# Event Class Handling

## Pass 5 and Prior Implementations (ST v9r4p2 and earlier)

- Prior to DC2, there was a separate event class designation for each section of the LAT where the event converted, e.g., DC1::FRONT, DC1::BACK were two distinct classes for otherwise identical event selections. These were given values of 0 and 1, respectively, in the EVENT_CLASS column of FT1. The CONVERSION_TYPE column was set to have the same values, indicating a front (thin) or back (thick) section conversion.
- For DC2, there were two distinct sets of cuts that divided the events into two classes, A and B. This resulted in 4 possible values for the EVENT_CLASS column, and two values for CONVERSION_TYPE:

| event class name | EVENT_CLASS | CONVERSION_TYPE |
|---|---|---|
| DC2::FrontA | 0 | 0 |
| DC2::BackA | 1 | 1 |
| DC2::FrontB | 2 | 0 |
| DC2::BackB | 3 | 1 |

  Besides still containing redundant information, this required special code in gtselect and the Likelihood tools to handle.
- For Pass 5, Bill defined a new classification tree variable, CTBClassLevel, and proposed three different cuts corresponding to what he envisaged as the corresponding kind of science analysis:

| CTBClassLevel cut | science analysis designation |
|---|---|
| > 0 | "transient" |
| > 1 | "source" |
| > 2 | "diffuse" |

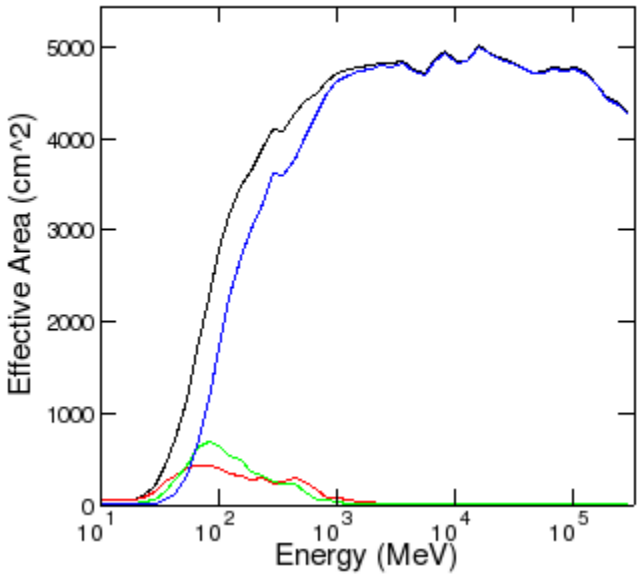
  Even though the CTBClassLevel value did partition the event data just as the DC2 class A and class B designations did, the mechanism for handling classes that was used for DC2 was unwieldy, and so it was not modified to handle these new classes. A temporary ad hoc procedure was adopted in which a CTBCLASSLEVEL column was added to the FT1 files (outside of the FFD definition), and fcopy or some other tool besides gtselect was used to make the CTBCLASSLEVEL cuts. The EVENT_CLASS and CONVERSION_TYPE columnss simply indicated front (0) vs back (1), and three separate sets of IRFs were generated for each of the cuts. The principal disadvantage of this scheme is that gtdiffrsp needs to be run separately for each CTBCLASSLEVEL cut, and there was no DSS keyword handling implemented to account for which cut was applied for any given dataset.

## Proposed Scheme (25 Mar 2008)

We will use the EVENT_CLASS column to designate the single event class to which are particular event belongs. In the case of Pass 5, the CTBCLASSLEVEL values minus 1 (in order to have them start at zero) can be used. Separate front vs back conversion type information will not be encoded in the EVENT_CLASS value, but it will instead just reside in the CONVERSION_TYPE column, thereby factoring out the redundant information that had been contained in previous implementations. This scheme will have several consequences:

- Separate IRFs for each class will need to be generated, but instead of generating the IRFs using events that have a minimum CTBClassLevel, as we currently do, the IRFs will need to be generated for significantly smaller numbers of events for which CTBClassLevel is a specific value, e.g., CTBClassLevel==1, CTBClassLevel==2, CTBClassLevel==3. In this case, only the IRFs for CTBClassLevel==3 would be the same as an original

Pass 5 set (diffuse class). There would be substantially fewer events for CTBClassLevel==0 or 1. Here is a plot of the effective area for the front section for the three CTBClassLevel values using the AllGamma_v13r9p3_Lyon data:




| EVENT_CLASS | CTBClassLevel cut | curve | number of events |
|---|---|---|---|
| 0 | ==1 | red | 0.27 M |
| 1 | ==2 | green | 0.24 M |
| 2 | ==3 | blue | 4.30 M |
| 0,1,2 | > 0 | black | 4.81 M |

The smaller number of events overall and the significant dearth of any events above 1 GeV will cause difficulties for characterizing the PSF and energy dispersion for the EVENT_CLASS 0 and 1 values.

