

# DIRC FAST MONITORING DOCUMENTATION

[Important directories for the Fast Monitoring](#)

[Why is the table showing the last 200 processed runs not updated?](#)

[How to restart the Fast Monitoring?](#)

[How to look at JAS plots remotely?](#)

[How to process a Fast Monitoring run in standalone?](#)

[The DrcOfmPostProcess package](#)

[Brief description of the spew file contents](#)

[How to update the DIRC JAS references](#)

[How to change the tag of DrcMonTools in IR2](#)

[How to create new JAS references](#)

[How to rerun the Fast Monitoring on a xtc file](#)

[Useful Links](#)

A PDF version of the documentation is available [here](#) (last update: 2006/06/29).

## Important directories for the Fast Monitoring

- For each run, an hbook file containing the Fast Monitoring data for all subsystems is archived in the directory `/nfs/bbr-srv02/u4/Monitoring/OutputArchive/` which is visible from any `bbr-dev` or `bbr-farm` machine.
- The subdirectory structure is quite obvious: if you're looking for a file from June 2005, just do
 

```
cd 2005/06/LiveFastMon
```

 to access the proper area.
- Files have the following generic name:
 

```
LiveFastMon-00NNNNN-YYYYMMDD-HHMMSS.hbook
```

  - `NNNNN` is the run number;
  - `YYYYMMDD` is the day the run was taken;
  - `HHMMSS` is the time the file was created; it should be less than 2 minutes after the beginning of the run.
- The processing of a Fast Monitoring hbook file starts automatically as soon as the run has ended.

Since end of Run 5a, the processing of the DIRC data is included in the generic BaBar **post-processing** framework. Having a better synchronization between the Fast Monitoring data taking and their processing increases the reliability of the whole chain and makes sure that DIRC oncall experts get the Fast Monitoring postscript files as early as possible.

The post-processing system has some additional nice features. For instance, if a problem occurs, it can be re-run later on old runs it would have missed. Such intervention is done centrally and does not require any action at the DIRC level. The current manager of the post-processing software is [Jim Hamilton](#). Contacting him in case of problem with the DIRC Fast Monitoring is the right thing to do.

- The DIRC post-processing is done by the package `DrcOfmPostProcess` which has been developed under the `babardrc` account to make code changes and transitions between experts smoother. The current (03/2006) running directory is `/nfs/bbr-srv02/bfdist/Builds/18.6.1-DrcOfmPostProcess/DrcOfmPostProcess`.
- Fast Monitoring data are stored in the directory `$BFROOT/www/Detector/DIRC/DircOperations/FastMonitoring/` which is expected to be the permanent location of these files. Postscript files access is easy thanks to webpages which are updated automatically when a new run gets processed.
  - A detailed [table](#) providing various information on the last 200 runs which have been processed by the Fast Monitoring.
  - The [list](#) (without any detail) on all runs processed in 2005.

If you're looking for the fast monitoring data of a run taken before the setup of the post-processing framework -- or if the storage area has been cleaned to save some disk space -- you can always [recreate](#) a postscript file manually.

- The post-processing logfiles can be found in the directory `/nfs/bbr-nfs01/logfiles/Oep/YYYY/MM/FastMonPostProc/<Machine>` where `YYYY` and `MM` are the current year and month. `<Machine>` is the machine on which the post-processing is running (currently `bbr-dev102`).

## Why is the table showing the last 200 processed runs not updated?

- The first reason is obvious: BaBar may not be running! Check the [logbook](#) first:
  - make sure that runs long-enough (at least a handful of minutes!) have been taken recently;
  - read also the 'Activities' and 'Problems' sections to see if some computing/dataflow work (maintenance or problem fix) is not ongoing on the Fast-Monitoring server.
- If everything looks normal, it is worth checking the recent logfiles of the DIRC post-processing. They can be found in the directory `/nfs/bbr-nfs01/logfiles/Oep/YYYY/MM/FastMonPostProc/<machine>`, where `YYYY/MM` are the current year/month and

<machine> is the IR2 machine where the post-processing framework is running. For instance, the path for logfiles in November 2005 was [/nfs/bbr-nfs01/logfiles/Oep/2005/11/FastMonPostProc/bbr-dev102](#).

