

User Interface

Presented by [Sébastien Incerti \(CNRS/IN2P3\)](#)

Based on slides kindly prepared by [Makoto Asai \(SLAC\)](#)

Geant 4

Contents

- Syntax of **U**ser **I**nterface commands
- Macro files
- Terminal commands

1) Syntax of **User Interface** commands

Geant4 UI command

- A **command** consists of
 - Command directory
 - Command
 - Parameter(s)
- A **parameter** can be a type of string, boolean, integer or double.
 - Space is a delimiter.
 - Use double-quotes (“”) for string with space(s).
- A parameter may be “omittable”. If it is the case, a default value will be taken if you omit the parameter.
 - Default value is either predefined default value or current value according to its definition.
 - If you want to use the default value for your first parameter while you want to set your second parameter, use “!” as a place holder.

`/run/verbose 1`

`/vis/viewer/flush`

`/dir/command ! second`

Command submission

- Geant4 UI command can be issued by
 - (G)UI interactive command submission
 - Macro file
 - Hard-coded implementation
 - Slow but no need for the targeting class pointer
 - Should **not** be used inside an event loop

```
G4UImanager* UI = G4UImanager::GetUIpointer();  
UI->ApplyCommand("/run/verbose 1");
```

- The availability of individual command, the ranges of parameters, the available candidates on individual command parameter **may vary** according to the implementation of your application and may even **vary dynamically** during the execution of your job.
- some commands are available only for limited Geant4 **application state(s)**.
 - E.g. **/run/beamOn** is available only for **idle** states.

Command refusal

- Command will be **refused** in case of
 - Wrong application state
 - Wrong type of parameter
 - Insufficient number of parameters
 - Parameter out of its range
 - For integer or double type parameter
 - Parameter out of its candidate list
 - For string type parameter
 - Command not found

2) Macro files

Macro file

- Macro file is an **ASCII file** contains **UI commands**.
- All commands must be given with their **full-path directories**.
- Use “**#**” for comment line.
 - First “**#**” to the end of the line will be ignored.
 - Comment lines will be echoed if **/control/verbose** is set to **2**.
- Macro file can be executed
 - interactively or in (other) **macro file**
/control/execute file_name
 - hard-coded

```
G4UImanager* UI = G4UImanager::GetUIpointer();  
UI->ApplyCommand("/control/execute file_name");
```


Available Commands

- You can get a list of available commands **including your custom ones** by
 - `/control/manual [directory]`
 - Plain text format to standard output
 - `/control/createHTML [directory]`
 - HTML file(s) - one file per one (sub-)directory
- List of built-in commands is also available in *section 7.1 of User's Guide For Application Developers.*

Alias

- Alias can be defined by

```
/control/alias [name] [value]
```

- It is also set with `/control/loop` and `/control/foreach` commands
- Aliased value is always treated as a **string** even if it contains numbers only.

- Alias is to be used with other UI command (and macro files).

- Use curly brackets, `{` and `}`.

- For example, frequently used lengthy command can be shortened by aliasing.

```
/control/alias tv /tracking/verbose
```

```
{tv} 1
```

- Aliases can be used recursively.

```
/control/alias file1 /diskA/dirX/fileXX.dat
```

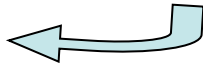
```
/control/alias file2 /diskB/dirY/fileYY.dat
```

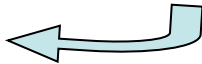
```
/control/alias run 1
```

```
/myCmd/getFile {file{run}}
```

Loop

- `/control/loop` and `/control/foreach` commands execute a macro file more than once. Aliased variable name can be used inside the macro file.
- `/control/loop [macroFile] [counterName] [initialValue] [finalValue] [stepSize]`
 - `counterName` is aliased to the `number` as a loop counter
- `/control/foreach [macroFile] [counterName] [valueList]`
 - `counterName` is aliased to a `value` in `valueList`
 - `valueList` must be enclosed by double quotes ("`\"`")
- on UI terminal or other macro file

```
/control/loop myRun.mac Ekin 10. 20. 2.
```
- in `myRun.mac` 

```
/control/foreach mySingleRun.mac pname "p pi- mu-"
```
- in `mySingleRun.mac` 

```
/gun/particle {pname}
/gun/energy {Ekin} GeV
/run/beamOn 100
```

Batch mode / interactive mode

- In your *main()*

```
int main(int argc, char** argv)
{ ...
  // get the pointer to the User Interface manager
  G4UImanager* UI = G4UImanager::GetUIpointer();

  if (argc != 1)
  { // batch mode
    G4String command = "/control/execute ";
    G4String fileName = argv[1];
    UImanager->ApplyCommand(command+fileName);
  }
  else
  { // interactive mode : define UI session
    G4UIExecutive* ui = new G4UIExecutive(argc, argv);
    ui->SessionStart();
    delete ui;
  }
}
```

Defining UI commands

- Geant4 offers you the possibility to **define simple or more complex UI commands**.
- For this, you will need to define a **messenger class** for your UI commands
- Each messenger class must be derived from **G4UImessenger** base class.
- A messenger class can handle more than one UI commands.
- A messenger class **should be instantiated by** the constructor of the **target class** to which commands should be delivered, and **should be deleted** by the destructor of the target class.

- Methods of messenger class
 - **Constructor**
 - **Define** (instantiate) commands / command directories
 - **Destructor**
 - **Delete** commands / command directories
 - void **SetNewValue**(G4UIcommand* command, G4String newValue)
 - **Convert** "newValue" parameter string to appropriate value(s) and **invoke an appropriate method** of the target class
 - This method is invoked when a command is issued.
 - G4String **GetCurrentValue**(G4UIcommand* command)
 - **Access to an appropriate get-method** of the target class and convert the current value(s) to a string
 - This method is invoked when the current value(s) of parameter(s) of a command is asked by (G)UI.

3) Terminal commands

Terminal commands

- **Interactive terminal** supports some Unix-like commands for directory.
 - **cd**, **pwd** - change and display current command directory
 - by setting the current command directory, you may omit (part of) directory string.
 - **ls** - list available UI commands and sub-directories
- It also supports some other commands.
 - **history** - show previous commands
 - **!*historyID*** - re-issue previous command
 - **arrow keys and tab** (TC-shell only)
 - **?*UIcommand*** - show current parameter values of the command
 - **help** [*UIcommand*] - help
 - **exit** - job termination
- Above commands are interpreted in the interactive terminal and are not passed to Geant4 kernel. You **cannot** use them in a macro file.