

Examine GUI



Root Framework

Root GUI

Java GUI

Root Framework



- ⌘ Interactive Framework via GUI
- ⌘ Uses Root to communicate with the GUI (Comm & Xserv classes)
- ⌘ Uses Root to store histograms
- ⌘ Receives commands from GUI
- ⌘ Sends histograms and status messages to GUI
- ⌘ Serves Multiple clients

Using Root Framework



- ⌘ Link with `root_framework` instead of `iframework` library
- ⌘ Link with main object `rframework.o` instead of `tframework.o` or `iframework.o`
- ⌘ Histogram initialization and booking
- ⌘ Sending messages to the GUI

Histogram Init. & Booking

```
#include "root_framework/Xserv.hpp"
```

```
Xserv *xserv = Xserv::instance();
```

```
_hbkMgr = new  
HepRootFileManager((root_file).c_str(),  
                    HepFileManager::REFRESH);
```

```
int catid = xserv->Add(_hbkMgr);
```

```
xserv->SetOption(catid, hname, "lego1");
```

```
xserv->SetOption(catid, hid, "lego1");
```

Sending Messages to GUI

To send `itc::String_message` to GUI which displays it asynchronously in a pop up window:

```
#include "root_framework/gstream.hpp"
if(!fmod(_count, ERMOD)) {
    ostringstream evmss;
    evmss << "Examine::processEvent: event #"
        << _count;
    itc::String_Message evmess = evmss.str();
    cgui << evmess << ends; // null byte
    cgui.flush();
}
```

Root & edm



Both define class Tkey. A statement like:

```
using namespace edm;
```

will have unexpected results if Root's Tkey class is called in its scope. Be explicit, for example:

```
edm::Tkey<CalDataChunk> calKey;
```

Root GUI



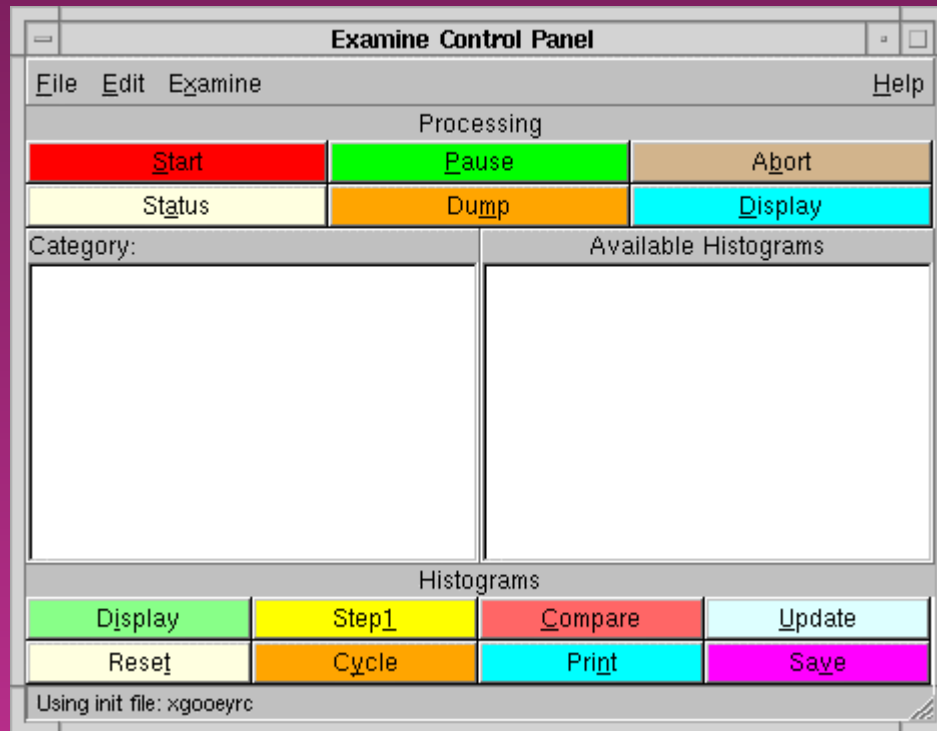
- ⌘ Creates and controls an Examine process
- ⌘ Observes data by requesting and displaying histograms from Examine
- ⌘ Only 1 creator, many Observers
- ⌘ Socket Communications => Remote location
- ⌘ User preferences & Examine choice
- ⌘ Examine startup script
- ⌘ DAQ or Root file input