DIRC post-processing logfiles obey the following convention: [DrcFastMonPostProc-YYYYMMDD-HHMMSS](#). If you're not familiar with the package, you should look for the statement '[DrcFastMonPostProc exited with code 0 for <run number> \(...\)](#)': return code '0' should mean that everything was OK.

If there is no logfile for the recent run (and if you're in the proper directory!) the post-processing framework may not be running at the moment. Contact Jim Hamilton and let him know the current situation. As he told me once, **'whatever it is, I'll fix it!'**

---

## How to restart the Fast Monitoring?

- As already mentioned, DIRC Fast Monitoring data are processed by the general post-processing framework. So the 'only' thing we can do on our side is to make sure that the [DrcOfmPostProcess](#) code is working. The running of the post-processing is controlled by Jim Hamilton.
- 

## How to look at JAS plots remotely?

There are two possibilities.

- The simplest method is to launch [RemoteBabarJas](#) from an IR2 machine ([bbr-dev](#) or [bbr-farm](#)). But this may consume a lot of resources of your computer, especially if you're accessing JAS plots from a remote site.
  - If you're using windows, you can install java and download JAS3: details can be found in this [hypernews](#). In this case, the client will run on your machine and you'll only access the data which will be much faster.  
Note: the JAS3 client is also useful to browse the ambient database.
-

## How to process a Fast Monitoring run in standalone?

This section assumes a good knowledge of the basic features of the BaBar offline framework which are explained in details in the [workbook](#).

- Choose a machine from the IR2 network, for instance `bbr-dev100`.
- Create a new test release in your working area.  
As `DrcOfmPostProcess` is a standalone package, any (recent) release could be chosen. The list of available *offline* and *online* releases can be found in the directories `$BFDIST/releases` and `$BFDIST/online_releases` respectively. The release on which a lettered offline release or a online release are based is defined in the `.current` file located in the top directory of the release.
- After the `srtpath`, checkout an up-to-date version of the `DrcOfmPostProcess` package: look at the most recent tag in its [CVS repository](#).
- Add the `workdir` package and setup it properly.
- **Important:** go to the `DrcOfmPostProcess` directory and edit the file `localSetup.tcsh`. Replace the line similar to  

```
setenv WORKDIR /nfs/bbr-srv02/bfdist/Builds/18.6.1-DrcOfmPostProcess/workdir
```

 by  

```
setenv WORKDIR <the absolute path of your workdir>
```

 Setting correctly this variable is mandatory to make sure that the files created when you run the post-processing in standalone will be stored in your release area. If you don't change this setting, files will be created and stored in the running area!  
 Source this file to get the proper environment variables defined:  

```
source localSetup.csh
```
- Then, compile the package using the instruction `gmake bin`. The compilation is very simple: it copies several scripts (without any extension) from the package to the `bin/` directory and make them executable -- the 'mangling' phase.
- Finally, go to the `workdir` and create two directories: `tmp/` and `spew/` (*troubleshooting: respect the naming convention*). The former is used to store temporary files and the latter will contain the postscript files produced by the analysis of Fast Monitoring data.
- You can now analyze a Fast Monitoring hbook with the simple command

`DrcOlocalRunProcess`. It uses normally three arguments: `<YYYY>`, `<MM>` and `<run Number>`. Indeed, to find the hbook location, the script needs to know the run number (5 digits) and when it was taken (year and month). The [Shift Run info](#) provides the DQM logbook information for a given run (and hence its date). An even faster way to get the time information is to type the command `ir2runs <run Number>` in a shell window. For a first test of the framework, you can also run the command without argument as all three parameters have default values (run 59577 taken in November 2005).

