Go4 Workshop May/June 2011 Schedule

Time	Action	Tutors
10:00 – 10:30	Go4 V4 Overview (presentation) Main Features Runtime environment and event loop Analysis framework Introduction to GUI	Jörn Adamczewski-Musch
10:30- 12:00	Working with Go4 GUI (practice) Go4 browser Viewpanel and graphic options Analysis client/server control Conditions and markers Remote object monitoring Hotstart	Sergei Linev
12:00- 13:00	LUNCH break	
13:00- 13:30	Analysis in batch mode (practice) The go4analysis executable	Sergei Linev
13:30- 15:00	Go4 Analysis code (practice) How to edit and compile the user analysis Analysis, Processor and Event classes How to define new histograms/conditions/... Unpacking of Mbs events Structuring the analysis code	Jörn Adamczewski-Musch
15:00- 15:30	COFFEE break	
15:30- 16:00	Go4 Advanced Features (presentation) Analysis Steps Go4 Event Store with ROOT Trees Dynamic Histograming Go4 Fitter	Sergei Linev
16:00- 17:30	Go4 Advanced Features (practice) How to add an analysis step Multistep analysis setup How to work with Parameters How to use macros How to implement user defined event sources	Jörn Adamczewski-Musch
17:30- 18:00	Discussion and Questions Special topics on request	
	CLOSING	

Working with Go4 (practice)

1 Preparation

1.1 Log in with gsi account or guest account "dvgast01"

1.2 Set go4 working environment

```
[shell] . go4login head
```

1.3 Create working directory

On normal GSI linux account create directory like "/u/user/go4workshop/ "

```
[shell] cd
[shell] mkdir go4workshop;
[shell] cd go4workshop
```

On guest account, create directory with your name inside like "/u/dvgast01/Go4workshop2011/MyName/ "

```
[shell] cd ~/Go4workshop2011
[shell] mkdir MyName
[shell] cd MyName
```

1.4 Copy example analysis:

```
[shell] cp –r $GO4SYS/Go4Example2Step .
[shell] cd Go4Example2Step
```

1.5 Compile example:

```
[shell] make clean
[shell] make
```

1.6 Startup Go4 GUI:

```
[shell] go4
```

1.7 Launch example analysis:

Select “Analysis -> Launch analysis” menu. In dialog window specify:

Host:	localhost
Name:	MyName
Dir:	<directory with analysis> (can be selected via file browser)
Lib:	analysis library name (field can be left empty)
Args:	<empty>
Starting mode:	exec
Shell mode:	Qt window

Press “Start analysis client” button (green tick) to start analysis process.

1.8 Configure analysis:

In “Analysis configuration” dialog change following fields:

Event source: MBS Random

Press “Submit+Start” button to run analysis.

1.9 Generate hotstart file:

When analysis running, generate hotstart file by selecting menu “Settings->Generate hotstart”. Specify filename for hotstart file (like mygo4) and press “Save” button.

1.10 Analysis start with hotstart:

Next time one could start complete analysis from the shell with only command:

```
[shell] go4 mygo4.hotstart
```

1.11 Available in workshop event sources:

MBS Random:	build in go4 random event generator
MBS Files:	/s/go4/Workshop11/*.lmd
MBS Stream Server:	depcp002

2 Working with analysis in GUI

2.1 Start/stop analysis

2.2 Submit/Close analysis

2.3 Browser content

2.4 Display/refresh/monitor objects from analysis

2.5 View panel and different draw options

2.6 Conditions

2.7 Picture

2.8 Workspace, export objects

3 Working with analysis in batch

3.1 Get arguments list with –h option:

```
[shell] go4analysis -h
```

3.2 Run analysis with different event sources:

```
[shell] go4analysis -random
[shell] go4analysis –stream depcp002
[shell] go4analysis –file "/s/go4/Workshop11/*.lmd"
[shell] go4analysis –file /s/go4/Workshop11/gaussfiles.lml
```

Hint: running analysis can be always normally stopped by Ctrl-C pressing.

