Table of Contents

The official CMS Luminosity Calculation Tools (all instructions below are valid for run until 211831 inclusive)

[[$link][]]

- **LumiDB datatag v12**
  Tue, 03 Dec 2013 19:53:32 +0100
  reload some 2010 runs fixing L1 prescale.list
- **Minor release tag V04-02-10**
  Tue, 03 Sep 2013 11:08:39 +0200
  removed --check-forupdate from lumi scripts
- **switch 2012 official to use pixel lumi (pixelLumiCalc.py)**
  Thu, 29 Aug 2013 09:57:31 +0200
  load 2012 full year pixel lumi, new norm tag PIXELV2, data tag pixelv2, excluding per-bunch
- **Minor release tag V04-02-09**
  Wed, 14 Aug 2013 16:21:28 +0200
  fix pixelLumiCalc.py header display
- **LumiDB datatag v11 more operations**
  Fri, 05 Jul 2013 15:27:17 +0200
  reload 53 2010 runs fixing wrong per-bunch lumi. file list Has effect only on perbunch data.
- **Minor cvs tag V04-02-08**
  Thu, 13 Jun 2013 10:13:17 +0200
  improved timestamp precision in report
- **LumiDB datatag v11**
  Tue, 11 Jun 2013 10:11:42 +0200
  reload run 149294 fixing wrong per-bunch lumi. Has effect only on perbunch data of this run.
- **Minor cvs tag V04-02-07**
  Fri, 24 May 2013 09:28:51 +0200
  fix lumibylsXing return also runs with no trg;fix pixelLumiCalc.py lumibyls --hltpath;fix estimatePileup_makeJSON.py
- **Minor cvs tag V04-02-03**
  Fri, 12 Apr 2013 19:19:06 +0200
  adopt to python2.73 in slc5_amd64_gcc472 environment
- **Minor cvs tag V04-02-02**
  Tue, 12 Mar 2013 15:13:37 +0100
  fix lumibyls --hltpath output

Table of Contents

Set up

- Lumi script toolkit uses a small number of stable external libraries from CMSSW, the exact version of CMSSW shown in the examples is **indicative**.
- The Lumi script toolkit version is identified by the git tag announced by the lumi group. Users are required to checkout/compile/run the scripts from a **local** cmssw project area.

```
ssh to lxplus.cern.ch
scramv1 p CMSSW CMSSW_5_3_9
```

```
cd CMSSW_5_3_9
cmsenv
git clone https://github.com/cms-sw/RecoLuminosity-LumiDB.git $CMSS
cd $CMSSW_BASE/src/RecoLuminosity/LumiDB
git checkout V04-02-10
scramv1 b
cmsenv
```

or alternatively

```
cd CMSSW_5_3_9/src
mkdir -p RecoLuminosity
curl https://codeload.github.com/cms-sw/RecoLuminosity-LumiDB/tar.g
mv RecoLuminosity/RecoLuminosity-LumiDB-* RecoLuminosity/LumiDB
cd RecoLuminosity/LumiDB
scramv1 b
cmsenv
```

- ***Note*** when using tags<=V04-02-09, please always use option **--without-checkforupdate**
- ***Note*** recheckout/recompile or rerun from a different local cmssw area should be followed by 'cmsenv' to reset the running environment.
- ***Note*** Use unix command 'where lumiCalc2.py' (tcsh)/'which lumiCalc2.py'(sh) to check the true path of the scripts. If the path does not point to the correct local area, use cmsenv to correct the environment.
- ***Note*** pixel has limited data of pp runs until end of 2012

Scripts

- lumiCalc2.py (HF), pixelLumiCalc.py (Pixel), lumiNorm.py (corrections and versions), lumiTag.py (data versions), lumiContext.py (L1T,HLT,beam), lumiPlot.py
- pileup estimation tools analyze the output of lumiCalc2.py, instruction see here
- release area
- source code browser

Feature Overview

- report luminosity out of the box
  - ♦ report HF measurements of LHC delivered,CMS recorded luminosity and recorded luminosity in a given hlt path
  - ♦ report luminosity calculated from pixel vertex reconstruction, CMS recorded luminosity and recorded luminosity in a given hlt path
  - ♦ report luminosity with different granularity: per fill, per run, per lumisection, per bunch crossing, per hltpath and can be further filtered by beamenergy,amodetag and timestamp
  - ♦ data in a relational database with distributed web cache (frontier) frontend to ensure worldwide accessibility
  - ♦ tools: lumiCalc2.py, pixelLumiCalc.py

