ISBN: 978-3-95450-261-5 ISSN: 2673-7000 doi: 10.18429/JACoW-NAPAC2025-MOP091

THE BEAMLINE STEERING SOFTWARE FOR THE APS UPGRADE (APS-U) ACCELERATOR STORAGE RING*

Hairong Shang†, Louis Emery, Martin Smith, Siniša Veseli
Argonne National Laboratory, Lemont, IL, USA

Abstract

A new beamline steering software system has been developed for the Advanced Photon Source Upgrade (APS-U) accelerator storage ring. This system comprises three main components: the main steering server, which performs the actual beamline steering; the beamline steering server, which monitors user steering requests and forwards them to the main steering server; and an operational steering application. The underlying steering functionality is managed by the APS Data Acquisition (DAQ) PV Group (PVG) module. This new software offers significant improvements over the previous system, including parallelization capabilities and enhanced efficiency, and is compatible with any kind of global orbit correction running independently.

INTRODUCTION

Historically, the APS provided steering through a relatively slow protocol that involved manual interactions through floor coordinators and operators, resulting in substantial delays. The initial improvements introduced a web-based steering request interface [1], which reduced response times to a few minutes.

The APS-U steering software significantly advances these protocols by integrating real-time control and automation using EPICS [2,3] through the Python-based DAQ PV Group module [4], facilitating immediate steering response and improved accuracy.

APS-U steering software includes the steering server and user steering interface, which is operated by the MCR team on a daily basis. To enhance user convenience, a beamline steering tool has also been developed. This tool includes a beamline steering server to monitor users' steering requests and MEDM screens for users to input steering requests. The actual steering is performed by the steering server.

The APS-U Steering Server is a crucial innovation, centralizing control via a graphical user interface (GUI) with dedicated tabs for insertion device (ID), bending magnet (BM), and canted undulator (CU) steering. This system provides:

- **Sector-specific server management**, to allow efficient allocation of computational resources.
- **Integrated Channel Access (CA) [5] connections** for efficient communication with steering process variables (PVs).
- **Automated limit-checking mechanisms** to ensure operations remain within predefined safe ranges.
- **Comprehensive Save-Compare-Restore (SCR) file management**, to provide robust data archiving for user-specific steering configurations.

The server performs detailed checks to ensure operational safety, including verifying beam position monitor (BPM) limits, corrector limits, and potential operational conflicts. It also provides real-time feedback and notifications to both beamline personnel and operators.

DAQ PV GROUP

The DAQ PV Group [4] is a Python package that enables efficient management and monitoring of a set of related EPICS Channel Access (CA) Process Variables (PVs). The software uses PvaPy [6] for communicating with EPICS PVs and for hosting EPICS PV Access (PVA) servers. The PVG package can be used via its Python APIs or through included command-line interfaces (CLIs).

The PVG *create* command uses a list of PVs and other group parameters specified either on the command line or via the input SDDS [7] configuration file, which will be described in more details in the next section in the context of the steering server. The *create* command starts the PVG controller process responsible for controlling the group PVs. The controller process establishes CA monitors for all group PVs, and it starts EPICS PVA [8] and remote procedure call (RPC) servers that enable interaction with users. Optionally, it can also start a local EPICS IOC hosting several designated control PVs that can be used for setting the group PV values and reporting the group status. The PVA server is used for hosting a channel containing values of all group PVs, as well as various other group information and metadata. The PVA group channel can be accessed using standard PVA command-line tools and APIs. The custom PVG RPC channel can be accessed using the provided Python API and command-line interfaces. Those interfaces offer users the ability to retrieve and set group PVs, retrieve group status and performance information, change group configuration at runtime, and send the shutdown signal to the group controller. For user convenience, a limited set of control and monitoring features is also offered via a set of control and configuration Channel Access PVs that can be hosted locally by the PVG controller or by a remote IOC. This allows users to set group PV values, update and read the most frequently used group settings by direct CA put and get operations, and use any of the standard EPICS CLI and API tools.

* Work supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-06CH11357.
† shang@anl.gov

The PVG controller supports several modes of controlling the underlying group PVs using value and gain inputs provided by the user, as well as assigned group PV weights and offsets. These modes include:

- **Delta**: new=current+value*gain*weight
- **Absolute**: new=value*gain*weight
- **Set**: new=value
- **Offset**: new=offset+value*gain*weight
- **Multiply**: new=current*value*gain*weight
- **Differential**: new=initial+value*gain*weight

The controller ensures PV values remain within desired ranges using lower and upper limits specified either in the configuration file or obtained from the EPICS database fields.