Root GUI Features

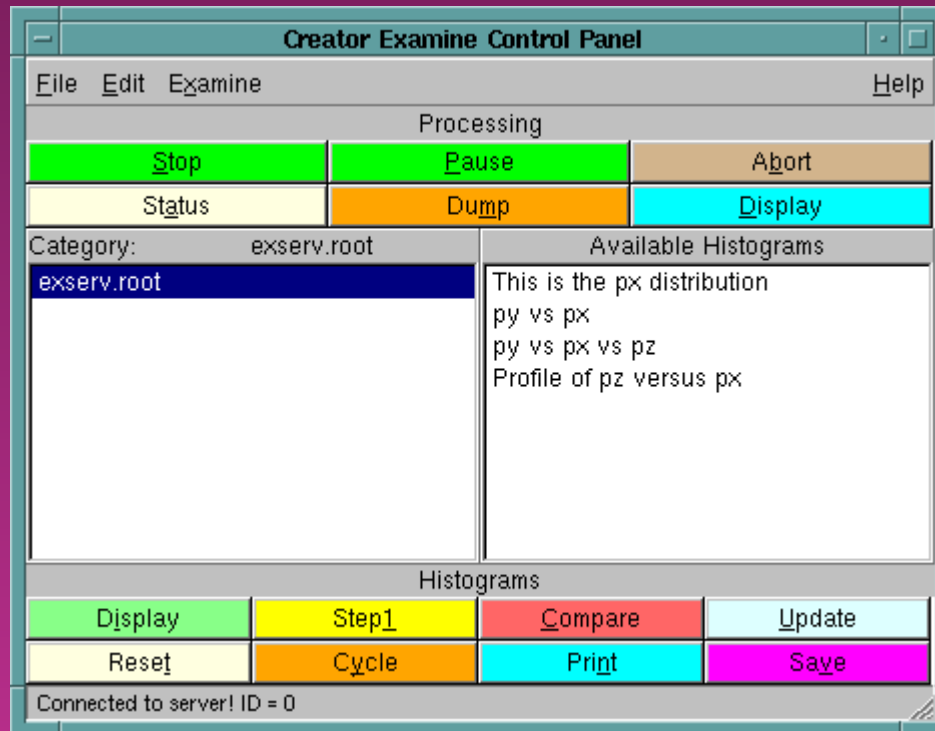


- ⌘ Start/Stop, Pause/Resume
- ⌘ Abort, Status, Dump, Display
- ⌘ Choose category, select histograms
- ⌘ Display, Step, Cycle, Update
- ⌘ Print, Save, Reset, Compare
- ⌘ Help & Status bar

