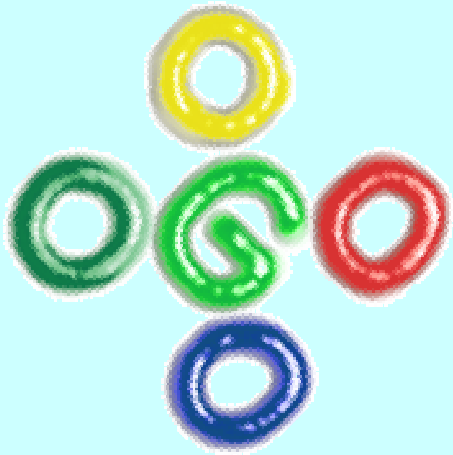


GO4 Version 4.4

J.Adamczewski-Musch, H.G.Essel, S.Linev

Lectures

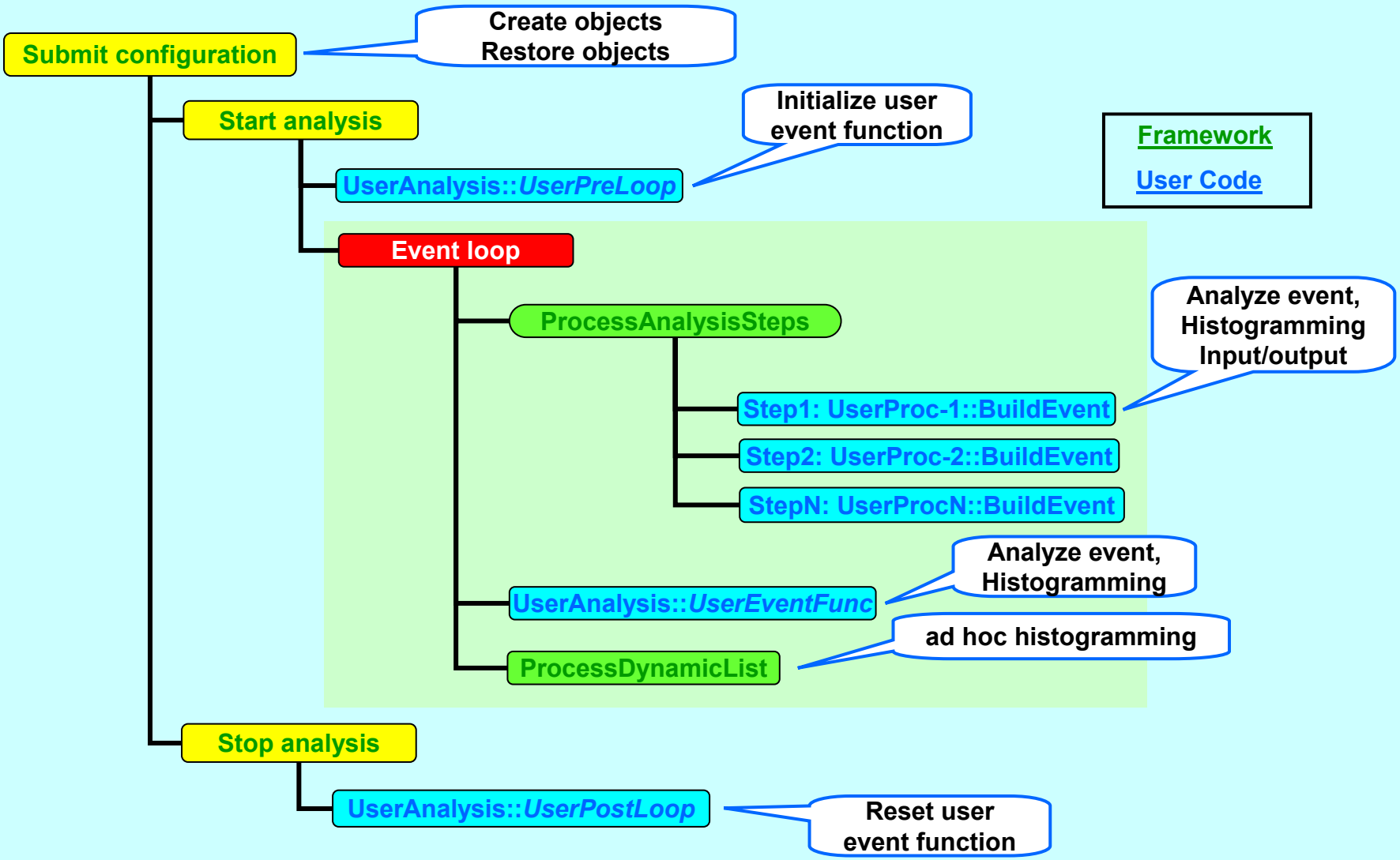
Steps, event and processor objects
are typically created in the code (MainUserAnalysis or UserAnalysis)
Now, the steps must be configured:

