

Requirements for the Beam Monitor Data Channelling Process.

Brett Viren
Brookhaven National Lab

August 20, 2004

Revision : 1.10

Abstract

This document lists the current understanding of and requirements for the process which will channel the beam monitoring data into the offline. It defines what data is to be collected, how it will be processed and dispatched.

Contents

1 Overview	2
2 Data Source	2
2.1 XML-RPC overview	2
3 Data Source Configuration	3
4 Data Acquisition Methods	3
4.1 A Note on Security	3
5 Timing of Data Acquisition	4
6 Data Output	4
6.1 Details of Raw Data Format	4
6.2 Block IDs	5
7 Non Requirements	6
8 Open Questions	6

1 Overview

Beam monitor data comes from three loosely separate sources: Primary proton beam monitoring devices provided by Beams Division, target and horn monitoring devices and finally, secondary hadron and muon monitors (PIC arrays). All are read out on a spill-by-spill basis. The overall job of the process described here is to get this data into the offline data stream.

Additionally, there is "environmental" monitoring (temperatures, DAQ electronics health) which is not taken spill-by-spill. This is considered the responsibility of the DCS and not of this process. The exact definition of what data is pertinent to this device is given below.

(Note: in this document the requirements or delineated **[requirement]**:*like this*. There are also some things which are unknown at this time and they are marked **[fixme]**:*like this*. This document is subject to change.)

2 Data Source

The source of all data considered for this process is ACNET. ACNET frontends acquire data from DAQ hardware and this is accessed via Data Acquisition Engines (DAE). A DAE communicates to a Data Acquisition Client (DAC) via Java Remote Method Invocation (RMI). This is all in the domain of the Beams Division¹.

The method recommended to access this is through a special DAC which is also an XML-RPC server. This server is provided by Beams Division (contact: Charles King).

2.1 XML-RPC overview

XML-RPC, like Java RMI, is a method for making remote procedure calls (RPC). It uses extensible mark-up language (XML) as the interchange data format. The actual network connection uses hypertext transport protocol (HTTP). The connections are stateless and initiated by an XML-RPC client which sends a function name and any arguments to a server via an HTTP POST. The server responds with any values that are returned by the function. The roles of client and server can be mixed, but for any connection there is but one of each. More information is available in the XML-RPC HOWTO document².

The XML-RPC protocol specification³ is open and simple but there are many free implementation software libraries available that take care of the HTTP connection and the translation from native data structures in to and out of XML. These are available for a variety of platforms and languages. After extensive testing the Python implementation is chosen as the most desired.

[requirement]:*All data are accessible via XML-RPC.*

¹See http://minos.phy.bnl.gov/~bishai/numi_sync.jpg for a diagram of this.

²<http://xmlrpc-c.sourceforge.net/xmlrpc-howto/xmlrpc-howto.html>

³<http://www.xmlrpc.com/spec/>

[**requirement**]: *Implementation language is Python.*

3 Data Source Configuration

Each device available via the Beams Division’s XML-RPC server has an “AC-NET” name, a scalar value (or an array of values) and a unit. For example, there is a device named “M:OUTTMP” which gives the outside temperature in degrees.

[**requirement**]: *The process must be able to be configured with a list of devices to read back. This configuration only happens at startup time.*

4 Data Acquisition Methods

There are currently two ways of acquiring data via the Beam Divisions XML-RPC system: by polling and by setting up a callback. In **polling** mode an XML-RPC client simply sends a list of device names to the server and the response contains a corresponding list of their values. In **callback** mode a client gives the Beams Division’s XML-RPC server a device to monitor for some state change (which could be a periodic timer or an accelerator event), a list of devices to collect data from and finally a URL and method of an XML-RPC listener which is called in order to return the values when the given device’s state changes.

A mix of the two are possible and may be desired in order to sample devices that are read out in the middle of a spill. This was thought necessary to read out pedestal data from the muon/hadron monitor SWIC scanners, however now the goal is to have this done in hardware read out both in- and out-of-spill data once per spill.

[**requirement**]: *Use callback access method, allow for mix of callback and polling.*

[**requirement**]: *Independence from near and far detector running. This process should run irrespective of any other data acquisition.*

4.1 A Note on Security

There is no inherent security in the XML-RPC protocol. Data is passed over the wire in plain text. Also, anyone who can reach the server’s port can execute the remote procedure calls.

In the case of the Beams Division server, all remote procedure calls are “read only” in the sense that they only return data and do not allow for any control.

