ApiMenu

Version : 1.0
Date : 19/08/2006
Author : ¨Patrick Germain Placidoux

SUMMARY

1 OBJECTIVE

This document describes the Apimenu purpose.

Note: apimenu is an OpenSource project at sourceforge.net, contact : apimenu@gmx.com.

2 INTRODUCTION

Apimenu is a tools that allows you to compose simple and complexe multidimensional terminal menus either from an xml interface or promatically using the repoz Api.

Apimenu aims to work on any plateform (Unixe(s), Windows (TM), Macs (TM), ... every where python works). This is why the design stays so simple;
No advanced graphical effect (like colors, hightlighting, blinking, …) because most of them are plateform specific.

Apimenu focuses on helpful functionalities like :

- Multidimensional menu : menus can be indefinitly imbricated inside each other.
- Interactive support, menu can accept input values, proceed to their integrity check and provide hem as variables for commands formating.
- Help system and mulilanguage support.

3 PRERIQUISITES

<u>Oses</u> (every where python works)

Linux, AIX ®, Windows ®

<u>Langages</u>

Python >= 2.4 < 3

4 INSTALLATION

Download apimenu at www.sourceforge.net

Unzip the file apimen_#.##.zip in the directory of your choice.

Put the path <APIMENU_INSTALL_PATH>/bin in your path environment variable.

Type :

> \> apimenu on Windows
> or
> \> apimenu.sh on Unixes
> (dont forget to run chmod ugo+x *.sh in the bin directory)