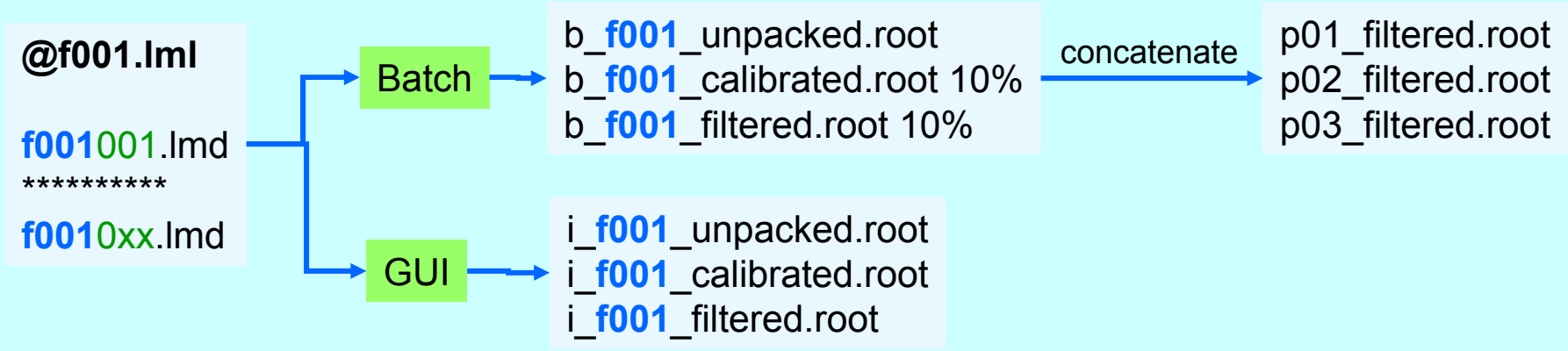
- **Enable/disable steps**
- **Specify file names for event source and store**
- **Enable file reading/writing**
- **Specify overwrite mode**

Autosave filename, interval must be specified.

There are several methods, how this can be done.

Go4 event loop





Data source and data store directories from shell variables

Order of setups

There are several methods to configure the analysis which can be combined in a defined order:

Constructor of user analysis class or MainUserAnalysis

All steps must be created.
One may use user arguments given by go4analysis command (behind **–x**) or in the **Launch analysis** panel (**Args**).

Batch: optional Go4 preferences file

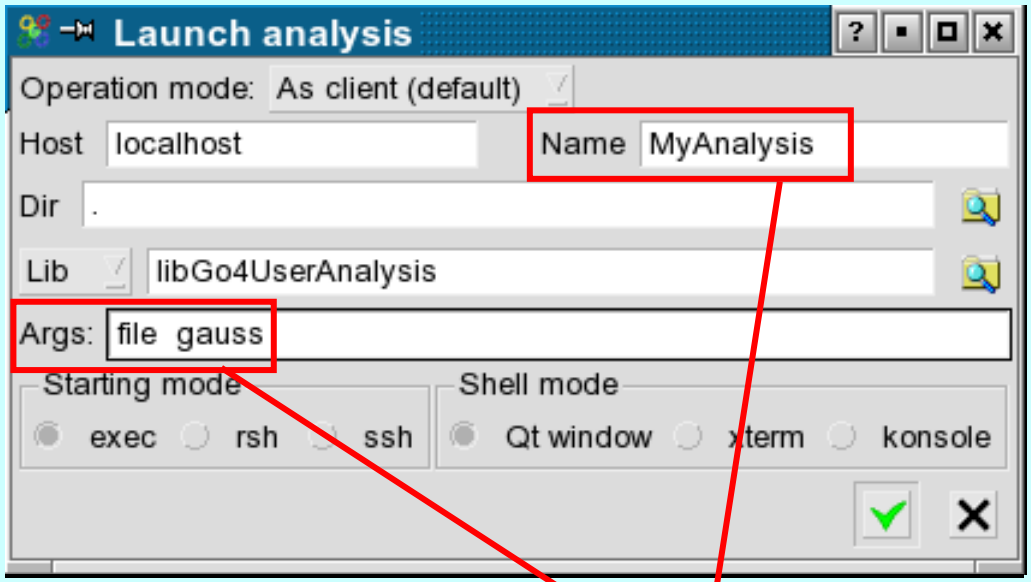
With –prefs <file> one can enable loading a preference file.
Then the arguments of go4analysis (before **–x**) overwrite the settings.

