ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA095

MANAGING ARCHIVER RULES FOR INDIVIDUAL EPICS PVS IN FRIB'S DIAGNOSTICS SYSTEM*

B. S. Martins†, S. Cogan, S. M. Lidia, D. O. Omitto,
Facility for Rare Isotope Beams, East Lansing, USA

Abstract

The Beam Instrumentation and Measurements group at the Facility for Rare Isotope Beams is responsible for maintaining several EPICS IOC instances for beam diagnostics, of different IOC types, which end up generating tens of thousands of PVs. Given the heterogeneity of Diagnostics devices, the need to archive data for scientific and debugging purposes, and space limitations for archived data storage, there is a need for having per-PV (as opposed to per-Record) archiving rules in order to maximize utility and minimize storage footprint. This work will present our solution to the problem: "IOC Manager", a custom tool that leverages continuous integration, a relational database, and a custom EPICS module to allow users to specify regular-expression based rules for the archiver in a web interface.

INTRODUCTION

FRIB's Beam Instrumentation and Measurements department is responsible for a myriad of devices used in FRIB's beamline for data collection and monitoring of operational parameters, such as Allison Scanners, Profile Monitors, Cameras, Beam Position Monitors, Beam Current Monitors, among many others. All of these devices' parameters are made available to operators, scientists and engineers through the EPICS [1] framework as Process Variables (PVs). Currently, there are more than 200,000 PVs in FRIB's diagnostics system alone. Managing this many PVs across dozens of different IOCs presents numerous challenges, especially with regards to data archival: FRIB diagnostics PVs archival requirements are difficult to implement given some current limitations of the EPICS framework and of the EPICS Archiver Appliance [2]. These limitations and the solutions to them will be presented in this paper.

Archiver Configuration Enforcement

Given the limited features of the Archiver Appliance API, it is essential that there is a mechanism in place to enforce that the desired archiver configuration is in fact, in place.

Non-Regular PV Aliases

Different users of FRIB's diagnostics system expect to have different views over the same devices: controls engineers are typically interested in operational parameters of the devices themselves, while operators care about the readings and actuation capabilities that devices provide.

For example, µTCA crates used in FRIB's diagnostics system, such as the one shown in Fig. 1, can host several types of fast acquisition cards, each having a number of acquisition channels. Controls engineers might be interested in the state of the card itself, whereas operators might be interested in the readings of a particular channel attached to a particular physical device. EPICS provides an aliasing mechanism that can be used in this case to provide one PV name to engineers and a second, aliased name to operators. However, aliasing PVs in batches can be cumbersome if the aliases are not uniform.

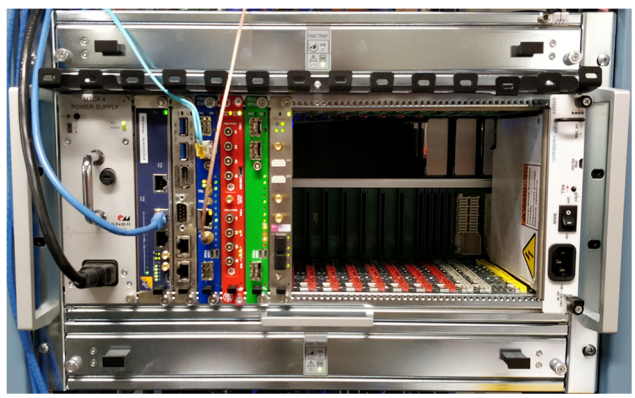

Figure 1: A µTCA crate with MCH, CPU, event receiver, Pico8, BPM and BCM cards.

Non-Regular PV Archival Policies

The second challenge faced by FRIB's controls engineers is the need for having data archival rules on a per-PV basis, rather than on a more typical per-Record basis. For instance, all ADC channels on a CAENels Pico8 picoammeter card have the same kind of EPICS Record that provides the channel readout, but the archival policy for a particular channel depends on what it is connected to: channels connected to Faraday Cups may have a different policy than channels connected to Halo Monitor Rings, for example. In other words, the archival policy must be based on a PV *alias*, not on the underlying *record*.

