Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Description</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 13, 2006</td>
<td>1.0</td>
<td>Initial specification</td>
<td>Chris Slominski</td>
</tr>
<tr>
<td>February 21, 2006</td>
<td>1.1</td>
<td>Modified budget constraints</td>
<td>Chris Slominski</td>
</tr>
<tr>
<td>February 27, 2006</td>
<td>2.0</td>
<td>Incorporated review-I comments</td>
<td>Chris Slominski</td>
</tr>
<tr>
<td>March 3, 2006</td>
<td>2.1</td>
<td>Incorporated review-II comments</td>
<td>Chris Slominski</td>
</tr>
<tr>
<td>March 9, 2006</td>
<td>2.2</td>
<td>Incorporated review-III comments</td>
<td>Chris Slominski</td>
</tr>
<tr>
<td>March 13, 2006</td>
<td>2.3</td>
<td>Incorporated review-IV comments</td>
<td>Chris Slominski</td>
</tr>
</tbody>
</table>

Table of Contents

1. Introduction 4
   1.1  Purpose 4
   1.2  Definitions, Acronyms, and Abbreviations 4

2. Overall Description 4

3. Specific Requirements 4
   3.1  Functionality 4
       3.1.1  Archive Channel Values 4
       3.1.2  Channel Value Resolution 5
       3.1.3  Channel Meta-data 5
       3.1.4  Channel Disconnection 5
       3.1.5  API Data Requests 5
       3.1.6  Designating Channels 6
       3.1.7  Event Logging 6
       3.1.8  Data Culling 6
       3.1.9  Automatic Channel shutoff 6
   3.2  Usability 6
       3.2.1  Administration 6
       3.2.2  Archiver Instances 7
       3.2.3  The Archiver API 7
       3.2.4  The API Environment 7
       3.2.5  The Configuration Tool 7
       3.2.6  Channel Set Organization 7
   3.3  Reliability 8
       3.3.1  High Availability Archiver 8
       3.3.2  Fault Tolerance 8
       3.3.3  Channel History Backup 8
       3.3.4  Core Channel Set Preservation 8
       3.3.5  Automatic Startup 8
       3.3.6  Watchdog 9
   3.4  Performance 9
       3.4.1  EPICS Channel Capacity 9
       3.4.2  Prefer Recent History 9
       3.4.3  API throughput 9
       3.4.4  API Data Latency 9
3.4.5 Offline Channel History 9

3.5 Design Constraints 10
  3.5.1 Budget 10
  3.5.2 Schedule 10
  3.5.3 Environment 10
  3.5.4 Lifespan 10
Software Requirements Specification

1. Introduction
This document is yet another attempt to define the needs of the Jefferson Lab accelerator group, with regard to keeping records of the past state of the accelerator control system. These records are critical for monitoring and debugging the system and are used equally by scientists, engineers, technicians, and software analysts. Attempts in the past to get representatives of the user community to collaborate on the requirements specification have failed, so the less than optimal choice of the software developer as author has prevailed.

1.1 Purpose
This document will serve as the high-level requirements of the Jefferson Lab EPICS channel archiver. There is no requirement for this system to be used outside Jefferson Lab, however the design should not intentionally be made to preclude such use. This document will be disseminated amongst the various organizations within Jefferson Lab so that consensus is achieved before a product is released for the users.

1.2 Definitions, Acronyms, and Abbreviations
- API – Application Program Interface.
- MTBF – Mean time between failures.
- MTTR – Mean time to recovery.
- Jlab – Jefferson Lab.
- FTE – Full time equivalent (a man year of effort).

2. Overall Description
The EPICS control system spans a number of distributed computer systems, each having a set of variables made available to the rest of the controls system via the EPICS Channel Access protocol. The term archiver in this document refers to the set of hardware and software components that is used to store and catalogue historical records of individual control system variables, referred to henceforth as channels. One historical record of one channel consists of a timestamp and the value of the channel at that time. The EPICS control system will notify clients when the value of a channel changes, and provides a timestamp indicating when the change occurred. The job of the archiver is to collect and store channel updates from the EPICS control system.

