** [8] to determine the emittance and beta functions. Using this method allowed us to quickly develop a very realistic simulation of the quadrupole scan without the need to embed complex physics of an accelerator simulation into a portable channel access server.

The second example aided development of software related to orbit correction and lattice calibration. In this case, **sddspcas** hosted process variables for corrector kicks and BPM readings. To model the orbit behavior, **sddscontrollaw** was used with a model-forward response matrix to translate corrector changes into orbit changes. This permitted developing and debugging applications to

perform response matrix measurement, orbit correction configuration, response matrix inversion, and orbit correction. By using a perturbed lattice to generate the model response matrix, it was possible to develop and test lattice calibration software.

ACKNOWLEDGEMENTS

The applications recounted in the subsection titled "Software Development Applications" were performed by M. Borland as a consultant for Lyncean Technologies, Inc. We appreciate permission from Lyncean Technologies to recount this work here. SDDS is available under an Open Source license from ANL.

REFERENCES

[1] J. Hill, "A Server-Level API for EPICS," Proc. ICALEPCS 1995, Chicago, Illinois, USA, pp. 136-141 (1996).

[2] M. Borland, L. Emery, "The Self-Describing Data Sets File Protocol and Toolkit," Proc. ICALEPCS 1995, Chicago, Illinois, USA, pp. 653-662 (1996).

[3] R. Soliday, M. Borland, L. Emery and H. Shang, "Use of a Simple Storage Ring Simulation for Development of Enhanced Orbit Correction Software," Proc. PAC 2003, Portland, Oregon, USA, pp. 3476-3478 (2003), www.jacow.org.

[4] C. Saunders, M. Borland, "APS runControl Library," www.aps.anl.gov/asd/oag/manuals/APSRunControl/APSRunControl.pdf

[5] H. Shang, R. Soliday, L. Emery and M. Borland, "New Features in the SDDS-Compliant Epics Toolkit," Proc. PAC 2003, Portland, Oregon, USA, pp. 3470-3472 (2003), www.jacow.org.

[6] Lyncean Technologies Inc., private communication.

[7] M. Borland, Advanced Photon Source Light Source Note LS-297, September 2000.

[8] M. Borland, L. Emery, H. Shang, R. Soliday, "SDDS-Based Software Tools for Accelerator Design," Proc. PAC 2003, Portland, Oregon, USA, pp. 3461-3463 (2003), www.jacow.org.