Centrally Managing Archiver Rules

Lastly, given the variety of device types and the sheer amount of diagnostics PVs, it is important to make it easy for controls engineers to add, modify and remove archiver rules in a central place (as opposed to in each individual IOC), as well as to allow the engineer to assess the coverage of each rule.

* This material is based upon work supported by the U.S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661, the State of Michigan and Michigan State University.
† martins@frib.msu.edu

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA095

PV AUTO PROVISIONER AND ARCHIVER TAGS

The PV Auto Provisioner is a standalone program developed at FRIB to solve the problem of enforcing the archiver configuration. It scans EPICS PV info tags present in Channel Finder [3], builds the desired archiver settings from the relevant info tags with key "archive" and then updates the configuration to the Archiver Appliance. It is also capable of generating reports for discrepancies it finds between the desired state expressed in Channel Finder and the current state found in the Archiver Appliance.

In order to make use of this system, the controls engineer must simply set an "info" tag on a PV with the desired archiver configuration. For example, Listing 1 shows a record which expects to have its value archived whenever it changes ("monitor"), at the maximum rate of 1 Hz ("1.0"), with the data being retained for 3 months ("3mo"). Then, at IOC startup, the recsync [4] EPICS module will push this information to the Channel Finder service, which in turn will be scanned by PVAP.

Listing 1: Example of a PVAP Info Tag

```
record(ai, "EXAMPLE") {
    info("archive", "monitor:1.0,reten-
tion:3mo")
}
```

RETOOLS

In order to solve the aliasing and the per-PV archiver rules problems, an EPICS module was developed to allow info tags to be attached to PVs that have names matching a regular expression. This EPICS module, named retools (Regular Expression TOOLS) [5], leverages C++11's standard regex library to perform regular expression matching and substitution. Among the functions provided by retools, two are the most important: `reAddAlias` and `reAddInfo`.

reAddAlias

`reAddAlias` adds aliases to the PVs that match a given regular expression. It is useful for mapping engineering PVs to operation or scientific names.

For instance, FRIB uses Pico8 ammeter cards to read low current signals. Many Pico8 cards can be put into a µTCA crate and each card has eight channels. The engineering PV name prefixes capture this architecture by having the pattern `DIAG_MTCAxx:PICOy_CHz`, where `xx` is the number of the µTCA crate, `y` is the number of the Pico8 card in the crate and `z` is the channel number. Listing 2 shows a few existing PVs in an IOC.

Listing 2: Example of Existing Base PVs

```
epics> dbgrep DIAG_MTCA01:PICO8_CH0:*
DIAG_MTCA01:PICO8_CH0:DESC_RD
DIAG_MTCA01:PICO8_CH0:TOF_CSET
DIAG_MTCA01:PICO8_CH0:TRIP_EN_CMD
DIAG_MTCA01:PICO8_CH0:LAVGNSAMP_CSET
...
```

PVs with engineering names, however, are not particularly useful for the operators and scientists that use them; instead, they care about the devices that are connected to certain Pico8 channels. In this case, Listing 3 shows how an IOC engineer can use retools to map *all* PVs with a certain engineering prefix to the device prefix associated with the Faraday Cup number D0796 installed in the LEBT section.

Listing 3: Example of Adding Aliases

```
epics> reAddAlias
"DIAG_MTCA01:PICO8_CH0:(.*)" \
                    "FE_LEBT:FC_D0796:$1"
epics> dbgrep FE_LEBT:FC_D0796:*
FE_LEBT:FC_D0796:DESC_RD
FE_LEBT:FC_D0796:TOF_CSET
FE_LEBT:FC_D0796:TRIP_EN_CMD
FE_LEBT:FC_D0796:LAVGNSAMP_CSET
...
```

reAddInfo

`reAddInfo` adds info tags to the PVs that match a given regular expression. In the context of this paper, this is useful for adding tags that later will be read by the PV Auto Provisioner. For example, Listing 4 shows how `reAddInfo` creates an info tag with key "archive" for all Faraday Cup, Halo Monitor Ring, Ion Chamber and Neutron Detector current average PVs so they can be archived at 2 Hz for 3 months.

