TriggerDecision and its Maker

History

- `TriggerDecision` is intended as a user interface to the trigger **configuration** and **decision** for each event. It exists since release 11.0.5 (Monika).

- Up to 12.0.1, different `TriggerDecision` objects were produced for level 2 and event filter ("by hand" only).
  - Level 1 information had to be gathered from a different class, CTP_Decision.

- `TriggerDecisionMaker` was introduced in 12.0.1 to produce a `TriggerDecision` object containing all satisfied signatures for level 1, and HLT.

- In release 12.0.2, `TriggerDecision` was modified to contain all trigger levels in different internal maps and add some functionality.

- Notes:
  - `TriggerDecisionMaker` and `TriggerDecision` are consistent **for each release** and work as a pair.

  - A bug was found in `TriggerDecisionMaker` in release 12.0.2 (thanks to Katrin Facius) which `attributes event filter signatures to level 2` in `TriggerDecision`. This is corrected in 12.0.3 nightlies. More info in the Wiki. Not all methods are affected.

Documentation

- Wiki pages with information and example code for TriggerDecision: https://uimon.cern.ch/twiki/bin/view/Atlas/TriggerDecision
- and TrigDecisionMaker: https://uimon.cern.ch/twiki/bin/view/Atlas/TriggerDecisionMaker

- Some info on trigger-aware tools in previous talk in PESA Algorithms and Performance, 29 June 06: http://indico.cern.ch/materialDisplay.py?subContId=2&contribId=s0t0&materialId=slides&confId=a0638

Example code

Below is some example code to retrieve and manipulate the TriggerDecision object. These are snippets from a (private code) class called TrigDecisionChecker. ***The code is attached to this page*** (note: twiki changed the name of the job options file to end in `.txt` this should be changed back to `*.py`). If you use it in your own package, remember to add this algorithm to the `components/*_entries.cxx` file.

TrigDecisionChecker is an Algorithm class which has been used to help developing TriggerDecision and TriggerDecisionMaker. In the `execute()` method, it checks if a TriggerDecision exists with a certain key with the line: `(m_storeGate->contains( m_trigDecisionKey )` . Then it retrieves all TriggerDecision objects in StoreGate (there may be more than one if the trigger has been re–run on ESD to produce a AOD, for example). This is done with the lines:

```
// retrieve all TriggerDecision objects
 log <<MSG::INFO  <<"Retrieving all TriggerDecision objects"  <<endreq;
 const DataHandle trigDec;
 const DataHandle lastTrigDec;

 sc = m_storeGate->retrieve(trigDec,lastTrigDec);
 if (sc.isFailure()) {
   log  <<MSG::INFO  <<"No TriggerDecision found"  <<endreq;
   return  StatusCode::SUCCESS;
 }

 log  <<MSG::INFO  <<"TriggerDecisions retrieved"  <<endreq;
```

Once the (usually one, but maybe several) TriggerDecision objects have been retrieved, it can be queried to determine if the trigger, has accepted the event:

```
 log <<MSG::INFO  <<"TriggerDecisions retrieved"  <<endreq;

 for (int i=0; trigDec != lastTrigDec; ++trigDec, ++i) {

   log  <<MSG::INFO  <<"Looking at TriggerDecision "  <<i  <<endreq;

   if (trigDec->isTriggerPassed()) {
     log  <<MSG::INFO  <<"Trigger passed"  <<endreq;
   } else {
     log  <<MSG::INFO  <<"Trigger failed"  <<endreq;
   }
```

Or if the event was was accepted by level 1 or level 2:

```
   if (trigDec->isDefinedL1()) {
     log <<MSG::INFO  <<"L1 defined "  <<endreq;

     if (trigDec->isPassedL1()) {
       log  <<MSG::INFO  <<"L1 passed"  <<endreq;
     } else {
       log  <<MSG::INFO  <<"L1 failed"  <<endreq;
     }
```

Conclusions

- TriggerDecision and TriggerDecisionMaker are working with no known problems in 12.0.3 nightlies
- They should be very useful to many physics analysis for CSC notes

```
TriggerDecisionMaker            INFO Initializing TriggerDecisionMaker...
TriggerDecisionMaker            INFO Properties:
TriggerDecisionMaker            INFO  doL1            = True
TriggerDecisionMaker            INFO  doL2            = True
TriggerDecisionMaker            INFO  doEF            = True
TriggerDecisionMaker            INFO  TrigDecisionKey = MyTriggerDecision
TriggerDecisionMaker            INFO  TrigConfigL2Key = storeL2Location
TriggerDecisionMaker            INFO  TrigConfigEFKey = storeEFLocation
TriggerDecisionMaker            INFO  IgnoreItemNames = dummy0 dummy1
TriggerDecisionMaker           DEBUG Retrieving Level 1 configuration
TriggerDecisionMaker           DEBUG L1 map has 4 items
TriggerDecisionMaker           DEBUG TrigerItem id=0; signature label=iEM01
TriggerDecisionMaker           DEBUG TrigerItem id=1; signature label=iL1_2EM15I
. . .
TriggerDecisionMaker            INFO REGTEST Run summary:
TriggerDecisionMaker            INFO REGTEST  Events processed    : 10
TriggerDecisionMaker            INFO REGTEST  Level 1 passed      : 10
TriggerDecisionMaker            INFO REGTEST  Level 2 passed      : 1
TriggerDecisionMaker            INFO REGTEST  Event Filter passed : 0
TriggerDecisionMaker            INFO REGTEST  Unprocessed events  : 0
TriggerDecisionMaker            INFO REGTEST  Level 1 errors      : 0
TriggerDecisionMaker            INFO REGTEST  Level 2 errors      : 0
TriggerDecisionMaker            INFO REGTEST  Event Filter errors : 0
TriggerDecisionMaker            INFO REGTEST  StoreGate errors    : 0
```