Table of Contents

How to find out the value of $\mu$.......................................................................................................................1

1. From the run database......................................................................................................................................1
2. For a given run, using PVSS. The longwinded way. NB this section is out of date and doesn't work anymore. It would require some work to update it.................................................................1
3. In real time, running a job in the monitoring farm....................................................................................4
4. From routing bits...............................................................................................................................................5
How to find out the value of mu

For a definition of mu, see here.

1. From the run database

   1. Using the link from the LHCb home page to the web interface of the run db, select a run. Click on the circled i just in front of the runnumber. It will give you the average value of mu for that run and more.

2. For a given run, using PVSS. The longwinded way. NB this section is out of date and doesn't work anymore. It would require some work to update it.

   1. You need an account on the online cluster.
   2. For a given run, look up the runnumber from the run database. Note the fillnumber for this run and also note the start/end times of this run.
   3. From Windows, start a terminal server session to lbts01, then another one to ui01w.
   4. From Linux, do an ssh to lbgw01, then ssh to the "ui" machine. (This is a machine with PVSS where you can run PVSS user interfaces without disturbing the ECS).
   5. Open the shortcut LHCCOM_UI_RunPlan in G:\online\ecs\Shortcuts38\LHC\LHCCOM (Windows) or /group/online/ecs/ShortCuts38/LHC/LHCCOM/LHCCOM_UI_RunPlan.sh (Linux)
6. Click on “Summary per fill number” and wait.

7. Click on the Fill number and wait.

2. For a given run, using PVSS. The longwinded way. NB this section is out of date and doesn’t work anymore.
4 windows will pop up. Close all, except the "Luminosity Page". The orange curve shows the value of mu as a function of time. The period with stable beams is delimited by the vertical red lines. Move your cursor on this curve to the time of the run you are interested in and right click. The small yellow window will pop up containing the value of mu at that time. Mu is calculated using the instantaneous luminosity and the number of bunches.

2. For a given run, using PVSS. The longwinded way. NB this section is out of date and doesn't work anymore.
3. In real time, running a job in the monitoring farm

The value of \( \mu \) is also published as a DimService, using the DIM DNS tcf002. You can subscribe to it in your C++ program:

```cpp
DimClient::setDnsNode("tcf002");
std::string serviceName="Mu";
m_MuSvc = new DimInfoMu(serviceName.c_str());
m_Mu=m_MuSvc->getMu();
```

where DimInfoMu is a DimInfo C++ client:

```cpp
#include "DimInfoMu.h"
#include "RTL/rtl.h"
#include <iostream>
#include <string>

DimInfoMu::DimInfoMu(std::string MuSvcName) :
DimInfo((MuSvcName).c_str(),-1) {
  m_hasData=false;
  m_Mu=0;
  m_data=0;
}
```

3. In real time, running a job in the monitoring farm
void DimInfoMu::infoHandler()
{
    m_data = getFloat();
    if (m_data !=0) {
        if ((m_data!=m_Mu)&&(m_data!=-1)) m_Mu=m_data;
    }
    if ((m_data!=0)&&(m_data!=-1)) setMu();
}

int DimInfoMu::getMu()
{
    int ntries =0;
    if (m_data==0) m_data=m_Mu;
    while (ntries<25) {
        if (m_hasData==true) {
            return m_data;
        }
        else {
            lib_rtl_sleep(1000);
            ntries++;
        }
    }
    return 999;
}

void DimInfoMu::setMu(){
    m_hasData=true;
}

The header file:

#include "dic.hxx"
#include <string>

class DimInfoMu : public DimInfo {
public :
    DimInfoMu(std::string MuSvcname);
    virtual ~DimInfoMu();
    int getMu();

private:
    void infoHandler();
    float m_Mu;
    float m_Mu;
    bool m_hasData; // true after histo has been filled first time
    void setMu();
};
4. From routing bits