Everything you'd like to know about

Subdetector EXamine

but was afraid to ask

Examine for subdetectors

- determine the histograms you want to see
- find the data chunks containing the necessary data
- write a package to access your data and fill your histograms
- put your package in CVS

Creation of new package

- setup n32 (at least for the latest D0RunII versions)
- setup D0RunII *version_name*

  at the moment use t00.53.00
- setup d0cvs
- newrel -t *version_name directory_name*

  this will create a new directory tree
- cd *directory-name*

Creation of new package

- ctnewpkg -lob *package_name*

  to create directories and ctbuild files for the new package

```
my_examine
GNUmakefile   SUBDIRS       bin           my_examine  src
LIBDEPS       VERSION       doc           script
my_examine/bin:
BINARIES   LIBRARIES
my_examine/doc:
index.html
my_examine/my_examine:
my_examine/src:
COMPONENTS         OBJECT_COMPONENTS
```

Creation of new package: Files required by CTBUILD

- Write your code and put

  *.hpp in my_examine/my_examine

  *.cpp in my_examine/src
- edit VERSION, LIBDEPS, COMPONENTS, OBJECT_COMPONENTS and GNUmakefile

Creation of new package: Files required by CTBUILD

- VERSION - the current version of package. Change it every time you update package
- LIBDEPS - list the packages you directly call inside your package
- COMPONENTS - list the cpp files, which you want to include in the library
- OBJECT_COMPONENTS - object files stored outside library. Usually framework registration.

Creation of new package: Files required by CTBUILD

If you want to build executable

- cd my_examine/bin
- create cpp file (for example testMyExamine)- might be empty
- edit BINARIES, LIBRARIES and OBJECTS
  - BINARIES - list the cpp files which become executables (testMyExamine)
  - LIBRARIES - indicate additional the libraries needed by executable
  - OBJECTS - list the object file to link

LIBDEPS:

- geoman
- HepTuple
- rcp
- ErrorLogger
- histoscope

LIBRARIES:

- my_examine
- framework
- io_packages
- simpp_evt
- gtr_evt
- d0_mcpp
- stream_ds

GNUmakefile

```
# Generic GNU makefile for CTBUILD under D0 SRT.
#
# David Adams
# 21jan99
#
# Copy this file to GNUmakefile in the main package directory.
#
# For help with CTBUILD at D0, see
# http://www.bonner.rice.edu/adams/cvs/d0/ctbuild/doc/d0faq.html
#
override CPPFLAGS+=-I$$HISTO_INC
include ctbuild/srt/srt.mk
```

OBJECTS:

- RegMyExamine
- geometry_management
- ReadEvent
- LoadDSPACK
- LoadEVPACK

Examine and framework

- Examine for every subdetector is an "ordinary" framework package:
  - the corresponding class must inherit at least from fwk::Package and fwk::Process
  - you must provide the means to register the package with the framework
  - write rcp file for your package

Framework interface

```
class MyExamine: public fwk::Package, public fwk::Process
{
public:
    //  constructors/destructor
  MyExamine(fwk::Context* context);
  ~MyExamine();
    // Overridden hook methods
  fwk::Result processEvent(edm::Event &event);
    // Overridden package methods
  std::string packageName() const {return “MyExamine”;}
  static const std::string version( return “VXX.XX.XX”);
private:
 // what ever you need
};
```

Framework interface: header files

```
#include "framework/Package.hpp"              must be here
#include "framework/hooks/Process.hpp"
#include "framework/Result.hpp"
#include "framework/hooks/JobSummary.hpp"     you might want to include if derive from
                                              corresponding class
///
class MyExamine: public fwk::Package, public fwk::Process
{
public:
    //  constructors/destructor
  MyExamine(fwk::Context* context);
  ~MyExamine();
    // Overridden hook methods
  fwk::Result processEvent(edm::Event &event);
    // Overridden package methods
  std::string packageName() const {return “MyExamine”;}
  static const std::string version( return “VXX.XX.XX”);
private:
 // what ever you need
};
```