If the script runs well, you should see a postscript file (currently 19 pages) in the [spew/](#) directory.

You can find [here](#) the (uncommented) log of a session during which the steps described above were performed in the same order. When kumac macros get executed, you'll see the error message `Incorrectly built binary which accesses errno or h_errno directly. Needs to be fixed`. Don't worry, it is not a concern as it is always there when one runs PAW!

## The DrcOfmPostProcess package

- The package contains three main types of files:
  - the executable scripts (= the files without extension);
  - the PAW kumac macros which read the Fast Monitoring hbook file and create the postscript QA file;
  - the php scripts (.php and .inc files) which update the webpages.
- `DrcFastMonPostProc` is the top script which is called automatically each time the post-processing is started in IR2. This short perl script gets the run number and selects the correct hbook file among the various streams which are received in argument. Then it calls the main script, `processLiveFastMon`, which steers the analysis of the DIRC Fast-Monitoring data.
- `DrcOprocessLiveFastMon` starts with an initialization phase. In addition to setting a few environment variables, it executes the special command `searchFile DrcOfmPostProcess` which is required by the online post-processing environment: it ensures that the job is properly setup and that the directory from where the code should be run is found.
- Then, the PAW macros are run one after the other; the postscript file is created in the master macro `spew.kumac` which calls a different kumac for each page. The postscript created is called `spew-00<run number>.ps` and is stored in the directory to which the environment variable `DRCOFMPPSPEW` points. This file is finally compressed (gzipped) to save disk space. Among the various temporary files stored in the directory `DRCOFMPTMP`, `paw.txt` contains the logfile of the macros ran in PAW.
- The next step of the post-processing is the update of the DIRC post-processing 'database', a simple text file: `$DRCOFMPOSTPROCESSOUTPUTDIR/localDB.txt`. A perl script,

- [DrcOappendLocalDB](#), deals with this issue and extracts the required information from the IR2 DB using the [ir2runs](#) script. After being formatted, these data are written in the text file.
- Then, the webpages are updated (in fact recreated from scratch using php scripts). A subset of runs (currently either the last 200 processed or those taken in 2005) is extracted from the DB and used as input for the php scripts. New webpages are written in the [\\$DRCOFMPOSTPROCESSOUTPUTDIR](#) directory.
  - Finally, the last part of the script provides a very simple interface between the DIRC post-processing and epics (the slow monitoring system used in BaBar).
    - The idea is the following: quantities computed during the post-processing and which are important for the data quality and/or for the DIRC are put into epics variables. If any of them goes outside its nominal range, it would trigger an alarm on the ALarm Handler (ALH) which will be noted immediately by the pilot. In this way, important problems should be found more quickly.
    - Currently, there is only one variable monitored: the mean of the core Gaussian used to fit the hit corrected timing distribution by a  $g$  (signal) +  $p_0$  (background) function. But this number should grow in the future.
 

*As the DIRC has the best timing resolution among all subdetectors, knowing where the distribution of the signal hit timings peaks allows one to find shifts in the master trigger crate's timing which occurred a couple of times during Run 5a. Such shifts are multiple of 16.667 ns and are clearly visible in the fit results of the DIRC timing plots.*
    - Some tests are done before updating the epics records: to avoid false alarms, one checks that the results are statistically significant and that BaBar was taking colliding data during the run currently analyzed -- different conditions of data taking obviously lead to different results!
    - Any change in this part of the script should involve both the DIRC epics expert and the package coordinator of [DrcOfmPostProcess](#) as some epics work is likely to be needed to match the changes in the DIRC post-processing. For instance, monitoring the evolution of an additional variable via epics requires the creation of a new record which would need to be included in the ALH once its alarm thresholds would have been set.
  - Running the DIRC post-processing scripts in [standalone mode](#) follows a similar logic although the scripts are different to avoid conflicts: [DrcFastMonPostProc](#) ↔ [DrcOlocalRunProcess](#) and [DrcOprocessLiveFastMon](#) ↔ [DrcOlocalProcess](#).
  - Finally, the script [setupFramework.tcsh](#) can be sourced to setup an environment similar to the one in which the post-processing scripts run in IR2. Calling it allows you to execute [DrcFastMonPostProc](#) or [DrcOprocessLiveFastMon](#) as if you were running the BaBar post-processing main script. As executing these commands may interfere with the DIRC post-processing running area, one should be cautious when using them.