Additionally, the archiver consists of the means to programmatically extract values of a channel, over a desired time span, for analysis. An API will be developed for use by the various applications used in the analysis and viewing of the history data. Another component of the archiver facility will provide users with the means of configuring which channels are to be archived.

These requirements refer to a core set of channels being archived. The core set covers those channels that are critical to analyzing and diagnosing the control system, and are given deference over those channels outside of the core set. Modifications to the core set may only be made by an archiver administrator. The core set must be insulated from the other channels and may be granted greater fidelity, resolution, and history span. It is anticipated that most archived channels will be in the core set, with the non-core set considered a sandbox area, used for fast turn around needs and short term diagnosing of accelerator systems.

3. Specific Requirements

3.1 Functionality
3.1.1 Archive Channel Values
The archiver is expected to record all primitive EPICS channel types including DBR_CHAR, DBR_STRING, DBR_ENUM, DBR_SHORT, DBR_LONG, DBR_FLOAT, and DBR_DOUBLE. In addition, arrays of these data types must be recordable. No other data associated with the channel or its
underlying EPICS record is archived on change in the channel value, except the EPICS timestamp associated with the change. Note, however, the requirement on a channel’s meta-data described in 3.1.3. This includes the status and severity fields normally associated with the .VAL field of an EPICS record. The values obtained from the control system are considered unitless numbers by the archiver, and the interpretation of the values is the responsibility of the user of the data. Underlying changes to the units of a channel will not be noted by the archiver in any way. Note that EPICS enumerations will be archived as their numeric value that is given by default from the EPICS control system, not the associated text string that is stored in the associated EPICS record.

A logical EPICS channel is characterized by channel name, channel type, and channel arrayness. These must remain constant, or the archiver will not continue to archive the channel. It will however generate alerts to the archiver administrator as soon as possible, noting suspected changes to the definition of a channel, or that a channel may have been removed from the control system. Conversely, the archiver will never be required to collect or maintain data for two logically independent channels having the same name. The various scenarios of channel definition change will be enumerated in the administrator’s guide, along with suggested corrective actions to be taken.

3.1.2 Channel Value Resolution

The archiver must be able to capture every change in a channel value that is noted by the EPICS control system. Furthermore, it must be configurable to filter out insignificant value changes as a form of data reduction. What it should consider as a significant change can be made on a per channel basis, and will be specified at the time the decision is made to archive the channel. Changes to the significance threshold may be made afterwards, but will only take effect on subsequent data collection, not effecting previously collected historical records of the channel.

3.1.3 Channel Meta-data

The archiver will maintain information about a channel’s EPICS definition, how it is being archived, and administrator’s notations. These notations will include information about the units of a channel, as provided by custodians of the channel, as well as the history of changes to the channel’s definition. Note that the enumeration strings associated with an EPICS enumerated data type are considered meta-data, and must be programmatically accessible to users of the API, such that text strings may be associated with the enumeration indices archived for a channel.

The archiver will note changes to a channel’s meta-data at the time of connection to the channel, whenever possible, however other forms of meta-data will be entered into the archiver by some external means. The archiver will never monitor or sample any source of channel meta-data.

3.1.4 Channel Disconnection

There are times that the value of a channel is not available to an EPICS client. This can be due to numerous reasons spanning the source computer of the channel, network connectivity, and client host components. The archiver must note periods of unavailability, so that data analysts are not mislead about the continuity of the channel history records.

3.1.5 API Data Requests

Historical records of a channel will be made available, via the archiver API, to application programs. Requests may be made for all values attained during a time span specified by the requesting application. In addition, the application may request the value of a channel at a specific time. This will be interpreted one of two ways, selectable by the API user. One interpretation is the value of the channel last known to the archiver before the specified time, skipping any intermediate disconnection events. The other interpretation is the last recorded event prior to the specified timestamp, whether it be a data update event or a disconnection event. In addition, the API may be queried to determine how many channel history records will be returned if a request for the history is made.

When a user’s requested time span covers times when a channel’s value is unknown, the API will identify segments of data unavailability. In addition, information as to why the channel history is unavailable will
be provided to users of the API. Reasons for data unavailability will include channel data not being requested, channel disconnection, archiver shutdown, data moved to offline storage, and data purged from archives.