5 QUICK VIEW

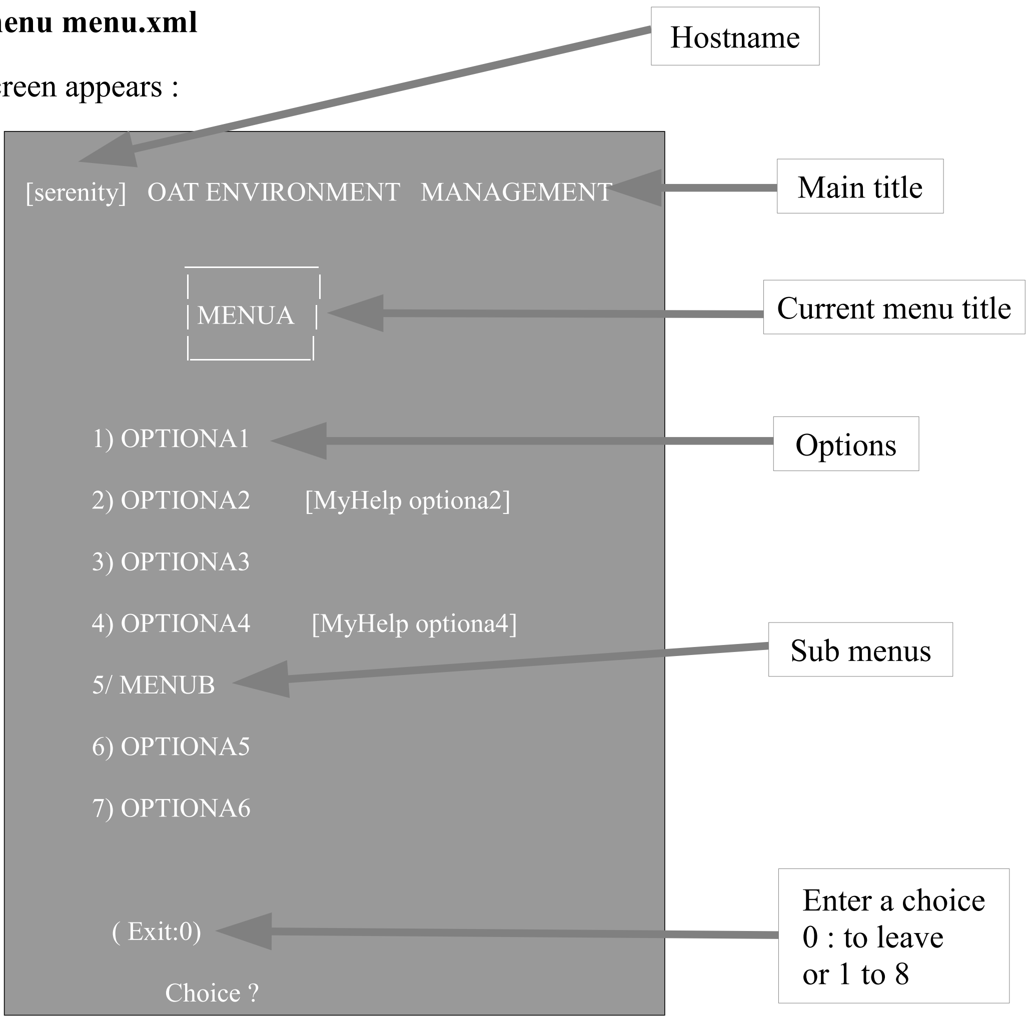
5.1 RUNNING APIMENU FOR THE FIRST TIME

Enter the directory <APIMENU_INSTALL_DIR>/samples/

Note: I f you are running apimenu from kikonf (see the kikonf project at sourceforge.net) replace this directory by: <KIKONF_INSTALL_DIR>/samples/apimenu

Type:
$ apimenu menu.xml

This screen appears :



```
[serenity]  OAT ENVIRONMENT  MANAGEMENT

                 ___________
                |           |
                | MENUA     |
                |___________|

        1) OPTIONA1

        2) OPTIONA2       [MyHelp optiona2]

        3) OPTIONA3

        4) OPTIONA4       [MyHelp optiona4]

        5/ MENUB

        6) OPTIONA5

        7) OPTIONA6


          ( Exit:0)

               Choice ?
```

Figure 1.

6 USING THE XML INTERFACE

6.1 STANDARD MENUS

When the command **apimenu** is called with an xml file as parameter, you are using the xml interface.

e.g.:
$ apimenu menu.xml

This is the content of <APIMENU_INSTALL_DIR>/samples/menu.xml :

```
1. <config title='OAT Environment Management'>
2.
3.      <menu title='menua'>
4.         <option name='optiona1' command='ls -ltr'/>
5.         <option name='optiona2' command='dir' confirm='True' help='MyHelp optiona2'/>
6.         <option name='optiona3' command='dir'/>
7.         <option name='optiona4' command='ls -ltr' help='MyHelp optiona4'/>

8.         <menu title='menub'>
9.            <option name='optionb1' command='dir'/>
10.            <option name='optionb3' command='ls -ltr'/>
11.            <option name='optionb4' command='dir'/>
12.            <option name='optionb5' command='ls -ltr'/>
13.        </menu>

14.        <option name='optiona5' command='ls -ltr'/>
15.        <option name='optiona6' command='dir'/>
16.     </menu>
17.
18. </config>
```

This file shows a simple sample of 2 imbricated menus.

Note:
The allowed schema for an Apimenu xml file is described into a file called the **Apimenu descriptor file** at : **<APIMENU_INSTALL_DIR>/core/lib/menu.desc.xml**.
For an extensive view on what kind Attributes are allowed for a guiven tag please consult this file.

Line 1 : The special tag : config set the preference Attributes for whole menus.
The available Attributes to set up Apimenu preferences are listed into the menu descriptor file or into *"chapter 8 Annexe : The ApiMenu Pydoc/ 8.1 class Config"*.

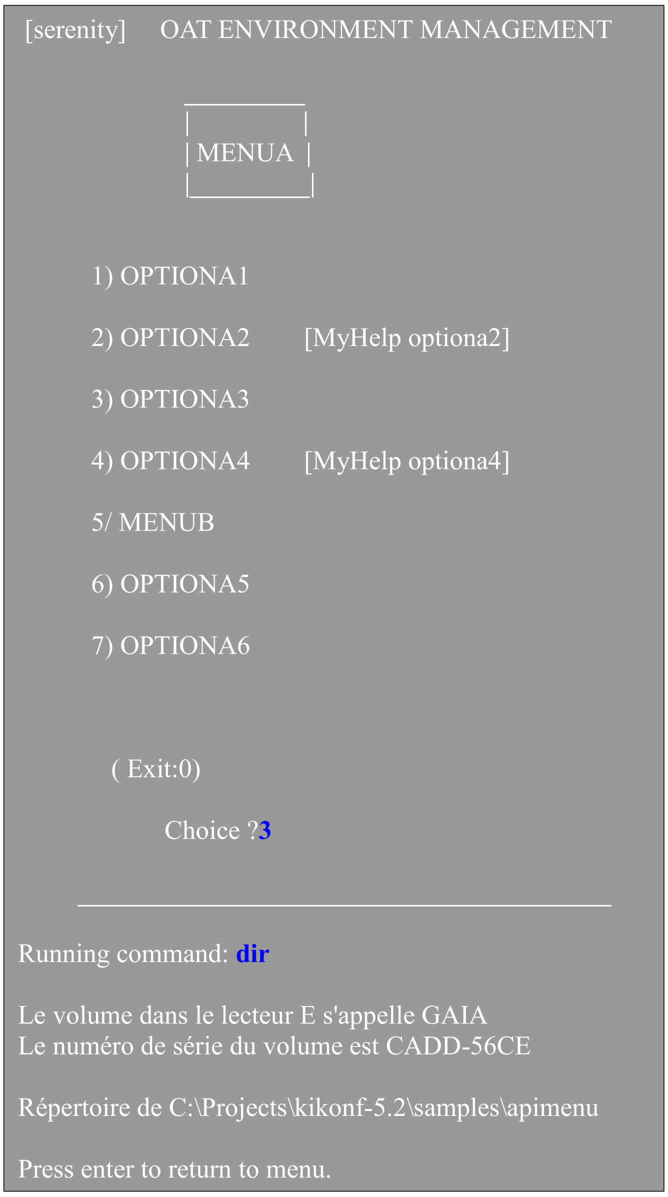
Note : every config Attributes have default values, so they are not required except: **title** which is required.