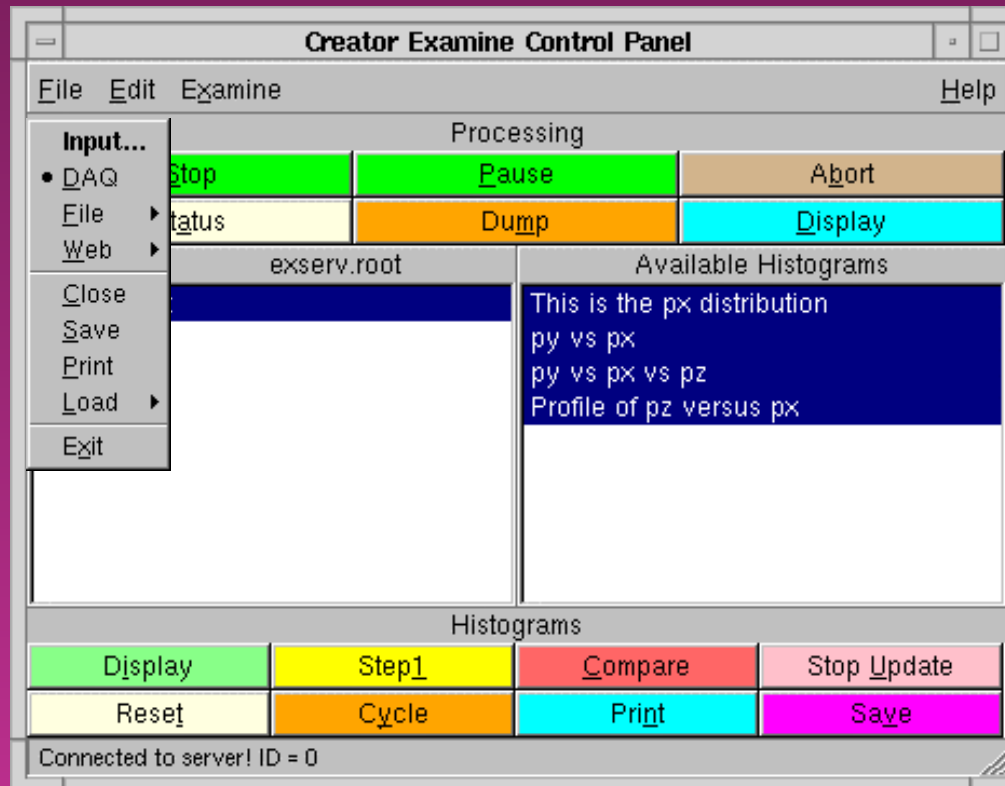
Examine Control Panel



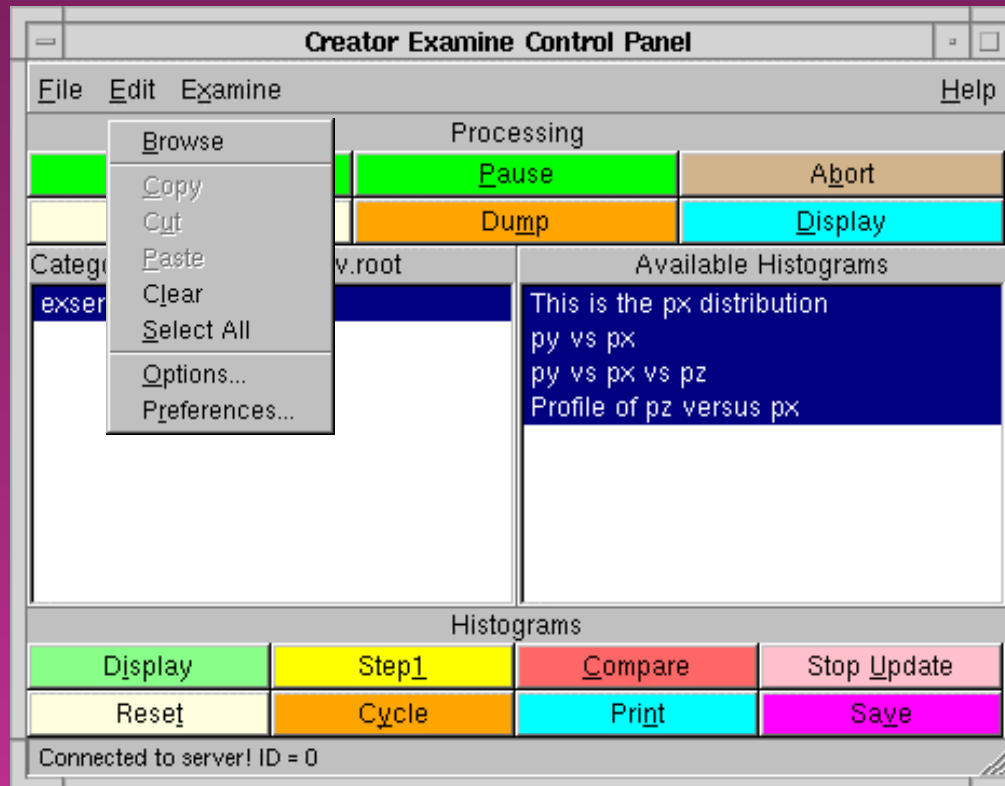
Examine Control Panel



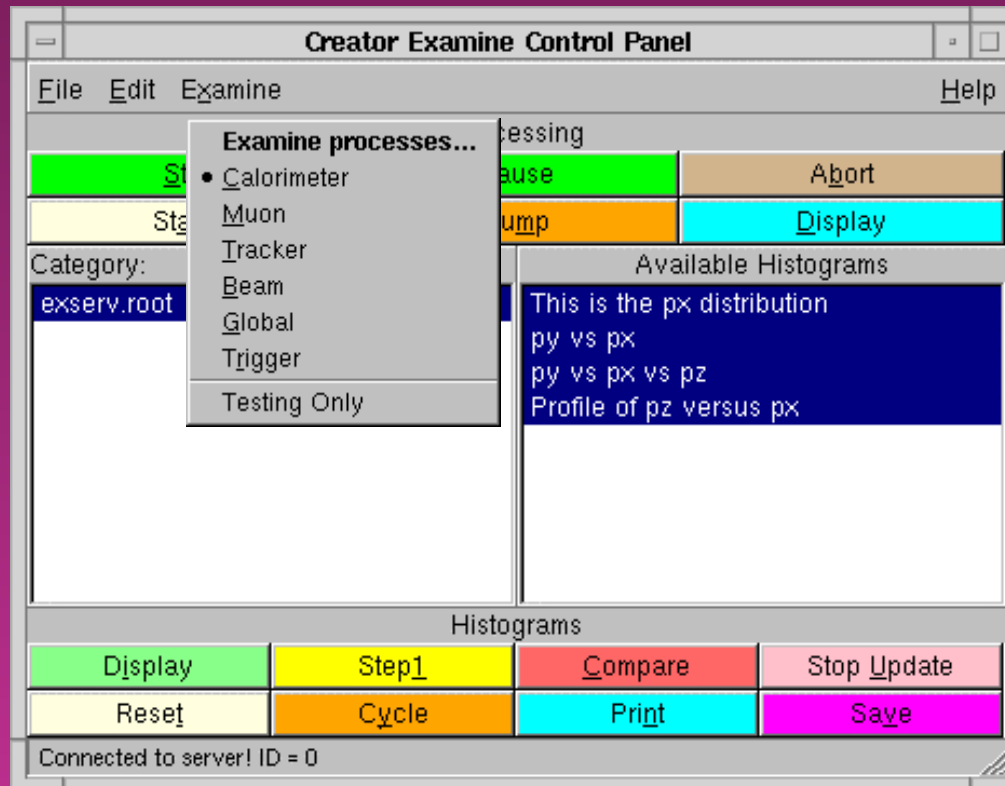
Examine Control Panel



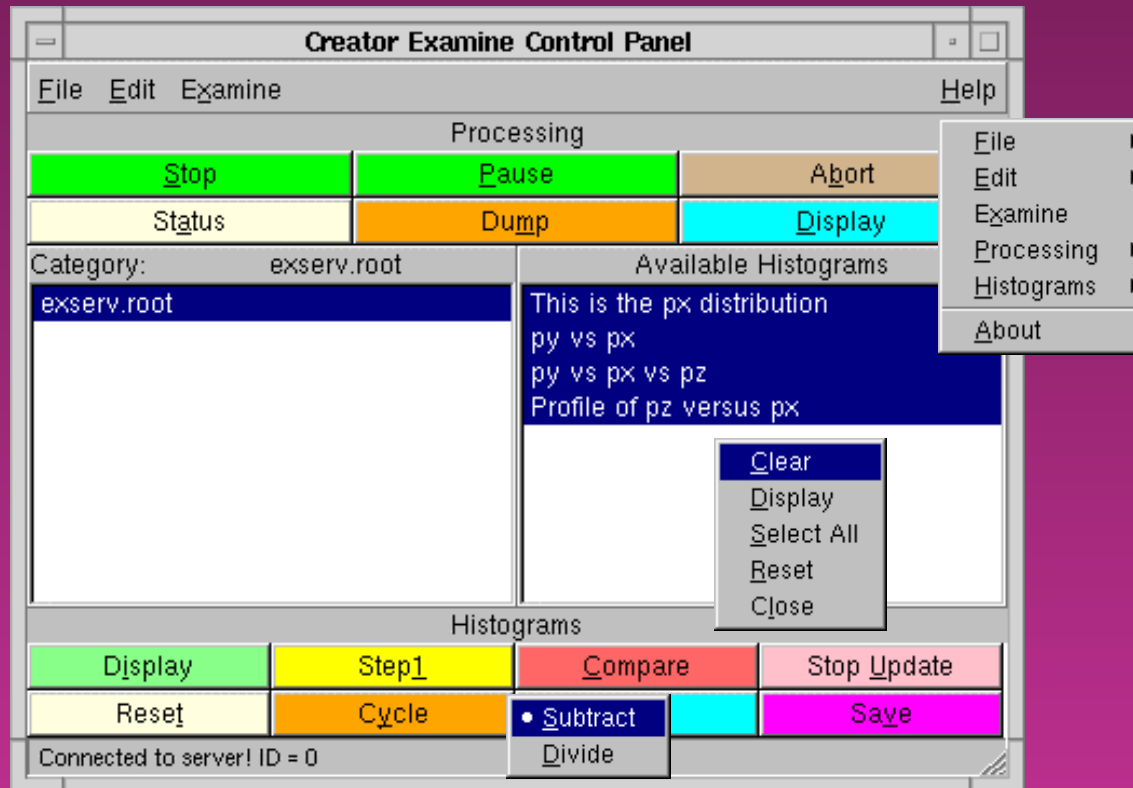
Examine Control Panel



Examine Control Panel



Examine Control Panel



Examine Control Panel

ECP Preferences

Display | Examine

Histogram Grid

X: 2 Y: 2

Cycle/Update period (s): 2.0

Flashable Buttons

Audible Warnings

Ok Cancel

This screenshot shows the 'Display' tab of the 'ECP Preferences' dialog box. It features a 'Histogram Grid' section with two dropdown menus for 'X' and 'Y', both set to '2'. Below this is a text field for 'Cycle/Update period (s)' set to '2.0'. At the bottom, there are two checked checkboxes: 'Flashable Buttons' and 'Audible Warnings'. The dialog has 'Ok' and 'Cancel' buttons at the bottom.