- raw data are versioned
  - ♦ there is always a current data tag open which represents the current best quality raw data
  - ♦ raw data in LumiDB, including HF lumi, pixel lumi, trigger,hlt , automatically acquire a version tag when written
  - ♦ all scripts using raw data have an option **--datatag**
  - ♦ data tags are created by lumi operation when judging necessary: e.g. before re/post load full/partial raw data of a run
  - ♦ tool: lumiTag.py

- luminosity corrections are parametrised, versioned and configurable
    - corrections refer to all the parameters/functions to apply to the raw lumi measurements out of the inital apparatus: e.g. calibration,afterglow, drift,linearity corrections.
    - a named run number sequence marking the validity of the corrections is refered to as a norm tag. The sequence is open and can be appended to.
    - there is always one default norm tag for each lumi type : HF, PIXEL. There can be non-default norm tags
    - all scripts reporting luminosity have an option **--normtag**
    - norm tag is created and appended to by lumi group when judging necessary: e.g. append new calibration due to beam energy change
    - lumi corrections can be completely turned off with the --without-correction switch.
    - tool: lumiNorm.py

Note : combination of --datatag,--normtag reproduces an old result.

Note : users should keep note of the datatag and normtag and specify them to reproduce past results.

- report L1trigger, HLT and beam conditions used by luminosity calculation
    - report trigger menu content on run basis
    - report hlt,l1trigger prescales, counts and deadtime fraction
    - report beam condition and beam intensity
    - tool: lumiContext.py

- produce interactively or luminosity plots according to selection condition
    - tool: lumiPlot

lumiCalc2.py and pixelLumiCalc.py

Feature highlights

- **lumiCalc2.py** reports HF measurements of LHC delivered,CMS recorded luminosity and recorded luminosity in a given hlt path
    - HF lumi are available for: pp8TeV,pp7TeV, pp900GeV,pp2760GeV,HI and selected totem fills.
    - HF lumi data end user availability: ~6 hours to frontier after run closing

- **pixelLumiCalc.py**, Pixel reconstructed CMS recorded luminosity and recorded luminosity in a given hlt path
    - identical options as lumiCalc2.py except for **-b**, **--xingMinLum** filter, and lack of **lumibylsXing** action
    - **delivered** in this case is not true LHC delivered but rather the luminosity CMS would have recorded without trigger deadtime.
    - pixel lumi are available for : pp7TeV(2011). pp8TeV(2012)

actions

- **overview** : integrated luminosity delivered by LHC and recorded by CMS
- **delivered** : integrated luminosity delivered by LHC and the beam conditions.
- **recorded --hltpath** : recorded luminosity in a selected hlt path.
- **lumibyls** : integrated luminosity delivered by LHC and recorded by CMS in each lumi section.
- **lumibyls --hltpath** : recorded lumi by CMS and in the hltpath in each lumi section.

- **lumibylsXing** : integrated luminosity delivered by LHC and recorded by CMS (/ub) in each lumi section and instataneous lumi (Hz/ub) in each bunch crossing. Unavailable for pixelLumiCalc.py.

input

- run/ls filters
  - **-r** runnumber, **-f** fill number, **-i** selection json, **--amodetag** accelerator mode (PROTPHYS,IONPHYS), **--beamenergy** nominal single beam energy in GeV.
  - **-b** selects lumi sections by their beam status. Note: pixelLumiCalc.py does not accept the -b argument (since the pixel detector is only on during stable beams anyway.)
  - **--begin**, **--end** selectors are automatically recognized as either run number, fill number or a time string. In case of a time string, it filters to the lumi section level. Except for lumiContext.py where timestamp filters to run.
  - above filters can overlap each other. For example, **-i** selection can be further narrowed down combined with **--begin**, **--end**, **-r**,*-f*.

- Be aware that the delivered lumi returned by **-i** selected LS means their corresponding luminosity before CMS trigger deadtime. To get the true LHC delivered luminosity for an entire run/fill, other general range filters, e.g. **-r**, **-f**, **--begin**, **--end**, should be used instead.