GUI: Go4 preferences file

When launched from GUI settings are loaded from Go4AnalysisPrefs.root.
With hotstart file the complete setting from that file is used and overwrites the settings.

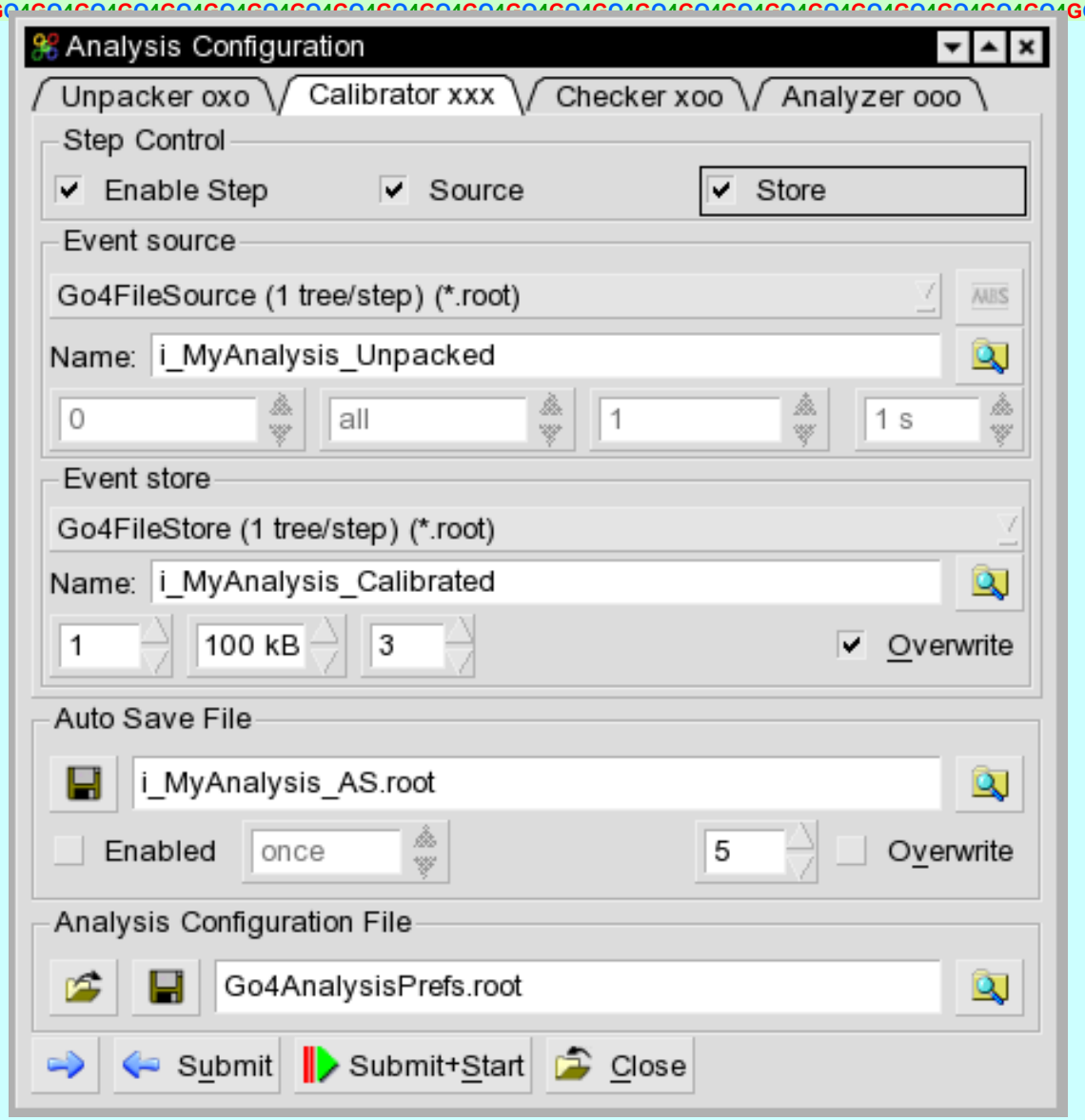
To print final setup in UserAnalysis::*UserPreLoop()*
{ Print(); }

