Gaucho v2r0 Installation and User's Guide

LHCb Technical Note

Reference: LHCb-2004-096
Author: Eric van Herwijnen
Version: v2r0
Date: 29 march 2005

http://lhcb-comp.web.cern.ch/lhcb-comp/ECS/Gaucho/default.htm

Please send any comments, bugs, suggestions or requests for help to the author (tel. 022-7679887, or eric.van.herwijnen@cern.ch).

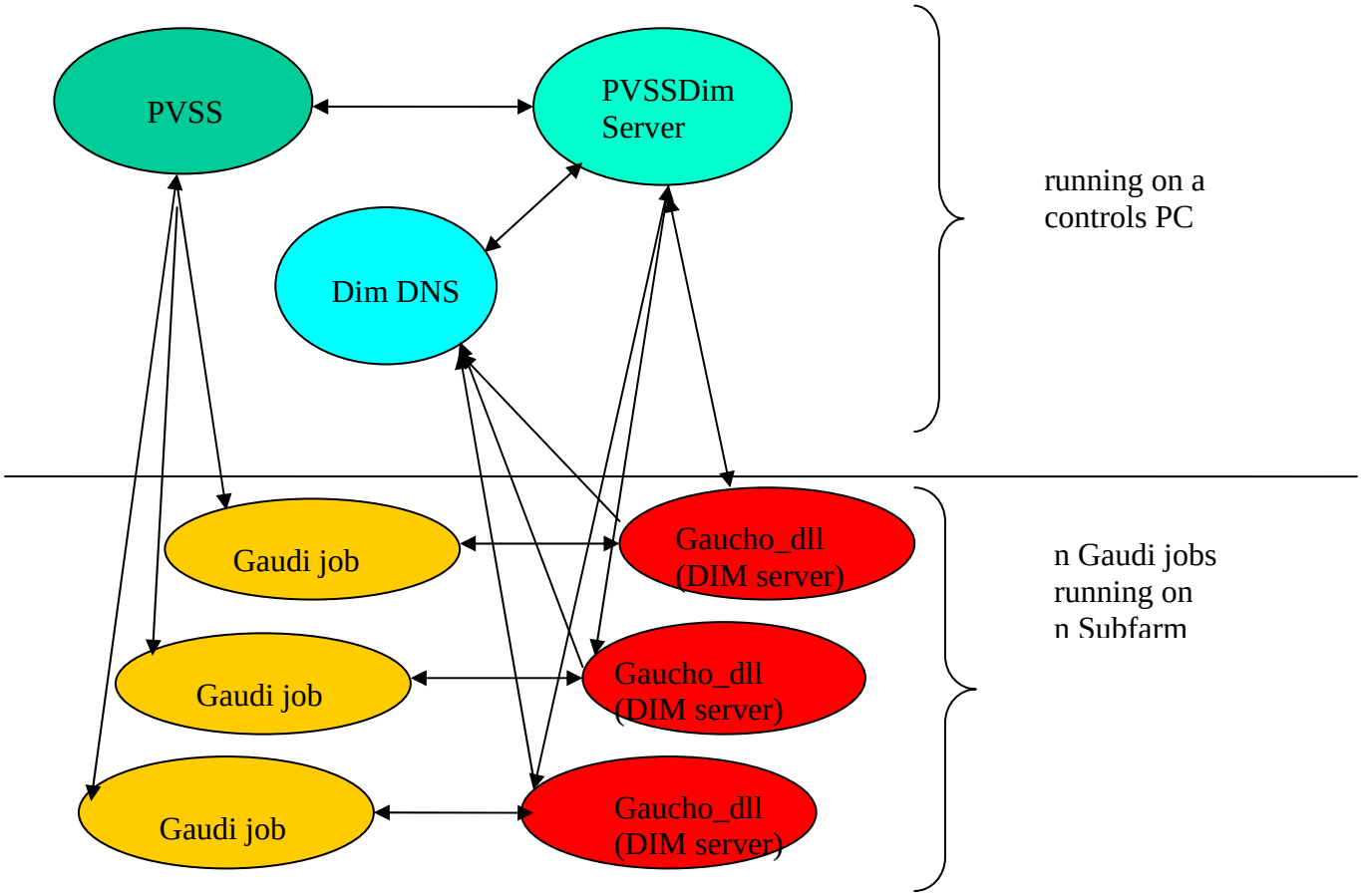
Introduction

Gaucho (GAUdi Component Helping Online) was designed and developed by Philippe Vannerem who has now left CERN. The Gaucho architecture is described in http://lhcb-doc.web.cern.ch/lhcb-doc/presentations/conferencetalks/postscript/2003presentations/VanneremCorrea.ppt

This document describes:

1. How to modify your Gaudi code to publish counters and histograms so they can be viewed in real time using PVSS (page 3).
2. How to install PVSS and the LHCb framework containing the Gaucho component (page 9).
3. How to submit a job from the Gaucho PVSS interface and monitor the published information from PVSS (page 17).

The Gaucho components are shown in the Figure below.



Installing Gaudi prerequisites

From Gaudi release 15r3 and GaudiKernel v18r0 on, no Gaudi prerequisites need to be installed for Gaucho v2r0. Gaucho v2r0 is backwards incompatible with previous versions.

1. Dim (current version v15), obtain by getpack Online/DIM v15r0
2. Gaucho (current version v3r0), obtain by getpack Online/Gaucho head
3. GauchoJob (current version v2r0), obtain by getpack Online/GauchoJob head

To compile Gaucho (and GauchoJob), you do as usual:

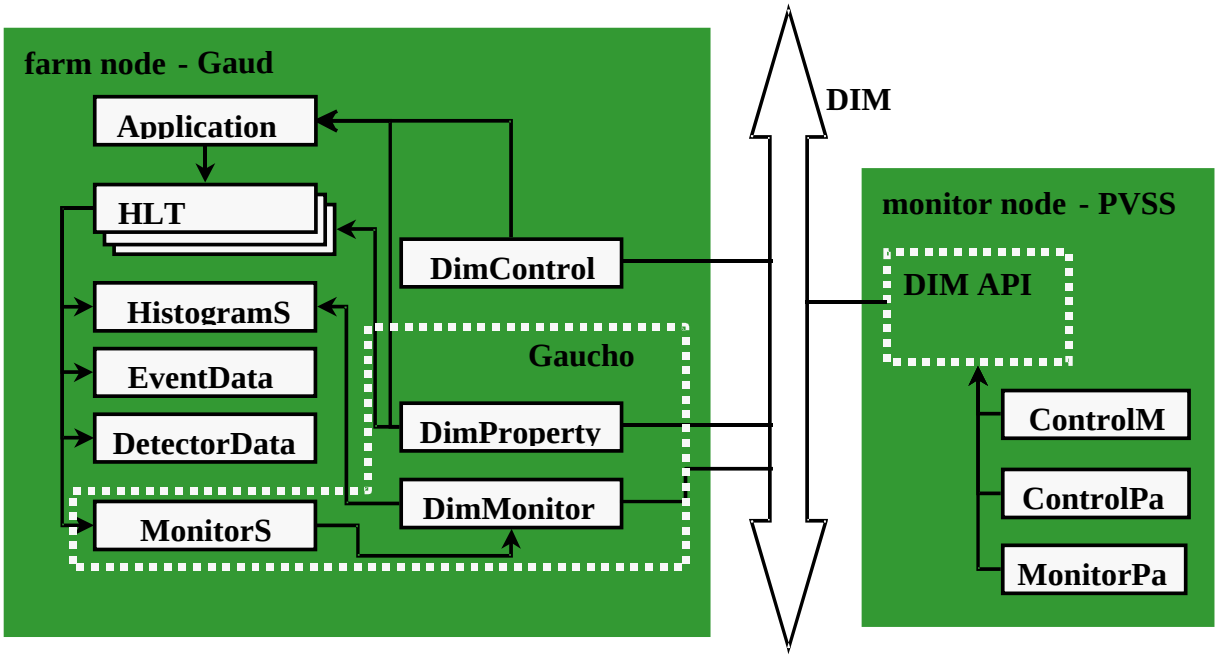
1. cd cmt
2. source $LHCBHOME/scripts/CMT.csh
3. GaudiEnv v15r3
4. source setup.csh
5. type make

The GauchoJob package contains a modified standard example of a Gaudi Job that publishes histograms and counters. The following sections explain how these files work.

To test your installation, run the script GauchoJob/v2r0/cmt/startgaudijob.sh.

Modifying your Gaudi job

The Gaudi services communicate with DIM and PVSS as follows:



The GaudiMain program (Linux) should contain a pointer to **the SvcLocator** interface (6,24), and should create an instance of the **GaudiDimController** (17) and run it (37), as shown in the figure below for a simple GaudiMain program. The Application Manager is steered by the GaudiDimController program.

```
1: #include "GaudiKernel/SmartIF.h"
2: #include "GaudiKernel/Bootstrap.h"
3: #include "GaudiKernel/IAppMgrUI.h"
4: #include "GaudiKernel/ISvcLocator.h"
5: #include "GaudiKernel/IProperty.h"
6: #include "GaudiDimController.h"
7:
8: //--- Example main program
9: int main ( int argc, char* argv[] ) {
10:  char* nname;
11:  nname=new char[60];
12:
13: // Get the input configuration file from arguments
14: std::string opts = (argc>1) ? argv[1] "../options/JobOptions.opts";
15: #ifdef WIN32
16:  int errcode;
17:  errcode=wins::gethostname(nname,60);
18: #else
19:  gethostname(nname,60);
20:  char *pch;
21:  pch = strtok(nname,".");
22: #endif
23:
24:  char * pid=new char[10];
25:  sprintf(pid,"%d",getpid() );
26:  strcat(nname,"/");
27:  strcat(nname,pid);
28:
29:  // create an instance of Dim Control Manager for host and process ID
30:  GaudiDimController* gaudimctrl = new GaudiDimController(nname);
31:
32:  // Create an instance of an application manager
33:  IInterface* iface = Gaudi::createApplicationMgr();
34:  SmartIF<IProperty>     propMgr ( IID_IProperty, iface );
35:  SmartIF<IAppMgrUI>     appMgr  ( IID_IAppMgrUI, iface );
36:  SmartIF<ISvcLocator>   svcLctr ( IID_ISvcLocator, iface );
37:
38:  if( !appMgr.isValid() || !propMgr.isValid() ) {
39:  std::cout << "Fatal error while creating ApplicationMgr " << std::endl;
40:    return 1;
41:  }
42:
43:  propMgr->setProperty( "JobOptionsPath", opts );
44:
45://pass serviceLocator?
46:  gaudimctrl->run(svcLctr);
47:
48:// All done - exit
49:  return 0;
50:}
```

Notice that on SLC3, the gethostname command returns the domain name and country code in addition to the hostname, whereas only the hostname is required (lines 16-22). The process id is used to give the GaudiDimController a unique name, thus allowing multiple Gaucho jobs to be run on the same host.

The Job Options file

An example of a job options file for a Gaucho job is shown below. The quantities which are published in this example are calculated in the HelloWorld and PartentAlg algorithms.

```
1: // services needed by a standard job
2: ApplicationMgr.ExtSvc = { "EventSelector" };
3: ApplicationMgr.ExtSvc += { "MonitorSvc" };
4:
5: // DLLs used by a standard job
6: // must not be used by a statically linked program
7: ApplicationMgr.DLLs = { "GaudiAlg", "GaudiAud", "Gaucho" };
8:
9: AuditorSvc.Auditors = { "ChronoAuditor"};
10:
11: // Private Application Configuration options
12: ApplicationMgr.TopAlg = { "Sequencer/TopSequence" ,"ParentAlg" };
13: ApplicationMgr.TopAlg  = { "ParentAlg" };
14: // Set output level threshold
15: (2=DEBUG,3=INFO,4=WARNING,5=ERROR,6=FATAL)
16: MessageSvc.OutputLevel = 3;
17:  ApplicationMgr.OutputLevel  = 2;
18: // Event related parameters
19: ApplicationMgr.EvtMax = -1; // events to be processed
20: ApplicationMgr.EvtSel = "NONE"; //do not use any input events
21:
22: // Algorithms Private Options
23:// Setup the next level sequencers and their members
24: TopSequence.Members = {"Sequencer/Sequence1","Sequencer/Sequence2"
};
25: TopSequence.StopOverride = true;
26: Sequence1.Members = {"Prescaler/Prescaler1", "HelloWorld",
"EventCounter/Counter1" };
27: Sequence2.Members = {"Prescaler/Prescaler2", "HelloWorld",
"EventCounter/Counter2" };
28:
29: HelloWorld.OutputLevel = 2;
30: Prescaler1.PercentPass = 50.;
31: Prescaler2.PercentPass = 10.;
32: Prescaler1.OutputLevel = 4;
33: Prescaler2.OutputLevel = 4;
34: MonitorSvc.OutputLevel = 2;
35: DimEngine.OutputLevel = 2;
36: ParentAlg.OutputLevel = 2;
```

The GaudiDimController program

The GaudiDimController program takes as argument the hostname where the program is running, and starts a DimServer called "HLT"+hostname+"/"+pid.
The GaudiDimController allows the Gaudi application to be steered via commands (**config**, **start**, **pause** and **stop**) that it receives from DIM (via a DimCommand called with the name=hostname). It publishes the state of the program (**configured**, **processing**, **paused** or **stopped**) as a DimService called hostname+"/"+pid+"/**status**". After configuring the Application Manager, the GaudiDimController sets itself to sleep until the next command arrives. The GaudiDimController tells the Application Manager to execute the eventloop when it receives the command "**start**".

If you are happy with this behaviour, you will not need to modify the GaudiDimController program.

Algorithms

The HelloWorld algorithm is a standard algorithm, in the case of our example it doesn't do anything. In our example, the work is done in the ParentAlg, as shown in the Figure below. We show an extract of this file to highlight the important points:

- the declaration of the information to be published (counter1, fraction, status and eventtype) during initialization via the method: **m_publishsvc->declareInfo(const std::string& name, const type& variable, const std::string& desc, const IInterface* owner)**
- (type can be bool, char, int, long, double, string, or AIDA::IHistogram)
- the calculation of the variables during execution
- the undeclaration of the information during finalization via the method: **m_publishsvc->undeclareInfo(const std::string& name, const IInterface* owner)**

```
// Include files
#include "GaudiKernel/MsgStream.h"
#include "GaudiKernel/AlgFactory.h"
#include "GaudiKernel/DataObject.h"
#include "GaudiKernel/IDataProviderSvc.h"
#include "ParentAlg.h"
# define mysleep() usleep(100000)

// Static Factory declaration
static const AlgFactory<ParentAlg>  Factory;
const IAlgFactory& ParentAlgFactory = Factory;
// Constructor
ParentAlg::ParentAlg(const std::string& name, ISvcLocator* ploc)
: Algorithm(name, ploc), m_publishsvc() {
  m_publishsvc = 0;
}

StatusCode ParentAlg::initialize() {
  MsgStream log(msgSvc(), name());
  StatusCode sc;
  sc = service("HistogramDataSvc", m_histosvc, true );
  sc = serviceLocator()->service("MonitorSvc", m_publishsvc, true );
```

```
  if( !sc.isSuccess() )   {
    log << MSG::FATAL << "Unable to locate IPublish interface" << endreq;
    return sc;
  }

  counter1=0;
  frac1=0.0;
  status=new char[20];
  status=strcpy(status,"initializing");
  myhisto = m_histosvc->book("1", "eventtype", 5, 0.5, 5.5 );
  m_publishsvc->declareInfo("counter1",counter1,"All events",this);
  m_publishsvc->declareInfo("fraction",frac1,"Ratio 2:1",this);
  m_publishsvc->declareInfo("status",status,"Trigger status",this);
  m_publishsvc->declareInfo("eventtype",myhisto,"Event type",this);
  time(&time_old);

  // use Random Number Service to generate trigger events
  sc = random.initialize(randSvc(), Rndm::Flat(0.,1.));
  if ( !sc.isSuccess() ) {
    return sc;
  }
  return StatusCode::SUCCESS;
}

StatusCode ParentAlg::execute() {
MsgStream        log( msgSvc(), name() );
StatusCode sc;
std::vector<Algorithm*>::const_iterator it  = subAlgorithms()begin();
std::vector<Algorithm*>::const_iterator end = subAlgorithms()->end();
  for ( ; it != end; it++) {
    sc = (*it)->execute();
    if( sc.isFailure() ) {
      log << "Error executing Sub-Algorithm" << (*it)->name() << endreq;
    }
  }

  counter1++;
  // use Random Number Service to get generate trigger events
  float dice1=random();
  float dice2=random();
  float tfdice;
  float bincons[80];
  int binnr,i;
//eventtype histo
  if (dice1<0.5) {
    myhisto->fill(1.0);
  }
  else if(dice1<0.95){
    myhisto->fill(2.0);
  }
  else if(dice1<1.0){
    myhisto->fill(3.0);
  }
  if (counter1 % 50 == 0) {
        status=strcpy(status,"trigger1");
  }
  if (counter1 % 100 == 0) {
```

```
        status=strcpy(status,"trigger2");
  }
  if (counter1 % 50 == 0) {
        time(&time_new);
    frac1=counter1/(time_new-time_old);
  }

  int dumint;
  float dumfloat;
  char* dumstring;
  dumstring=new char[50];
//  delay
  mysleep();
  return StatusCode::SUCCESS;
}

StatusCode ParentAlg::finalize() {
  MsgStream log(msgSvc(), name());
  m_publishsvc->undeclareInfo("counter1",this);
  m_publishsvc->undeclareInfo("fraction1",this);
  m_publishsvc->undeclareInfo("status",this);
  m_publishsvc->undeclareInfo("eventtype",this);

//  m_publishsvc->undeclareAll( this );

  log << MSG::INFO << "finalized successfully" << endreq;
  return StatusCode::SUCCESS;
}
```

The MonitorSvc

The MonitorSvc service is to instantiate various objects (the DimEngine and the DimPropServer). It implements methods for declaring integers, floats, strings and histograms as DIM services. You should not have to modify this code.