## Brief description of the spew file contents

- First, what is the Fast Monitoring hbook made of? Well, it contains data from about 20% of the L1Accept events. This percentage depends on the average event size: the bigger the events, the lower the fraction of processed events. More details can be found in this [e-mail](#) from Jim Hamilton.
- **Page 1**
  - This plot shows the number of hits accumulated by each PMT over the whole run: the 'hotter' the color of a dot, the more hits that PMT integrated. The color code (from blue to black) is not absolute: the range covered by each color depends on the number of hits collected during the run.
  - Black dots correspond to hot spots (for instance the central ring-shaped region of the DIRC where the majority of hits show up during colliding beams data taking). Noisy PMTs also appear like black dots which come and go from run to run.
  - 'Dead' PMTs (i.e. phototubes without any single hit during that particular run) are marked by magenta stars. They are more visible in page 2 where all PMTs but the unefficient ones are printed in blue.
  - The hit map is different for
    - [cosmics](#)  
Cosmic rays come from the sky and so cross BaBar from top to bottom. Therefore, the sectors with the highest occupancies are #11, #0, #5 and #6. In addition, the hit rate is lower than during physics runs: more PMTs are likely to have accumulated very few hits, in particular along the outer borders of the sectors.
 and for
    - [collision data](#).  
The injection pattern makes the background hits more numerous in sectors #2 to #5. The ring structure visible in the example plot should always be visible. To compare HER and LER backgrounds, have a look at [LER](#) and [HER](#) single beam data. During normal data taking we're dominated by the HER background.
  - When you check the hit map, look for new and unusual patterns (clusters of PMTs with the same color etc.): they are likely to trigger a problem in the DIRC [hardware](#). The table below aims at summarizing the main problems which should be clearly visible on the hit map.

Hit map pattern	Most likely problem

2 consecutive sectors (with even and odd numbers counting clockwise) show only magenta stars (all PMTs unefficient).	HV crate or ROM OFF
1 sector completely unefficient	Front-End crate OFF or DCC not working
1 'radius' of unefficient PMTs	Dead DFB
1 'compact' block of 16 unefficient PMTs	1 HV channel OFF or at a voltage lower than the nominal value
A set of unefficient PMTs (a multiple of 8) belonging to the same DFB	Likely a faulty DFB component: check with DIRC electronics experts!
A 'discontinuity' in the color map in one sector, for instance some blue (i.e. inefficient) PMTs surrounded by black ones.	FEE or HV problem
1 Christmas tree (see below)	A PMT has a hole in its glass cover and is sending light in the DIRC.

and so on and so forth...

- Information about the mapping of DFBs and HV channels in a DIRC sector can be found in the [DIRC numerology](#) webpage.
- A Christmas tree is usually visible on the occupancy map: see this [picture](#) for instance. It shows 2 Xmas Trees: one in sector #0 (row #7; column #1) and sector 5 (row #38; column #22).

Another way to identify a Xmass tree is to take a [standalone run](#): for instance, a Xmas tree is visible in sector 2 of this [plot](#) taken in March 2005: few PMTs in this sector show a rate two orders of magnitude higher than the average one in all other sectors.