Users of the API will be able to issue queries to the archiver to ascertain the archiver’s configuration and any meta-data associated with a channel. This includes searches of channels being archived and any logical grouping of channels defined by the archiver. Searches by wildcard will be allowed.

Channel data is collected as values with timestamps, lending themselves to analysis by channel versus time. The API will also provide capability to assist users in collecting correlation data, so that trends in a channel’s value may be analyzed with respect to values attained by another channel.

3.1.6 Designating Channels

The set of channels that are archived will not be static. The archiver must discover which channels are to be archived at both startup and during normal operation. Users and administrators of the archiver must be able to specify which channels are to be archived, however there is a core channel set that must be protected from the normal user. The core set should be partitioned in such a way that regular users may view channel values, but not interact with the archiving configuration of these channels.

Users will interact with the channel set configuration via a custom tool, henceforth referred to as the configuration tool. Actions initiated by users on the non-core channel set will be automatically logged so that administrators may be able to ascertain what changes to the configuration were made, and by whom. There will be a command line interface for the configuration tool, offering access to the basic configuration features. More sophisticated configuration capabilities will be available through a graphical user interface.

User’s data collection requests will be limited to a finite lifespan, both in the span in data collection and in the span of data retention. However, the user must be able to save their configuration information so that reinstatement in the future is made easy.

3.1.7 Event Logging

The archiver will operate without observation for long periods, not requiring operator interaction. Significant events must be logged for later review by administrators.

3.1.8 Data Culling

The history of channels may only be relevant for only a defined time period. The archiver will maintain configuration information for each channel, which will declare its period of applicability. Collected channel data that lapses beyond a channel’s relevancy time span will be discarded by the archiver. A channel may have a relevancy span of infinity, meaning no culling of collected data will occur, except in the scenario described in 3.3.2 as self preservation.

3.1.9 Automatic Channel shutoff

The configuration of each channel will include an optional shutoff date. Those channels having a shutoff date specified will have the collection of new data for that channel terminated once the shutoff date has lapsed. This will have no effect on already collected data. The granularity of the shutoff date will be one day, and the user of the archiver will have no control over when within the specified day that the actual shutoff will occur.

3.2 Usability

3.2.1 Administration

Administration of the archiver is not intended as a normal user task. Installation, execution, and maintenance activities will be performed by key personnel only. However, an archiver administrator is not required to have any specialized expertise outside the realm of normal system administration duties. Administrators must have the capability to easily monitor the archiver to ascertain its current status and health. Administration procedures will be clearly documented in an Administrator’s Guide, which will be
3.2.2 **Archiver Instances**

One logical archiver may be comprised of multiple archiver instances. An archiver instance is one deployment of the archiver software on one host computer. An instance will collect and store its own specified channel updates. Creating archiver instances may be done to facilitate segmentation of the archiving load and provide scalability. No duplication of channel archiving will occur amongst any archiver instances in one logical archiver. More than one logical archiver may exist with the Jlab network to allow for independent archiving capabilities, channel history redundancy, and archiver development. The API must be configurable to choose which logical archiver will be in scope. The API will always see the channel history provided by all archiver instances within the selected logical archiver.

3.2.3 **The Archiver API**

The history API will be clearly designed and documented so that the normal staff programmer can make use of it in their applications with no unwarranted difficulties. The level of expertise required to use the API should be bounded to that which is the standard usage patterns of the computer programming language that implements the API. The API documentation will be reviewed during acceptance test procedures.

When history clients cannot obtain their desired time sequences of channel data, they must have the capability of obtaining notification as to why their request was not granted. This notification must convey meaning and indicate corrective action for both the software debugger and the history tool users.

The history API must be designed in an extensible way, such that layers of software may be built upon it to gain additional capabilities not provided by the basic API. Examples of this are various forms of data reduction.

3.2.4 **The API Environment**

The API must be available for UNIX based systems commonly used within the accelerator group. This includes the HP and Linux platforms. Applications using the PERL, tcl/tk, and C++ programming languages will be supported. In addition, a tool for the UNIX shell user will be provided to fetch channel history.