Framework interface

- put in the implementation (cpp) file of MyExamine class the line:

```
FWK_REGISTRY_IMPL(MyExamine,"$Name:$")
```

- create separate file RegMyExamine.cpp with just two lines:

```
#include "framework/Registry.hpp"
FWK_REGISTRY_DECL(MyExamine)
```

Framework interface: your package RCP

- Write rcp file

```
// Author: Mister X
// Date:   05/15/98
//
// $Id: my_examine.rcp,v 1.1 1999/04/09 16:10:44  Exp $
//
// rcp for my_examine package
//
   string PackageName = ”MyExamine"
// whatever your package needs. For example
//histogram file
string hbk_file=”MyExamine.hbook"
```

Framework interface: main RCP

```
string InterfaceName = "process"
string Interfaces = "generator decide builder filter modify analyze
   process jobSummary runEnd runInit"
string Flow = "generator decide builder filter modify process
   analyze dump"

// Packages

string Packages = "read geom cft_examine  CalExamine MyExamine"

RCP read = <ReadEvent.rcp>
RCP geom = <geometry_management.rcp>
RCP cft_examine = <cft_examine.rcp>
RCP CalExamine=<CalExamine.rcp>
RCP MyExamine=<my_examine.rcp>
```

Interface to histogramming packages

- We use HepTuple as a tool to define fill and store histograms
- Two options left for browser GUI:
  - HISTOSCOPE
  - ROOT

Interface to histogramming packages: header file

```
//  HepTuple classes
#ifdef ROOT_HIST
    class HepRootFileManager;
#else
    class HepHBookFileManager;
#endif
class HepHist1D;
class HepHist2D;
class HepHistProf;
class MyExamine: public fwk::Package, public fwk::Process
{
private:
#ifdef ROOT_HIST
    HepRootFileManager* _hbkMgr;
#else
    HepHBookFileManager* _hbkMgr;
#endif
 HepHist1D* _oneDimHistos[numOneDim];
 HepHistProf* _profileHistos[numProfDim];
 HepHist2D* _twoDimHistos[numTwoDim];
};
```

Interface to histogramming packages: constructor

```
MyExamine::MyExamine(Context* context):
  Package(context), Process(context), JobSummary(context)
{
     RCP rcp=packageRCP();  //  get RCP for MyExamine package
#ifdef ROOT_HIST
    _hbkMgr= new HepRootFileManager((rcp.getString("hbk_file")).c_str());
#else
    hs_initialize("My Examine Histograms");
    _hbkMgr= new HepHBookFileManager((rcp.getString("hbk_file")).c_str());
#endif
...
int nBin=rcp->getInt("hist0_low");
float low=rcp->getFloat("hist0_low");
float up=rcp->getFloat("hist0_up");
_oneDimHistos[0] = &  _hbkMgr->hist1D(“histogram # 0”,nBin,low,up);
...
hs_hbook_setup("//PAWC");
}
```

Interface to histogramming packages: filling

```
Result MyExamine::processEvent(Event &event)
{
  TKey<MyChunk> digiKey;
  THandle<MyChunk> digiChunk=digiKey.find(event);
//
  if( digiChunk.isValid()) // always check validity of the handle !!!
    {
    // fill your histograms here
     _oneDimHistos[0]->accumulate(myData);
    }
    else // do not use assertions!!!
   {
     error_log(ELwarning,"MyChunk missing")<<" MyExamine skips histogramming\n";
   }
...
  #ifdef ROOT_HIST
    _hbkMgr->write();
  #else
    hs_update();
  #endif
  return Result::success;
}
```

Interface to hisogramming packages: destructor

```
MyExamine::~MyExamine()
{
  delete _hbkMgr;
}
```