- effective luminosity and hltpath
  - action **recorded** or **lumibyls** combined with a --hltpath selection gives the effective recorded luminosity in that path
  - --hltpath argument can be either the exact name or matching by unix filename pattern rules: "*,?,[seq],[!seq] ".
  - hltpath seeded by a single L1 bit is supported; with seed expression **X OR Y**, the looser prescale is used; **X AND Y**, the tighter one is used.
  - Other format of L1 seed expression cannot be calculated by the tool.

- **-n** scales the lumi result by a constant. Typical use case is to show average inst lumi.

- **--datatag** (re)produce results with a specific raw data tag

- **--normtag** (re)produce results with a specific norm tag

- **--without-correction** to switch off all the corrections(including calibration) on the raw data.

- **--without-checkforupdate** to switch off the default check for software update. The switch will be removed soon.
- **--nowarning** to switch off the warning messages. Warnings are shown when a LS is selected by the filters but does not exist in the database.

- **-i** input file selection mode and the **"+"** operator
  - multiple files can be given to **-i** option: [0-n] result csv files and [0-1] selection file.
  - result files should be marked with **+** at the end of the name, e.g. "163668-delivered.csv+"
  - the result files must come from the -o output of the same action.

output

- effective luminosity and prescale results
  - meaning of effective luminosity output: luminosity expected in a HLT path taken into account prescale factors and deadtime.
  - meaning of hlt prescale=0: the path is switched off but still in the menu. This is used for paths which should be off at higher luminosity.

    - rule for hltpath seeded by more than one L1bit: With seed expression **X OR Y**, the looser prescale is used; **X AND Y**, the tighter one is used.
    - Other format of L1 seed expression cannot be calculated by the tool.

- **-o** option
    - only screen printout calculates/shows the grand total
    - the output file is normally a file on disk in csv format
    - if the output file name is "stdout", the result is then dumped to stdout. This feature is designed for piping data to further processing by programs.
    - the output file can be used in the **-i** option of the same action
- exit code
    - 11 : undefined 'CMSSW_BASE', 12: unresolved norm/correction, 13: no qualified data
    - get exit code from shell: echo $? ; get exit code from python taking the upper 8 bits from output status: exit_code=status>>8

units

- Time format:
    - in arguments as in output, the format is **"mm/dd/yy HH:MM:SS"**. It is in UTC time
- Lumi units:
    - in screen output, unit is print with the result
    - in csv file output, unit is always 1/ub
- Energy units :
    - single beam energy in **GeV**

examples

- **overview** of delivered/recorded lumi

```
lumiCalc2.py -f 2684 overview

lumiCalc2.py --begin  "04/17/12 05:11:00" --end "06/01/12 00:00:00" overview

lumiCalc2.py overview -i Run2011B_Nov8RerecoJSON_HLTMu40eta2p1_v1.txt

pixelLumiCalc.py -r 65364 overview

pixelLumiCalc.py -f 1795 overview

pixelLumiCalc.py overview -i Run2011B_Nov8RerecoJSON_HLTMu40eta2p1_v1.txt
```

- get LHC **delivered**

```
lumiCalc2.py -f 1718 --begin 162925 --end 162929 delivered

lumiCalc2.py -r 162929 -b stable delivered

lumiCalc2.py -r 162929 -b stable delivered -o 162929.out
```

- **per-LS** lumi

```
lumiCalc2.py -r 165364 lumibyls

pixelLumiCalc.py -f 1795 lumibyls
```

- **effective lumi in hltpath** by run or by ls

```
lumiCalc2.py -r 162924 --hltpath HLT_R032_v1 recorded

lumiCalc2.py -r 162924 --hltpath "HLT_R032_*" recorded

lumiCalc2.py -r 162924 --hltpath HLT_R032_v1 lumibyls

lumiCalc2.py recorded -i Run2011B_Nov8RerecoJSON_HLTMu40eta2p1_v1.txt --hltpath  "HLT_Mu4

pixelLumiCalc.py recorded -i Run2011B_Nov8RerecoJSON_HLTMu40eta2p1_v1.txt --hltpath  "HLT_
```

