

# How to Use CSG Event Flags

---

Reiner Hauser, rhauser@fnal.gov

v1.0, 15 May 2005

## Contents

|   |   |
|---|---|
| <a href="#">1 Introduction</a>                | 1 |
| <a href="#">2 Existing Documentation</a>      | 1 |
| <a href="#">3 Preparing your executable</a>   | 1 |
| <a href="#">4 Updating your framework RCP</a> | 2 |
| <a href="#">5 Example</a>                     | 2 |

## 1 Introduction

Both the skimming and p17 reconstruction processing puts a number of flags into each event. These flags allow a fast selection without actually unpacking any of the real data. This document describes how to add this to an analysis program.

## 2 Existing Documentation

If you are familiar with the D0 framework and RCPs, have a look at the documentation in the `io_packages`:  
[http://www-d0.fnal.gov/d0dist/dist/packages/io\\_packages/devel/rcp/FilterEventFlags.rcp](http://www-d0.fnal.gov/d0dist/dist/packages/io_packages/devel/rcp/FilterEventFlags.rcp) <[http://www-d0.fnal.gov/d0dist/dist/packages/io\\_packages/devel/rcp/FilterEventFlags.rcp](http://www-d0.fnal.gov/d0dist/dist/packages/io_packages/devel/rcp/FilterEventFlags.rcp)>

This should be enough to get you started.

## 3 Preparing your executable

You can use the `FilterEventFlags` package with any executable you normally use. You should add the following line in the `bin/OBJECTS` file:

```
FilterEventFlags
```

Then recompile and relink as usual.

## 4 Updating your framework RCP

You should add the following two lines to your framework RCP:

```
string Flow = "generateUserHeader filterUserHeader removeUserHeader generator decide builder merge filter
bool OverrideFlow = true
```

Then, in your list of packages you execute, add a new entry (`filter` in our case):

```
string Packages = "geo filter read config ..."
```

```
RCP filter = <mypackage FilterEventFlags>
```

You can copy over the `io_packages/rcp/FilterEventFlags.rcp` to your area and modify it.

## 5 Example

Let's say, you are interested in what was formerly the 2MU skim. This is now stored as part of the `MUinclusive` skim. So you add the `SKIM_2MU` tag name to the `FilterEventFlags.rcp` file:

```
string PackageName = "FilterEventFlags"

string Combine = "any" // "any" or "all"

string EventFlags = ( "SKIM_2MU" ) // Positive logic event flags.

string NotEventFlags = ( ) // Inverse logic event flags.

bool InvertResult = false; // Invert final result if true.
```

You can find all the names for event flags from the CSG skimming page.

Now run your executable as usual, e.g. with `d0tools`. There are two things to look out:

- The `events.read` file will not give you a correct account of the number of events in the input files. The reason is that all events that don't have the `SKIM_2MU` tag in our example are filtered away and won't be read at all.
- To get the real number of events plus all those that were filtered, look in the `...out` file for lines that start with `Headers read`:

An example output might look like this:

```
ReadEvent: Closing /sam/cache/boo/recoT_all_0000204941_mrg_249-255.raw_p17.03.03
First header read: Run number: 204941, Event Number: 48052248
Last header read: Run number: 204941, Event Number: 48476131
```

---

Headers read: 15511

First event read: Run number: 204941, Event Number: 48052209

Last event read: Run number: 204941, Event Number: 48476131

Events read: 10290

The time spent on skipping an event is very small, so the total time to run over the full `MUinclusive` skim should be comparable to the old separate skims. It will be a bit slower, since the all the files still have to be delivered to you by SAM.