A PROPOSAL FOR INTROSPECTION IN EPICS

L.R.Dalesio, D.Dohan, BNL/NSLS II, Upton, NY, U.S.A.
R.Lange, HZB/BESSY II, Berlin, Germany

Abstract

Introspection provides some mapping of function to process variables. To implement this in EPICS, a service is required to define these relationships that may be accomplished external to the control system as an extension to the directory service. This paper outlines the requirements and design.

INTRODUCTION

In the Channel Access protocol, the client connects to Process Variables by providing the name. This requires that each client specify the name of each PV that is needed.

Several high level applications have developed directory services that provide this functionality and are embedded in the tools. XAL[1] has a PV selector that goes through the lattice hierarchy and filters through the functions and locations. By using some advanced directory service, more general purpose clients could be developed. This would allow operators to connect to PVs by some combination of function and location. This is not introspection in the Object Oriented sense, where methods are dynamically exposed. It is a service that allows a system engineer to provide attributes for PVs. Queries can then be made by attribute that return a list of PVs that meet the criteria.

This paper discusses the performance goals for the advanced directory service, the possible data sources for the SQL based directory service, and the applications that may be developed using this service..

PERFORMANCE GOALS

The directory service is designed for use in Client programs that are used for operational tools. It is envisioned that a typical application would be connecting to approximately 300 PVs. The typical application will display the data in some table or plot. From request to rendering, no more than 4 seconds should pass. The typical operator station will be a Linux box that does not have any special hardware. The relational database that contains these relationships is expected to have entries for up to 150,000 PVs that each have up to 10 attributes. The goal is to return the list of PVs in under 1 second. This leaves 75% of the time budget for the Channel Access Client connections and rendering of the display.

The displays must support up to 600 channels being updated at 5 Hz. The most demanding application that we have considered is the orbit display, which plots 300 BPMs, reading X position and Y position. Tests are underway to evaluate the performance of various aspects of Control System Studio from DESY[3] and SNS[4]. Of concern are the Eclipse framework and the communication data plugs.

DATA SOURCES FOR THE ADVANCED DIRECTORY SERVICE

The directory service is planned as an independent SQL database that provides a set of interfaces for construction and inquiries. Two possible sources of information are scripts that sue the naming standard to populate the table or scripts that extract the information from the IRMIS database.

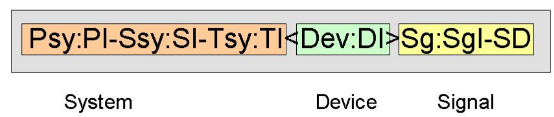

Figure 1: NSLS II nomenclature standard.

NSLS II has a nomenclature standard[5] in place. With a good name standard, an attribute table may be derived. The system, subsystem, and signal could be used to derive functions from the signal (high vacuum, Xpos, Ypos), as well as the system: (linac, booster, storage ring). This approach would likely produce more entries in the advanced directory service then are desired. A method could be used to limit which signal domains or signal types are desired. No work has been done on this approach.

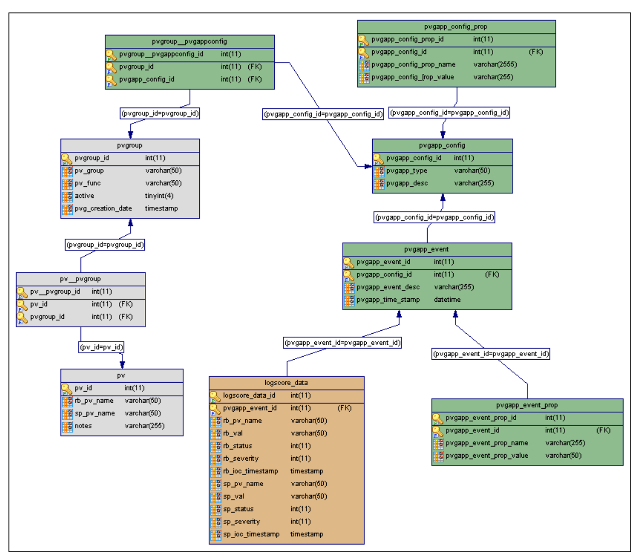

Figure 2: IRMIS ERD – grey are the PV tables.