Note: --hltpath does not work with lumibylsXing

- **average inst** lumi in Hz/ub

```
lumiCalc2.py -r 195265 -b stable lumibyls -n 0.0429
```

  - ***Note***: factor 0.0429 (=1/23.31) is to convert the result to Hz

- **per-bunch** lumi (lumiCalc2.py only action)
  - dump lumi to csv file

```
lumiCalc2.py -r 165472 -b stable  --xingMinLum 5.0e-03 -o 165472-perbunch.csv lumib
```

  - dump lumi to stdout then pipe the data to my own analysis

```
lumiCalc2.py -r 175832 -b stable --xingMinLum 5.0e-03 -o stdout | myanalysis
```

- compare lumi data before and after patch

```
lumiCalc2.py -f 2684 overview --datatag v0

lumiCalc2.py -f 2684 overview --datatag v1
```

- compare lumi results with different corrections

```
lumiCalc2.py -f 2684 overview --datatag v0
```

- reproduce an obsolete lumi result 2 years ago (that is imagine we are in 2014...)

```
lumiCalc2.py -f 2684 overview --normtag HFV2 --datatag v0
```

output explained

Header:

```
**********************************************************************************
* Mon Jun  4 13:39:09 2012 UTC
* lumitype: HF , datatag: v1 , normtag: HFV2 , worktag: V04-00-00
*
* by: /home/CMSSW_5_0_1/bin/slc5_amd64_gcc434/lumiCalc2.py
*
* update: NONE
**********************************************************************************
```

- line1: execution time in UTC time
- line2: lumi type, lumi datatag used to produce this result, normtag used to produce this result, release tag of the script producing the result
- line3: absolute path of the script producing the result

- line4: if there are any update version of the script available.

Note **It is required to always quote the Header when reporting problems or claiming results.**

Note use --headerfile to write the header into file instead of on screen

overview:

```
| Run:Fill | Delivered LS | Delivered | Selected LS |  Recorded |
```

- col1: Runnumber:FillNumber
- col2: n lumi sections, in the run, delivered by LHC or selected by the json filter
- col3: total integrated luminosity, in the run, delivered by LHC or selected by the json filter
- col4: lumi sections recorded by CMS (central daq) or selected by the filters.
- col5: total integrated luminosity, in the run, recorded by CMS or selected by the filters.

```
= Total: | Delivered LS | Selected LS | Delivered | Recorded |
```

- col1: total number of LS delivered by LHC or selected by json filter
- col2: total number of LS selected by CMS trigger or by json filter
- col3: total integrated luminosity delivered by LHC or by json filter
- col4: total integrated luminosity recorded (after deadtime) by CMS or by json filter

lumibyls :

```
| Run:Fill |  LS | UTCTime |  Beam Status | E(GeV) | Del(unit) | Rec(unit) | avgPU |
```

- col1: Runnumber:FillNumber
- col2: lumiLSNumber:CMSLSNumber, CMSLSNumber=0 means cms daq not working during this lumi section.
- col3: UTCTime, lumi section start time
- col4: beam status flag
- col5: single beam energy during this lumi section
- col6: average integrated LHC delivered luminosity during this lumi section. For lumibyls,if there is -i or other filters, only selected LS are printed.
- col7: average integrated CMS recorded luminosity during this lumi section. Only selected LS by -i or other are printed and counted.
- col8: average pileup. The default min bias cross-section for 8ReV pp run is 69300ub.

Note use --minBiasXsec to customize the cross-section in the average pileup calculation. Recommended values:

amodetag	egev	minBiasXsec
PROTPHYS	4000	69300 (default)
PROTPHYS	3500	68000
PAPHYS	4000	2100000

```
= Total: | Delivered LS | Selected LS | Delivered(unit) | Recorded(unit) |
```

- col1: total lumi sections delivered by LHC or selected by the json filter.
- col2: total lumi sections selected and recorded by CMS

- col3: total integrated luminosity delivered by LHC or selected by the json filter
- col4: total integrated luminosity recorded and selected by the json filter

lumibyls --hltpath :

```
| Run:Fill |  LS | HLTpath | L1bit | HLTpresc | L1presc | Recorded(unit) | Effective(unit) |
```