3.2.5 **The Configuration Tool**

The channel set configuration tool should not be daunting to the normal accelerator group staff member who may have need to request the archiving of a channel, when updating the non-core channel set. Normal users are not allowed to modify the core channel set configuration, and must initiate such an update via a formal request mechanism. The configuration tool will contain appropriate help information so that typical channel requestors can perform the common channel request tasks within less than 10 minutes, on their first experience with the tool. User configuration requests must be handled promptly. A request to archive an additional 100 channels, once fully specified by the user, should take no longer than 10 seconds to complete.

The archiver is required to notify users of the configuration tool whenever deadlines for termination of data collection, or deletion of their requested data approaches. Only one notification will be made, on the first occurrence, for regularly scheduled clipping of data.

Multiple archiving configurations of the same channel are not allowed within a logical archiver. When the archiver is presented with requests to archive the same channel with different characteristics, it will choose the less limiting parameters, such as the smaller archive delta or the longer period of relevancy.

3.2.6 **Channel Set Organization**

The set of channels being archived will be large. Users of the archiver must be able to view and manage the channel set in logical groupings. Arrangements by a channel’s functional area or a user’s area of interest must be available. Groupings are considered views into the channel set, and any individual channel may
appear in more than one group. Archive groups will be associated with an owner. Only the owner and the archiver administrator may make modifications to an archive group.

3.3 Reliability

3.3.1 High Availability Archiver

The archiver is intended to be an almost continuously running facility, where it runs for months at a time with no planned down time. The development effort must focus on this issue and choose hardware configurations and software techniques accordingly. The expected MTBF is one year with a MTTR of 12 hours to restart the archiving of control system channels. An additional 24 hours of repair time may be expended in restoring backups of the history set, when the fault has compromised the channel history set.

Accelerator operations are brought to a halt for scheduled down times throughout the year. All archiver maintenance activities should occur only during these times. This includes installation of software updates, operating system patching, and any applicable hardware maintenance. Scheduled maintenance of the archiver is expected to be completed in less than two hours in most cases.

3.3.2 Fault Tolerance

The archiver will run inconspicuously most of the time, not requiring operator or administrator monitoring and interaction. Because of this and its high availability requirement, the archiver must tolerate exceptional conditions. It must log detection of these conditions and continue whenever possible, even when a reduction in the quality of service must be incurred.

For example, the archiver adds to an ever-growing channel history set. The archiver must be designed so that running out of disk capacity does not cause a system crash. An acceptable degradation in this situation is to throw away older channel data in deference to recording the present state of the control system. Note that data removed from the archiver as a form of self preservation, not due to lapse in relevancy, must be maintained in some form of offline storage. Another example is the history API, which exposes the archiver’s history set via an API to clients. Poorly behaved clients must not be able to sabotage the archiving of control system data.

3.3.3 Channel History Backup

The channel history collected by the archiver is a valuable asset to the lab and must be protected from loss. The archiver must employ a backup strategy that ensures the safety of its collected data. The backup feature will be selectable on a per archiver instance basis, so that not all channel history will have to be backed up. The purpose of the backup is to restore the history set to its present state whenever a catastrophic loss of data is incurred. It is sufficient to have a granularity of one day, meaning a restoration from the backup may only contain channel history prior to one day before the catastrophic event.

3.3.4 Core Channel Set Preservation

The core channel set must not be put in jeopardy by occurrence within the non-core channel set. In situations where a reduction in performance is permanently or temporarily encountered, degradation in the non-core channel set is preferred.

3.3.5 Automatic Startup

The archiver must automatically restart whenever the hosting computer system is booted following a normal shutdown. The archiver will not automatically restart if it cannot ascertain the hosting computer system is up in its normal state. Automatic restarts after other source of break in operation are not required, including shutdown by administrative command or software crash. Automatic restart after a system crash is not desirable because of the unknown state of the system. Further archiving should not be attempted until an archiver expert has analyzed the situation and knows that a restart is not detrimental to the archiver system or its accumulated history set.
3.3.6 **Watchdog**

A watchdog system will be developed that will keep an eye on one or more archiver instances. This system must be independent from the archiver, so that it may determine the health of any hardware or software component of the archiver. It will be able to detect failure in a component within one hour of the failure’s occurrence. It will report all failures to the archiver administrator immediately upon detection.