In the case of the callback listener described above, there is the potential for data corruption if a malicious process connects to the listener and inserts false data. Only allowing the Beams Division system access to the listener’s port should be sufficient.

[**requirement**]: *Sufficient security must be in place to assure the integrity of the data and invulnerability to denial of service attacks.*

5 Timing of Data Acquisition

For primary beam, target and horn monitoring, and to a lesser extent hadron and muon monitoring, it is necessary to be able to associate this data with that collected in the near detector for the same spill.

The DAE timestamps each ACNET device readout. Tests show that the latency and jitter are typically around 10 ms with worse case rarely exceeding 1 second.

The absolute time for both the MINOS DAQs and the DAE timestamp synchronized via separate NTP servers, each sync'ed to separate GPS devices. The relative timing of these two time standards has not yet been checked.

[requirement]:*Provide data needed to correlate beam monitoring data with the near detector data on a spill-by-spill basis. It is not in the scope of this process to do this correlation.*

6 Data Output

To go into the offline data stream, "raw" data must first be packaged in a simple class derived from a RawDataBlock. This is done by making a connection to a running rotorooter⁴ daemon and feeding it blocks of data.

[requirement]:*Output block data via connection to rotorooter.*

6.1 Details of Raw Data Format

The design of the data format borrows from what already exists for DCS data and follows what is needed to send data to the rotorooter. The way the raw data blocks will be structured is detailed here.

For each beam spill, blocks holding a header and in or out-of spill data will be sent as a contiguous record:

Header block
Data payload block

The block id (see below) can be checked to resolve whether the data block holds in or out-of spill information.

The header block contains this structure:

size in words
check sum
block id
seconds
nanosecs
spill count

⁴http://minos.phy.bnl.gov/software/sid/WebDocs/howto_roto.html

Where the seconds and nanoseconds provide an absolute time stamp representative of the entire block. What event this will actually stamp has yet to be determined. Individual devices also have absolute time stamps (set by the DAE) which in general will differ slightly from the block timestamp. The spill count gives a relative enumeration of the number of callbacks (it remains to be seen if this will correlate one-to-one with the actual beam spills). For out-of-spill data, the spill count from the most recent callback is used.

The payload blocks are structured like:

size in words
check sum
block id
number of devices
ACNET name part 1
ACNET name part 2
seconds timestamp
msec timestamp
number of values
device value 1
device value 2
device value 3
...
ACNET name part 1
ACNET name part 2
seconds timestamp
msec timestamp
number of values
device value 1
device value 2
device value 3
...

ACNET names are 8 bytes and in the format X:YYYYYY. The “X” indicates node (I = Main Injector, E = NuMI line), the “YYYYYY” encodes device, location and a suffix indicating a property or function of the device. See MI-NOTE 0189 for some details. Except for the handron/muon montiors most devices will be scalar valued. The “:” is overwritten in the output in order to encode if the device contains floating point or integer data.

6.2 Block IDs

Each block (header or data) has a block ID which contains 4 values packed into a 32 bit word. The values used will be:

csf = 0	CFN = 001	S = 1	major id = 0x300 - 0x3ff	minor id = version
---------	-----------	-------	--------------------------	--------------------

Where “csf” sets the simulation flag to Data, “CFN” sets the detector to Near and “S” sets the source flag to DCS. The Major ID is described below and the minor id is used to version this schema.

The Major ID uniquely determines the semantics of the data in the block. Currently 0x300 has been allocated for beam monitoring data from ACNET. This identifier space will be broken down into:

ID	Usage
0x310	Record Header
0x320	In-spill data
0x330	Out-of-spill data

Other IDs in the range 0x300 - 0x3ff are reserved for future use.

7 Non Requirements

Some specific things that this process will *NOT* be responsible for.

- Data not accessible via XML-RPC (with the possible exception of timing data needed to correlate with near detector events, or if the fully asynchronous method is needed) will not be channelled by this process.
- No data quality checking or other manipulation will be done, except for anything related to the actual channelling of the data. That is, no semantic value is attributed to the data. If this is to be done it should be done with independent code acting on the records placed in the offline data stream.
- There will be no communicating output data to any other processes except the rotorooter daemon and Beams Division’s XML-RPC server.
- There will be no means to change configurations during run time. New configurations require the process to restart.
- There will be no controlling of any beam device with this process.

8 Open Questions

Besides the [fixme]:’s above there are some open questions:

- How to handle startup, shutdown and configuration?
- How to interact with any kind of system monitoring, report status?
- What computer to run on? Just about any computer can access the Beams Division’s XML-RPC server as well as an instance of the rotorooter, so there is likely very little constraints.