- col1: Runnumber:FillNumber
- col2: lumiLSNumber:CMSLSNumber, CMSLSNumber=0 means cms daq not working during this lumi section.
- col3: selected HLT path
- col4: selected L1 bit
- col5: prescale of selected hlt path
- col6: prescale of selected L1 bit
- col7: average integrated CMS recorded luminosity during this lumi section
- col8: effective integrated CMS recorded luminosity during this lumi section in the selected path, recordedlumi/(hltPrescale*L1Prescale)

```
= Total: | Selected LS | HLTPath | Recorded(unit) |
```

- col1: total lumi sections selected by the filters
- col2: qualified hlt path
- col3: total integrated luminosity recorded and selected by CMS
- col4: total effective integrated luminosity recorded in the hlt path

recorded --hltpath :

```
|  Run:Fill |  SelectedLS | Recorded | HLTpath(Presc) | L1bit(Presc) | Effective |
```

- col1: Runnumber:FillNumber
- col2: lumi sections seen by CMS or selected with -i and all other ls filters.
- col3: CMS recorded luminosity without prescale, in lumi sections selected by -i and other ls filters.
- col4: selected hlt path name (hlt prescale values, multiple if dynamic prescaling)
- col5: L1 seed bit name (L1 bit prescales)
- col6: recordedlumi/(hltPrescalexL1Prescale)

```
= Total: |  HLTPath | SelectedLS | Recorded | Effective |
```

- col1: qualified hltpath
- col2: total lumi sections seen by CMS or selected with -i and other filters
- col3: total integrated luminosity recorded and selected by CMS
- col4: total effective integrated luminosity recorded and selected in the hlt path

delivered:

```
| Run:Fill | N_LS | N_CMSLS | Delivered | UTCTime | E(GeV) |
```

- col1: Runnumber:FillNumber
- col2: n lumi sections delivered by LHC or selected by json filter in this run.
- col3: n lumi sections recorded by CMS and/or other ls filters in this run.

- col4: n integrated luminosity delivered by LHC or by the json filter LS in this run
- col5: run start UTC time. It is not affected by the LS filters
- col6: average single beam energy during stable beams

```
= Total:| Delivered LS | Total CMS LS | Delivered(unit) |
```

- col1: total lumi sections delivered by LHC or selected by json filter
- col2: total lumi sections recorded by CMS (central daq) and/or selected by the command filters
- col3: total integrated luminosity delivered by LHC or selected by other filters before CMS trigger deadtime

Note: Larger number of of delivered lumi sections shows the effect of lumi daq continuous mode.

Note: this action requires no trigger information, therefore is the fastest command for luminosity calculation.

lumibylsXing: No screen printout, -o outputfile is required

- file_field1: Runnumber:FillNumber
- file_field2: LS number
- file_field3: LS start UTC Time
- file_field4: integrated luminosity(/ub) delivered in the LS.
- file_field5: integrated luminosity(/ub) recorded inthe LS.
- file_field6: bunch index, **inst lumi(Hz/ub)** pair for all the selected bunches.

Note: the default algorithm for BX lumi is OCC1, to change the default use option --xingAlgo (OCC1,OCC2,ET).

Note: the default filter for BX lumi is 0.001, to change the default use option --xingMinLum

Note: Since the --lumibylsXing option includes also the luminosity observed in nominally inactive bunch crossings, one should not expect the sum over all crossings to add up to the reported luminosity. Also: please be careful using the information from these inactive crossings, since the afterglow correction applied is not strictly speaking fully correct for the inactive crossings.

lumiNorm.py

Features

- **lumiNorm.py** describes and manage the lumi corrections. only **list** action is usable by general users, other actions are for tag manipulations by lumi group.

actions

- **list**

examples

- list all tag names

```
lumiNorm.py list
```

- inspect correction values by normtag name

```
lumiNorm.py list --name HFV2
```

- inspect current default normtag for given lumitype

```
lumiNorm.py list --lumitype HF
```

output explained

header :

```
 = | Name | Type | IsTypeDefault | Comment | CreationTime |
```

- col1: norm tag name
- col2: lumi type. Can be HF or PIXEL
- col3: if this tag is the default of this lumi type
- col4: comment on this correction set, if any
- col5: tag creation time

norm tag detail:

```
| Since  | Func  | Parameters | amodetag | egev | comment |
```