There are two ways to fix a Xmas tree. First, one can turn off the HV channel which is powering it. This can only be a temporary fix as 15 additional PMTs get unefficient. During the next access, the Xmas tree PMT must be disconnected to get the recover the other PMTs. Finding the culprit PMT can be tricky: look for an unefficient PMT surrounded by PMTs showing a huge activity. The Xmas tree is likely to be the silent one. The only way to know whether the correct PMT has been unplugged consists in powering the channel and seeing if it trips or not. Do not forget to label the PMT you unplug (when? Who? Why? Which PMT? etc).

- [Page 2](#)

'Dead' PMTs are marked by red stars on this map -- all other PMTs are printed in blue -- see an [example](#) here. The number of 'dead' PMTs should remain pretty constant provided that enough events have been integrated in the run. The criterion to identify whether a PMT is 'dead' or not is

explained below in the 'Page 3' section.

- **Page 3**

The quantity used to decide whether a given is 'OK' or 'dead' during a given run is the ratio between the number of collected hits and the number of events processed by the FM. We cut on this ratio: PMTs below the threshold are called 'dead' whereas the other ones are declared 'healthy'. Normalizing the number of hits per PMT makes the judgment 'independent' from the run duration.

This [page](#) displays the histogram of this quantity for the whole DIRC (top plot; the current cut is visible as well) while the bottom plot shows the number of 'dead' PMTs per sector.

These quantities appear pretty stable over wide run-ranges and so significant variations should trigger a problem with some PMTs (Xmas tree...)

- **Page 4**

Four plots are displayed -- during [collision data](#), they look like [this](#).

- [Plot \(a\)](#) shows the number of hits integrated by each sector during the whole run. Sector 4 has usually the highest occupancy during colliding data taking. This effect is also visible on plots (c) and (d).
- [Plot \(b\)](#) shows the histogram of the event FX (*'Feature Extracted'*) status. Most of the events have an FX status equal to 0 which means they are OK (dataflow error-free); note that the y-axis of this plot has a logarithmic scale. Entries in any other bin trigger some errors -- 'fatal' flags are above 15. The correspondance between bins and errors can be found [here](#). During the data taking, few errors can randomly appear, especially in bin #11 (*'TdcDataTruncated'*: some DIRC TDC chips couldn't store all the hits they collected due to a limited buffer size). The same plot is displayed in the DIRC JAS webpages browsed several times by the DQM during each run. Instructions are usually to page experts when such damages show up. To avoid unnecessary pages, thresholds have been setup on JAS -- in particular for bin #11 where entries cannot be avoided, especially when the background is high. Thus, the DIRC oncall expert is contacted only if one bin exceeds its 'tolerance' value during a run.
- [Plot \(c\)](#) shows the average number of PMT hits per event in each sector. This is a 'normalized' version of plot (a): each of the 12 bins has been divided by the number of events processed by the Fast Monitoring.
- [Plot \(d\)](#) is a 2D-plot showing the distribution of the (corrected) hit timing in the 12 sectors; signal hits have a timing close to 0. The shape of the plot is roughly sector independant: additional background (hits with random timing) is normally visible in sector #4 and also in sectors #3 and #5.

During [cosmics data taking](#), plots (a), (c) and (d) look quite [different](#). The number of integrated hits has a strong  $\Phi$ -dependence and the hit corrected timings are shifted in the negative directions by a few tens of ns (this last difference will be clearer on the next two pages).

- **Page 5**

There are two plots on this page -- see examples for [collisions](#) and [cosmics](#) data taking.

- The first plot (a) shows the distribution of hits per module (= couple of DIRC sectors). There are thus six entries per event in this plot. The higher the background, the stronger the tail in the distribution -- note the log. scale on the y axis.  
If the run ended with a beam loss, there can be entries at or around 0 meaning that some modules were almost empty in a few L1 accepted events: -- see an [example](#) here.  
Rerunning offline the Fast Monitoring on the [xtc](#) file clearly showed that these strange events are the last ones recorded just before the beams get lost. Hence, we believe this is induced by the way the beams are lost rather than by a problem on the DIRC.
- The second plot (b) shows the distribution of the corrected TDC hits in ns. On top of the histogram, there is a fit of the distribution by a Gaussian + a zero-order polynom accounting for background.

