

Data Logging

Collection and Storage

Fermilab
Beams Division
Accelerator Controls Department

Kevin Cahill

Version 1.1
October 7, 2002

Architecture

Components

There are several data logging domains utilizing varying strategies to collect accelerator data for viewing.

Distribution

The data logging components are distributed across the control system to maximize the control system's available CPU and disk resources.

Operating Systems and Languages

The older data logging components are written in the C language and run under the VMS operating system. The newer data logging components are written in Java and run under the Solaris operating system

Stored Data

The VMS data loggers log single precision DEC floating-point values with an integer timestamp with one second resolution.

The Java data loggers log double precision IEEE floating-point values with a long timestamp with one millisecond resolution.

Lists

A data logger supports twelve (12) lists of devices where each list may contain sixty (60) device names. Collection rates and space allocation is list based.

Buckets

Data loggers write and read a bucket of data (250 timestamps and values) to improve write and read performance.

Disk Management

Each list is assigned a number of buckets, e.g. 50,000. As each bucket of data is written, a bucket number assigns a place in a circular, ordered list of buckets where the next bucket written for a list overwrites the oldest logged data for that list.

The sum of the bucket assignments of all the lists determines the number of Gigabytes of disk used by the data logger.

Database

The VMS data loggers utilize a commercial, source code based hierarchical file system.

The Java data loggers utilize MySQL, a public domain relational database. A single MySQL database table is limited to four (4) Gigabytes. Twelve (12) lists of 50,000 buckets results in a database table size of four (4) Gigabytes. Java data loggers are being converted to utilize one database table per list. Most converted Java data loggers will be restricted to use no more than the available thirty (30) Gigabytes for data logging.

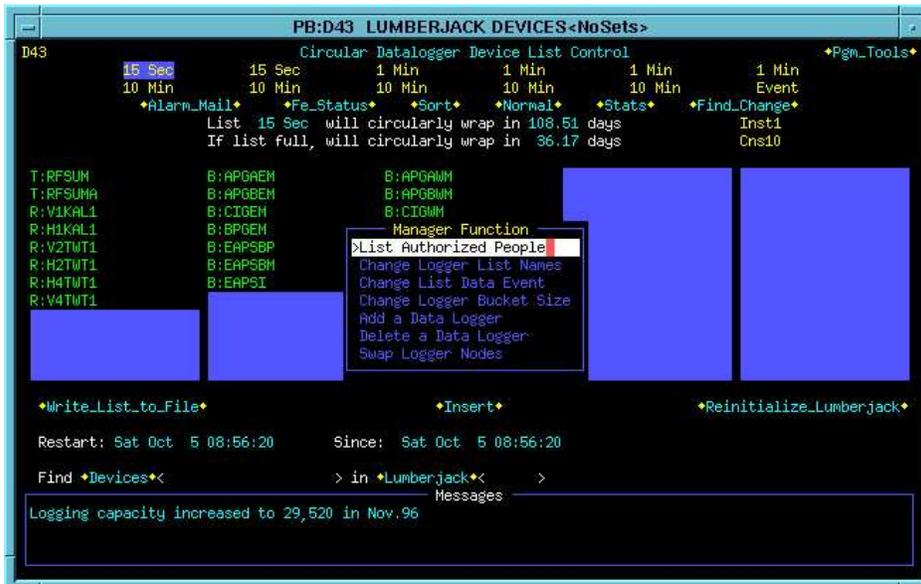
Backups

Data loggers are not backed up. Disk problems are infrequent but can cause a loss of data.

Configuring a Data Logger

Page D43, Lumberjack Devices

This application page displays the configuration of a data logger. Use the Pgm_Tools menu to list authorized people to modify data logger configurations.



Adding a Data Logger

Steps include:

- Select an available node. New data loggers are assigned to Java nodes.
- Choose a name.
- Choose list rates and names.
- Allocate the number of buckets for each list.

Increasing a List's Bucket Allocation

A list will circularly wrap in a longer period of time when assigned more buckets. Be aware the increase will not take effect until the current bucket assignment's value passes the prior bucket limit.