- col1: since which run the correction are applied
- col2: correction function short name. This is a real function and it is called as is.
- col3: correction parameters
- col4: amode tag of the runs. This is a description, it has no real effect on the correction application.
- col5: beam energy of the runs. This is a description, it has no real effect on the correction application.
- col6: further description.

lumiTag.py

- **lumiTag.py** describes and manage the lumi raw data tags. The tags can be used as --datatag input to calculation scripts. Only **list** action is usable by general users, other actions are for tag manipulations by lumi group.
- there is always an open current default data tag.

actions

- **list** describes the lumi raw data tag. --lumitype option is required, no default assumed

examples

```
lumiTag.py list --lumitype HF
lumiTag.py list --lumitype PIXEL
lumiTag.py list --lumitype HF --name v0
```

output explained

list:

```
| Name | Min Run | Max Run | Creation Time |
```

- col1: tag name
- col2: minimum run included in the tag
- col3: max run included in the tag. If max run is "open", it's the current open tag.
- col4: tag creation time

list --name:

```
| Run | Data Id | Insertion Time | Patch Comment |
```

- col1: runnumber
- col2: data id. lumiid-trgid-hltid
- col3: data insertion time
- col4: comment for patch reload/postload. Normally inserted data do not carry comment.

lumiContext.py

- shows run,trigger,hlt and beam conditions for the lumi calculation.
- **actions** : trgbyls , hltmenu, hltbyls,beambyls,runsummary

trgbyls

- list L1 trigger deadtime fraction, selected trigger bit count and prescale per lumi section.
- Trigger bit can be selected by exactly name or a pattern rule.
- If no --name specified, will list all bits.

trgbyls examples

```
lumiContext.py trgbyls -r 195757
lumiContext.py trgbyls -r 195757 -i Jet_Run2012A_PromptReco_v1_AOD.json --name "L1_ZeroBias
lumiContext.py trgbyls -r 195757 -i Jet_Run2012A_PromptReco_v1_AOD.json --name "L1_ZeroBias
```

trgbyls output explained

```
| Run | LS | dfrac | (bitname,count,presc) |
```

- col1: runnumber
- col2: lumi section number
- col3: deadtime fraction
- col4: selected trg bit name,count and prescale

hltmenu

- list hlt menu content including the l1 seed expression and bit name of selected run
- Hlt path can be selected by the exact name or a pattern rule.
- If no --name specified, all paths are listed

hltmenu examples

```
lumiContext.py hltmenu -r 195757
lumiContext.py hltmenu  --name "HLT_HT650_v3"
lumiContext.py hltmeny --name "HLT_HT*"
```

hltmenu output explained

- `| Run | hltpath | l1seedexpr | l1bit |`
    - col1: runnumber
    - col2: hlt path
    - col3: corresponding seed expression
    - col4: L1 bit considered as the seed (used by lumi calculation)

hltbyls

- list prescale,l1pass,hltaccept info of selected hltpath
- Hlt path can be selected by the exact name or a pattern rule.
- option --name is required

hltbyls examples

```
lumiContext.py hltbyls  --name "HLT_HT650_v3"
lumiContext.py hltmeny --name "HLT_HT*"
```

hltbyls output explained

```
| Run | LS | (hltpath,presc,l1pass,hltaccept) |
```

- col1: runnumber
- col2: ls number
- col3: selected hltpathname, prescale,l1passed trigger count,hlt accepted count

beambyls

- list beamstatus,beamenergy,ncolliding bunches of selected run/ls
- dump beamintensities file if --with-beamintensity --minintensity are specified

beambyls examples

```
lumiContext.py beambyls  -r 196531
lumiContext.py beambyls -r 196531 --with-beamintensity --minintensity 1.0e11 -o 196531_beam
```

beambyls output explained

```
| Run | LS | beamstatus | egev | ncollidingbx |
```

- col1: runnumber
- col2: ls number
- col3: beam status

- col4: single beam energy in GeV
- col5: number of colliding bunches of the run
- col6 (only in output file and only if beamintensity is asked for) : bunch index(0-3563),beam1 intensity,beam2 intensity

runsummary in plain printout without table view.