3.3 Produce histograms over first 10000 events:

All histograms, filled by analysis, saved in autosave file. If one enable it, histograms can be inspected later with go4 gui:

```
[shell] rm –f asf.root
[shell] go4analysis –random –asf asf.root –number 10000
[shell] go4 asf.root
```

Same can be done with any other event sources.

3.4 Enable output for the first step:

One very often use batch mode to convert original data into ROOT tree format, which can be analyzed directly with ROOT macros. To enable store:

```
    [shell] go4analysis –stream depcp002 –number 10000 –store
output.root
```

Created ROOT tree can be browsed with standard ROOT browser:

```
    [shell] root –l output.root
    [root] new TBrowser
  <select tree in browser, double click on leaf to fill histogram>
    [root] .q
```

3.5 Enable output for second step:

By default all arguments applied for the first analysis step. To specify arguments for other step, this step should be explicitly selected. Lets try to enable output of second step:

```
    [shell] go4analysis –stream depcp002 –number 10000 –step
Analysis –store output2.root
```

And to produce output for both events simultaneously:

```
     [shell] go4analysis –stream depcp002 –number 10000 –step
Unpack –store output1.root –step Analysis –store output2.root
```

3.6 Printout of event source data:

Sometimes it is useful to see original MBS data as is. For that just:

```
    [shell] go4analysis –stream depcp002 –print hex
```

For printout of MBS events no user analysis is required. More info about printout command one can get with:

```
    [shell] go4analysis –help print
```

3.7 Run analysis in server mode (advanced):

Analysis can be started as server from batch and than monitored/steered from the GUI.

```
    [shell] go4analysis –random -server
```

On debug output one should see message:

Waiting for client connection on PORT: 5000

Here is important port number 5000 (can be 5001, 5002 and so on). This port number should be specified when connecting to analysis from the GUI:

```
[shell] . go4login head
[shell] go4 –observer localhost 5000
```

Connection arguments can be specified also via gui menu command “Analysis -> Connect to running server”. With same command one could connect to any running analysis of your colleague - just type host name of remote computer instead of localhost. GUI can also be connected with administration privileges, which allows reconfiguring and shutdown of analysis:

```
[shell] go4 –admin localhost 5000
```

4 Exercise: Rename and modify analysis

4.1 Rename existing code:

`$GO4SYS/build/rename.sh XXX MyName`

4.2 Modify analysis class to disable second step

Start editor: `kate TMyNameAnalysis.cxx`
Commment definition of second step
Recompile: `“make”`

4.3 Start Go4 GUI with modified analysis

What has changed?:

5 Exercise: Add New histogram in first step

5.1 Add histogram pointer in header file:

`kate TMyNameUnpackProc.h`

5.2 Add Add histogram definition and fill action:

`kate TMyNameUnpackProc.cxx`
Use “MakeTH1” statement in constructor on new pointer
Fill new histogram in BuildEvent
Recompile: `“make”`

5.3 Start Go4 GUI with modified analysis

What has changed?:

6 Exercise: Add New condition in first step

6.1 Add condition pointer in header file:

`kate TMyNameUnpackProc.h`

6.2 Add condition definition and fill action:

`kate TMyNameUnpackProc.cxx`
Use "MakeWinCond" statement in constructor on new pointer
Use new condition in BuildEvent to set a gate for new histogram
Recompile: "`make`"

6.3 Start Go4 GUI with modified analysis

What has changed?: Modify condition during monitoring analysis

7 Exercise: Arrays of histograms and condition in first step

7.1 Extend previous histogram and conditions to array over channel number in header file

`kate TMyNameUnpackProc.h`

7.2 Modify definition, test and fill action

`kate TMyNameUnpackProc.cxx`
Use "*Form*" statement in constructor to label each object with channel index
Use channel index in BuildEvent to test/fill the correct condition/histogram
Recompile: "`make`"