The DimPropServer

The DimPropServer implements the rpcHandler method of the DimRpc class.
It allows the configuration of algorithms through RPC commands. This code should not have to be modified. It allows inspection of the algorithm tree, and reading and setting of an algorithm's properties. The initialize method of an algorithm is called when its properties are changed. This is not guaranteed to work, as the initialize method often can not be called without errors after the algorithm has started.
The DimPropServer can also be used to browse the transient histogram store.

The DimCmdServer

The DimCmdServer allows dynamic publication of histograms found on the transient store. This means the user does not have to call the declareInfo method.

Installing PVSS prerequisites

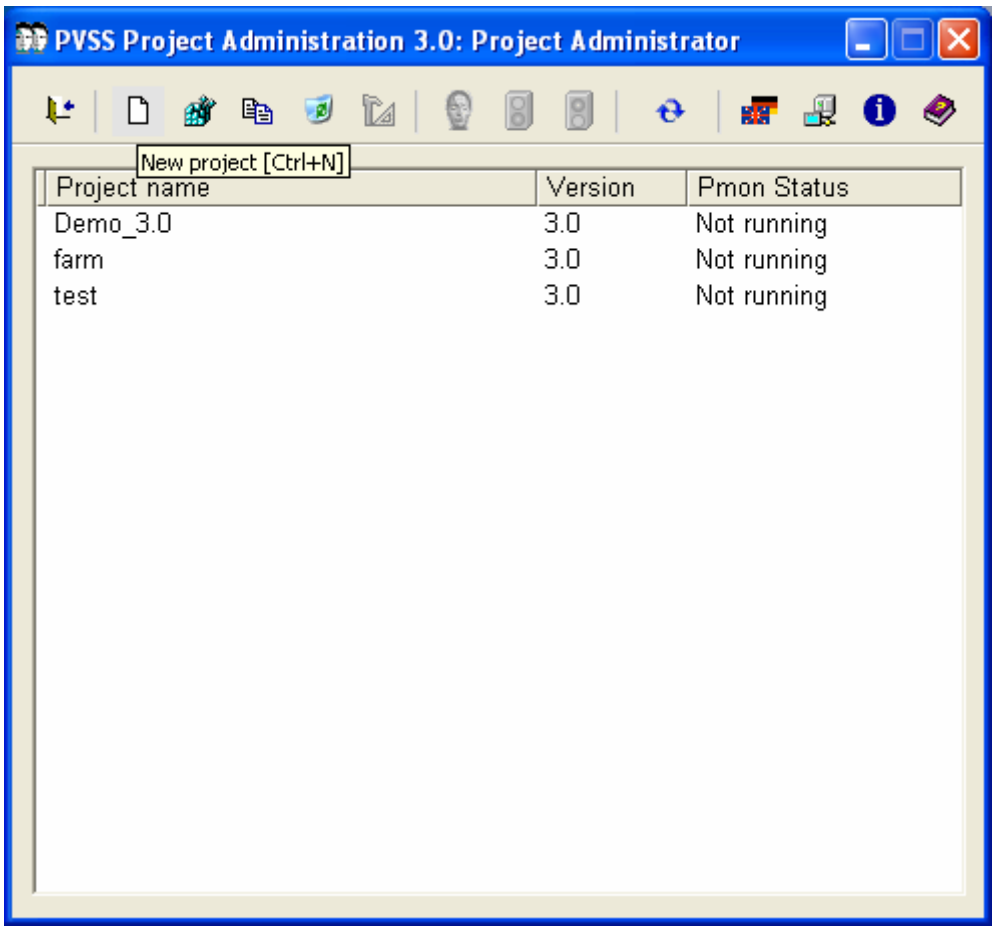
To configure the correct PVSS environment you need to carry out a number of steps in the order given in this chapter. These instructions assume that you will be installing PVSS on Windows; for Linux the procedure should be similar.

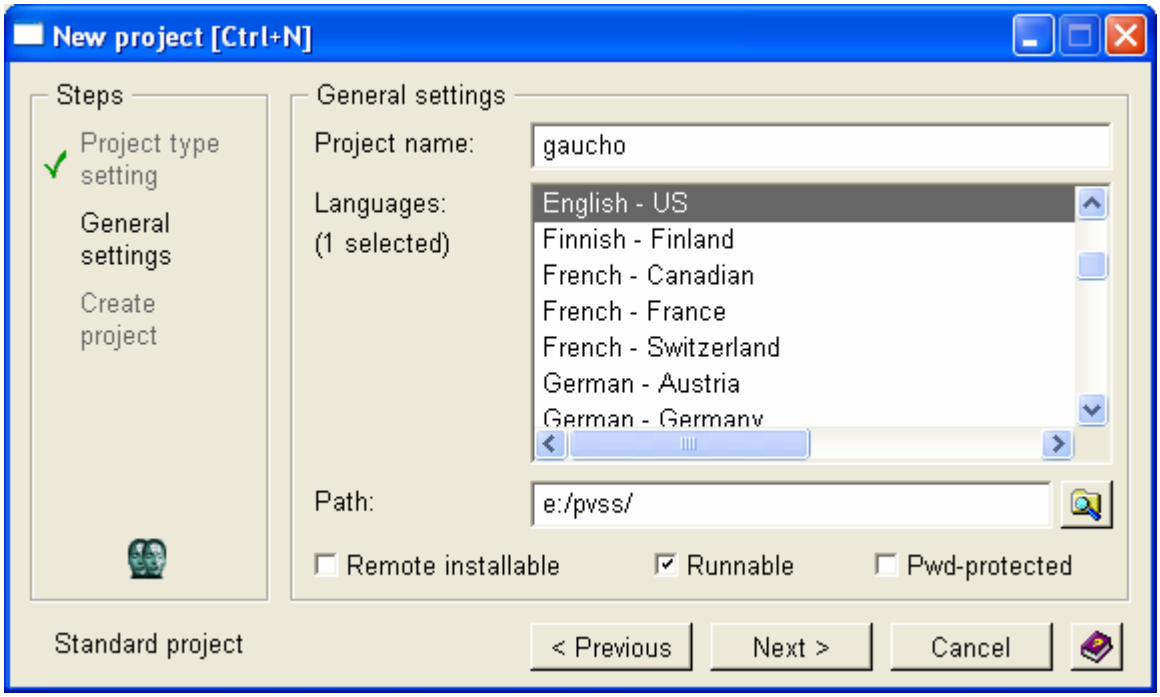
Installing PVSS 3.0

First you need to install PVSS 3.0 (see the instructions on http://itcobe.web.cern.ch/itcobe/Services/Pvss/) , and create a new project in the usual way. I called my project "**gaucho**". If you already used PVSS 2.12.2, make sure your project name is not the same as one of a previously existing PVSS 2.12.2 project (bug in PVSS).

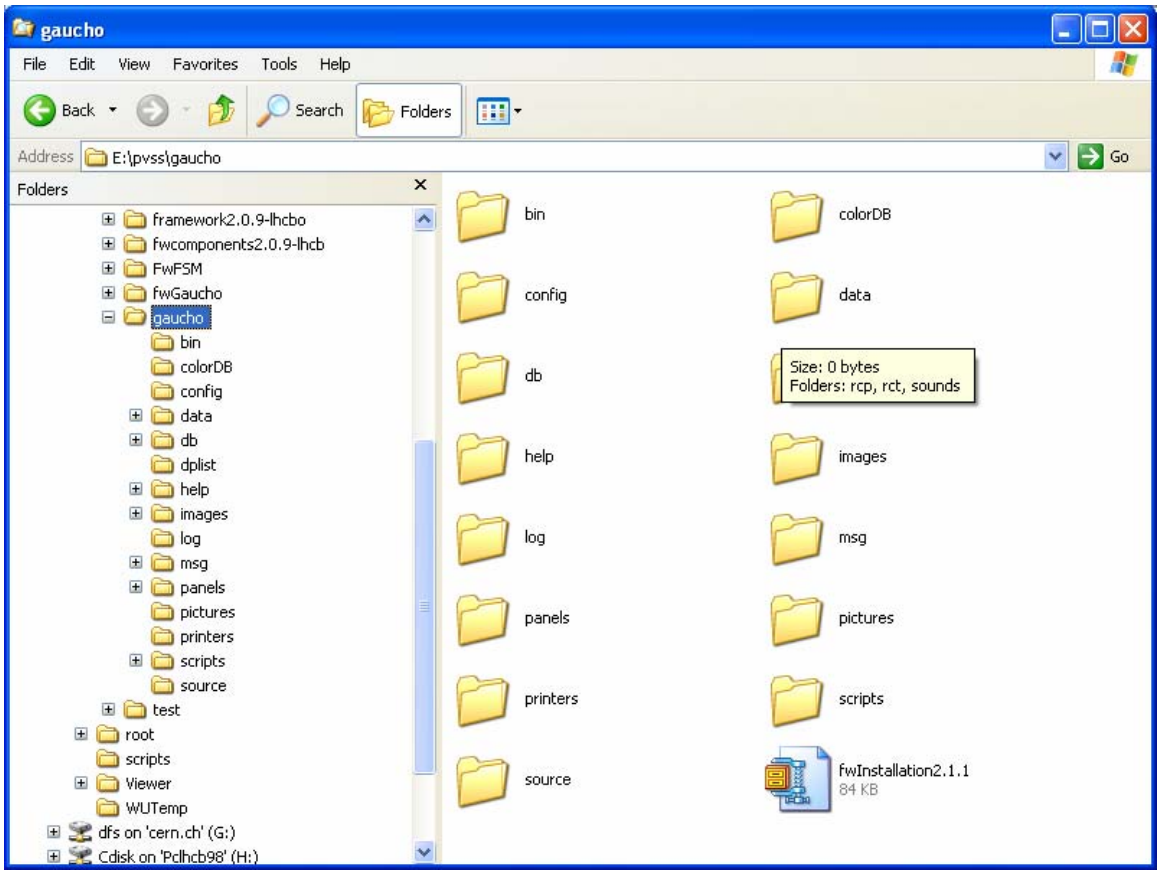
Installing the LHCb framework and the Gaucho component

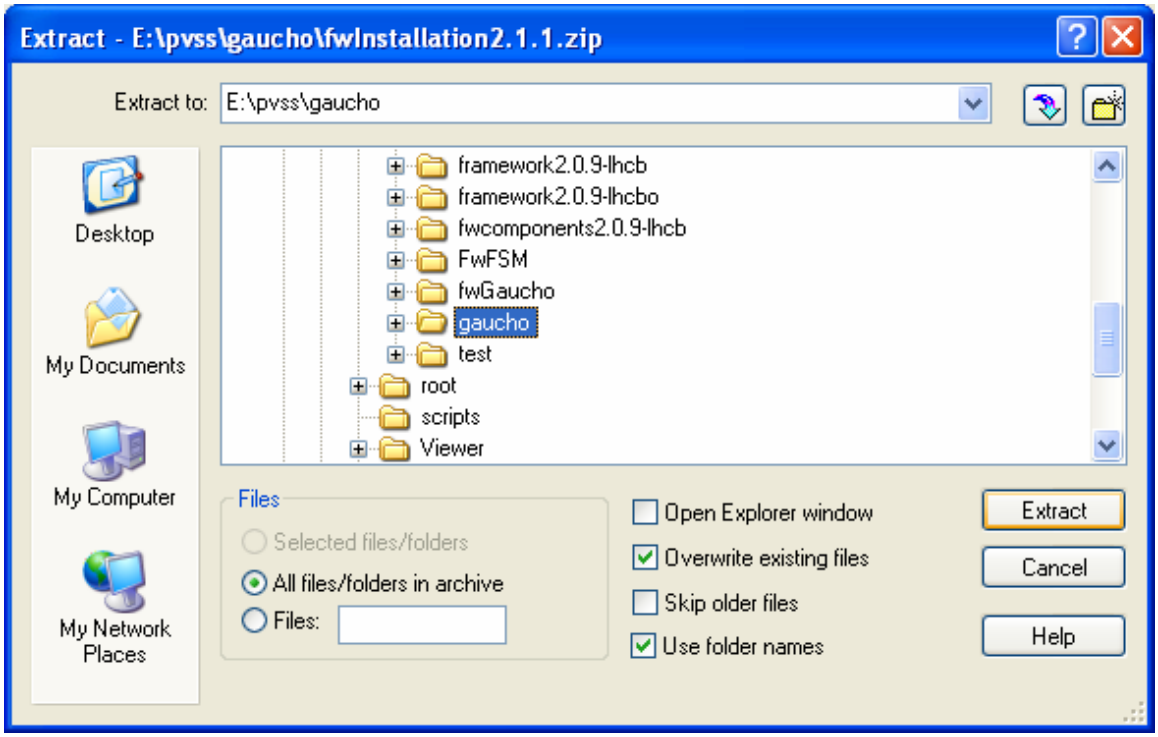
1. Download the component installation tool and the LHCb Framework zipfiles from the web page http://lhcb-comp.web.cern.ch/lhcb-comp/ECS/lhcb-fw/ .The current release is framework2.0.9-lhcbv1r0.
2. Unzip framework2.0.9-lhcbv1r0.zip
3. Download the installation tool from the web page under point 1.
4. Create a new PVSS project, e.g. 'gaucho' from the Project Administrator tool, by clicking the 'new project' icon:



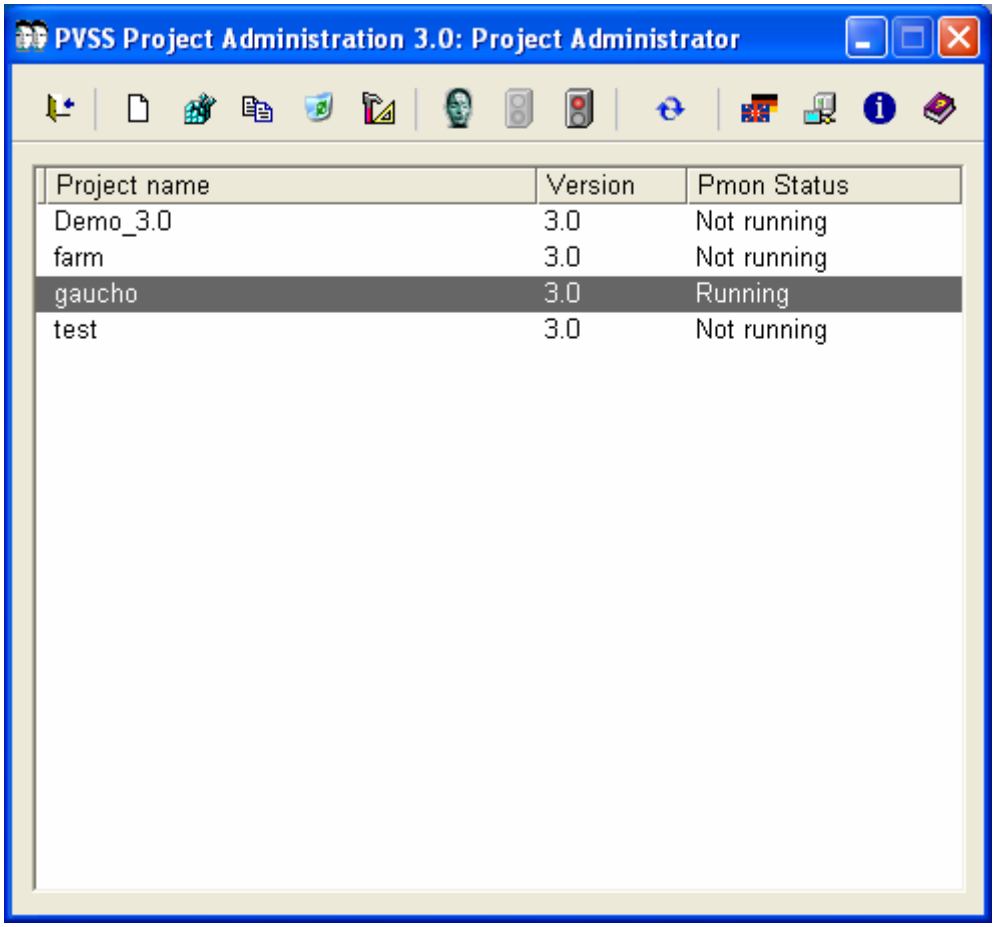


5. Unzip the installation tool into the root directory of your ‘gaucho’ project.

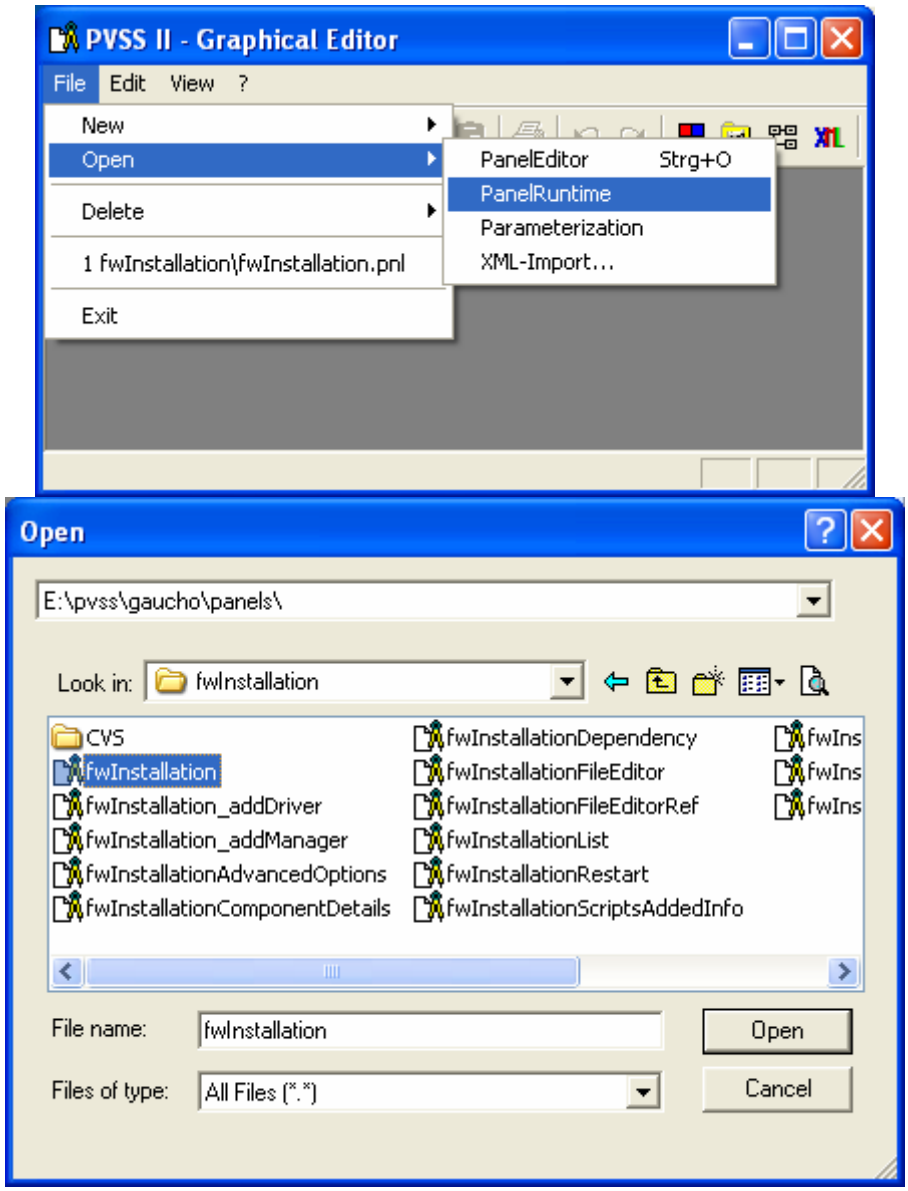




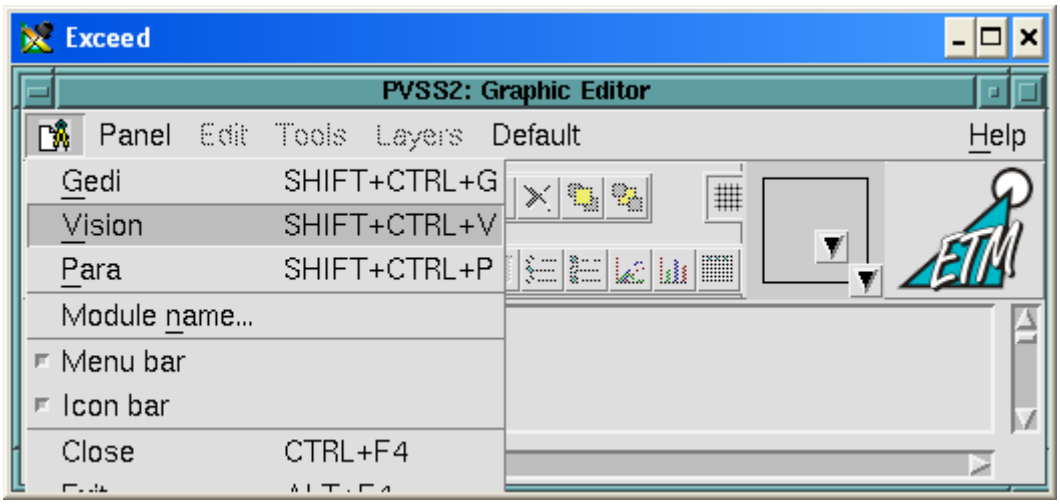
6. Start your project from the Project Administrator by clicking the green light:



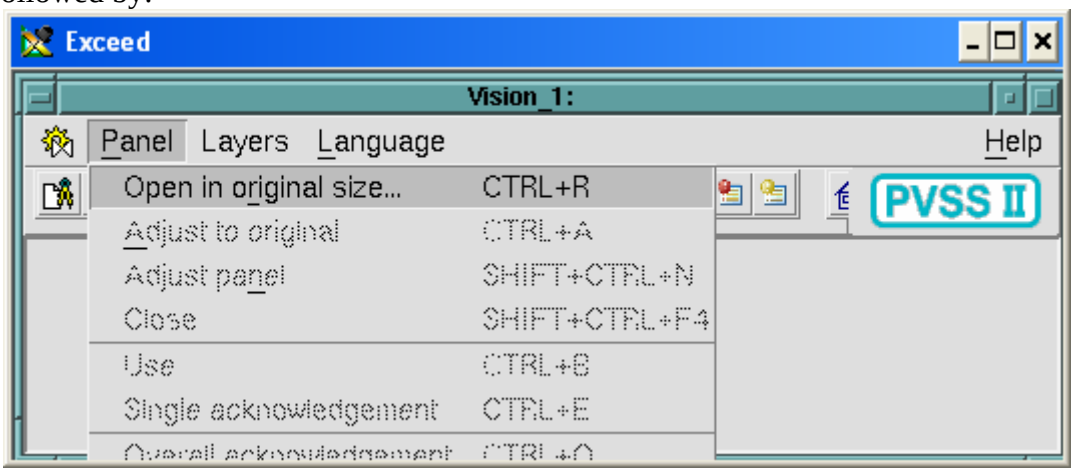
7. Open the framework installation tool from the Graphical Editor:


PVSS II - Graphical Editor
File Edit View ?
New
Open
Delete
1 fwInstallation\fwInstallation.pnl
Exit
PanelEditor Strg+O
PanelRuntime
Parameterization
XML-Import...
Open
E:\pvss\gaucho\panels\
Look in: fwInstallation
CVS
fwInstallation
fwInstallation_addDriver
fwInstallation_addManager
fwInstallationAdvancedOptions
fwInstallationComponentDetails
fwInstallationDependency
fwInstallationFileEditor
fwInstallationFileEditorRef
fwInstallationList
fwInstallationRestart
fwInstallationScriptsAddedInfo
File name: fwInstallation
Files of type: All Files (*.*)
Open
Cancel

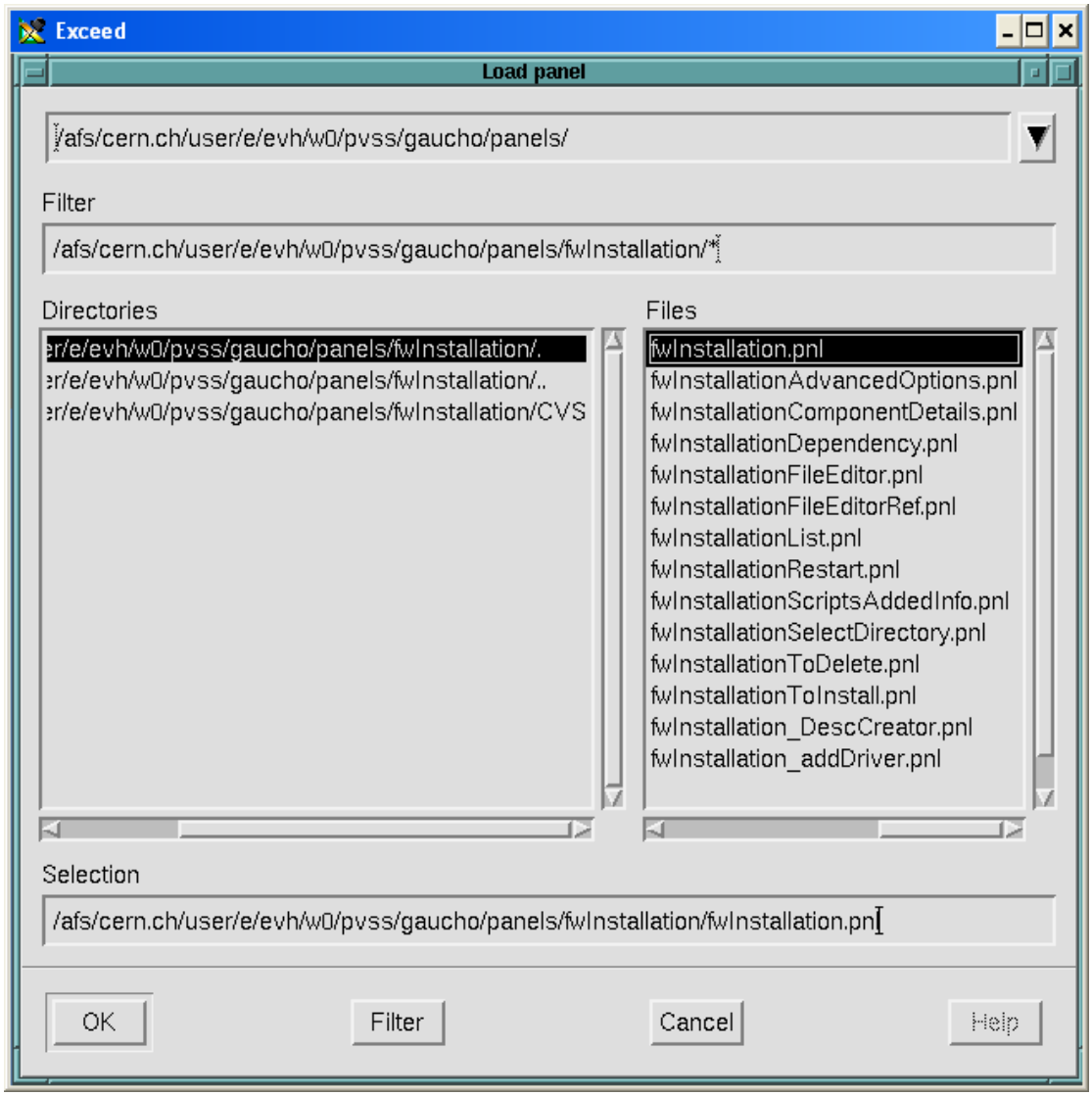
On Linux:


Exceed
PVSS2: Graphic Editor
Panel Edit Tools Layers Default Help
Gedi SHIFT+CTRL+G
Vision SHIFT+CTRL+V
Para SHIFT+CTRL+P
Module name...
Menu bar
Icon bar
Close CTRL+F4

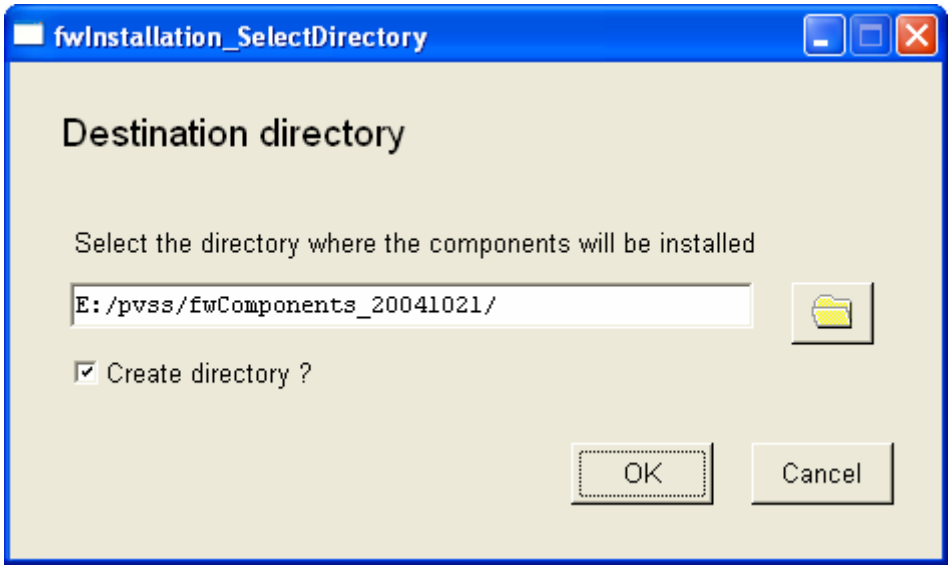
Followed by:



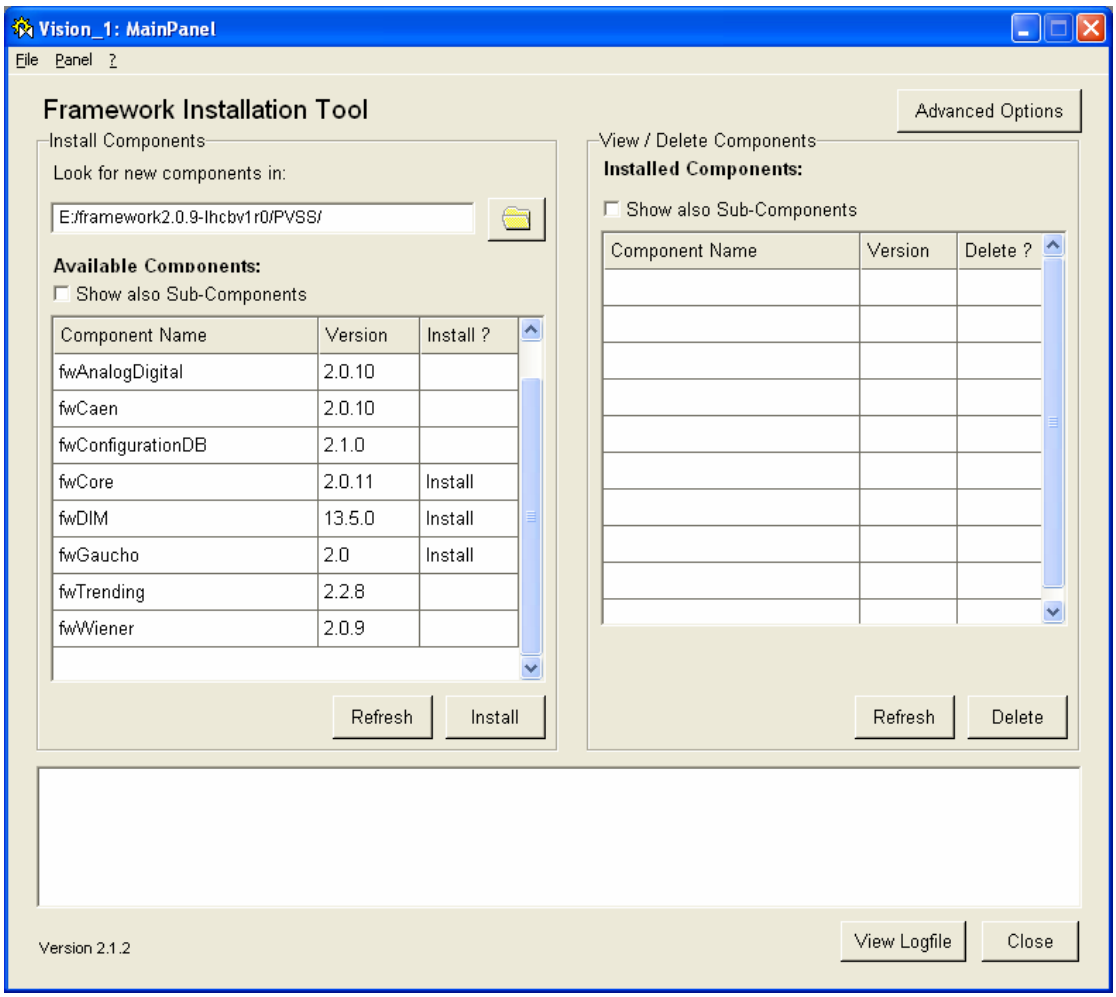
And:



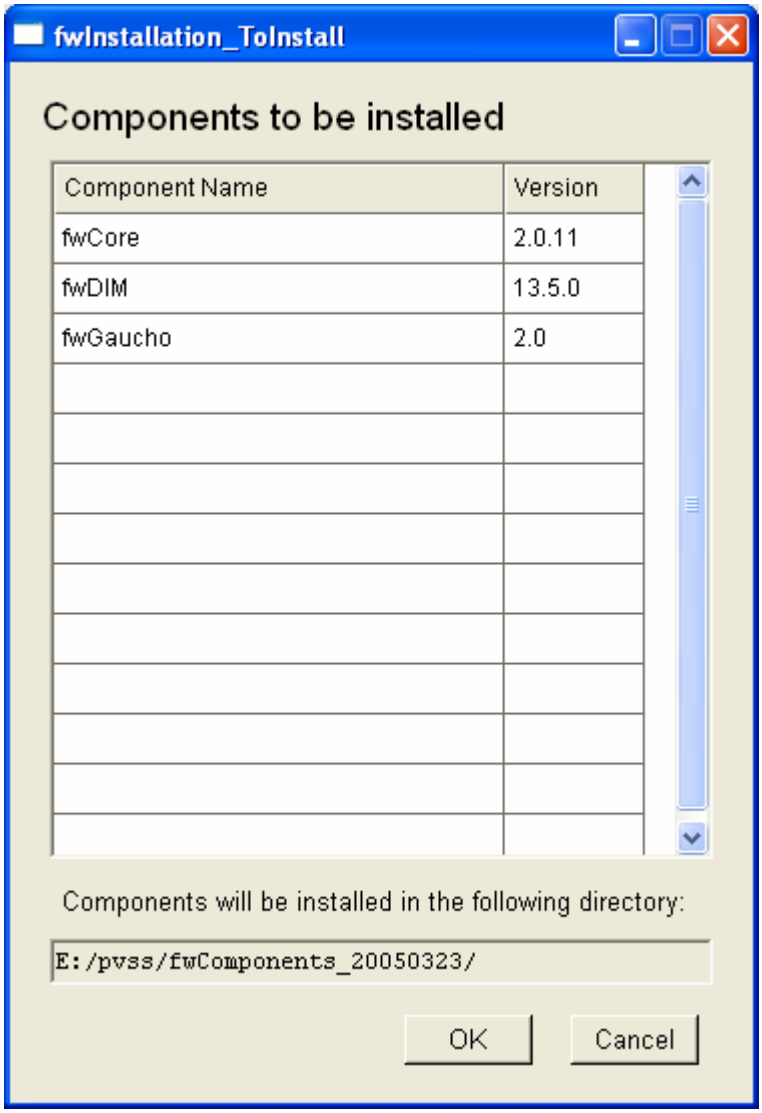
8. Click on OK, or change the directory name where you want the framework components to be installed:



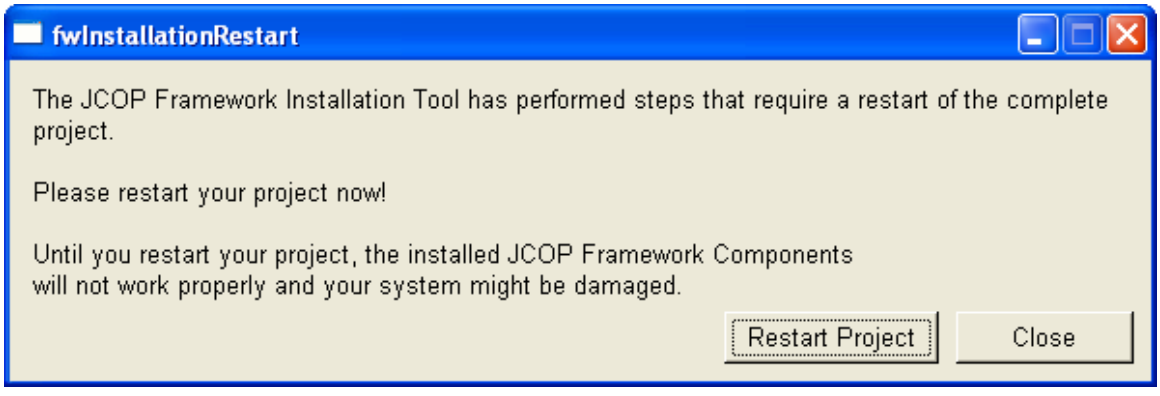
9. Under “Look for new components in:” put the name of the directory where you unzipped framework2.0.9-lhcbv1r0.zip under step 2 above:



10. To install the Gaucho framework component, click on fwCore, fwDIM, fwGaucho in the Install ? column. (You may install other components as you need, now, or later). Then click on Install. After the window:

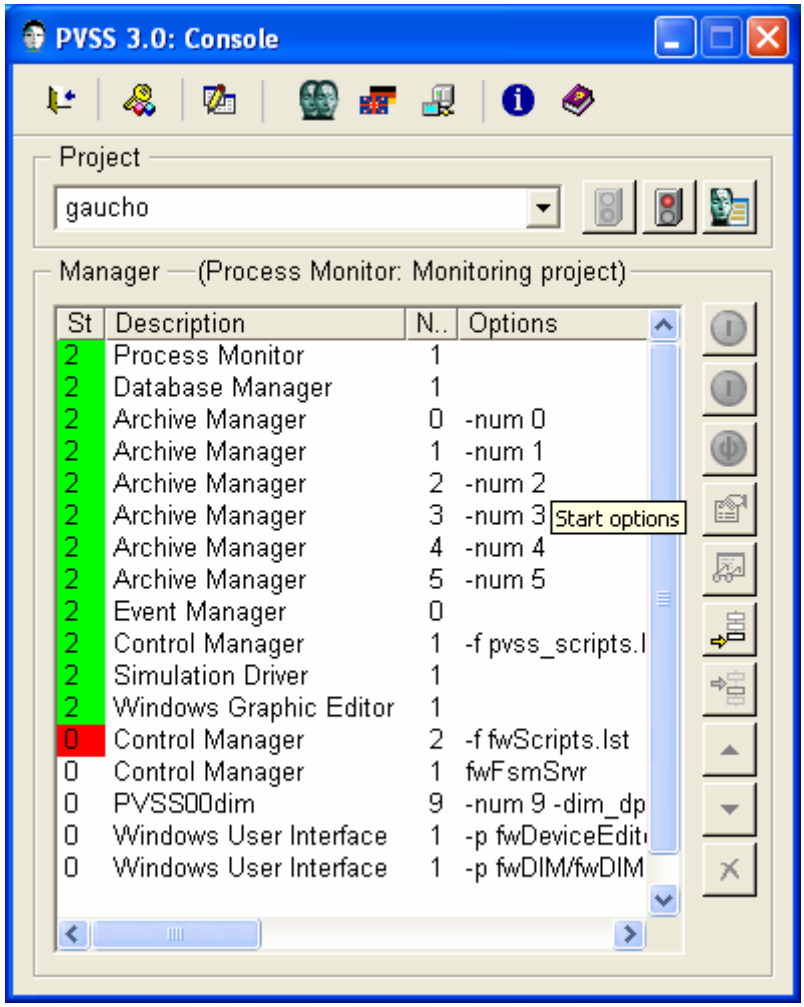


Click on OK. You should get:

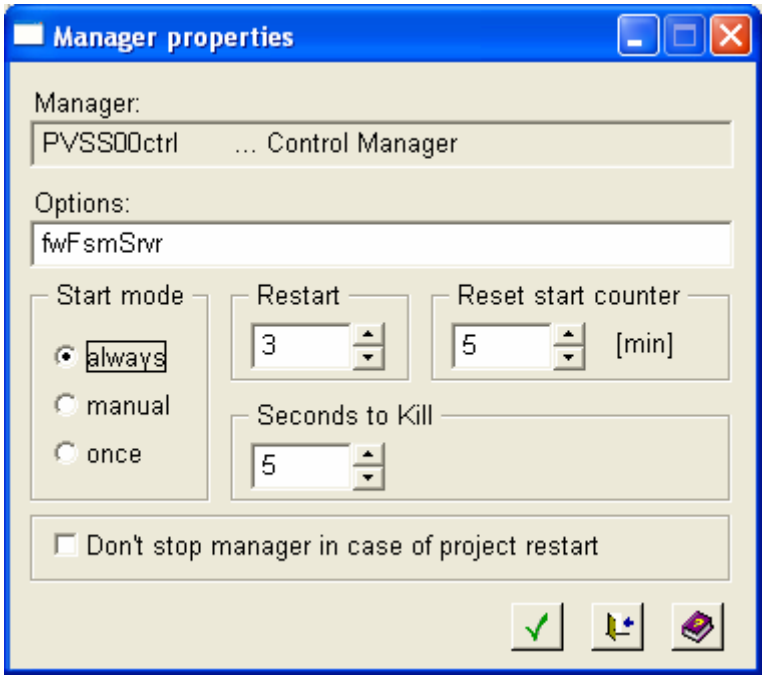


Click on “Restart Project”.

11. Your Project Console should look like:

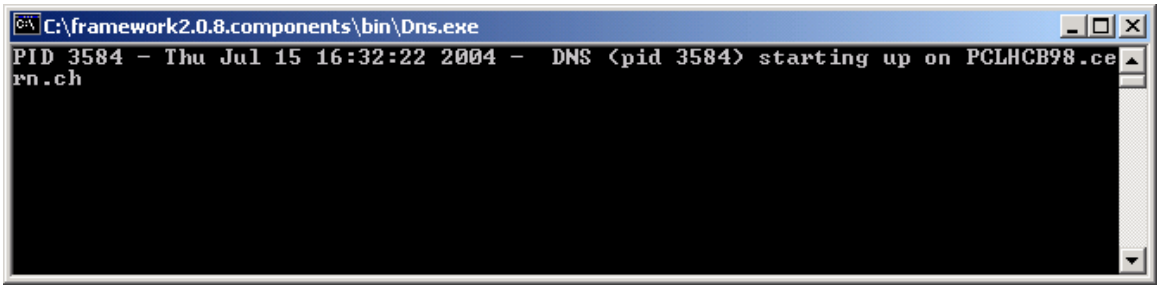


The installation tool added 4 managers. By default they have to be started manually. To change this, double click on them:



Click on ‘always’ to have the manager start up when you start your project. Before starting the PVSSdim manager, install the DIM DNS.

12. After installing the framework components, you should have a directory framework2.0.9.components/bin (or similar). Inside it you will find the DIM Domain Name Server, **Dns.exe**. DIM needs this program to find out which services are subscribed. Execute it by double clicking on it. A cmd window should appear:



On Linux:

setenv DIM_DNS_NODE yourhost.yourdomain.yourcountry
componentspath/bin/dns &

Modifying the FarmCTRL.ctl library (to allow jobsubmission from your account)

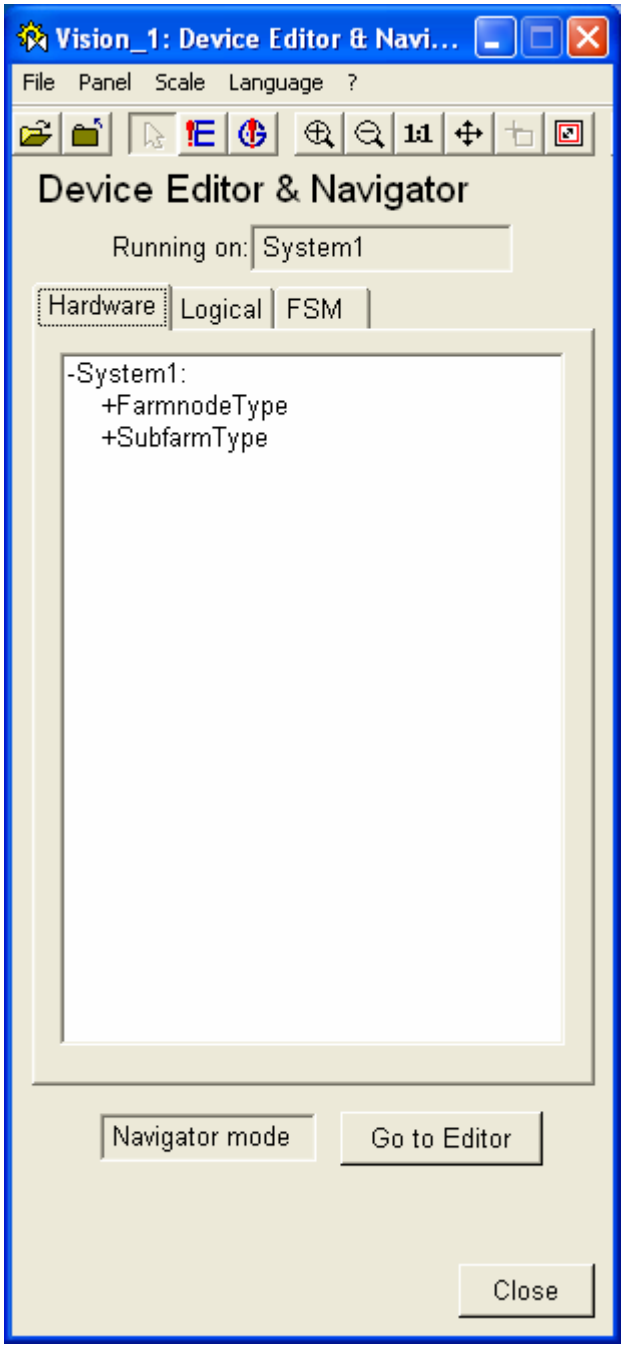
To submit jobs that run under your account on LXPLUS, you need to change the FarmCTRL.ctl library as follows. From a PVSS Graphical Editor window, select Edit->Libraries, and select FarmCtrl.ctl from the drop down menu in the top right hand corner.
In the startGaudijob function, update the path of your startgaudijob in the variable scriptname, edit your afs userid and password xxxxx in cmdname, and remove /k in syscmd if you don’t want to see the cmd window pop up on your screen when you submit a job.

```
  scriptname="~/cmtuser/Online/GauchoJob/v2r0/cmt/startgaudijob.sh";
  cmdname="ssh your_id@"+nodename+" "+scriptname+" &";
  system(cmdname);
```

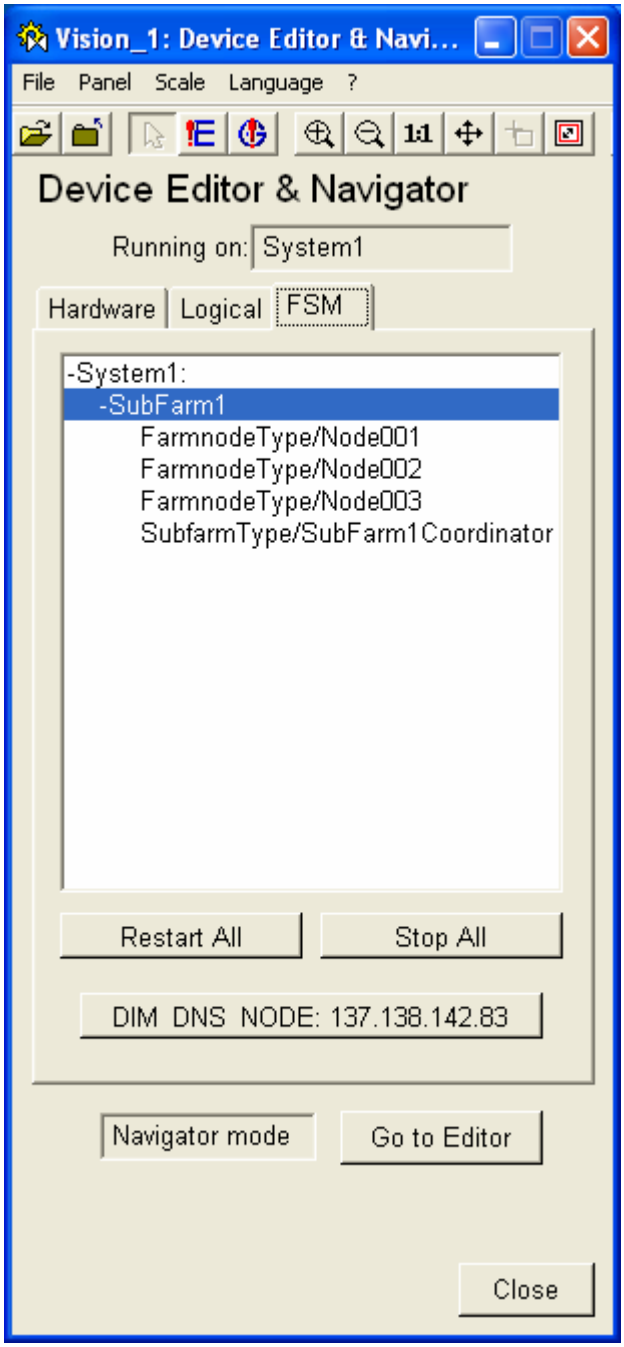
For Windows, this example uses “plink.exe” that you may need to download from http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html if it is not yet installed on your system. Remember to update your path to point to the place where plink.exe is installed. For Linux this is not necessary when working at CERN.