3.4 **Performance**

3.4.1 **EPICS Channel Capacity**

The control system for the accelerator is large with many defined channels. The archiver must be designed to handle the number of channels that will need to be archived as well as the data throughput mandated by updates to these channels. From past experience with channel archiving, there is need to archive up to 100,000 channels. The average number of channel updates experienced among this channel set is around 25,000 per second. Furthermore, the archiver must be designed in such a way that it can scale to increased demands that future control system needs may dictate. The archiver design must support scaling of both the number of archived channels and data throughput by a factor of ten. This means the original archiver design with minimal modification can be used to meet future need with additional or replacement hardware.

3.4.2 **Prefer Recent History**

The normal usage pattern of channel history favors recent data over past data. The archiver design should take advantage of this to enhance performance and capacity. The boundary between recent and past history will be placed no sooner than one year old. Performance criteria cited in the remainder of this section will apply only to recent channel history. There are no stated requirements on performance for past history.

3.4.3 **API throughput**

The API must be able to quickly access a channel’s historical records and provide the data to applications running on the various systems used within the accelerator group. API users are responsible for assuring their application will be able to hold the amount of requested data. Compiled applications running on typical accelerator group UNIX workstation can expect a minimum average data rate of 25,000 history records per second for scalar channels. This data rate assures a busy day of accelerator data, having significant changes in value once every second, can be obtained in 3.5 seconds. The archiver must also be able to provide the value of 10,000 different channels at a selected time in one minute. A configuration or meta-data request for one channel must be met in no more than 0.1 seconds.

The API allows for concurrent access to channel history. Users of the API can expect to see degradation in performance due to client load, however the performance must not degrade more than linearly with the number of clients. History requests are not expected to exceed a rate of a few requests per minute on the average, but peak loads may surpass this significantly.

3.4.4 **API Data Latency**

There will be no more than two minutes of latency between the presence of data in the control system and its availability to the history API clients.

3.4.5 **Offline Channel History**

The archiver is not required to hold the entire span of history for all channels in local disk storage. Data sent to offline storage must be accessible to data analysts by request, but may be presented to the user in as much as 15 business days from the request. The requested data must be converted to a form such that normal archive data analysis tools may make use of it.
3.5 Design Constraints

3.5.1 Budget

No additional staff resources will be created for this effort, therefore the design and development effort of the archiver is expected to be absorbed into the current staff workload. One FTE will be expended over the specified design and development period.

Hardware and software purchases will be limited to no more than $30,000.00 for the product development in this fiscal year. If the design requires additional funds than can be provided in this fiscal year, the deployment of the archiver will be delayed until additional funding is available, however the total purchases for the development of the archiver must not exceed $100,000.00. Note the budgetary limitations of the production archiver system should not impact the software design and development. The archiver should be designed to work on less capable systems in a reduced capacity, and scale to the needs of a full production system.

3.5.2 Schedule

There is already an archiver in place that is marginally meeting the accelerator group needs, therefore there is no emergency need to be filled. The release of the product specified in this requirements document should be available by the fall of 2006, to coincide with the start of the next fiscal year and, hopefully, greater accelerator operations.

3.5.3 Environment

The operating environment of the Jefferson Lab accelerator group is assumed to be the target for the archiver. There will be no major change in the type of systems used including hardware and software products. In general, history API applications and the configuration tool will run on UNIX systems such as the hpux-11-11-parisc and rhel-4-0-ia32 systems. Current operational procedures and normal user patterns must be respected.

3.5.4 Lifespan

The archiver, as discussed in this document, is intended for use at Jefferson Lab for an extended period of time. Its design must make use of techniques and technologies that will persist throughout the target product lifespan, so that minimal investment in resources will be required to continue to use the archiver in the face of product updates and maintenance. The target lifespan of the archiver is ten years.