Line 3 : The nex t container under config is a **menu** node titled: "menua".
Because this is the fist menu instance, it is also the first one shown at starttup (see figure 1).
This menu host two kind of child :

- options : from optiona1 to optiona6
- one menu: titled "menub"

line 4 : Options are single commands,

e.g. if "3" was typed on figure 1., this would run the command "dir", which is the definition for **option** optiona3 on line 4.

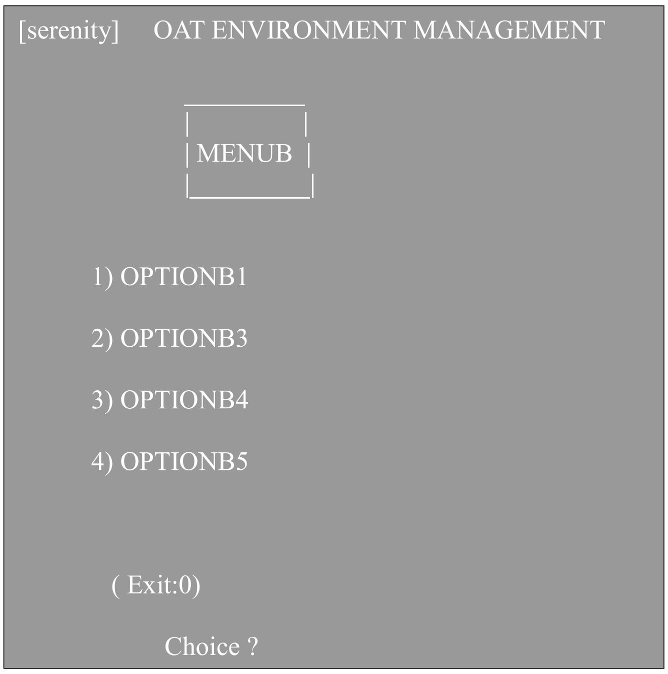


```
[serenity]    OAT ENVIRONMENT MANAGEMENT

                 _________
                 |         |
                 | MENUA  |
                 |_________|


        1) OPTIONA1

        2) OPTIONA2        [MyHelp optiona2]

        3) OPTIONA3

        4) OPTIONA4        [MyHelp optiona4]

        5/ MENUB

        6) OPTIONA5

        7) OPTIONA6


          ( Exit:0)

               Choice ?3

       _______________________________________

Running command: dir

Le volume dans le lecteur E s'appelle GAIA
Le numéro de série du volume est CADD-56CE

Répertoire de C:\Projects\kikonf-5.2\samples\apimenu

Press enter to return to menu.
```

Figure 2.