Using Gaucho

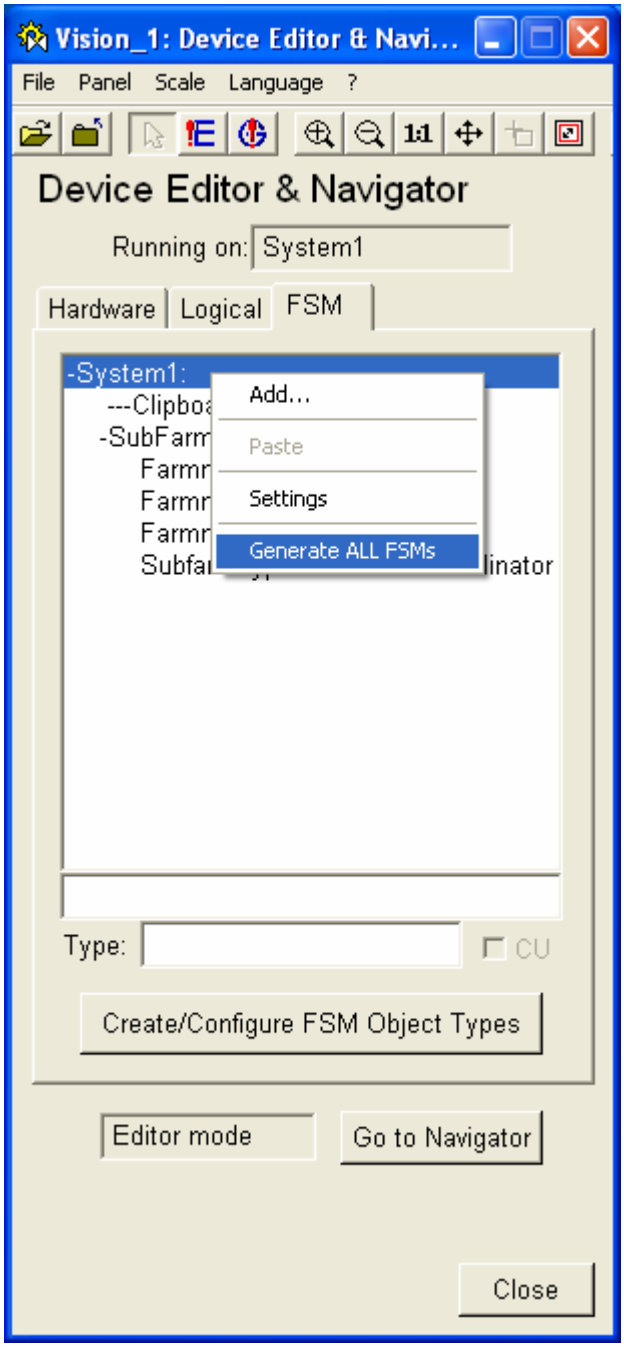
One of the managers you added to the console should start the Device Editor & Navigator:



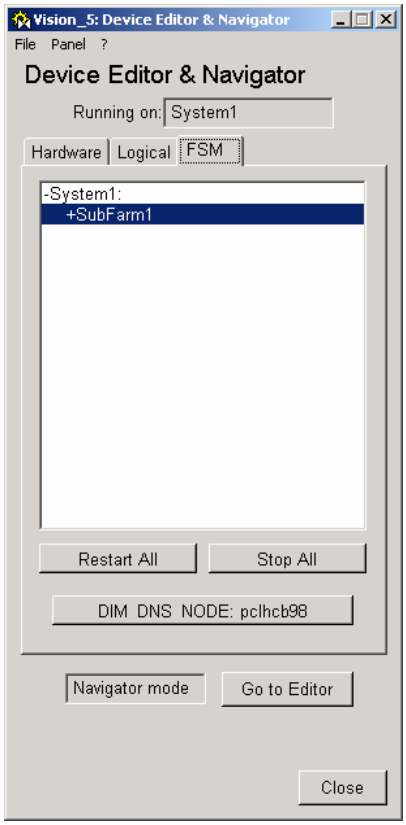
Click on the “FSM” tab.



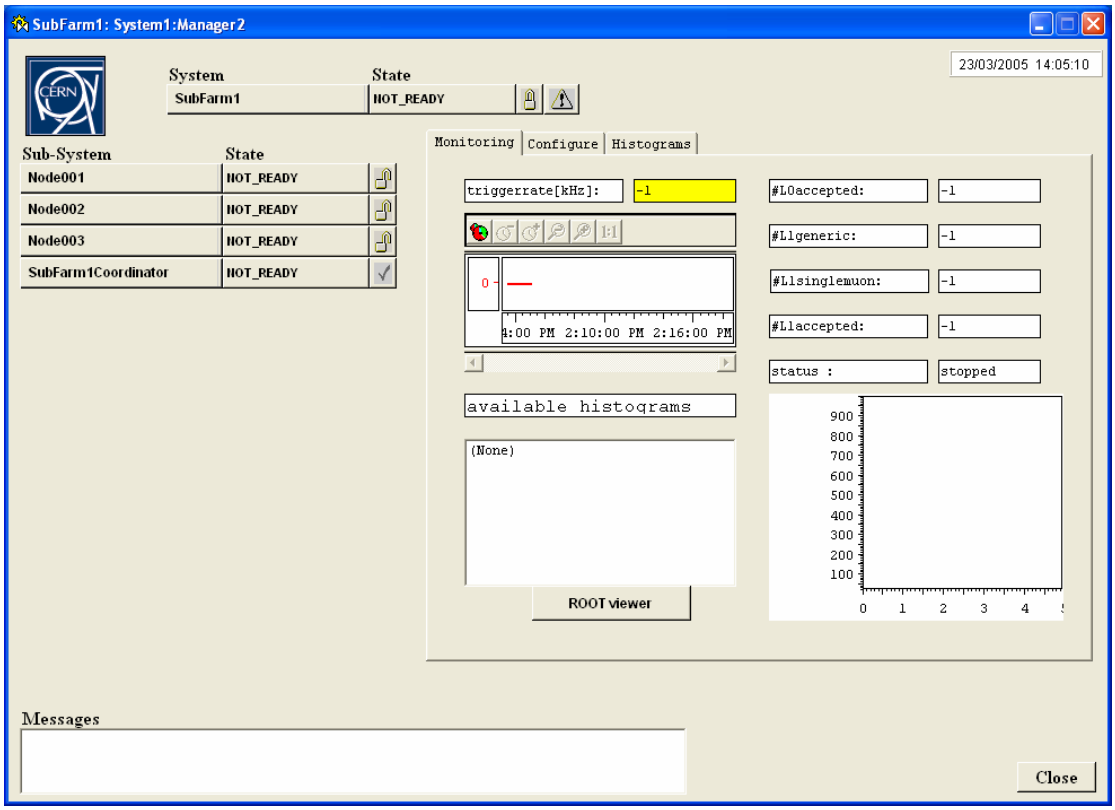
Click on “Go to Editor”, then select and right-click on System1, choose “Generate all FSMs”:



Now click on “Go to Navigator”, click on “Stop All”, then “Restart All”.

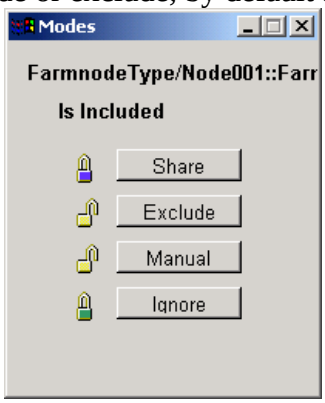


Right-click on SubFarm1, choose “View”:

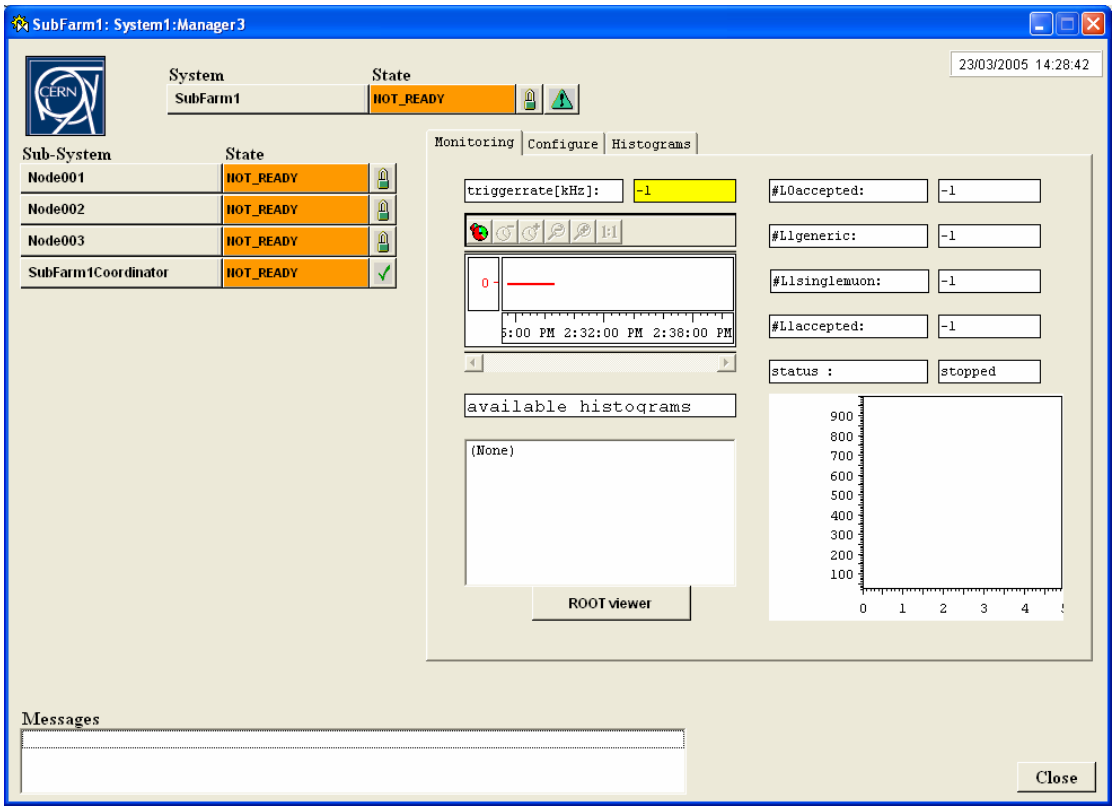


Click on the lock next to Subfarm1, NOT_READY, click on “Take”:

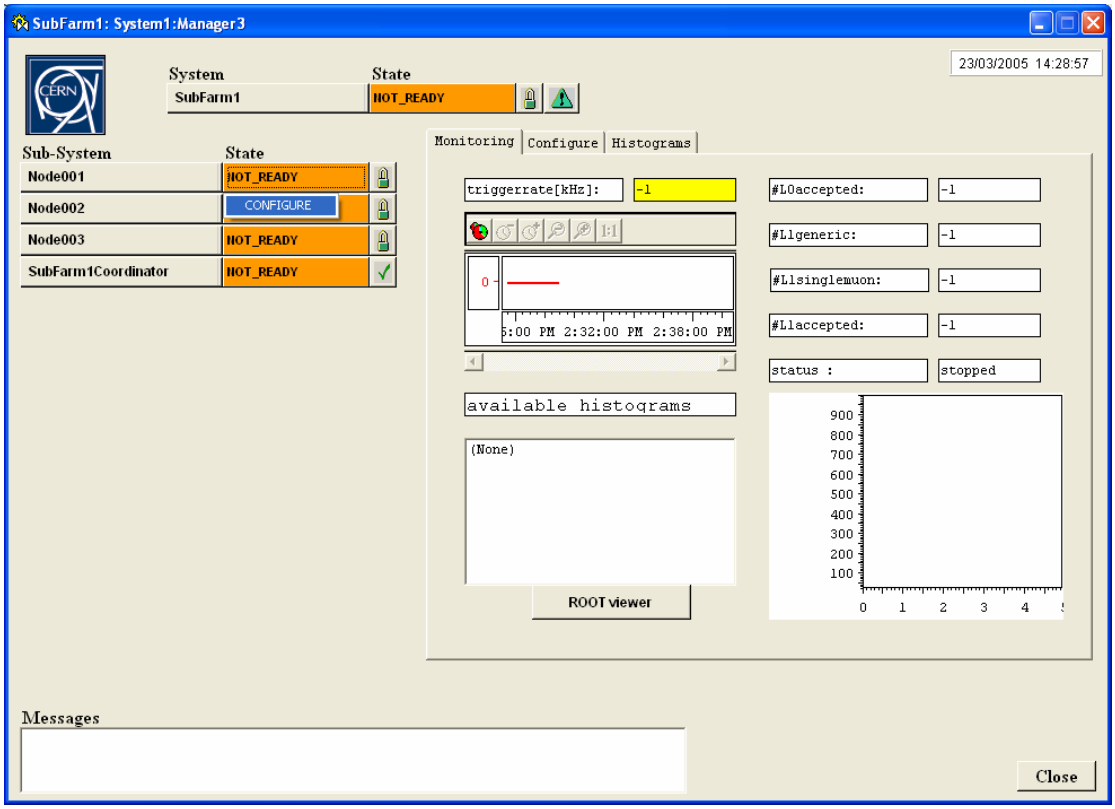
The colour of the NOT_READY fields changes to orange. Now you can decide which of the nodes you want to include or exclude, by default all are included:



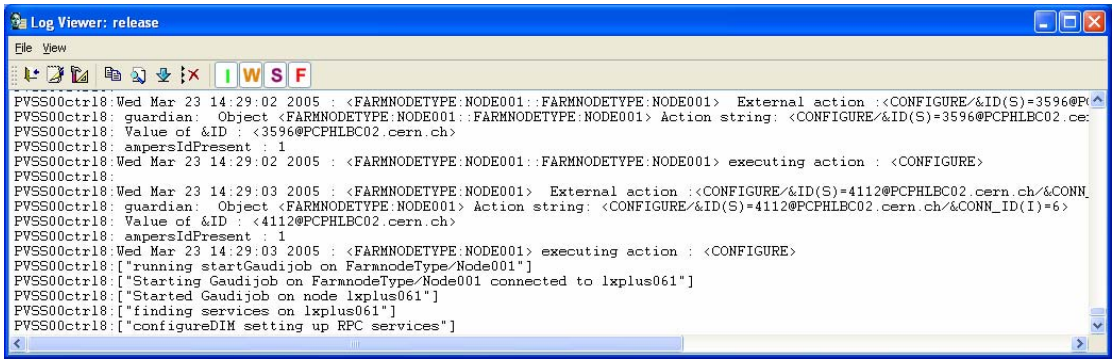
To submit jobs on all farms at the same time, click on the NOT_READY field next to Subfarm1.