- **Page 6**

It shows the distribution of corrected TDC hits in each of the 12 sectors. Obviously, each histogram should be similar to plot (b) in page 4 -- see examples for [collisions](#) and [cosmics](#) data taking.

Among the main differences between the two example plots, note the difference in the fitted meantime : it is shifted by a few tens of ns for cosmics. The occupancies also show larger variations between sectors in these conditions.

Note that the 12 plots have the same vertical scale: the numbers of hits integrated in each sector are directly comparable. The same convention applies in the occupancy plots of the next pages.

- **Page 7**

It shows the number of TDC hits per harness (= per DFB) in each sector. As the PMTs connected to a given DFB are distributed along a 'radius' (see the [DRC numerology](#) for details), the histogram should be pretty flat in a given sector (1 empty bin = 1 problematic DFB). The occupancy pattern in the whole DIRC obviously depends on the data collision mode: [collisions](#) or [cosmics](#).

- **Page 8**

It shows the number of TDC hits per HV group in each sector. These histograms are not expected to be flat. Indeed, the distribution has rather a 'bump-like' shape with a steep increase (roughly the 15 first groups) followed by a small plateau (untill around the HV group 20) and finally a softer decrease for high HV group numbers. Nothing physical in this shape, just a consequence of the way the HV groups are numbered in the sector -- see [DRC numerology](#) for details on this convention. Different patterns are expected between [collisions](#) and [cosmics](#) data taking.

- **Page 9**

It shows the mean time (in ns) per HV group in each sector. The histograms are expected to be flat. A mean value significantly higher or lower than the average is likely to trigger problems, for instance from a DFB. At least two bins in sector #4 show strange fluctuations which have been

present for years but which have not yet been studied in details -- see one [example](#).

- [Pages 10 and 11](#)

These pages show the chip errors per module (couple of DIRC front-end sectors read by the same ROM). They are usually empty.

These plots only look at DIRC errors (contrary to some of the following pages which get inputs from the whole detector). Chip errors are not equivalent to dataflow damages -- there are chip errors which are not dataflow damages and vice-et-versa. The most frequent DIRC chip errors are TDC buffer overflow or timestamp disagreement between a chip and the DAQ. More information should be soon available in the DIRC online documentation currently in prepar.

- [Pages 12 and 13](#)

These plots show the BaBar and DIRC trigger timing versus the trigger line. The y-axis have 4 values: 0, 1, 2 and 3. These are clock ticks: the BaBar DAQ clock works at 60 MHz whereas the trigger clock only works at 15 MHz ( $60 \text{ MHz} / 15 \text{ MHz} = 4$ ). There are several [trigger lines](#) in BaBar:

- lines #0 to #23 are used to select events;
- lines #24 and #25 scan the regions inside the LER and HER trickle injection inhibit windows and are thus thrown away by the Fast Monitoring;
- lines #26 and #27 are used for specific purpose and are of not interest here;
- finally, lines #28 and #29 are cyclic triggers running at 30 MHz.

So, in the first page, all trigger lines should be on the same clock tick line ('3'), apart lines #28 and #29 which can have entries on two clock tick lines ('1' and '3'). On the other hand, in the second page, DIRC trigger timings are all on the clock tick line '1'.

On both pages, the bottom two plots show events which are 'synchron' and others which are not. 'Out of synch' events are normally due to EMC -- DIRC data have always been 'synchron' so far. A (not 100% reliable) way to confirm that the DIRC is synchronized is to look at the bottom left plot in the second page: all 'out of synch' events should be on the clock tick line '1', i.e. where they are expected to stand for the DIRC. *Obviously, if the timing shifts were a multiple of four 60 MHz clock ticks, this particular plot couldn't show the problem.*

The sum of the two bottom plots should give the top plots.

Examples of plots for collision data: [first page](#) and [second page](#).