Line 8 : defines a **menu** titled : "menub".
When a menu entry is choosed, a new screen showing this menu is displayed.

e.g. if "5" was typed on figure 1., this would display the "menub" :



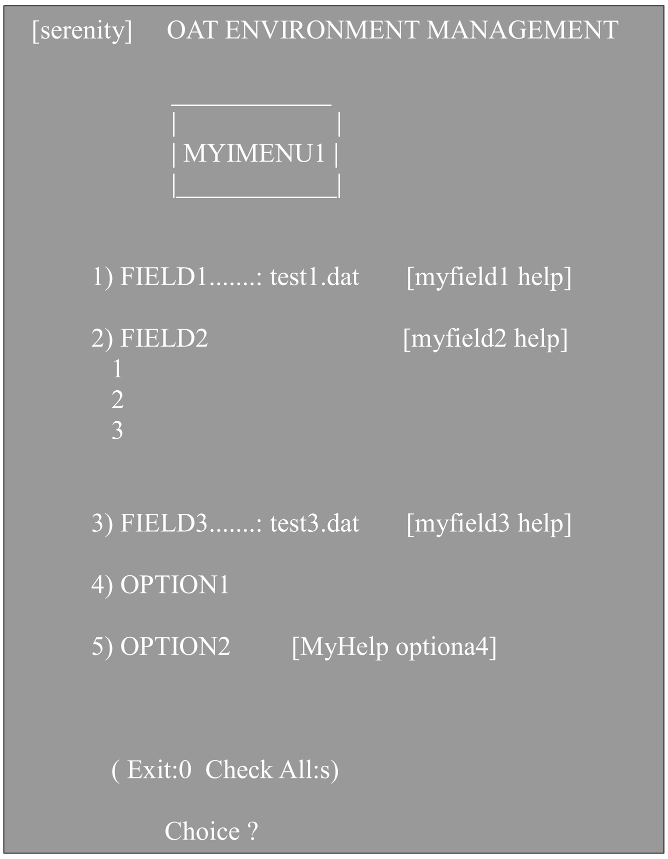
```
[serenity]    OAT ENVIRONMENT MANAGEMENT

                  __________
                 |          |
                 | MENUB    |
                 |__________|

        1) OPTIONB1

        2) OPTIONB3

        3) OPTIONB4

        4) OPTIONB5


          ( Exit:0)

               Choice ?
```

Figure 3.

6.2 INTERACTIVE MENUS

The sample menu1.xml shows a sample of an interactive menu.

e.g. type:
$ apimenu menu1.xml



```
[serenity]    OAT ENVIRONMENT MANAGEMENT

               _____________
              |             |
              | MYIMENU1 |
              |_____________|


      1) FIELD1.......: test1.dat      [myfield1 help]

      2) FIELD2                        [myfield2 help]
         1
         2
         3


      3) FIELD3.......: test3.dat      [myfield3 help]

      4) OPTION1

      5) OPTION2        [MyHelp optiona4]



        ( Exit:0  Check All:s)

              Choice ?
```

Figure 4.

This is the content of <APIMENU_INSTALL_DIR>/samples/menu1.xml :

```
1.  <config title='OAT Environment Management'>
2.      <imenu title='MyIMenu1' help='MyIMenu1 help.'
3.                  command='dir $field1 $field2 $field3' confirm='True'>
4.
5.          <ioption name='field1' help='myfield1 help' wk='{*type:str,*value:test1.dat}'/>
6.          <ioption name='field2' help='myfield2 help'>
7.                  {*type:list,*ltype:{*type:int},*value:[1,2,3]}
8.          </ioption>
9.          <ioption name='field3' help='myfield3 help' wk='{*type:str,*value:test3.dat}'/>
10.
11.         <option name='option1' command='dir'/>
12.         <option name='option2' command='ls -ltr' help='MyHelp optiona4'/>
13.         <option name='option3' command='ls -ltr' help='MyHelp optiona4'/>
14.
15.     </imenu>
16.
17. </config>
```

This file defines only one sole menu titled "MyIMenu1".

An interactive menu is a special kind of menu that accepts **input values**.
Input values are supported into interactive menus by the **ioption** tags.

Note: interactive menus also supports the regular **option** tag.