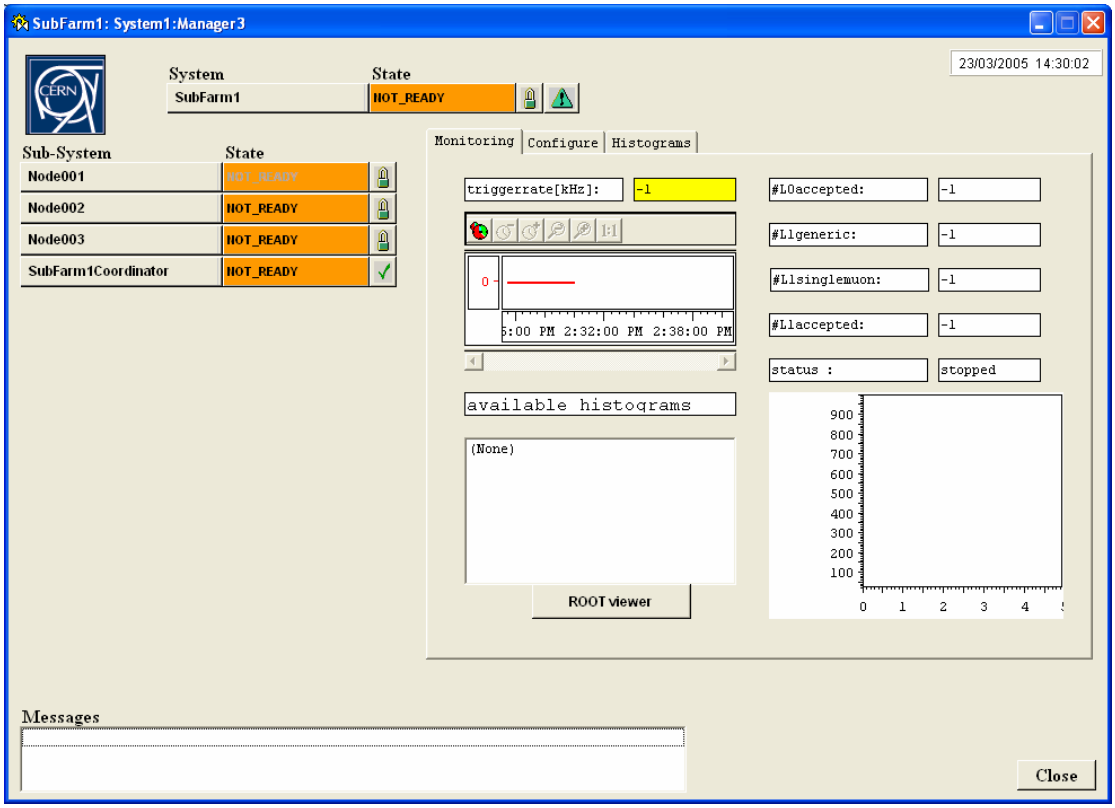
The CONFIGURE option will show up; select it. To submit a job on one farm at a time, go to the NOT_READY field of an individual node and click it.



Clicking on the CONFIGURE option submits a job to Node001. Look at the text in the logviewer. Before continuing, PVSS needs to know the hostname and the process ID of the Gaudi bob, so it can subscribe to the services that are published by it. PVSS will wait for a certain time, but if the job can not be submitted for some reason, the text in the log viewer may look like this:

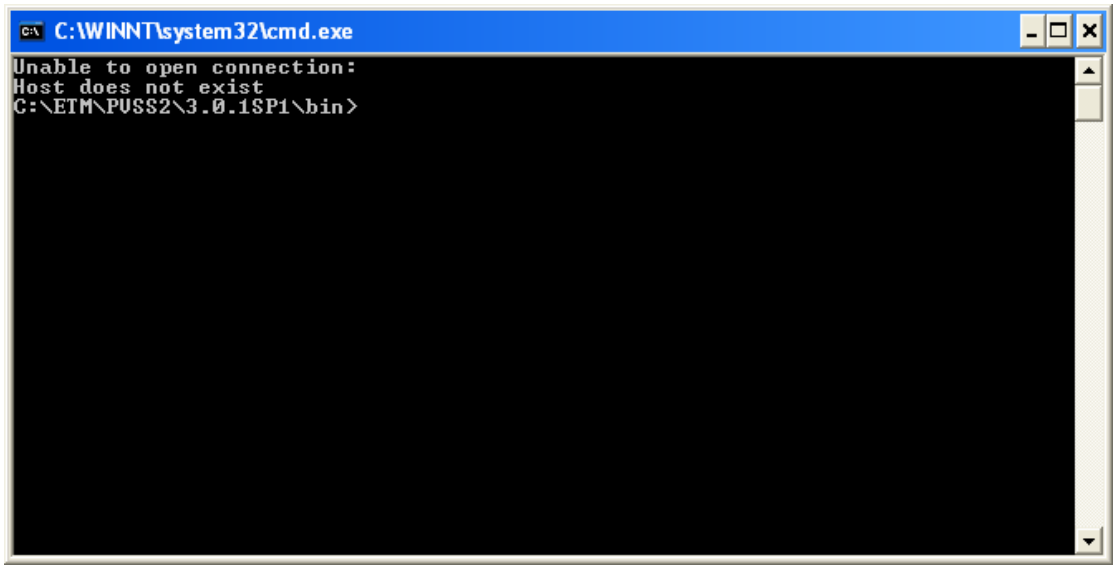


Notice that there is no mention of a PID, and the NOT_READY button remains gray:



If this happens, you should kill the window with your job and start again. You may need to reset the datapoint FramnodeType/Node001.Nodename to the value of the node without the /PID/, or choose the RESET command as explained below.

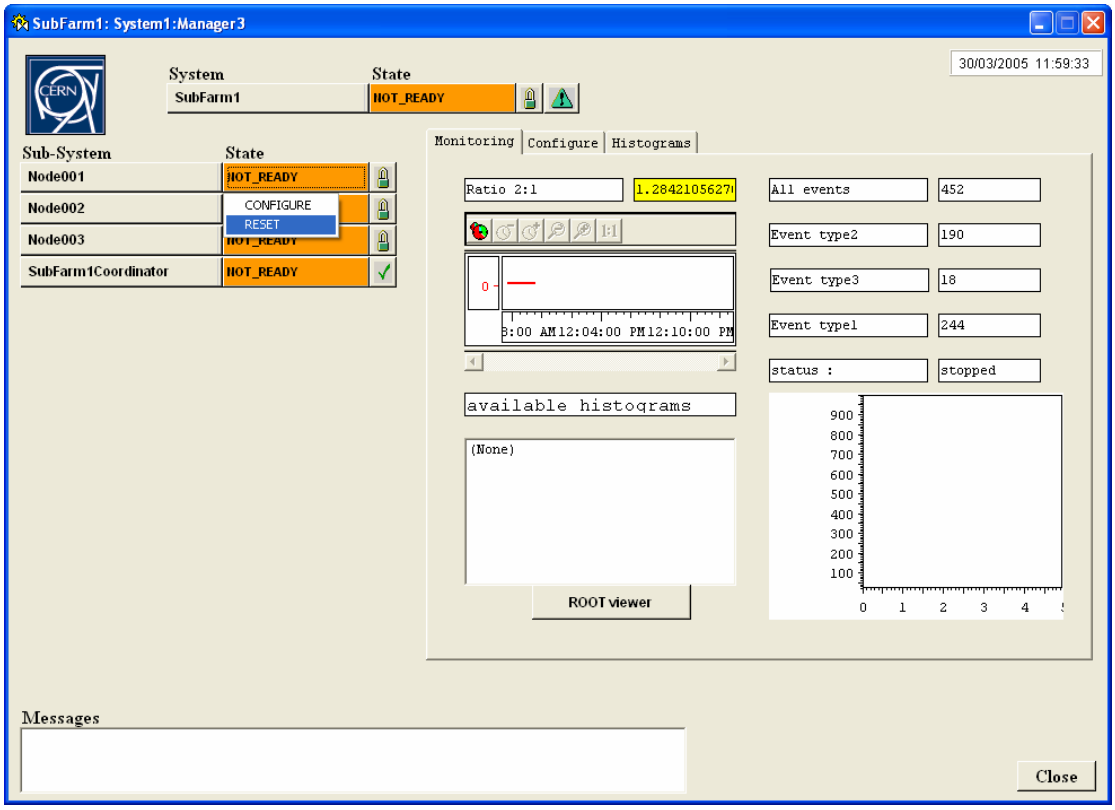
When a PID from a previous job is not correctly removed from the system by PVSS, you could get a message



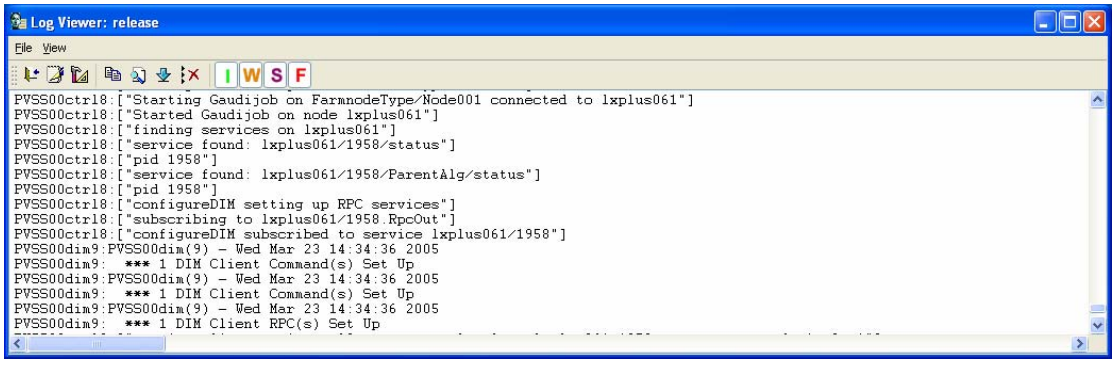
In the PVSS log file you would see something like:

```
PVSS00ctrl2:["Starting Gaudijob on FarmnodeType/Node001 connected to
lxplus061/20132"]
PVSS00ctrl2:Unable to open connection:
PVSS00ctrl2:Host does not exist
```

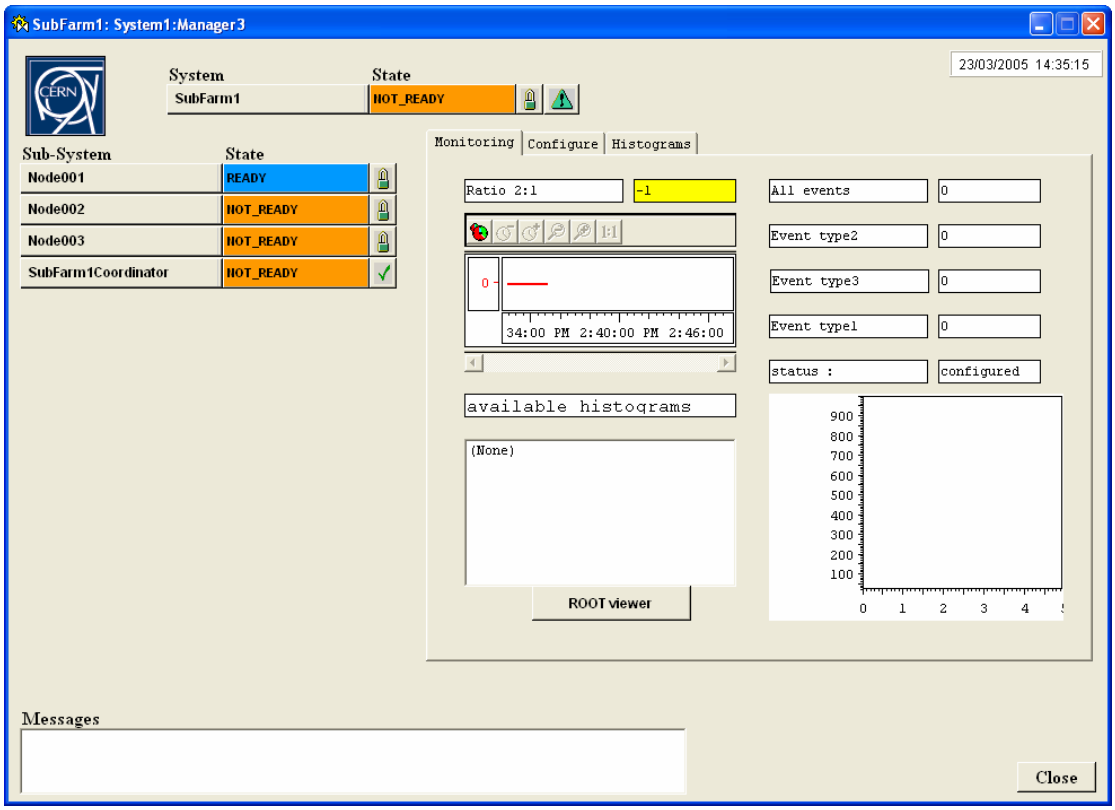
If this happens, choose the RESET command instead of CONFIGURE:



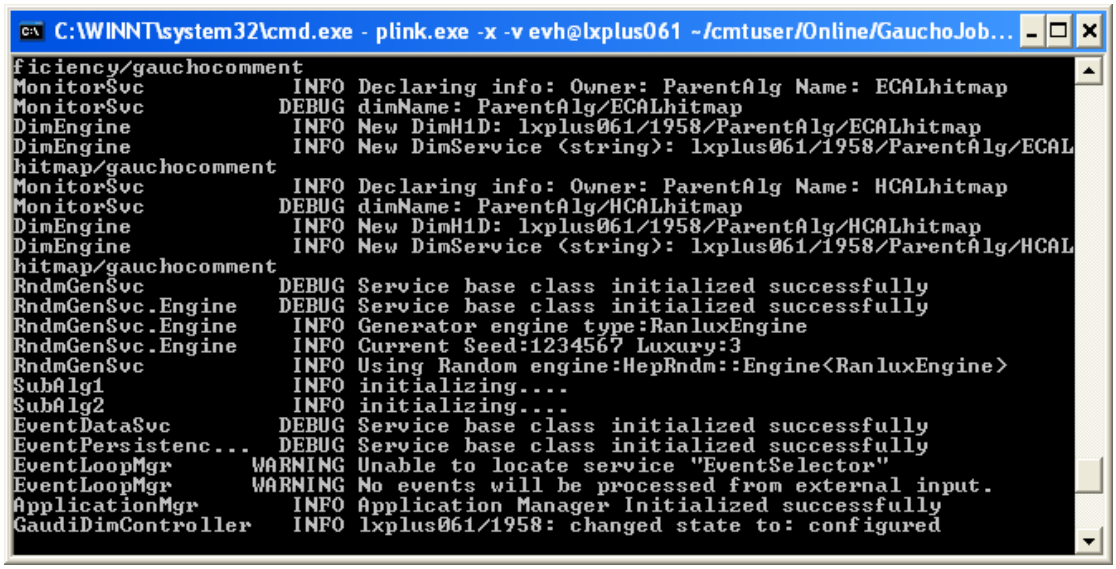
Try CONFIGURE again. If all goes well, the status of the node changes to "configured" and the usual Gaudi ouput appears in a cmd window:



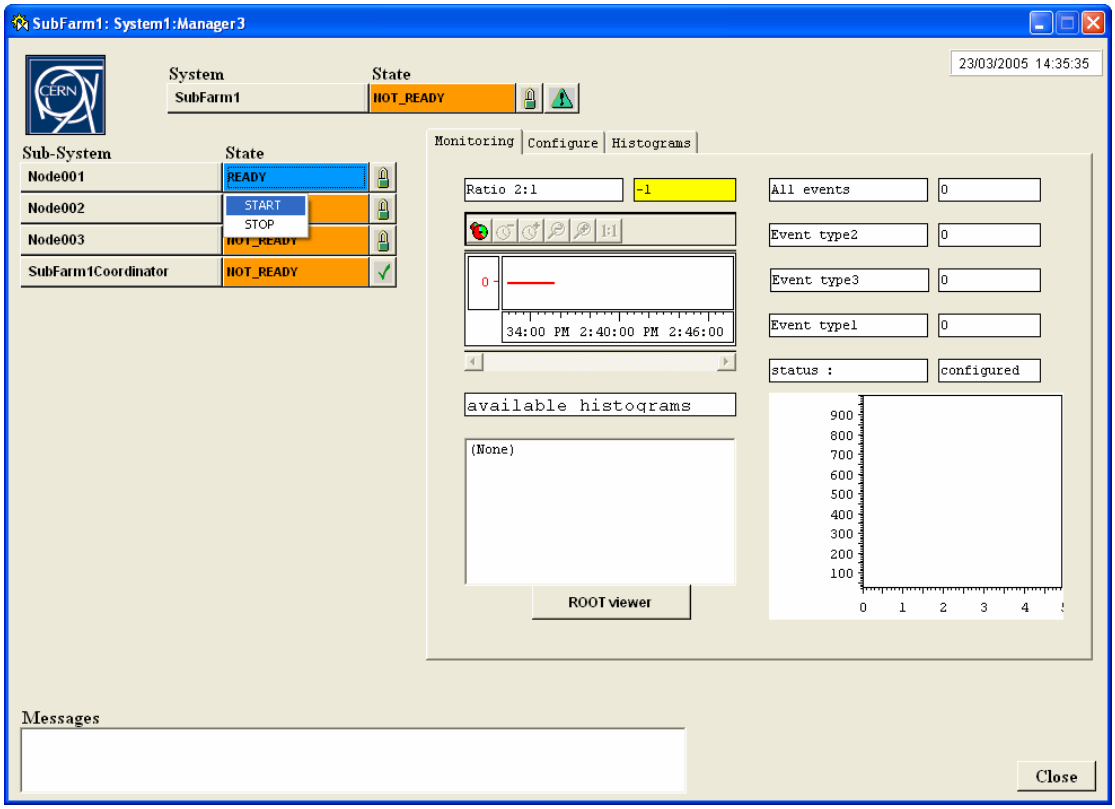
Here you can see that the PID was correctly found by PVSS and that it could subscribe to the required DIM services. The state of the node should become READY, the counters should reset themselves to 0 and the status field should read "configured":