To minimize the spread of channel set times, the PVG controller employs small delays and Python timers configured to invoke set operations nearly simultaneously. For larger groups, multiple CA worker processes can be employed to control subsets of group PVs, enhancing performance.

APS-U STEERING SERVER

The APS-U steering server is managed by the DAQ PV Group module [4] and SDDS configuration file. The steering configuration file must contain *ControlName* and *Weight* columns. The *ControlName* column defines the actual PVs being used in steering, and the *Weight* column defines the weight of each PV, which corresponds to the ID/BM bump coefficients computed by elegant [9].

The configuration file can also contain other optional columns such as *DeviceName* (provides the device name), *Offset* (defines PV offset), *LowerLimit* and *UpperLimit* (define the lower and upper PV limits). If the lower and/or upper limit columns are not provided, they will be obtained from the DRVL and DRVH EPICS database record fields that belong to the given PV.

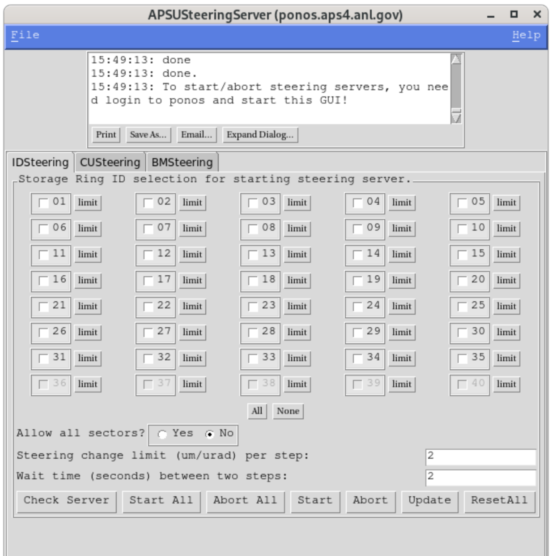
The PVA channel name is provided by the *ControlName* parameter in the configuration file. Other required parameters for steering configuration are:

- **SetValuePV**: PV name to set the PVA value.
- **SetRbvPV**: PV name for the accumulated steered value.
- **RunControlPV**: Run-control record for each steering server to ensure only one server is running.
- **SetStatusPV**: PPV name for the steering status.
- **KnobType**: Steering type (x, y, xp, or yp).

For each ID sector, there are four steering types: Xp, Yp, X, and Y, each as one PVA group. Therefore, each ID sector has four PVA channels, at least 16 PVs, and 4 run-control records. Currently, there are 35 ID sectors in APS-U, resulting in 140 PVA channels, 140 run controls, and over 560 CA PVs for ID steering. Since BM steering only has Yp steering, the number of PVs for BM steering is one-fourth of the ID steering.

An APS-U Steering Server GUI, shown in Figure 1, was implemented to start or abort the steering servers. It has three tabs for ID, CU, and BM steering, respectively. The gray check buttons are for sectors that do not exist; for example, for IDs, the nonexistent ID sectors are 36, 37, 38, 39, and 40. Due to significant computing resources consumption, the steering servers run on a dedicated server. The buttons on the GUI are:

- **Check Server**: Verifies if steering servers are running; green indicates active servers.
- **Start All**: Starts steering servers for all available sectors.
- **Abort All**: Stops all steering servers.
- **Start and Abort**: Controls servers for selected sectors.
- **Update**: Modifies the steering step size with the provided value.
- **Reset All**: Resets the steering status to good for sectors.


APSUSteeringServer (ponos.aps4.anl.gov)
File
Help
Print
Save As...
Email...
Expand Dialog...
IDSteering
CUSteering
BMSteering
Storage Ring ID selection for starting steering server.
01 limit 02 limit 03 limit 04 limit 05 limit
06 limit 07 limit 08 limit 09 limit 10 limit
11 limit 12 limit 13 limit 14 limit 15 limit
16 limit 17 limit 18 limit 19 limit 20 limit
21 limit 22 limit 23 limit 24 limit 25 limit
26 limit 27 limit 28 limit 29 limit 30 limit
31 limit 32 limit 33 limit 34 limit 35 limit
36 limit 37 limit 38 limit 39 limit 40 limit
All
None
Check Server
Start All
Abort All
Start
Abort
Update
ResetAll

```
15:49:13: done
15:49:13: done.
15:49:13: To start/abort steering servers, you nee
d login to ponos and start this GUI!
```

Allow all sectors? Yes [✓ No]
Steering change limit (um/urad) per step: 2
Wait time (seconds) between two steps: 2

Figure 1: APS-U Steering Server GUI.