- Since each event has a unique set of IRFs, gtdiffrsp will only have to be run once regardless of the downstream selections on EVENT_CLASS.
- The gtselect interface will need to be modified. I propose two new sets of parameters to handle selections on the EVENT_CLASS and CONVERSION_TYPE columns separately:

```
gtselect evclsmin=1 evclsmax=2 convtype=0
gtselect evclsmin=1 convtype=0
```

Either one of these will produce a set of events that are the same as would be produced using "CTBCLASSLEVEL > 1 && CONVERSION_TYPE==0" in fcopy or fselect. It is anticipated that users will only want to specify a range of EVENT_CLASS values and not wish to make non-contiguous selections such as "EVENT_CLASS==0 && EVENT_CLASS==2". It is likely that users will not want to make selections excluding the higher event classes, so evclsmin may be all that is needed. These will be hidden parameters and the defaults will be to make no selections on EVENT_CLASS or CONVERSION_TYPE.

- Depending on the number of event classes included, gtexpmap will run correspondingly slower. Since the Psf integral over the ROI is by far the most cpu-intensive part of the gtexpmap calculation and since it has to be performed separately for each set of IRFs, event selections that include three event classes (the proposed default for Pass 5) will take three times as long as the current implementation. (There may be a way of mitigating this by having the different PSFs share the cached values of the ROI integrals.)
- In order to enable backwards compatibility with old FT1 files, a new keyword has been introduced to the EVENTS HDU: EVCLSVER. If this is equal to "0" or missing, then the old implementation is assumed. If it is "1", then the new scheme will be used.
- The format of the IRF files will change slightly. In order to facilitate these changes, multiple event classes that partition the data will be contained within a single file and stored as separate rows in the binary tables. In keeping with the the organization of CALDB wherein the front and back sections of the LAT are different "detectors", separate IRF files for front and back IRFs will be maintained. Here is an example header for this new format. The only difference from the old headers is that NAXIS2=3, indicating IRFs for three event classes are present.
- The IRF names to be used by the tools via the irfs parameter will be standardized. These will be determined from the information in the caldb.indx file. I've written a tool in the irfs/irfLoader package that will allow users to discover the valid available names:

```
ki-rh2[jchiang] gtirfs
P5_v13 ( = P5_v13_0::FRONT + P5_v13_0::BACK + P5_v13_1::FRONT + P5_v13_1::BACK + P5_v13_2::FRONT +
P5_v13_2::BACK )
P5_v13_0::BACK
P5_v13_0::FRONT
P5_v13_0_diff ( = P5_v13_0_diff::FRONT + P5_v13_0_diff::BACK )
P5_v13_0_diff::BACK
P5_v13_0_diff::FRONT
P5_v13_0_source ( = P5_v13_0_source::FRONT + P5_v13_0_source::BACK )
P5_v13_0_source::BACK
P5_v13_0_source::FRONT
P5_v13_0_trans ( = P5_v13_0_trans::FRONT + P5_v13_0_trans::BACK )
P5_v13_0_trans::BACK
P5_v13_0_trans::FRONT
P5_v13_1::BACK
P5_v13_1::FRONT
P5_v13_2::BACK
P5_v13_2::FRONT
PASS4 ( = PASS4::FRONT + PASS4::BACK )
PASS4::BACK
PASS4::FRONT
PASS4_v2 ( = PASS4_v2::FRONT + PASS4_v2::BACK )
PASS4_v2::BACK
PASS4_v2::FRONT
PASS5_v0 ( = PASS5_v0::FRONT + PASS5_v0::BACK )
PASS5_v0::BACK
PASS5_v0::FRONT
PASS5_v0_DIFFUSE ( = PASS5_v0_DIFFUSE::FRONT + PASS5_v0_DIFFUSE::BACK )
PASS5_v0_DIFFUSE::BACK
PASS5_v0_DIFFUSE::FRONT
PASS5_v0_TRANSIENT ( = PASS5_v0_TRANSIENT::FRONT + PASS5_v0_TRANSIENT::BACK )
PASS5_v0_TRANSIENT::BACK
PASS5_v0_TRANSIENT::FRONT
ki-rh2[jchiang]
```

Setting chatter=3 or greater will cause the tool to print the irfs that are available prior to Pass 4. The P5_v13 IRFs show the Pass 5 IRFs under the new scheme. Any of these values should be valid inputs to gtrspgen. However, since the DSS keywords will be parsed by the Likelihood tools, only the combined versions should be used, e.g., irfs=P5_v13 is ok, irfs=P5_v13_0::FRONT is not.

- Packages affected:
  - Likelihood
  - pyLikelihood
  - pyExposure
  - dataSubselector
  - irfs/latResponse
  - observationSim
  - fitsGen
  - map_tools (? no idea since it probably doesn't use the standard interface to the response functions)
  - rspgen (Only if it parses the DSS keywords. Otherwise, the onus is on the user to choose the proper IRF and not make cuts that do not have a corresponding IRF representation.)