- list **Run** , **Fill** , **Amodetag** , **egev** , **runstarttime** , **runstoptime** , **L1Key** , **HLTkey** of the selected run

Luminosity Objects in EDM and lumi correction module

- `LumiSummary`, `LumiDetails` are standard EDProduct to be used in your analyzer and fetched by label **"lumiProducer"** from edm::LuminosityBlock object
- `LumiCorrectionSource` is an ESSource module that can be loaded at run time by your analyzer to correct/calibrate the raw lumi values from `LumiSummary`, `LumiDetails`

Note: LumiSummary objects produced by **CMSSW_5_2_X and older** are already corrected. The calibration factor used is 6.37 for unit Hz/ub, pp7TeV runs.

`LumiSummary`

- uncalibrated/uncorrected average luminosity in the lumi section.
- details of the class see inline comments in the header file: **DataFormats/Luminosity/interface/LumiSummary.h**
- injected into `RECO EDM` data since `RECO` and NOT before. Not present in RAW data.
- in `BuildFile.xml` add

```
<use name="DataFormats/Luminosity"/>
```

- code example:

```
#include "DataFormats/Luminosity/interface/LumiSummary.h"
...skip...
void YOURANALYZER::beginLuminosityBlock(const edm::LuminosityBlock&lumi, cor
      edm::Handle<LumiSummary> l;
      lumi.getByLabel("lumiProducer", l);
      // Check that there is something
      if (!l.isValid())  return;
      //use l
      float myavginstlumi=l->avgInsDelLumi();
      ...skip...
}
```

- comparison with script command
  - `LumiSummary::intgDelLumi()`

    should match

    ```
    lumiCalc2.py -i runls.json --without-correction lumibyls
    ```
- a comment on CMSSW version (**temporary**)

Above all, make sure the cmssw major version used to produce and analyze the edm data match. There is a data format change since `CMSSW_5X`.

When using CMSSW_4X, there is a bug in LS length calculation causing lumisummary::intgDelLumi() to appear exactly 10 times larger. One can either correct this by hand or use a patched version of DataFormats/Luminosity

```
cvs co -r V03-05-00 DataFormats/Luminosity
cd DataFormats/Luminosity
scram b; cmsenv
```

The problem is not in CMSSW5X.

LumiDetails

- luminosity per bunch crossing per lumi section in Hz/ub; beam 1 , beam 2 intensity
- details of the class see inline comments in the header file:
  DataFormats/Luminosity/interface/LumiDetails.h
- per-bunch luminosity is NOT corrected in any cmssw version
- injected into RECO EDM data since RECO and NOT before. Not present in RAW data.
- not present in AOD data because of its size. RECO is the best place to get it.
- in BuildFile.xml add

```
<use name="DataFormats/Luminosity"/>
```

- code example :

```
#include "DataFormats/Luminosity/interface/LumiDetails.h
...skip...
void YOURANALYZER::analyze(edm::Event const& evt, edm::EventSetup const& es)
      edm::Handle<LumiDetails> d;
      evt.getLuminosityBlock().getByLabel("lumiProducer",d);
      // Check that there is something
      if (!d.isValid())  return;
      //use d
      float myrawbxlumi=d->lumiValue(LumiDetails::kOCC1,evt.bunchCrossing())
      ...
}
```

A different style: if you are interested in seeing all the bunches.

```
float i_bx_B1_now[3564];
float i_bx_B2_now[3564];
float  i_bx_LUMI_now[3564];
for (int i=0;i<3564;++i)  {\\ Loop on bunch crossings
       i_bx_B1_now[i]=d->lumiBeam1Intensity(i);
       i_bx_B2_now[i]=d->lumiBeam2Intensity(i);
       i_bx_LUMI_now[i]=d->lumiValue(LumiDetails::kOCC1,i);
        //calibrated here but not corrected, in Hz/ub
 }
```

- comparison with script command
  - LumiDetails::lumiValue(LumiDetails::kOCC1,bunchidx)

    should match

    ```
    lumiCalc2.py -i runls.json --without-correction lumibylsXing  -xingAlgo OCC1
    ```
  - LumiDetails::lumiBeam1Intensity, lumiBeam2Intensity

    should match

    ```
    lumiContext.py --with-beamintensity -i runls.json beambyls -o stdout -minintensity
    ```

NEW All lumi detail data produced by CMSSW_5_2_X and higher are uncorrected, for release version CMSSW_6_1_0_pre3 and higher, please use LumiCorrectionSource for calibration/correciton..