- [Pages 14 to 17](#)

These 4 pages deal with the hit (corrected) arrival times for events selected by the 24 L1 trigger lines. Background hits are flat over time whereas signal hits peak around 0. These plots are of no particular interest on a daily basis but can be extremely useful if the DIRC has a problem.

- [Page 18](#)

This histogram shows the average extent of data per module. It should be roughly flat. One bin at 0 would obviously trigger a problem with a ROM or upstream it in the dataflow but it would certainly have shown up in previous Fast Monitoring pages.

- **Page 19**

For each of the 6 DIRC modules, this page shows a scatter plot: time (in  $\mu$  s) needed to feature-extract the ROM data versus the size of the xtc block of data. For the time being, there is nothing particular to look at in these plots.

- **Page 20**

This plot shows the 'relative F-Transition damages' which are exactly the 'dataflow damages'. *As the plot shows damages for the whole BaBar and not for the DIRC only, that page is likely to be removed soon: the dataflow has much better diagnostic tools!*

## How to update the DIRC JAS references

- To ease the DQM task and to make sure that problems are caught quickly, the DIRC is using reference histograms which are superimposed to the live data -- default references are normally for colliding beam data taking.
- When data taking conditions change significantly (higher lumi & higher background, long downtime due to PEP repairs during which only cosmics data are taken, non-transient change in the DIRC hardware configuration etc.) it may be necessary to change the references to match the data.
- The best place to make these changes is **IR2**  $\Rightarrow$  it should be done by the ops. manager or the oncall person, in agreement with the fastmonitoring expert. Creating new reference histograms is the responsibility of the fastmonitoring expert or the ops. manager -- usually this may not be needed because different sets of references are available.
- The reference files for the DIRC are stored in the directory [/nfs/bbr-srv02/u4/Monitoring/References/Drc/LiveFastMon](#). On 2006/03/25, the content of this directory was the following:

```
bbr-dev100|References/Drc/LiveFastMon>ls -lart
total 120
drwxrwsr-x 4 babar    bfactory 96 Apr 3 2001 ..
-rw-r--r-- 1 narnaud bfactory 24576 Feb 14 17:29
drcRefs_20060214.hbook
-rw-rw-r-- 1 narnaud bfactory 20480 Feb 15 21:06 fxstatusref.
hbook
-rw-r--r-- 1 narnaud bfactory 32768 Feb 15 21:14
drcRefs_lowLumi_20060215.hbook
-rw-r--r-- 1 narnaud bfactory 32768 Feb 15 21:15
drcRefs_cosmicsllpassthru_20060215.hbook
```

```
lrwxrwxrwx 1 narnaud bfactory      40 Mar 24 15:35 LiveFastMon.
hbook -> drcRefs_cosmicsllpassthru_20060215.hbook
drwxrwsr-x 2 babar    bfactory   8192 Mar 24 15:35 .
```

Like the fastmonitoring outputs, the reference histograms are stored in PAW hbook files (yeap!). The set of references currently in use must be called `LiveFastMon.hbook` and so **changing the set of references just requires to modify that symbolic link**.

Current references have all been created during Run 5b in 2006. You may find what you're looking for in the following files.