7.3 Start Go4 GUI with modified analysis

Modify condition(s) interactively

8 Exercise: Picture of new histograms and condition in first step

8.1 Combine new histograms and conditions into a TGo4Picture object

Each Condition should be in same pad as corresponding histogram
Take picture "condSet" as example for dividing the pads and setting graphic options
Use "for" loop over pad coordinates (2x4) to match channel numbers (0...7) in definition

8.2 Start Go4 GUI with modified analysis

How does picture look?
Modify embedded conditions in picture with marker editor

8.3 Set up the same picture view manually in GUI and save as hot start script

Open new viewpanel; use canvas divide tool to get 2x4 divisions
Drag and drop histograms to corresponding pads
Drag and drop conditions to corresponding pads
Generate hotstart script with new “Picture”
Test hotstart setup by shutdown and restart gui

9 Exercise: New variables in output event

9.1 Edit output event of first step

Open with editor: `kate TMyNameUnpackEvent.h` and `TMyNameUnpackEvent.cxx`
Declare Int_t variables for each crate to contain sum of all channels values
Implement `Clear()` for new member variables

9.2 Edit unpack processor to fill value

For each mbs event, accumulate sums and put into output event variables

9.3 Compile and run Go4

Build modified analysis “`make`”
Start Go4 in batch or GUI mode with output tree enabled
After end of processing, inspect tree with ROOT or Go4 viewer. Are new variables available?

9.4 Dynamic histograming from output event

Start Go4 in GUI mode. Configure step 1 with output tree
Use online tree viewer to define dynamic histogram from tree.
Monitor new histogram during Go4 run

9.5 Process values in second analysis step

Activate second analysis step in `TMyNameAnalyis.cxx`
Define and implement new histograms in second step to be filled from new variables

10 Exercise: Partial step analysis

10.1 Run first step without second step

In analysis set up (GUI or go4analysis batch mode) disable second step and switch on tree output of first step. Run from sample file.

10.2 Run second step without first step

In analysis set up (GUI or go4analysis batch mode) disable first step and enable second step. Use previous tree output of first step as second step event source.

10.3 Run both steps without intermediate tree

Compare speed (ev/s) for the three cases. Any influence of tree parameters? What about file size (lmd format, intermediate tree file)?

11 Exercise: Use Parameter object

11.1 Add Calibration values to Parameter class

Edit `TMyNameParameter.h , .cxx` and define different polynomial calibration coefficients (2^{nd} order) for each of the 8 channels of crate 1
Define Function `TMyNameParameter::DoCalibration(ch, adc)` that calculates energy from adc values for each channel.
Define a boolean switch "`recalibrate`" to indicate that calibration was changed by user

11.2 Add calibrated histograms to second step

Edit `TMyNameAnalysisProc.cxx .h`: Add new calibrated histograms for 8 channels of crate 1 array. Hint: Use function `InitCalibratedSpectra` to recreate these histograms based on the calibration values in parameter. Use `DoCalibration` function of parameter to set histogram boundaries!
Use Parameter calibration function to fill calibrated histograms from raw data.
Provide mechanism to recreate calibrated spectra with changed limits whenever the "recalibrate" switch in parameter object becomes true!

11.3 Compile and run example

View parameter object in GUI editor. Modify values and watch effect.

11.4 Save and recover parameter values

After changing inital values, produce an analysis macro to set current values.
Recover values by means of the `set_all.C` macro in analysis.

11.5 Improve Calibration function

Introduce "Randomize" function into parameter class to smear digital rounding artefacts at bin edges.

12 User defined event source

12.1 Copy and run Go4ExampleUserSource

Copy `$GO4SYS/Go4ExampleUserSource` to local work directory.
Rebuild "`make clean; make`"
Run from GUI with user source file
Look into code `TYYYEventSource` and `TYYYRawEvent`
Optionally discuss other event sources