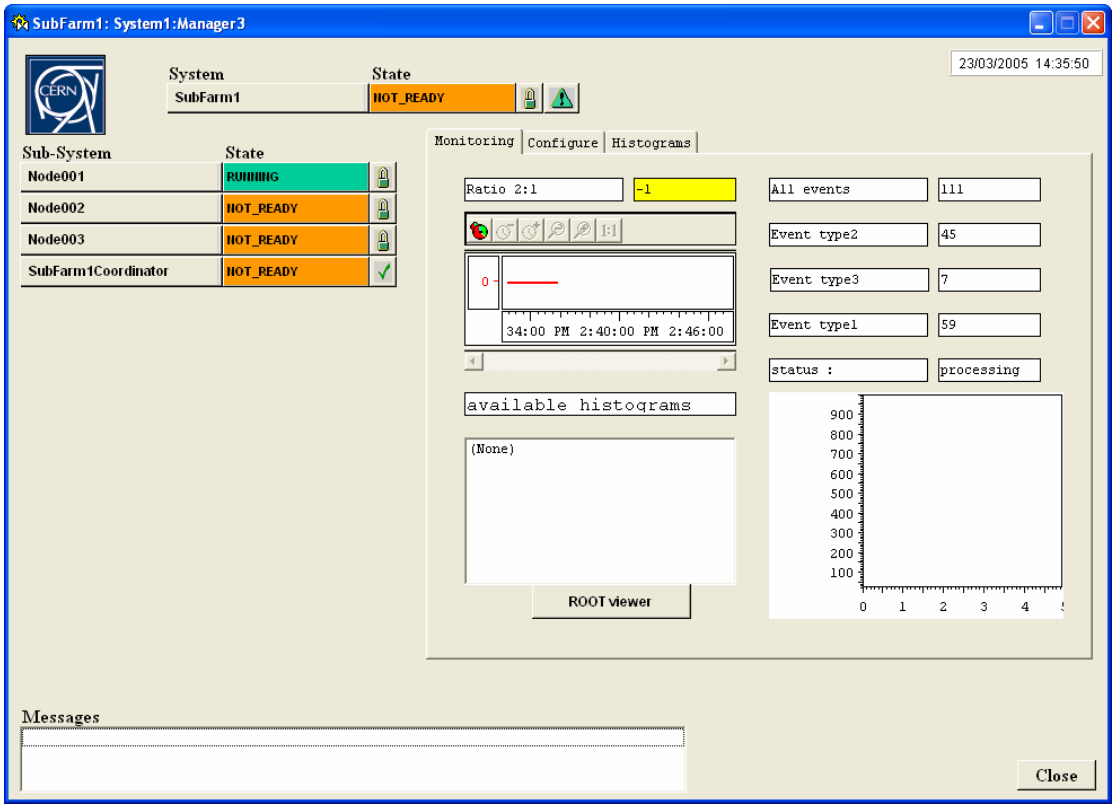
The output of the GaudiJob should look as follows (the last two lines should appear informing you that the Application Manager correctly initialized itself and the state of the job become ‘configured’):



To start a job, go to the READY field, click on it and choose START:



The node will go into the ‘RUNNING’ state and the status will display ‘processing’:

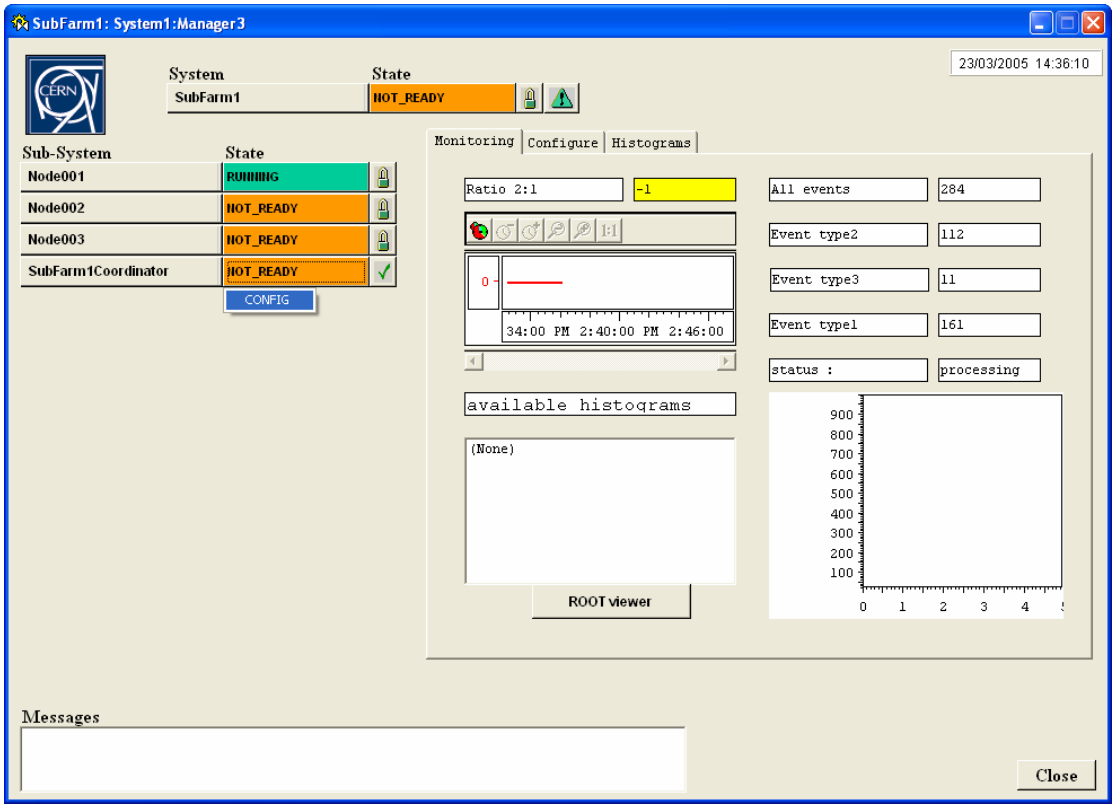


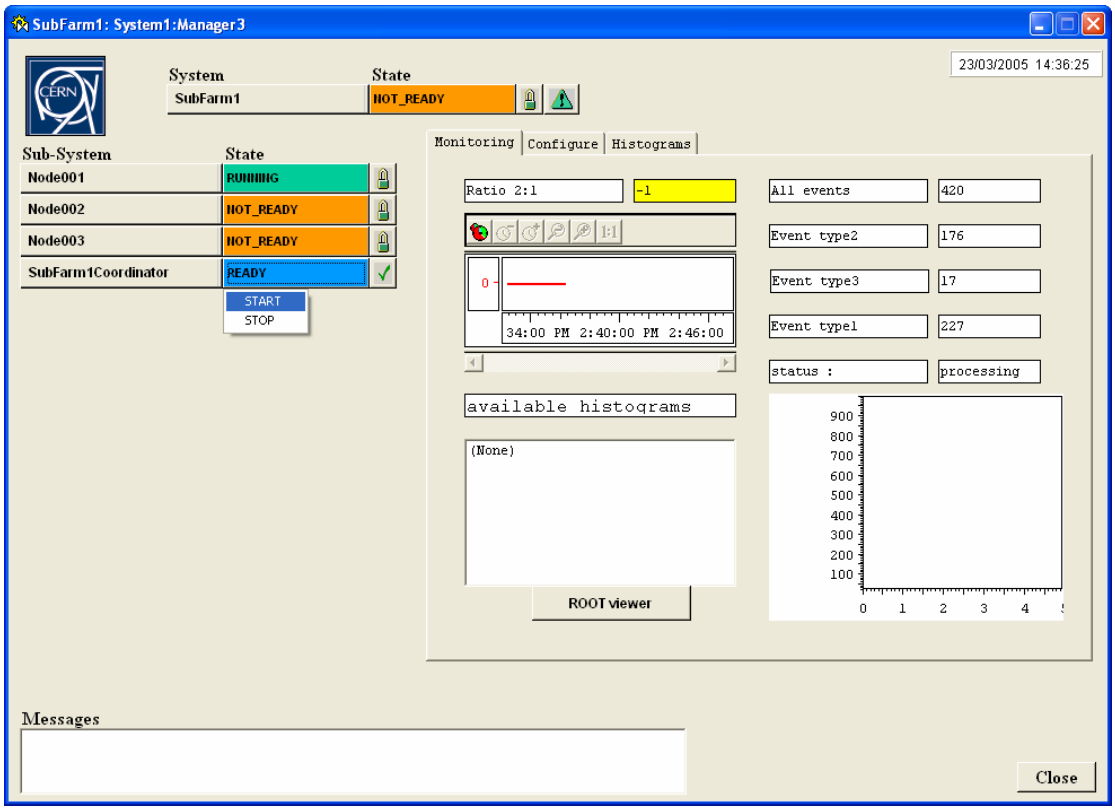
The text fields before the counters are filled with the description that was given when the MonitorSvc.declareInfo method was called in the GaudiJob.

Displaying published histograms

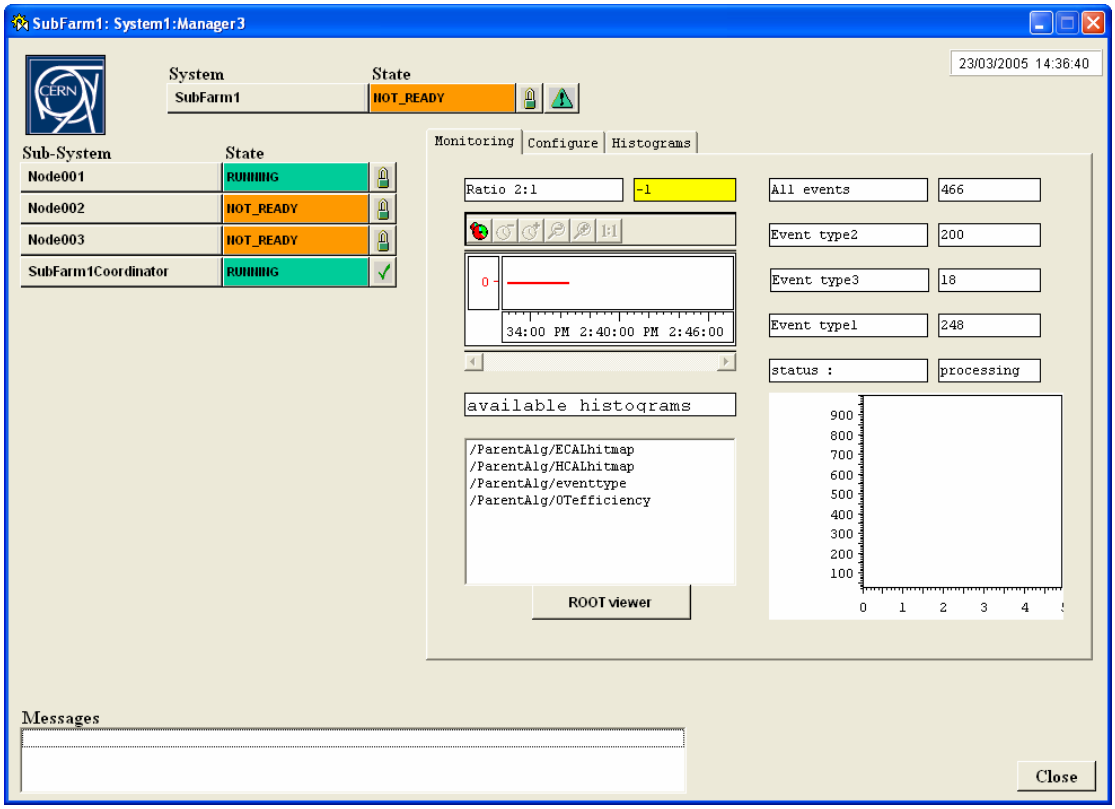
This section describes the facilities under the “Monitoring” tab.

To display published histograms (i.e. histograms that the GaudiJob published using the MonitorSvc.declareInfo methos), choose config in the SubfarmCoordinator, then start. The SubfarmCoordinator takes care of getting the results from all active nodes and adding them together.

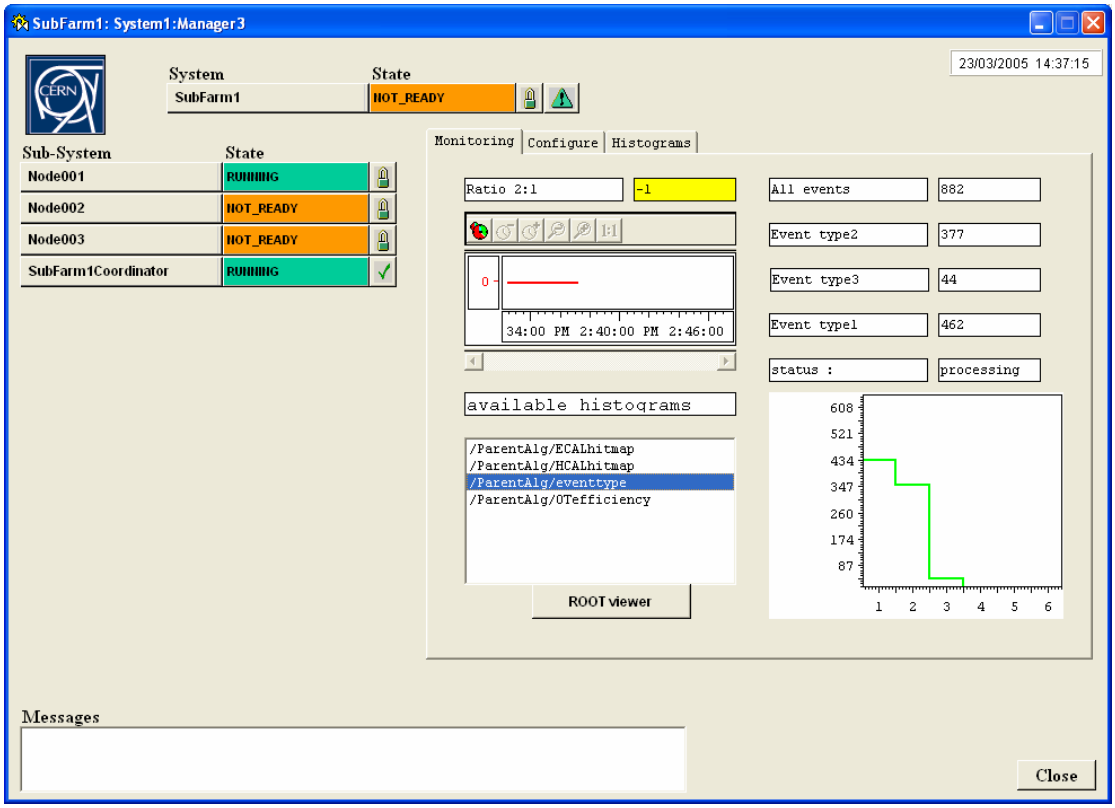

SubFarm1: System1:Manager3
23/03/2005 14:36:10
System
State
SubFarm1
NOT_READY
Sub-System
State
Node001
RUNNING
Node002
NOT_READY
Node003
NOT_READY
SubFarm1Coordinator
NOT_READY
CONFIG
Monitoring
Configure
Histograms
Ratio 2:1
-1
All events
284
Event type2
112
Event type3
11
Event type1
161
status :
processing
34:00 PM 2:40:00 PM 2:46:00
available histograms
(None)
ROOT viewer
Messages
Close


SubFarm1: System1:Manager3
23/03/2005 14:36:25
System
State
SubFarm1
NOT_READY
Sub-System
State
Node001
RUNNING
Node002
NOT_READY
Node003
NOT_READY
SubFarm1Coordinator
READY
START
STOP
Monitoring
Configure
Histograms
Ratio 2:1
-1
All events
420
Event type2
176
Event type3
17
Event type1
227
status :
processing
34:00 PM 2:40:00 PM 2:46:00
available histograms
(None)
ROOT viewer
Messages
Close

The list of available histograms will show up and can be selected by clicking on its name.