Decreasing a List's Bucket Allocation

Decreasing a list's bucket allocation may make more buckets available for other lists. The database size does not decrease until the space is reclaimed. In general, one should expect that the list's database table will be dropped and recreated resulting in the loss of logged data for that list when decreasing bucket size.

Managing Space

Thoughtful space consideration for the long term is suggested. Increased bucket allocations do not immediately effect wrap time, and decreasing bucket size is likely to clear a database table. A four (4) Gigabyte database can support 12 lists of 50,000 buckets. A thirty (30) Gigabyte available disk can support 4.5E6 buckets, but no list may contain more than 0.6E6 buckets (database table limit). A maximally sized 1 Hertz list with a full complement of 60 devices will wrap in $(600,000 \text{ buckets} * 250 \text{ points/bucket}) / (60 \text{ list entries} * 86400 \text{ seconds/day})$ or 29 days.

Wrap times are directly proportionate to the number of buckets assigned to the list and inversely proportionate to the number of items in the list and the rate items are being logged.

A device list may contain more than 60 devices if some of the entries specify array devices. Each element of an array consumes the space of a device.

Data Collection

Periodic

The fastest continuous rate is 1 Hz. The slowest continuous rate is unlimited.

Hard Event Collection

Collection on Tevatron clock event is possible if the target front-end supports collection on clock event on VMS logger Clock or any Java-based data logger.

Soft Event Collection

Soft event collection is utilized for target front-ends that do not support collection on Tevatron clock and for all collections on Tevatron clock event plus delay and for all collections on a state transition event (plus delay) since the data acquisition protocol cannot express collection on Tevatron clock event plus delay or on state transition.

Soft event collection is conducted using one-shot data collection upon detection of the event and consequently may suffer a loss of accuracy for very volatile signals. Since one-shot reads encompass more network traffic and front-end CPU cycles, Java data loggers limit soft event collection to no faster than one Hertz, VMS loggers even slower.

Specifying the Either Event Collection

A data collection event string such as “e,2,h,0” is ‘read’ as on hard event 02 with no delay. The string “e,2,s,0” is ‘read’ as on soft event 02 with no delay. The string “e,2,e,0” is ‘read’ as on hard event if the front-end supports it, otherwise on soft. When specifying list rates, a hard event string will not collect data from a front-end that does not support Tevatron clock, and a soft event string will always collect one-shot data shortly after detection of the event, but an either event string will proceed as hard when possible, soft otherwise.

Client Events

Lists may have a name and rate that specify a client event. Those lists do not have a data collection job assigned by the logger but receive values to log a client process. Examples include clock, state, and setting loggers.

Specialized Data Loggers

Arklv

The archive logger begins a data collection at midnight from other loggers for logged values at a 15 minute interval minimum.

BLM

This client logger logs Booster loss monitor and Chipmunk readings.

CBSDA

The client logger logs shot data from completed Collider shots.

EventC

This client logger logs Tevatron clock events.

EventV

This client logger logs software state transition events.

PBSDA

This client logger logs shot data from completed Pbar transfer shots.

Sets

This client logger logs settings across the control system.

Snap1

This logger logs snapshot data.

State

This client logger logs software state transitions (VMS).

Retrieval

Timestamps

Java-based data loggers with multiple databases when logging a device at multiple rates will return sets of data that are not strictly time-ordered. The databases will be searched serially, returning data from each database in turn as data matches the user's request.

The Java-based data loggers support the specification the list of interest for a retrieved device and support the specification of the desired logged event string in the request structure. Application programs do not yet implement this option.

Performance

Java-based data loggers are faster than their VMS counterparts, and retrieval to a VMS client from Java data loggers also suffers from date and floating point conversions

Data Return Ordering

Java-based data loggers use multiple databases to store data. The databases are searched in list order. If a device moves from a higher numbered list to a lower numbered list and retrieval is specified over a time period encompassing both list's data, the recent data will be returned before the older data. This effect is also possible in client loggers where list allocation moves at run time to spread the resource usage as equally as possible across all client logger lists.