USER INTERFACE AND OPERATIONAL WORKFLOW

The MEDM-based user interface provides simplified access through standardized screens tailored specifically for ID and BM steering applications. It reduces operational

ISBN: 978-3-95450-261-5 ISSN: 2673-7000 doi: 10.18429/JACoW-NAPAC2025-MOP091

complexity by allowing users to input steering values and request actions through intuitive controls. Automatic checks are performed upon request submission, streamlining the workflow.

Operational Workflow

The detailed operational workflow includes several critical phases:

1. **Steering Server Initialization**: Initiated via the APS-U Steering Server GUI and run on a dedicated server. The servers only need to be started once unless issues arise.

2. **Beamline Steering Server**: A Tcl/Tk server that monitors steering requests from the MEDM screen. Upon submission, it checks BPLD trip limits before initiating steering. If conditions are satisfied, it sets the PVA setpoint PV with the desired value. Only Xp and Yp steering are allowed in MEDM steering.

3. **Manual Steering**: A Tcl/Tk GUI that allows steering for selected sectors by clicking arrow buttons, typically performed by physicists or operators. All steering types (Xp, Yp, X, and Y) are allowed.

4. **System Validation**: The steering server automatically checks request parameters against operational limits, preventing invalid or potentially harmful configurations.

5. **Execution and Monitoring**: Validated steering requests are executed promptly, with continuous monitoring for compliance and accuracy.

6. **SCR File Management**: System updates are recorded into SCR files, archiving comprehensive steering histories for each beamline.

SUMMARY AND CONCLUSION

The APS-U steering software employs Python and Tcl/Tk scripting, utilizing the EPICS-compliant DAQ PV Group module to enhance communication efficiency and operational stability. The modular design facilitates code reusability, simplifies maintenance, and allows straightforward updates to accommodate future improvements.

This software represents a significant advancement in beamline steering technology, improving operational efficiency, user interaction, and system robustness. Extensive testing during APS-U commissioning validated its performance, establishing it as the operational standard for the APS upgrade.

ACKNOWLEDGEMENTS

The authors express gratitude to Greg Fystro and Lisa Marie Berkland for insightful discussions and extensive technical support.

REFERENCES

[1] L. Emery, G. I. Fystro, H. Shang, and M. L. Smith, “Beamline-Controlled Steering of Source-Point Angle at the Advanced Photon Source”, in *Proc. NAPAC’16*, Chicago, IL, USA, Oct. 2016, pp. 887–889. doi:10.18429/JACoW-NAPAC2016-WEPOB04

[2] L. R. Dalesio, M. R. Kraimer, and A. J. Kozubal, “EPICS Architecture”, in *Proc. ICALEPCS’91*, KEK, Tsukuba, Japan, Nov. 1991, pp. 278–282. doi:10.18429/JACoW-ICALEPCS1991-S07IC03

[3] L.R. Dalesio *et al.*, “The experimental physics and industrial control system architecture: past, present, and future”, *Nucl. Instrum. Methods Phys. Res. A*, vol. 352, pp. 179–184, 1994. doi:10.1016/0168-9002(94)91493-1

[4] S. Veseli, “APS DAQ PV Group software package”. https://git.aps.anl.gov/C2/daq/apps/daq-pv-group

[5] J. O. Hill, “Channel access: A software bus for the LAACS”, *Nucl. Instrum. Methods Phys. Res. A*, vol. 293, pp. 352–355, 1990. doi:10.1016/0168-9002(90)91459-O

[6] S. Veseli, “PvaPy: Python API for EPICS PV Access”, in *Proc. ICALEPCS’15*, Melbourne, Australia, Oct. 2015, pp. 970–973. doi:10.18429/JACoW-ICALEPCS2015-WEPGF116

[7] H. Shang, M. Borland, L. Emery, and R. Soliday, “New Features in the SDDS-Compliant EPICS Toolkit”, in *Proc. PAC’03*, Portland, OR, USA, May 2003, paper FPAG004, pp. 3470–3472.

[8] EPICS v4 Working Group, “EPICS pvAccess Protocol Specification”, https://github.com/epics-base/pvAccessCPP/wiki/protocol

[9] M. Borland, “ELEGANT: A flexible SDDS-compliant code for accelerator simulation”, Office of Scientific and Technical Information (OSTI), Aug. 2000. doi:10.2172/761286