instruction for LumiCorrectionSource

- `LumiCorrectionSource` is an ESSource module that can be loaded at run time by your analyzer to correct/calibrate the raw lumi values from `LumiSummary`, `LumiDetails`
- `LumiCorrectionSource` delivers `LumiCorrectionParam` object valid for the current run. `LumiCorrectionParam` contains the final lumi correction factor, as well as all the ingredients used to arrive at the factor, e.g. ncollidingbunches,corrFunc,afterglow corrections,correction coefficients.
- It is possible to use a specific normtag (optional) with LumiCorrectionSource. By default, if not specified, the current default lumi normtag is automatically used, and one can find out the tag used with method `LumiCorrectionParam::normtag()`
- There is also a configurable **datatag** parameter (optional) in LumiCorrectionSource, but it does not mean the actual lumi raw data version in edm here. It refers only to the version of the lumi data where ncollidingbunches is fetched from, which, in almost all the cases, is identical across different versions of lumi raw data. As we know the actual lumi raw data version in edm is known only to edm.
- The module is first released with `CMSSW_6_1_0_pre3`, who uses this version and higher can skip the following checkout/setup instruction. setup (where the exact CMSSW version is indicative)

```
scram p CMSSW CMSSW_5_3_0
cd CMSSW_5_3_0/src
cvs co -r V06-05-03 RecoLuminosity/LumiProducer
scram b
cmsenv
```

Note: for who uses also the LumiProducer in the same working area (i.e. older than CMSSW_6_1_0_pre3) to inject luminosity to his/her own data, extra checkout/compilation of package DataFormats/Luminosity is required:

```
cvs co -r V06-01-03 DataFormats/Luminosity
```

- example `EDAnalyzer`, `test/TestLumiCorrectionSource.cc`

```
edm::Handle<LumiSummary> lumisummary;   //get the raw lumi data from edm::LumiSumm
lumiBlock.getByLabel("lumiProducer", lumisummary);
float instlumi=lumisummary->avgInsDelLumi();
float correctedinstlumi=instlumi;
float recinstlumi=lumisummary->avgInsRecLumi();
float corrfac=1.;
edm::ESHandle<LumiCorrectionParam> datahandle; //get LumiCorrectionParam object f
es.getData(datahandle);
if(datahandle.isValid()){
  const LumiCorrectionParam* mydata=datahandle.product();
  std::cout<<"correctionparams "<<*mydata<<std::endl;
  corrfac=mydata->getCorrection(instlumi);           //get lumi correction factor
}else{
  std::cout<<"no valid record found"<<std::endl;
}
correctedinstlumi=instlumi*corrfac;                      //final lumi value=corre
float correctedinstRecLumi=recinstlumi*corrfac;  //Note: the correction factor is
```

Note: in order to correct the perbunch lumi(LumiDetails), and recorded lumi (LumiSummary), the input argument to `getCorrection()` method should always be `avgInsDelLumi` result from LumiSummary, i.e. the average inst lumi of the lumi section. In anotherword, the correction factor is the same for

lumiSummary and lumiDetails (delivered or recorded) for the same lumi section.

- example configuration `test/testLumiCorrectionSource.py`

```
process.source= cms.Source("PoolSource",
      processingMode=cms.untracked.string('RunsAndLumis'),
      fileNames=cms.untracked.vstring('file:/data/cmsdata/200786/FC7E661B-C3E8-E11
)
process.LumiCorrectionSource=cms.ESSource("LumiCorrectionSource",
      connect=cms.string('frontier://LumiCalc/CMS_LUMI_PROD'),
      normtag=cms.untracked.string('HFV2a')      //optional
)
```

Note: this configuration works fine standalone, but there may (or may not) be conflicts if you have another database module in the same configuration, e.g. global tag etc. It is currently being debugged.

Sandbox

- other links
  - ♦ checklumi.log
  - ♦ lumi offline HF,vertex analysis
  - ♦ LumiDBOperation
  - ♦ LumiCalcBetaTmp (obsolete)

-- Main.ZhenXie - 21-Apr-2010

This topic: CMS > LumiCalc
Topic revision: r188 - 2016-02-29 - ZhenXie