The IRMIS[6] database has been extended to support the name alias and attribute functionality. A snippet of the data contained in these tables would look like this:

S:C30-VA:G1<SGV:A>Pos-Opn.VAL
894 meters,
BeamIntercept, StorageRing

S:C06-D:G3<Screen>Pos-In.VAL
92 meters,
BeamIntercept, StorageRing
S:C06-D:G3<BPM:A>Pos-X.VAL
92.5 meters,
XPos, StorageRing
S:C30-D:G1<BPM:A>Pos-X.VAL
894.5 meters,
XPos, StorageRing

A script could easily copy this information into the SQL database used for the advanced directory service.

ACCESS TO THE ADVANCED DIRECTORY SERVICE

A Representational State Transfer (REST) style service is provided to allow network access to this directory service. This service layer is developed using JDBC.

APPLICATIONS FOR THE ADVANCED DIRECTORY SERVICE

The advanced directory service can be used to create general purpose clients without knowledge of all of the individual PVs. These applications could include user interfaces such as plots, save/restore sets, archive sets, and high level physics applications.

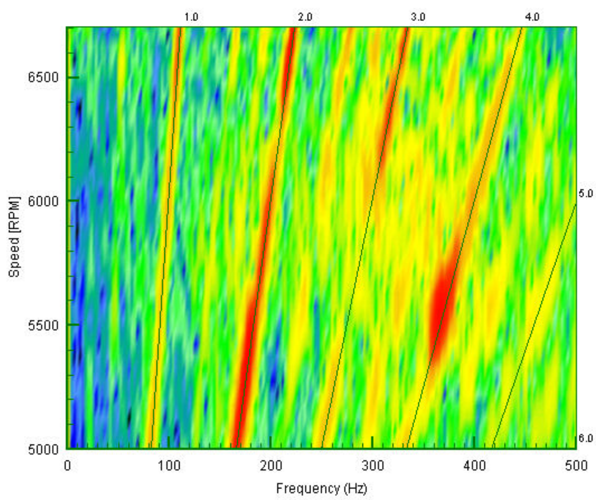

Figure 3: Possible application for adv dir serv.

For instance, a waterfall plot application could use an expression of functions and attributes to give a time elapsed view of a given subsystem. The call sequence would be:

advDirectorySearch(att_list, chan count, chan_list)
ca_serach_and_connect(each chan in chan_list)
ca_monitor(each chid returned from chan_list)
plot(chan_list,values connection information, values returned from channel access)

By providing this service, general purpose clients and scripts could be easily written to produce very versatile tools for the operators.

PERFORMANCE TEST

A demonstration SQL database was used to test the performance by populating the table with 150,000 PVs. Each PV was given six attributes: system, device, unit, position, counter, and cell. A REST service was created using Netbeans as the development environment and Glassfish as the web server. Using the web service, a request was made to return 2,000 channels. To return only the first 2,000 channels and their properties took approximately 100 milliseconds. The next test checked for specific attributes. The initial test for 2,000 channels with attributes, took between 2 seconds and minutes. This was the result of JPA classes creating bad SQL. JDBC was then used. This gives direct control over the SQL. These direct SQL calls need to be tested. The expectation is that the performance should be well within the goal of 1 second. We will also test the performance when all of the data is made memory resident.

ACKNOWLEDGEMENT

The work is being done as follows:
IRMIS: Don Dohan, Gabriele Carcassi, (BNL)
Implementation: Ralph Lange (HZB/BESSY II)
High Level Applications: Kunal Schroff (BNL)

CONCLUSIONS

The initial tests for using web services to search a table with PVs and attributes have shown that the desired performance is achievable. The architecture allows for different methods of creating the directory. Over the next 6 months we will complete the tools to create the table, determine which operator applications to develop and decide which environment will be used to deploy them.

REFERENCES

[1] J. Galambos, C et. al., "XAL – The SNS Application Programming Infrastructure", EPAC 2004, http://accelconf.web.cern.ch/AccelConf/e04/PAPERS/THPLT168.PDF.
[2] J.O. Hill, "Channel Access: A Software Bus for the LAACS", ICALEPCS '89, Vancouver, 1989.
[3] M.R.Clausen, et.al. (DESY) "Control System Studio - Integrated Operating, Configuration and Development", This conference. Paper THC002.
[4] D.J.Armstrong, J.D.Purcell*, K.-U.Kasemir (ORNL) X.H.Chen (ORNL RAD), "CSS - We didn't Invent It, We Made It Better.", This conference. Paper TUP010.
[5] Johnson, E., "NSLS II Nomenclature Standard", National Synchrotron Light Source II Internal Document, 2009.
[6] D.A. Dohan, N.D. Arnold, "IRMIS"; http://www.aps.anl.gov/epics/irmis.