Listing 4: Example of Adding Info Tags

```
reAddInfo
"^.*:(FC|HMR|IC|ND)_D.*[_:]AVG_RD$" \
          "archive" "monitor:2.0,reten-
tion:3mo"
```

The combination of these two retools functions allows a controls engineer to first associate certain base PVs with device-specific aliases and then add info tags in batches based on the aliased names.

IOC MANAGER

The last piece in this architecture, built to solve the problem of centrally managing archiver rules, is the IOC Manager: a web application backed by a SQL database that gathers all the existing diagnostics PVs and all the desired archiver rules in one place, allowing its users to add and remove archiver rules and to see, in real time, how many PVs are affected by a particular rule.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA095

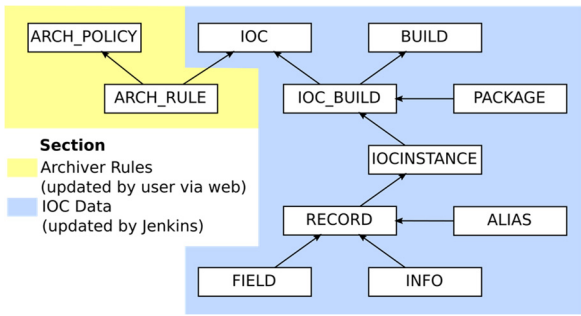

Figure 2: Simplified IOC manager database schema.

The database schema has two logical "sections" (sets of tables), as shown in Fig. 2. IOC data, shown in blue, and archiver rules, shown in yellow.

IOC Data These tables are populated by Jenkins [6] jobs that run every time the code in an IOC source control repository changes. Each relevant Jenkins job builds the IOC, executes it, collects all of its records (name, type, fields, info tags and aliases) via the dbDumpRecord funcion and then pushes the collected data to IOC Manager, along with metadata about the build.

Archiver Rules This section's tables are populated by users of IOC Manager via its web interface. It contains the desired archiver rules to be applied to an IOC. Each rule entry has two parts: a regular expression that identifies which PVs the rule should be applied to and the archiver policy to be used on the matching PVs.

Web Interface

The web interface exposed by IOC Manager, shown in Fig. 3, allows users to add, remove and edit archiver rules on specific IOCs and see the coverage of a particular rule in real time. All changes are saved in the database. The web interface also allows a user to automatically create a pull request for an affected IOC's repository with the updated set of archiver rules.

COMPLETE WORKFLOW

The workflow to configure archiver rules for a particular IOC is as follows. First, controls engineers specify archiver rules for a particular IOC via the IOC Manager's user friendly web interface. The same web interface allows the engineer to open a pull request on his behalf with an updated set of archiver rules for a particular IOC. The set of rules is implemented on the IOC via a sequence of calls to the retools' `reAddInfo` function. The engineer then proceeds to the Stash [7] web interface to review and approve the changes on the repository. The approval and merging of a pull request then triggers a Jenkins job associated with the IOC in question, which pulls the latest source code, builds it, executes the IOC, collects its data and pushes the collected data to IOC Manager. This entire process is illustrated in Fig. 4.

Meanwhile, the updated IOC source code is deployed to production via FRIB's continuous integration and delivery system [8]. When the IOC runs, it executes the specified list of `reAddInfo` commands to create the info tags that are then pushed to Channel Finder. Finally, the PV Auto Provisioner picks up the archiver tags from Channel Finder and configures the Archiver Appliance accordingly.

