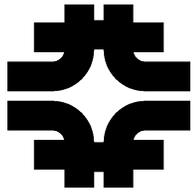


IFBeam Data Access Caching C++ API

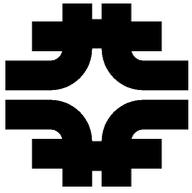
A.Norman, M.Mengel

FNAL-CD



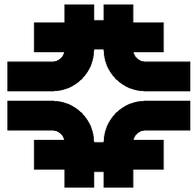
Overview

- Flexible beam data acquisition system has been in place for ~6 months
 - Monitors NuMI extraction lines
 - Provides physics normalization data bundles
 - Provides short-term (1 week) monitoring of all NuMI devices for beam studies
 - Recently extended to Booster Lines
 - Readouts out IRM data as block devices
 - Unpacks and stores IRM buffers in database as individual devices



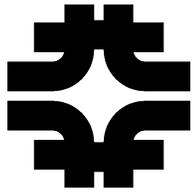
IFBeam Data Readout

- Beam data can be accessed through an extensible WEB API
 - Provides real time monitoring, plotting, dashboard functionality
 - See <http://dbweb0.fnal.gov/ifbeam/>
 - Data provides in multiple formats (csv, xml, json) with self describing tags
 - Not optimized for offline job processing
 - i.e. thousands of jobs requiring event by event beam spill records



IFBeam Data Offline Readout

- Designed to scale for offline job processing
 - Makes intelligent queries and caches results
 - Optimized around job flows that processes events that are consecutive in time (i.e. run/subrun structures)
 - Cache sizes are tunable by the calling job to further optimize query/memory overhead
 - Allows for single events to be matched to the closest spill record with a given Δt



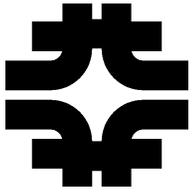
Data Records

- Data records can be retrieved as either:
 - Single devices with scalar readouts
(i.e. a toroid or a horn current monitor)
 - Single devices with vector readouts
(i.e. beam profile monitors)
 - Lists of devices with scalar readouts
(i.e. multiple horn current monitors)
- Can also retrieve
 - Lists of consecutive spills times
 - Lists of device names that were present for a spill



Integrating With Your Framework

- The IFBeam API is easy to integrate with current offline frameworks
 - Written in vanilla C++
 - Relies on NO external packages
- Check out from cvs
- Include “ifbeam.h”
- Compile and link with the package



Example Usage:

```
#include "ifbeam.h"

// Configure the parameters for the beam folder
const char* bundleName = "NuMI_Physics" // The bundle to access
const char* dbWebAddress = "http://dbweb3.fnal.gov:8080/ifbeam" // Address of the DB
int cacheTime = 3600 // Time window to cache in seconds

// Setup the beam data folder
BeamFolder* beamDataFolder = new BeamFolder(bundleName, dbWebAddress, cacheTime);

// Now make a call to retrieve the spill information
double spillTime = seconds.millisconds; // Time to retrieve
const char* deviceName = "E:TORTGT"; // Device name to retrieve
double value; // Value to retrieve into

// Retrieve the value for this spill
beamDataFolder->GetNamedData( spillTime, deviceName, &value);

// Retrieve a list of all the spills that there is data for
Std::vector<double> spillList = beamDataFolder->GetTimeList();

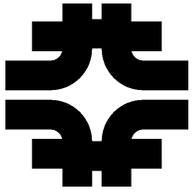
// Retrieve a list of all the devices that were readout
Std::vector<std::string> deviceList = beamDataFolder->GetDeviceList();
```

Configuration, there are sensible defaults

Device information

Retrieval methods
Make Sure to handle exceptions:

```
Try{
    Get()....
}catch{
    Opps...do this
}
```



Example Usage:

```
// Loop over all the available spills and histogram out a device
```

```
// Get the list of spills
```

```
Std::vector<double> spillList = beamDataFolder->GetTimeList();
```

```
Std::vector<double>::iterator spillList_itr;
```

Retrieve the spill List

```
// Loop over the spills and make a histogram
```

```
For(spillList_itr = spillList.begin();
```

```
    spillList_itr != spillList.end();
```

```
    ++spillList_itr){
```

Loop over the spill List

```
    try{
```

```
        beamDataFolder->GetNamedData(*spillList_itr, "MyDeviceName", &value);
```

```
    }catch(WebAPIException){
```

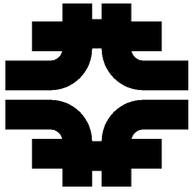
```
        continue;
```

```
    }
```

```
    histo->Fill(value);
```

```
}
```

Get device values and histogram them



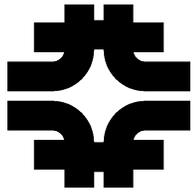
Usage Notes:

- The first call to the “get” function will contact the database and make take a while to return
 - Subsequent calls will return almost instantly if there is a cache hit
 - If there is a cache miss, the database will be contacted and the cache will grow
- If you know the approximate time interval that your job covers, you can pre-fetch the spill window at the start of your job
 - There are built in safeguards to prevent “bad” queries or excessive queries that would break the DB
Example: you request a 365 days of pre-fetch cache
- Multiple BeamFolder objects can be open at once (i.e. NuMI_Physics and BNB_Physics)
 - Each is completely independent
 - Each has a separate cache



Example Application:

- Simple beam info browser:
 - Select and browser multiple BeamFolders at the same time
 - Retrieves the list of devices that can be browsed
 - Retrieve individual values, vectors etc...
 - Histogram and time series graphs of values



Still to Come

- The API is in active development
 - Feature requests are welcome
 - Features in development currently
 - Spill matching w/ more robust Δt
 - Expanded error handling
 - Simpler vectored device retrieval
 - Convenience methods for time stamp translation
- Questions: contact Mark or me
mengel@fnal.gov, anorman@fnal.gov