# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>HltLine</td>
<td>1</td>
</tr>
<tr>
<td>Concept</td>
<td>1</td>
</tr>
<tr>
<td>HltLine in Python</td>
<td>1</td>
</tr>
<tr>
<td>HltLine in C++</td>
<td>1</td>
</tr>
</tbody>
</table>
HltLine

Most of this section is stolen from Gerhard Raven's presentation. (LHCb SoftwareWeek Thursday, March 19, 2009)

Concept

• An Hlt decision involves more than a single algorithm
  ♦ one might pick some L0 candidates
  ♦ another (sequence) may do some VELO reconstruction
  ♦ a third may match the above two
  ♦ ....

• In addition, it has some ‘standard’ pre/post processing:
  ♦ ‘entry’ filters (ODIN, and/or L0, and/or other HLT decision(∗))
  ♦ Prescale
  ♦ “The algorithm (sequence) which most people think of as the decision”
  ♦ Postscale

• And it needs to record the final yes/no result
• And possibly catch errors in the ‘hosted’ algorithms
• Let’s call this entire structure an ‘HltLine’

HltLine in Python

HltLines are created by calling some dedicated python code

• Enforces uniform naming convention
  ♦ might add some additional rules!
• Makes it easier to write HltLines
  ♦ Can ‘copy-and-modify’ entire lines to easily changes create variations:
    ◦ eg. clone, decrease threshold and increase prescale
      • For an example of cloning a line, see the section on Hlt2Lines.
  ♦ re-use pre-defined sequences (of sequences, of ... ) through ‘bindMembers’
    ◦ Useful, as each line SHOULD BE independent, and SHOULD not rely on the results of other lines.
    ◦ This last comment applies in Hlt2 as well as Hlt1. The decision of an Hlt2Line should be independent of that of any other Hlt2Line.
  ◦ Caveats:
    · It might of course make sense to create an Hlt2 line which only runs on the output of an Hlt1 line, and hence depends on the decision of that Hlt1Line.
    · You may wish to add heavily prescaled 'lines' to monitor the intermediate steps in an Hlt2Line.
    ◦ For an example of bindMembers, see the Hlt2SharedParticles page.
• Registers the existence of a line -- this is used to actually configure Hlt1/Hlt2.

HltLine in C++

• Invokes the various stages
  ♦ Each stage is an independent algorithm, HltLine only relies on ‘FilterPassed’
• Updates an entry in HltDecReports ‘as it goes along’
Catches exceptions and errors in the ‘hosted’ algorithms, updates HltDecReport entry accordingly, and recovers (hopefully)

♦ As a result, the status is recorded in TES (in HltDecReports) in uniform and reliable way:
  ♦ convention: all configured Hlt1 decisions appear in HltDecReports, regardless of their result
  ♦ at some point we may remove negative decision before conversion to rawbank -- but only after it has been shown that on readback we can reliably recreate the (relevant) missing information from the configuration

♦ Accept/Reject is recorded separately from ‘how far we got’: even if the event ‘fails’, it could be accepted, eg. because HltLine caught an exception: a (limited!) number of such errors will/could result in an ‘accept’.

-- StephanNies - 12 Apr 2009