Geometry

- Please do not build geometry inside you package, use geometry_management package instead.
  - Add geoman to LIBDEPS
  - Add geometry_management (for future releases RegGeometryManagement) to BINARIES
  - you'll need all geometry rcp's. Must be provided by RCP database, but currently you have to copy them (or make links) to your bin area.

```
cp $BFDIST/releases/$BFCURRENT/geometry_management/rcp/*.rcp .
cp $BFDIST/releases/$BFCURRENT/geometry_management/tests/*.rcp .
```

Committing package in cvs

- Ask the D0 release managers (d0-release-mgr@fnal.gov) to make a cvs directory for your package
- Make sure you tell them who you want to have the write privilege.
- newrel -t t00.\#\#.00 your-temp-area-name
- cd your-temp-area-name
- cvs checkout your-package-name

Committing package in cvs

- cp -R top-directory-with-your-codes-are-stored your-package-name

  This will copy the entire directory structure of the directory in which your development code are stored in.
- Repeat cvs-add for all directories first.
- For each file cvs add your-file.
- cvs -n update to check if somebody else modified code
- cvs commit At this stage you may want to setenv EDITOR to your favorate editor, because the cvs automaically invokes an editor for logging purpose.
- cttag tag will be taken from VERSION
- check that you do not miss something. The better way is to do this to checkout package in new area and to build it.

How to build and run cft_examine

- open two xterm windows
- in both windows:
  - setup n32
  - setup D0RunII t00.53.00
- in first window:
  - setup d0cvs
  - newrel –t t00.53.00 testExamine
  - cd testExamine
  - addpkg –h cft_examine

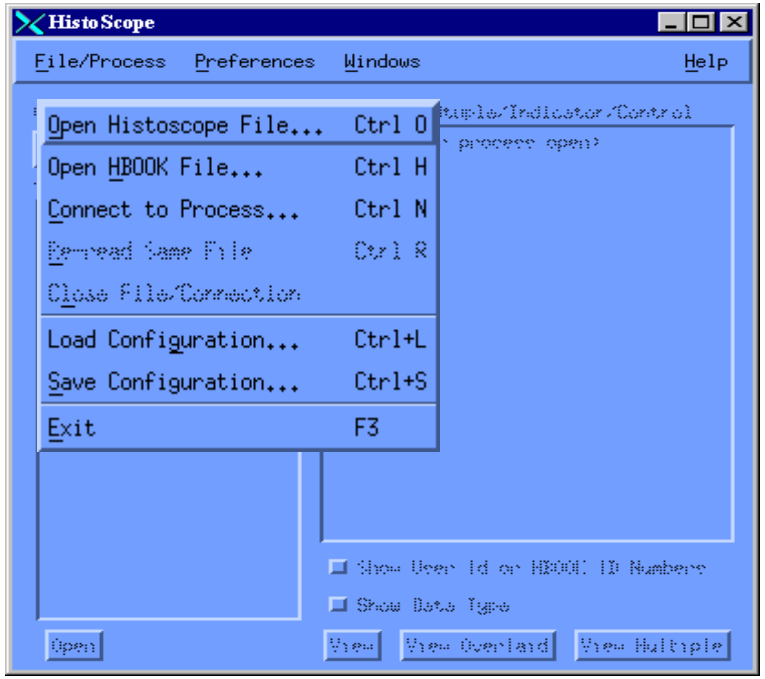
How to build and run cft_examine

- gmake
- cd bin/IRIX6n32-KCC_3_3/
- prepareRCP.sh
- testCftExamine –rcp framework.rcp

- in second window:
  - setup histo v5-0-2a-0
  - histo

How to build and run cft_examine

- When histoscope window will appeared go to file/process menu and click connect to process