GUI: Launch panel arguments



TUserAnalysis::TUserAnalysis(int argc, char argv) : TGo4Analysis(argc, argv)**

GUI: Configuration panel



With standard go4analysis

go4analysis -number <n> -[no]prefs [<name>] -asf <auto-save file>

-name <xxx>

-step UnPack -disable-step

-step Analysis -source <file>

-x <user argument list>

TUserAnalysis::TUserAnalysis(int argc, char argv) : TGo4Analysis(argc, argv)**
gets user argument list.
argv[0] is name <xxx>

Example1Step uses this name to build output file name
Default for input is coded, but overwritten by command line

User specific MainUserAnalysis programs
process their specific argument lists!

MainUserAnalysis or
UserAnalysis constructor: create event, processor, and step objects

```
TGo4StepFactory*  cali  = new TGo4StepFactory("CaliFact"); // any name
cali->DefEventProcessor("Calibrator","CaliProc");  // object name, class name
cali->DefInputEvent("Unpacked","UnpackEvent");  // object name, class name
cali->DefOutputEvent("Calibrated","CaliEvent");     // object name, class name
TGo4AnalysisStep* calistep = new TGo4AnalysisStep("Calibrator",cali,0,0,0);
AddAnalysisStep(calistep);
```

Factory is a utility class

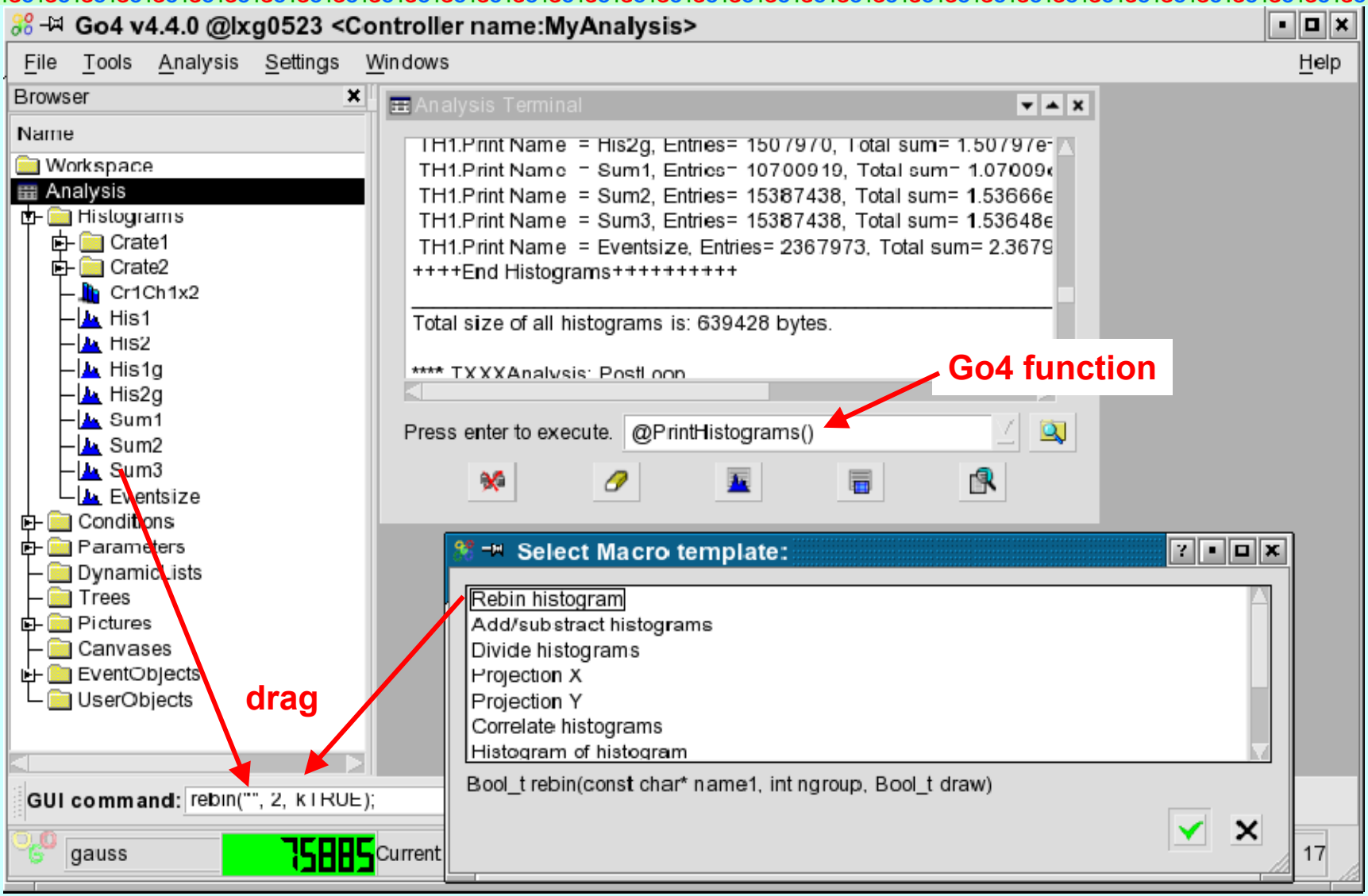
User setup macro: configure step, get a name as argument

```
// build filenames from argument

step = go4->GetAnalysisStep("Calibrator");
step->SetProcessEnabled(true/false);
step->SetEventStore(new TGo4FileStoreParameter(filename,...));
step->SetStoreEnabled(true/false);
// default event source is output event from previous step
if(<previous step disabled>) // event must be read from file by Go4
   step->SetEventSource(new TGo4FileSourceParameter(filename));
step->SetSourceEnabled(true);
```