- `drcRefs_cosmicsllpassthru_20060215.hbook` should be used for long period of **cosmics** data taking. These reference data were taken with the ORC config. set to `COSMICS_L1PASSTHRU`, all detectors included and the magnetic field ON. *Minor differences between live data and these reference plots will be visible if the conditions of data taking are different (no DCH or no EMC or no magnetic field etc.) but these references should work well for any type of cosmics.*
- `drcRefs_lowLumi_20060215.hbook` should be accurate for colliding beam data taking with low and moderate background -- and hence lumi not too high: around 5-6  $10^{33}$ .
- `drcRefs_bkg_20660419.hbook`: high lumi + high background.
- `drcRefs_highLumi_20660424.hbook`: high lumi + 'normal' background.
- Once `LiveFastMon.hbook` points to the proper file, the new DIRC references should be merged with the ones from the other subsystems. To do this, go to `/nfs/bbr-srv02/bfdist/Builds/MonitoringData` and execute `LfmUpdateGui`. The documentation of this nice GUI is available [here](#). Basically, the only thing you need to do is to click on `Merge References`, to motivate your changes in the popup window which appears then and to wait for the merging to be completed. Then, you should check that the file `LiveFastMon.hbook` in the directory `/nfs/bbr-srv02/u4/Monitoring/References/merged` has indeed been recreated.
- Last but not least, you should go to the DQM console (`BFO-CON05:0.0`) and restart the reference server on the `Fast Monitoring Control` window. This GUI should look like [this](#); restarting the reference server is done by left-clicking on the green `RUNNING` button at the top right. Once it's done, the new reference plots should be visible next time JAS refreshes the plots.  
Note: Restarting the reference server can be done anytime.
- In case of problem, do not hesitate to contact OEP experts, in particular Jim Hamilton.

---

**How to change the tag of DrcMonTools in IR2**

- Never do such update without being in IR2! You need to get the green light from the pilot and to make sure that the change is working as expected. The DIRC FM expert is responsible for providing and testing the new tag.
- Updating the version of `DrcMonTools` running in IR2 is simple.
  - Go to `/nfs/bbr-srv02/bfdist/Builds/MonitoringData` and run `LfmUpdateGui`, the GUI which is also used to [merge new references](#) -- documentation available [here](#).
  - Write the new tag in the dedicated field (note the current one somewhere in case you need to back out the changes) and click on the `Update` button.
  - Then, click on `Install HTML` which will do the work for you. Use the popup window to explain the reasons of the tag change.
  - Then you should be done; restarting `LiveFastMon` by clicking on the corresponding stream button (see the GUI on the DQM console [here](#)) should make the new code running.
- In case of problem, do not hesitate to contact OEP experts, in particular Jim Hamilton.

## How to create new JAS references

- Log as `babardrc` and go to the directory `~/JasReferencePlots`: the important script is `createReferences.pl`. The script should be well documented. Basically, one has to update a few parameters: the name of the output reference hbook file, the list of FM runs used as reference and the year and month when these data were taken (required to locate the directory where the FM files are stored).
- No more input is needed: simply running the perl script will produce the new reference file.
- Modifying the set of reference histograms stored in the hbook file (by adding/removing some of them or changing one particular plot) is trickier: Some additional knowledge of `PAW` will be required.

## How to rerun the Fast Monitoring on a xtc file

This recipe comes from Jim Hamilton (and works!)

- Setup a test release with a recent and known to be working in IR2 release.
- Checkout the `FastMonStreams` package and compile it (`gmake lib` and `gmake bin` should do the job).

- Copy the file [runFmcPlayback](#) in the top directory of your release.
- Now you can run on an xtc file. For instance, to run on 100 kEvents from Run 66126:  
ambientboot  
unsetenv CFG\_DEFAULT\_IMPL  
tcstage 0066126-001 [*to access the xtc file*]  
runFmcPlayback /nfs/babar/tcfiles/babar-0066126-001.xtc -n  
100000 <*user-defined path of the target hbook file*> [*the nfs path points to the location of staged xtc files*]

Running on xtc file is likely to take time  $\Rightarrow$  send your job in batch mode! Recent xtc files are directly available in the `/nfs/bbr-nfs102/pubxtc/theData/` directory.

---

## Useful Links

- [Fast Monitoring webpage](#)
- [DIRC Operations HyperNews Forum](#)
- [BaBar DIRC Home Page](#)

---

Last modified: Thu Jun 29 17:06:52 PDT 2006

This page is maintained by Nicolas Arnaud

This page was generated by the BaBar web unwrapping script. Original page was:

<http://www.slac.stanford.edu/BFROOT/www/Detector/DIRC/Monitoring/docFM.html>