ECP Preferences

Display | Examine

Server Address: localhost

Remote Start Script: /home/snow/cal_ex.sh

Calorimeter

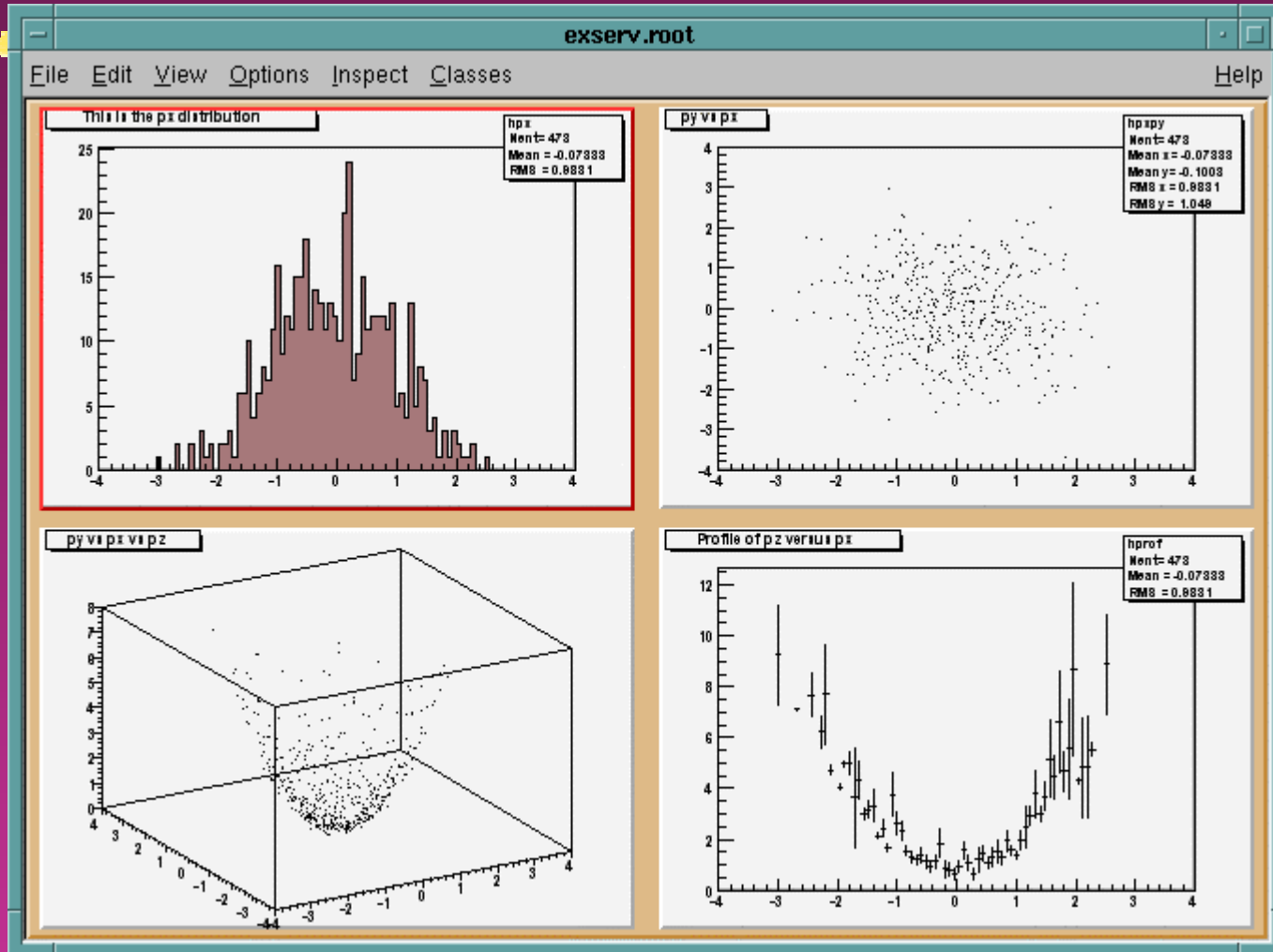
Server Port: 52105

RCP File: cal.rcp

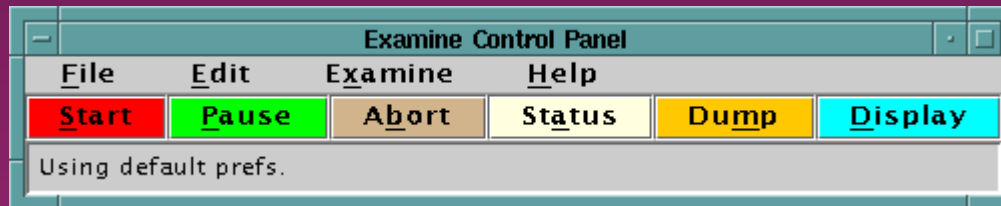
Ok Cancel

This screenshot shows the 'Examine' tab of the 'ECP Preferences' dialog box. It contains several text input fields: 'Server Address' (localhost), 'Remote Start Script' (/home/snow/cal_ex.sh), 'Server Port' (52105), and 'RCP File' (cal.rcp). The 'Calorimeter' section is currently collapsed. The dialog has 'Ok' and 'Cancel' buttons at the bottom.

ECP Histogram Canvas



Java GUI



- ⌘ Works with root_framework Examine
- ⌘ Same as Creator mode of Root GUI with only control functions, no histogram functions
- ⌘ Runs with Java VM v1.2 or better (Win32 or Linux)
- ⌘ 34 kb Jar file runnable from MSDOS mountable floppy (Win32 or Linux)

Examine GUI To Do



- ⌘ Implement process registry
- ⌘ Event Dump, Abort, Status, Event Display
- ⌘ Jgooney branch to operate with Root less iframework using CORBA
- ⌘ Add histogram rendering to Jgooney