Example2Step uses this mechanism

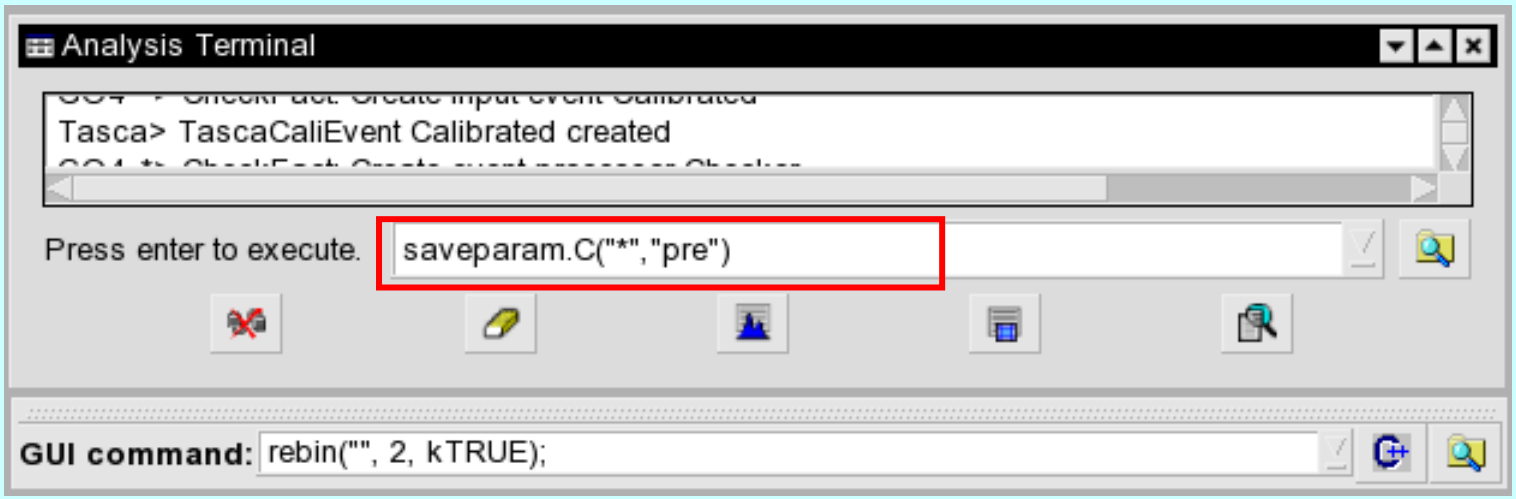
Macro execution



Access objects in analysis macro

```
// If histograms, parameters and conditions are in default folders, use:
MyParam *x =          go4->GetParameter("Param1"); // looks in folder  "Parameters" and subfolders
TGo4WinCond *wc = go4->GetCondition("cond1");     // looks in folder  "Conditions" and subfolders
TH1 *s =              go4->GetHistogram("histo1");    // looks in folder  "Histograms" and subfolders
// If histograms, parameters and conditions are in user folders, use:
MyParam *x = go4->GetObject("Param1"); // looks in all folders and subfolders
```

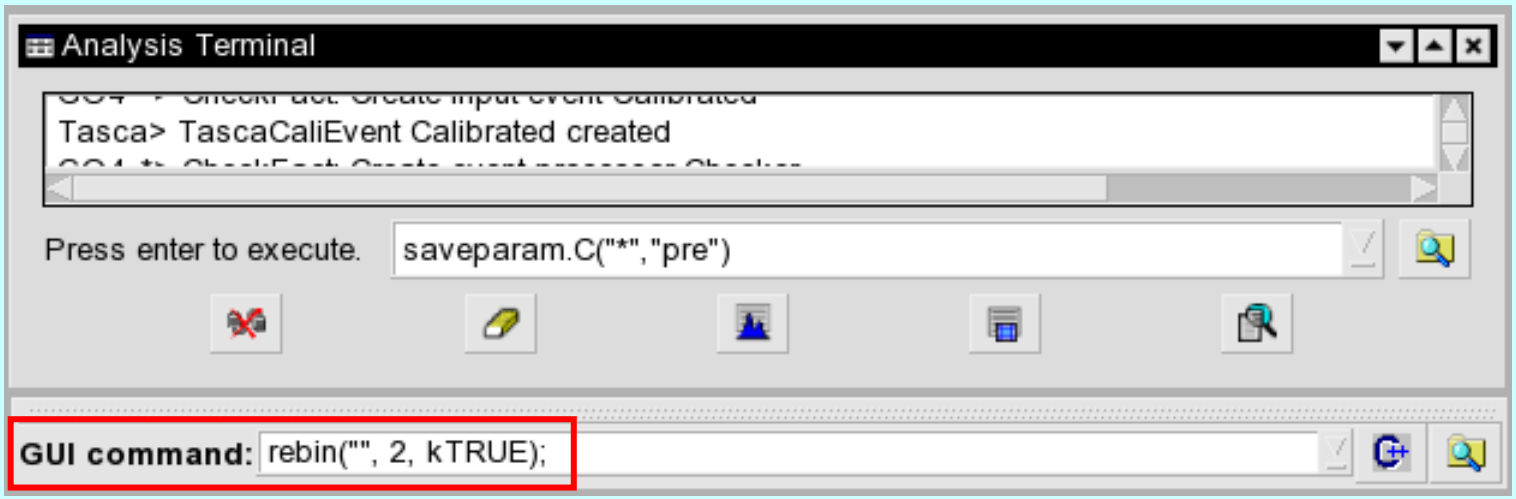
Called in user code (analysis or processors) or from GUI
gROOT->*ProcessLine*(".x macro.C()");



Access objects in GUI macro

```
TString fullname1 = go4->FindItem("histo1");
TH1* his=(TH1 *)go4->GetObject(fullname1,1000); // 1000ms timeout to get object from analysis
```

Called in GUI



Macro working in all environments

```
#ifdef __GO4MACRO__
// macro runs in GUI
TString fullname1 = go4->FindItem("histo1");
TH1* his=(TH1 *)go4->GetObject(fullname1,1000); // 1000ms timeout to get object from analysis
#endif

#ifdef __GO4ANAMACRO__
// macro runs in analysis
MyParam *x = go4->GetObject("Param1"); // looks in all folders and subfolders
#endif

#ifdef __NOGO4MACRO__
// macro runs from ROOT shell
TFile *f = TFile::Open("file.root","r"); // use standard ROOT to locate object in file
#endif
```