Proposal for a BS error svc

- following the schema adopted and proposed by the ID community for HLT (LVL2 & EF) & offline (see attached mail)
- thanks to Dmitri Emiliyanov, John Baines, Shaun Roe
- data access generalities (basically common between MS and ID)
  - LVL2 asks for selected data (RDO) to the RawDataProviderTool
    - RawDataProvider: uses the RODdecoder (to decode BS to RDO)
      - RODdecoder scan BS fragments and fill RDO collections + add to RDO container
  - EF asks for selected data (RIO=PRD) to the PrepDataProviderTool
    - PrepDataProvider used RawDataProvider (BS->RDO) and RDOdecoder (to decode RDO hits into offline hits), fill collections + add collections to RIO container
  - Offline: asks for all RDO to RawDataProviderTool + ask for all PRD to PrepDataProviderTool

LVL2

- RawDataProvider:
  - provide data `virtual StatusCode convert(const std::vector<IdentifierHash>&);`
  - provide bs errors (in the ID example) `virtual const std::vector<int>* fillPixelDataErrors() = 0; virtual const std::vector<int>* fillSCT_DataErrors() = 0;`
- In the LVL2 Algorithm:
  - if (m_providerTool->convert(...).isRecoverable())
  - {//bs errors detected
    - const std::vector<int>* errVec = m_providerTool->FillDataErrors(); //get the errors cached by the providerTool for transmission to LVL2 algos
    - // take some action: monitor, skip, etc ...
  - }

data request

error notification

error request

EF

- PrepDataProvider:
  - provide data `StatusCode decode( std::vector<IdentifierHash>& idVect, std::vector<IdentifierHash>& selectedIdVect )`
- In the EF Algorithm:
  - if (m_prdProviderTool->decode(...).isRecoverable())
  - {//rdo to prd decoding errors detected (maybe originating from bs errors)
    - xxx = m_ErrorSvc->GetDataErrors();
    - // take some action: monitor, skip, etc ...
  - }

data request

error notification

error request

OFFLINE

- tracking algorithms access prd collections
- they can check for BS/Conversion errors by interrogating the conditionSummarySvc (or the individual -technology specific- DataErrorSvc)
- before starting pattern recognition
- during pattern recognition and track finding to check for critical conditions being met in the data (missing station layer, etc ...)
- recover / apply protections against efficiency loss or crash ??

DataErrorSvc

- MdtDataErrorSvc
- RpcDataErrorSvc
- TgcDataErrorSvc
- CscDataErrorSvc
  - used directly (and individually) by the EF, since the EF already requests data to the PRD-provider tools individually
  - all inheriting from a common interface ?? MuonDataErrorSvc
- ConditionSummarySvc might give access to all of them
  - used in the offline for example !

RawDataProvider

- ask data decoding to the RODdecoder

```
288  for (ROBFragmentList::const_iterator itFrag = vecRobs.begin(); itFrag != vecRobs.end(); itFrag++)
289  {
290      //convert only if data payload is delivered
291      if ( (**itFrag).rod_ndata()!=0 )
292      {
293          std::vector<IdentifierHash> coll =
294                                      to_be_converted(**itFrag,collections);
295
296          if (m_decoder->fillCollections(**itFrag, *pad, coll).isFailure())
297              {
298                  // store the error conditions into the StatusCode and continue
299              }
300      }
```

- before reset m_cachedErrors (std::map<ErrorCode, HashId>)
- at each fillCollections if sc.isRecoverable() add an entry in m_cachedErrors
- implement a method to provide m_cachedErrors to the LVL2 algorithms

RodDecoder

- depends on the technology specific DataErrorSvc (RDO based and PRD based granularity are both provided in the same Svc)
  - holds a std::map<ErrorCodeRdo, RdoHashIndex> where ErrorCodeRdo is the most severe ErrorCode for that RDO collection, or a combination of the ErrorCodes detected
  - holds a std::map<ErrorCodePrd, PrdHashMax>where ErrorCodeRdo is the most severe ErrorCode for that RDO collection, or a combination of the ErrorCodes detected
  - this maps are event based (filled per event => reset at eventStart incident by the svc itself)
    - in a trigger+offline job at the last stage (offline reco) the maps will be entirely filled with info provided in the decoding on demand (via HLT algos) and in the subsequent decoding of the rest of the event (via offline decoding algorithms)
- while scanning the BS if an error is found
  - define the ErrorCode seen (integer Code)
  - define which RDO collection the error is related to
    - for each rdo collection affected by the error fill an entry in the internal map of the DataErrorSvc