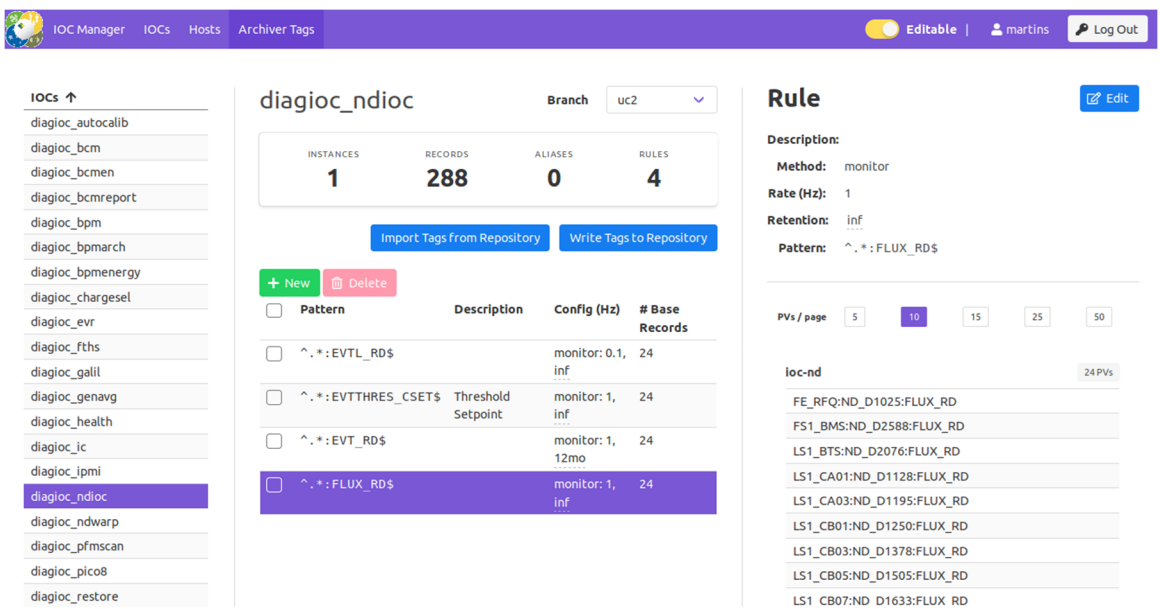

Figure 3: Archiver tags in IOC manager's web interface.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA095

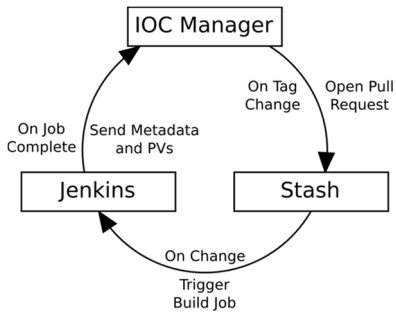

Figure 4: Archiver rules workflow.

CONCLUSION

The use of PV Auto Provisioner, Channel Finder, retools, IOC Manager, Jenkins and Stash allows for a very robust, flexible and convenient system for specifying and specifying archiver rules for the Archiver Appliance. A controls engineer that needs to create a new archiver rule for many similar PVs, or modify such a rule, simply has to input that rule in IOC Manager's web interface, verify that the rule covers the intended PVs and then approve the automatically opened pull request. Once the pull request is merged, a set of automated tools will make sure that the archiver rules are correctly reflected in the Archiver Appliance, and the specified PVs will be correctly archived.

REFERENCES

[1] EPICS, https://www.epics-controls.org

[2] The EPICS Archiver Appliance, https://slacmshankar.github.io/epicsarchiver_docs

[3] ChannelFinder, http://channelfinder.github.io

[4] EPICS Record Synchronizer, https://github.com/ChannelFinder/recsync

[5] EPICS Regular Expression Tools for IOCs, https://github.com/brunoseivam/retools

[6] Jenkins, https://jenkins.io

[7] Atlassian Stash, https://www.atlassian.com/software/bitbucket

[8] M. Konrad, D. Maxwell and G. Shen, "Continuous Integration and Continuous Delivery at FRIB", in *Proc. 11th Int. Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC'16)*, Campinas, Brazil, Oct. 2016, pp.145-147. doi:10.18429/JACoW-PCAPAC2016-FRITPLCO01