# Open Access Front Ends, Models, and Data Acquistion Redirection

Kevin Cahill
Fermilab
Accelerator Division Controls Department
MS 347
P.O. Box 500
Batavia, IL. 60510 USA

The task of building a front end or embedded model is formidable. The required understanding of data acquisition and network and error-reporting protocols complicate a simple task such as providing a calculated luminosity readback. Models have an additional burden of interfacing to existing programs.

This paper describes an architecture which supports the creation of front ends and models. The open access framework allows programmers to create clients at a high level of abstraction.

## HISTORICAL BACKGROUND

A data server in a control system must implement the functionality defined by the data acquisition protocols. This typically includes the decomposition of incoming network messages, initiating or canceling requests, scheduling data return on a periodic frequency or clock event, construction of network reply frames, error handling and diagnostics to aid debugging.

These details may be overwhelming. Consequently a new data readback summing two channels, for example, are often added to an existing front end that supports a variety of special data types. The disadvantage here is the delay incurred as the inventor of the channel explains the implementation and waits for the front end support person to code it. Also, delays may occur due to waiting for an opportune time to test the new code which could crash and threaten all the services of the front end. Another alternative is to implement a process which sets the calculated value to a channel in an existing front end. The disadvantage here is the insensitivity to data collection rates. Regardless of the users' interest in fast or slow collection or long periods of disinterest, the calculation and setting of the value proceeds at a constant, fixed rate.

## OPEN ACCESS FRONT END

The open access front end provides an architecture for rapid implementation of a data acquisition server. Further, it provides the foundation for model implementations.

The open access front end architecture consists of processes, database tables, applications, and libraries. The processes include the network addressable front end which is responsible for handling the data acquisition protocols, coordination of clients including their requests and replies, error handling and diagnostics. Other processes are the open access clients which are user-written client processes that implement a data acquisition feature. Sybase database tables hold information regarding named open access clients. Applications include the open access client registration program which manipulates database registration tables. Library support provides increasing levels of abstraction to the client while supporting the network-based connection to the open access front end.

## OPEN ACCESS CLIENT

As an example, suppose one wanted the Julian date available as a device reading. The simplest implementation steps would include:

- adding an open access client description to the database specifying:
  - the name, node, and class code for the open access client
  - the open access front end node to serve this client
  - an abstract of service definition, error code if not connected, and an EMAIL address of the maintainer
- creating a database entry for the channel
  - specifying the class code of the client and appropriate node
- creating an open access client
  - a mainline of one line containing a function call to initialize the open access client stream with client name, and optional reading, setting, periodic, and diagnostic function addresses
  - a reading function of one line to stuff the Julian date into a passed structure
- institutionalize the open access client
  - submit source to source code capture system
  - request service initiation to central services group who will create the appropriate command files for starting/stopping the client

The open access client library offers many services to clients. Synchronous and asynchronous replies are supported. Settings may be echoed to the database. Many process statistics and diagnostics are available to aid debugging. Process monitoring and transparent reconnection is provided.

## PROVISIONS FOR TESTING

Even before the open access front end architecture implementation began, the question of how to debug clients arose. Debugging in place is inconvenient to the user community. Creation of similar database entries and clients might also require changes to the driving application to reference the similar database entries. The ability to redirect data acquisition requests to a test client offers simplicity as well as support for models.

The data acquisition manager supports an API to redirect lists of devices or all datapool requests of an application to open access clients under test or to a model or a model under test. A general purpose redirection application and several specific model applications provide the user's interface to redirection. Redirection involves modifying the target front end node and front end specific addressing information when making a data request. An open access client running on any node other than its designated client node is considered to be a client under test. After connecting to the test open access front end node, any application within the control system may have some or all data acquisition redirected to the test client. Likewise, data acquisition for a program may be redirected to the model open access front end and its clients or to the test model open access front end and clients under test.

## OPEN ACCESS MODELS

The Fermilab accelerator control system actually has two open access front end nodes, one open access model node, one open access front end node supporting clients under test and one open access model node supporting clients under test.

One of the first model clients to emerge was the MIRROR model. MIRROR establishes a datapool of all channels that have been redirected, initializes the pool with current readings and settings from the true front end and simply reflects new settings into the reading pool. If a user initializes a parameter page without redirection, true readings are displayed. If the redirection application redirects all data acquisition for the parameter page application, MIRROR will receive the requests, populate from the real world and the parameter page will begin a static display of unchanging data. The look of the application does change as a yellow diagonal line is drawn across the application to indicate redirection. Now, the user may safely knob devices, turn devices on and off or otherwise test the application in a safe way since settings are simply reflected into readings within the MIRROR pool. MIRROR provides a safe environment to debug an application's settings as well as to explore an unfamiliar page.

Another model client is SRFILE. SRFILE establishes one or more datapools initialized with the readings and settings from a save/restore file. Utilizing redirection, any application can contribute to the save/restore system.

## LIBRARY SUPPORT

With each development in client or model support, reusable attributes are made available in a public library. SRFILE inherited the internal datapool management created by MIRROR and simply specifies an alternate initial population of the internal datapool. Another model is SDASRV, serving up collider fill data identical to SRFILE except for the initial population method. Auto population of a memory-based pool from real data acquisition, database settings, save files, or collider fill data is exploited by most open access clients and models.

Other open access model clients are emerging to support physics modeling of beam lines and operator training. Continued evolution of redirection includes using the save/restore system to support model saves and restores.

A dozen open access front end clients exist ranging from PSEUDO, supporting hundreds of devices set by application programs, to SCADA, the control system's port into a supervisory control and data acquisition system used at the site power substation.

## CONCLUSIONS

The open access front end and client architecture offers the user community easy solutions to data serving needs. The open access model and client architecture allows existing applications that use data acquistion to display historical data or to provide application support for model presentations without requiring changes to the application. The redirection capabilities in support of models and clients under test is a powerful extension to the data acquisition system. This single, encompassing framework has paid dividends through exploiting reusability to solve multiple problems.


## ACKNOWLEDGMENTS

The design and implementation of open access front ends, models, clients, supporting applications, and library routines represents the work of many individuals. Brian Hendricks, Bob Joshel, Boris Lublinsky, Junye Wang, and Tim Zingelman all made significant contributions.