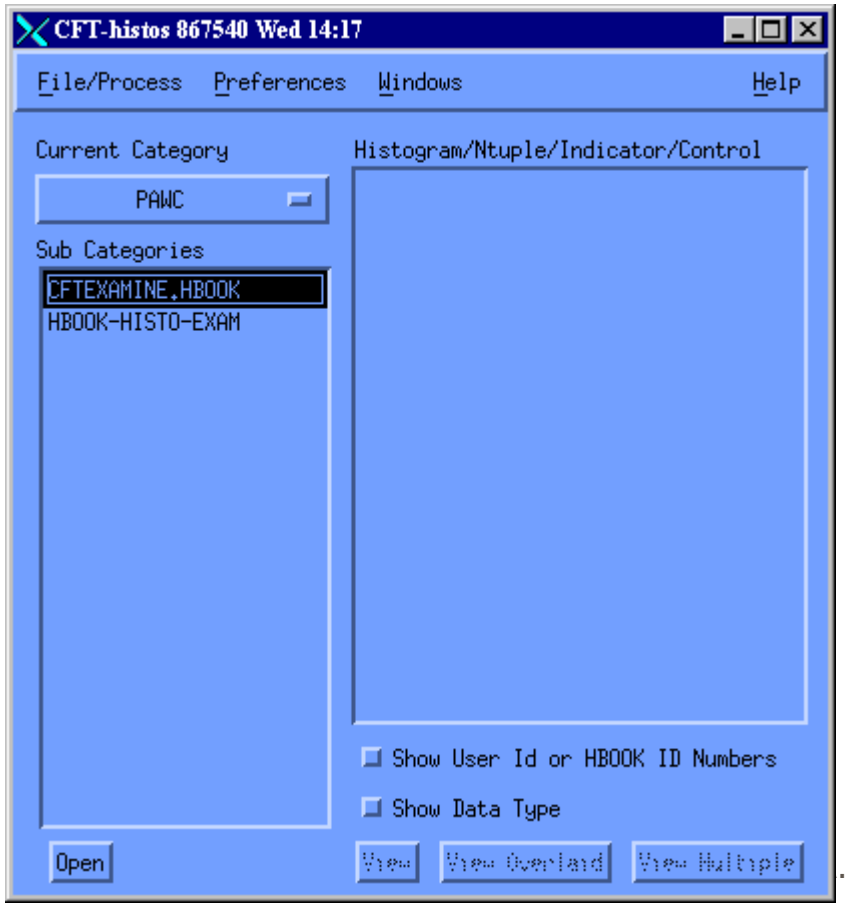


How to build and run cft_examine

- Choose the process to connect

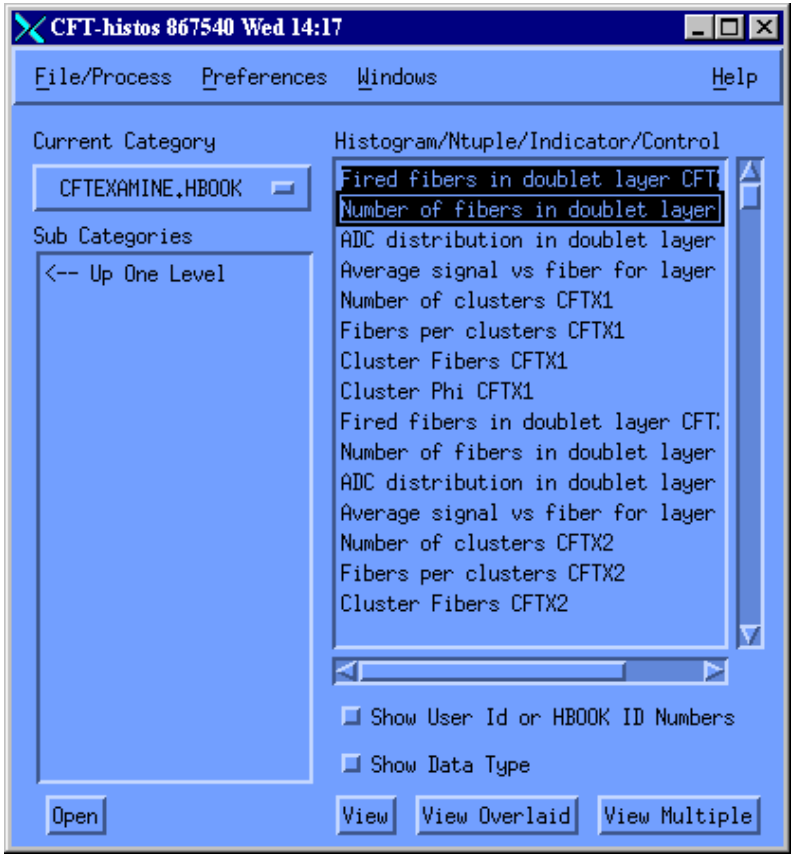
How to build and run cft_examine

- Choose the file with histograms and click open



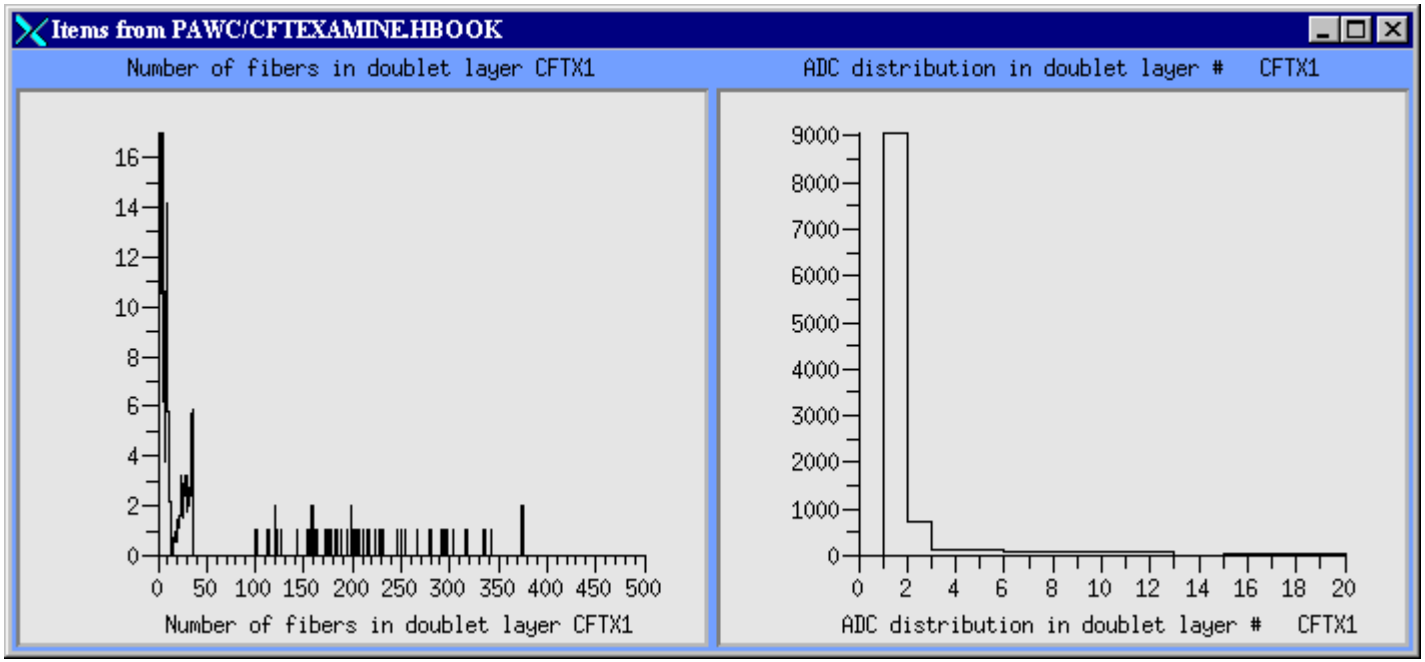
How to build and run cft_examine

- Select one or more histograms and click one of three view buttons:



How to build and run cft_examine

- enjoy your histograms



Useful links

- http://www-d0.fnal.gov/software/cmgt/getting_started_at_d0.html
- http://www.bonner.rice.edu:80/adams/cvs/d0/ctest/doc/
- http://www-d0.fnal.gov/d0dist/dist/packages/