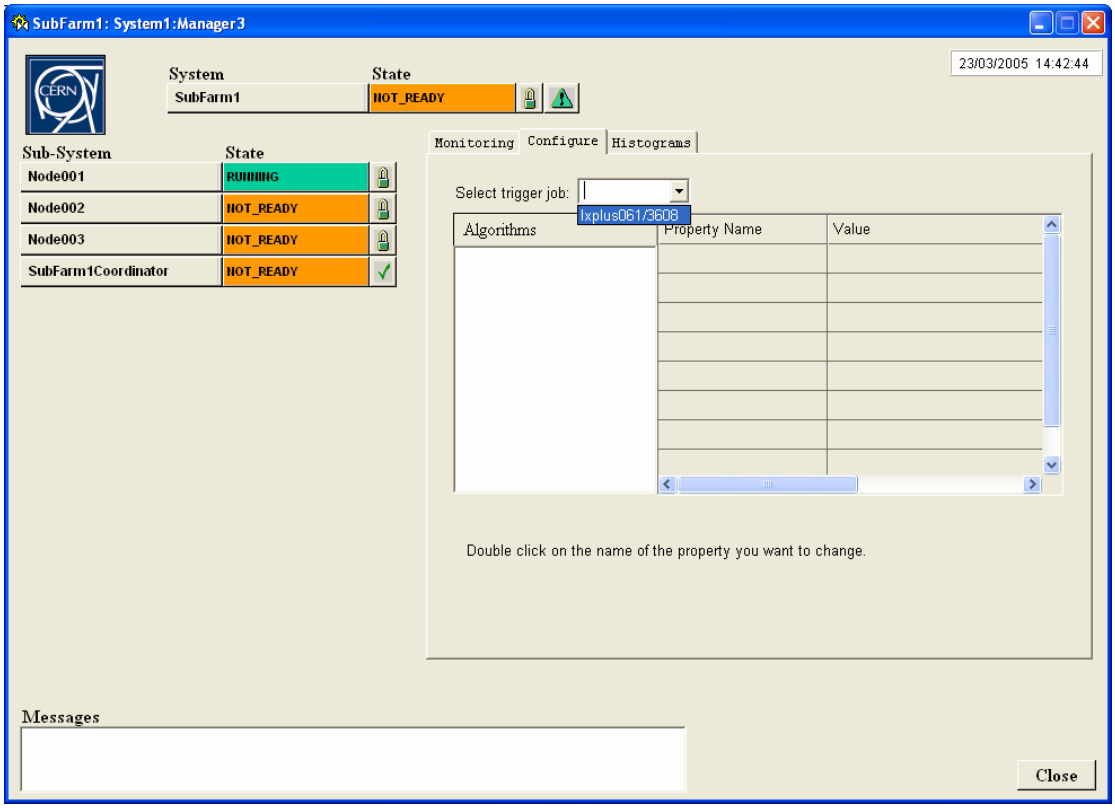
It may take some time before the command to send the data to PVSS reaches the job. As the jobs progress the values will be updated in real time. The histogram is the sum of the data coming from all active nodes.



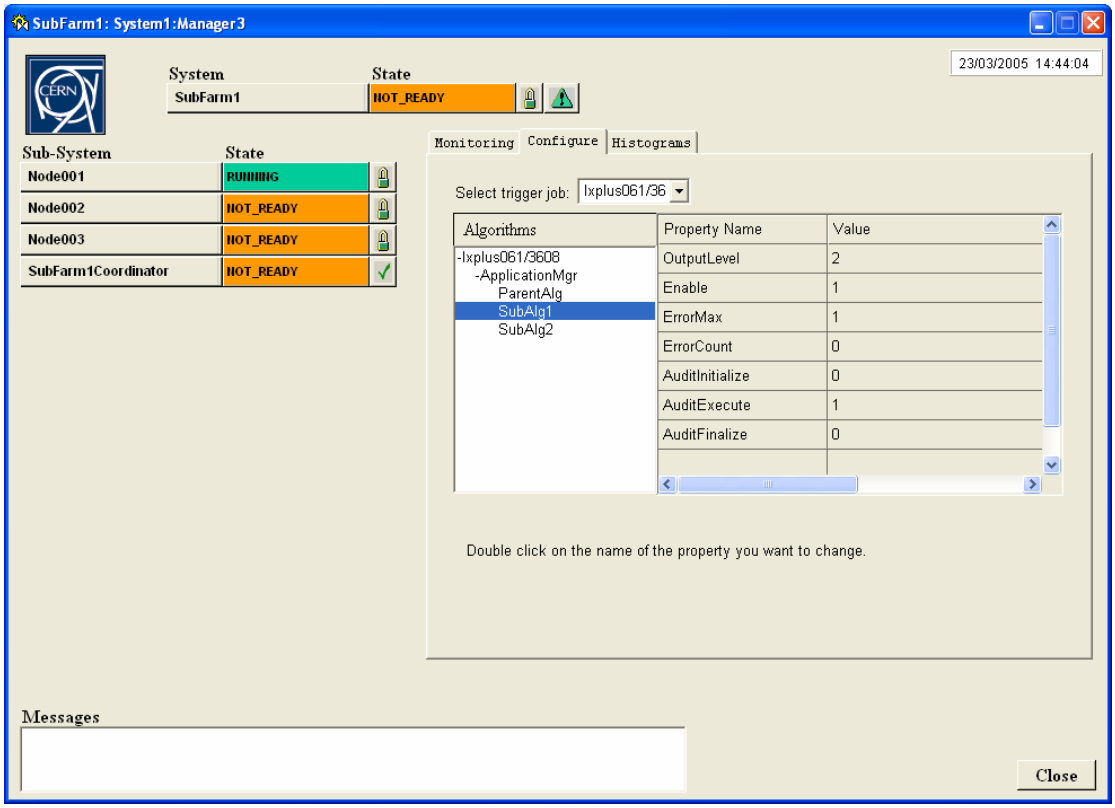
When you have finished, STOP the SubfarmCoordinator.

Inspecting the Algorithm tree

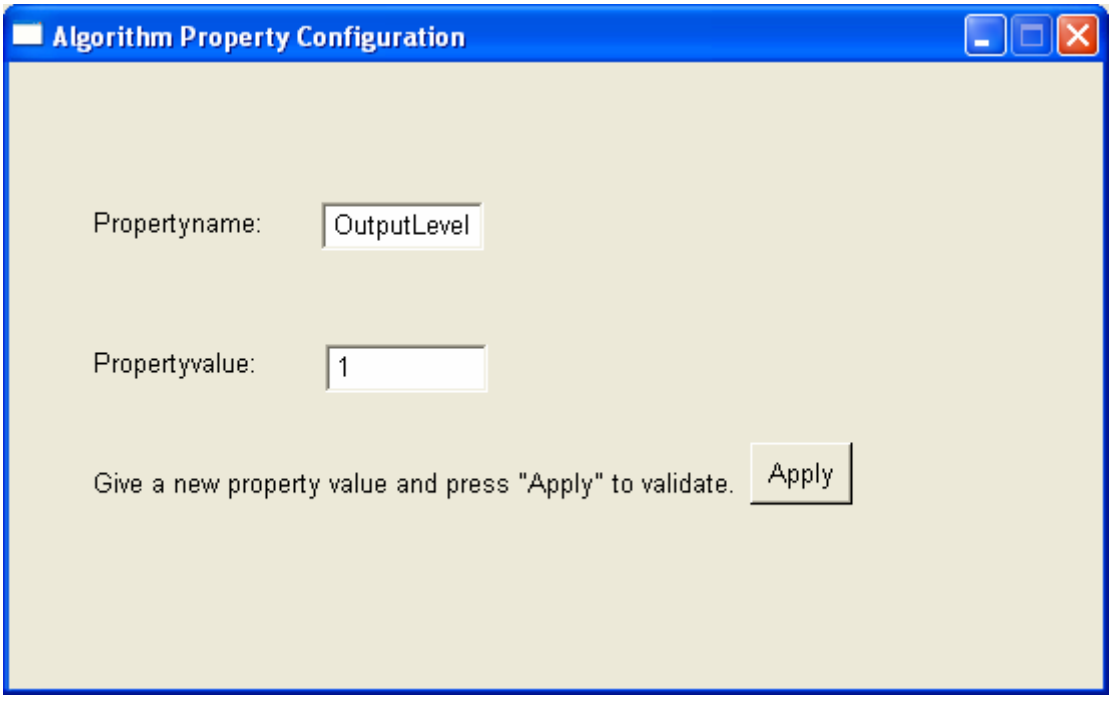
The features in this section are included as a ‘proof of concept’. Changing the options of an algorithm on the fly is dangerous and is not guaranteed to work.
To look at the algorithms inside a running GaudiJob, click on the “Configure” tab. The SubfarmCoordinator should not be running, as we are looking inside individual jobs. The pulldown menu next to “Select trigger job” will contain the list of jobs that are active. Select one by clicking on it:



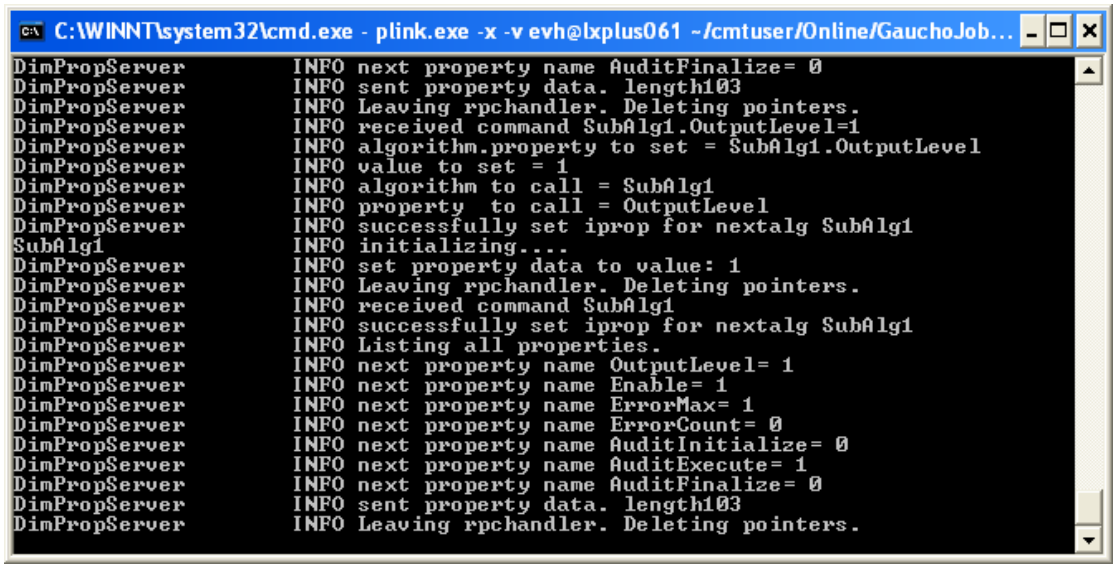
The algorithm tree should appear as a tree (click on the + to expand, click on the – to reduce). It may take a few seconds for the system to respond. Clicking on an algorithm will display its properties and values in the table next to it.



To change a property (option) double click on it. Fill out the panel:



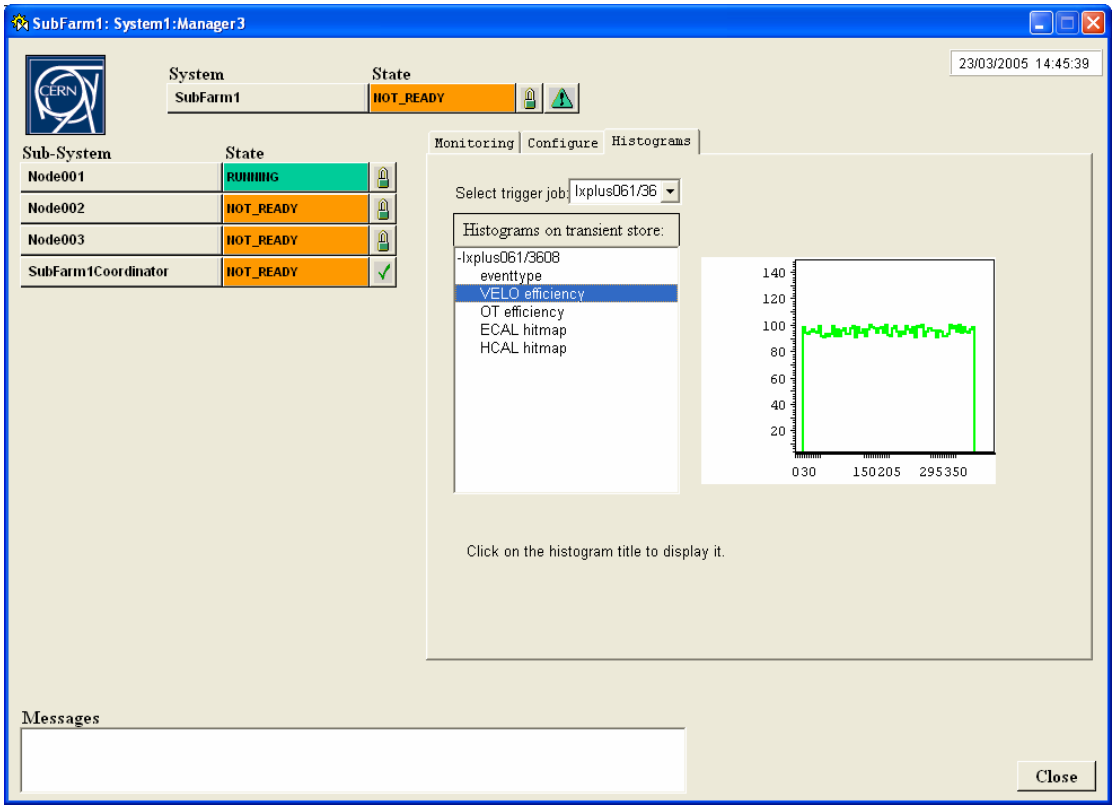
Clicking on the “Apply” button will call the setproperty and initialize methods of the algorithm:



As mentioned before, this should be used with care and is not guaranteed to work. In a future version of Gaucho, algorithms and their options should be selected before the job is run.

Inspecting the Transient Histogram Store

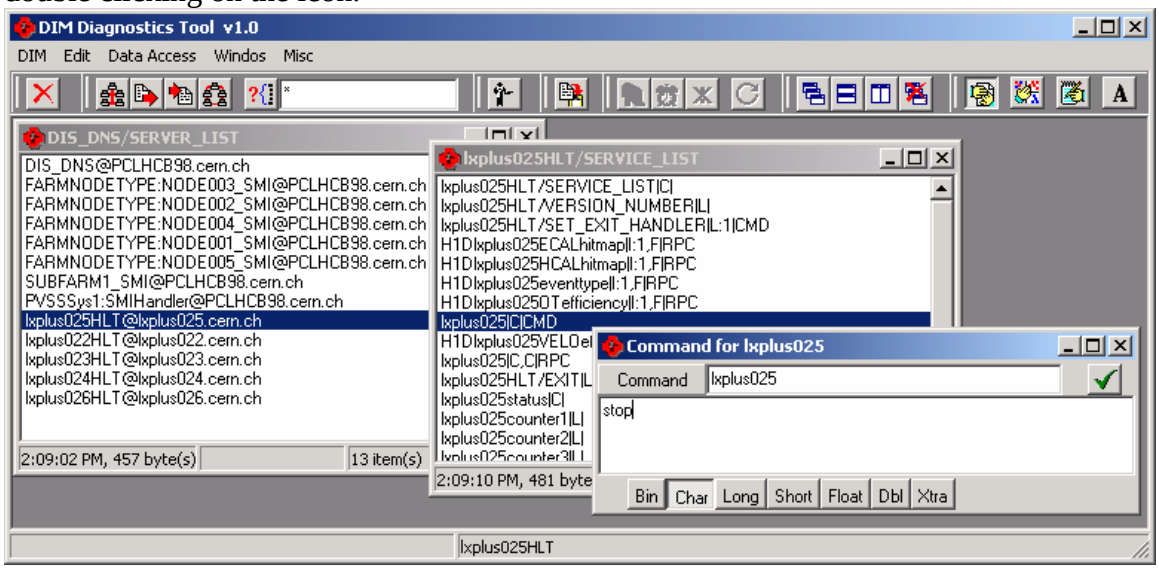
The third tab, labeled “Histograms” can be used to inspect which histograms exist on the transient store. Select a particular job as in the previous section and click on the job to see the titles of the histograms on the transient store:



By clicking on the title, you can dynamically publish it. It will take some time before the command reaches the job, the histogram is published, PVSS subscribes to the DIM service, and the job sends the values back to PVSS (can take about 20 seconds), so be patient!

Debugging with DID

If the PVSSDim server crashes, it is possible to send commands to the job using the Dim debugger, DID.
It can be found in the framework2.0.9.components/bin directory and is started by double clicking on the icon.



The servers are listed, to send a command to a job, click on the service nodename||C|CMD. A popup window will permit you to send the command you wish.
If you kill a job by simply killing its cmd window, the services will not be undeclared with the DNS and new jobs will not be able to start, unless you kill and restart the DNS.

References

For further references see:
http://lhcb-comp.web.cern.ch/lhcb-comp/ECS/Gaucho/default.htm