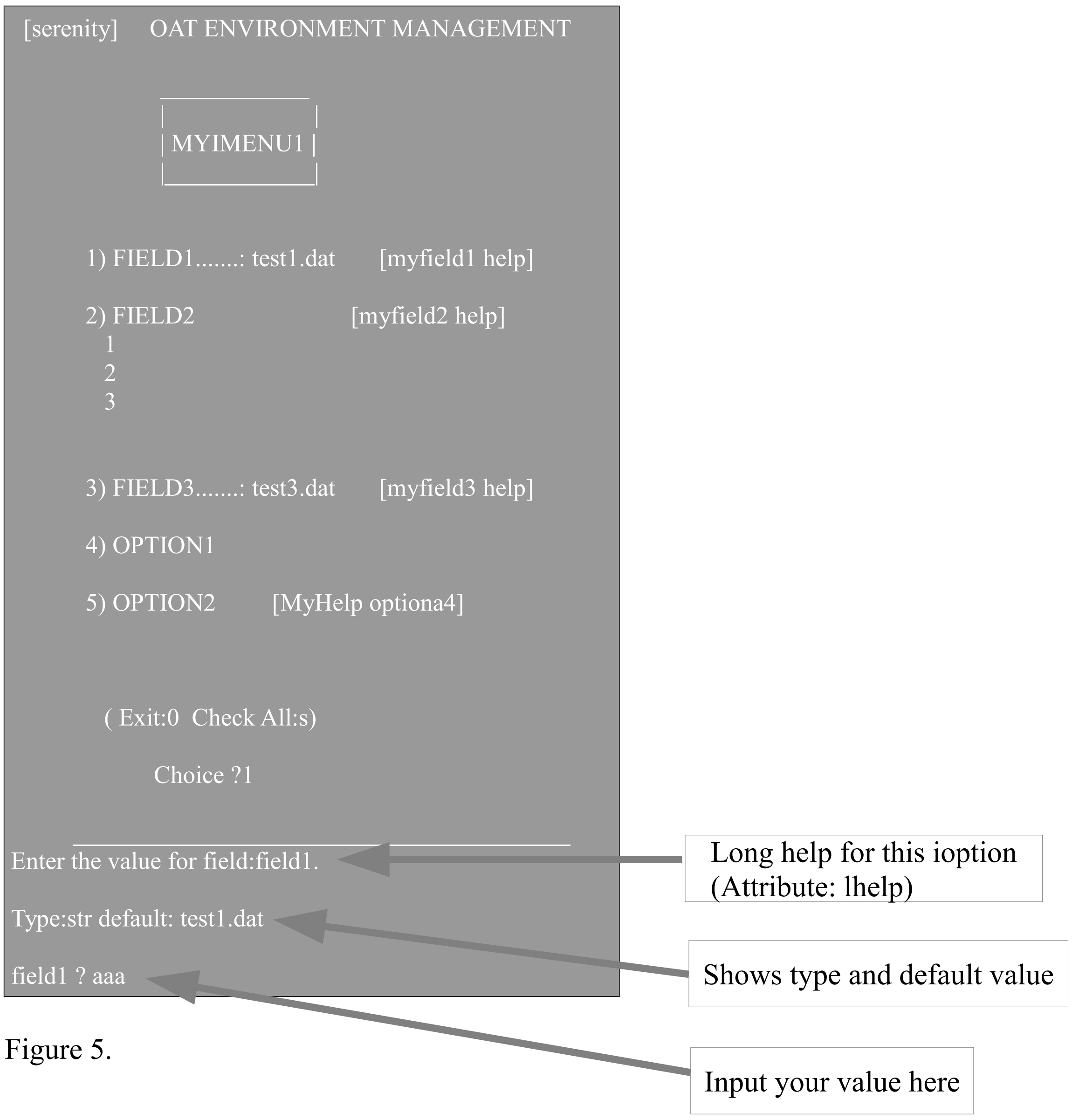
Line 2 : The **imenu** tag defines the **command pattern for the whole menu**.
The command pattern for menu "MyIMenu1" is: command='dir $field1 $field2 $field3'

All variable with this shape *$VARIABLE_NAME* into the command pattern will trigger a search througth this menu childs for an **ioption** named *VARIABLE_NAME*.
If found replacement occures with the values inputed by the user or the default.

Line 5 : The **ioption** named "field1" allows an input value of type string, wich default is: "test".
The input type check is controlled by the wk attribute here : "wk='{*type:str,*value:test1.dat}".

Note about wk:
For more information about the wk syntax please check the wk documentation into the directory <APIMENU_INSTALL_DIR>/docs or from the wk project at sourceforge.net.

If the user type option 1 on figure 4. this shows an input quiz for the **ioption** "field1:



```
[serenity]    OAT ENVIRONMENT MANAGEMENT

                 ____________
                |            |
                | MYIMENU1 |
                |____________|


         1) FIELD1.......: test1.dat      [myfield1 help]

         2) FIELD2                        [myfield2 help]
           1
           2
           3


         3) FIELD3.......: test3.dat      [myfield3 help]

         4) OPTION1

         5) OPTION2        [MyHelp optiona4]



           ( Exit:0  Check All:s)

                 Choice ?1

        ________________________________________
Enter the value for field:field1.

Type:str default: test1.dat

field1 ? aaa
```

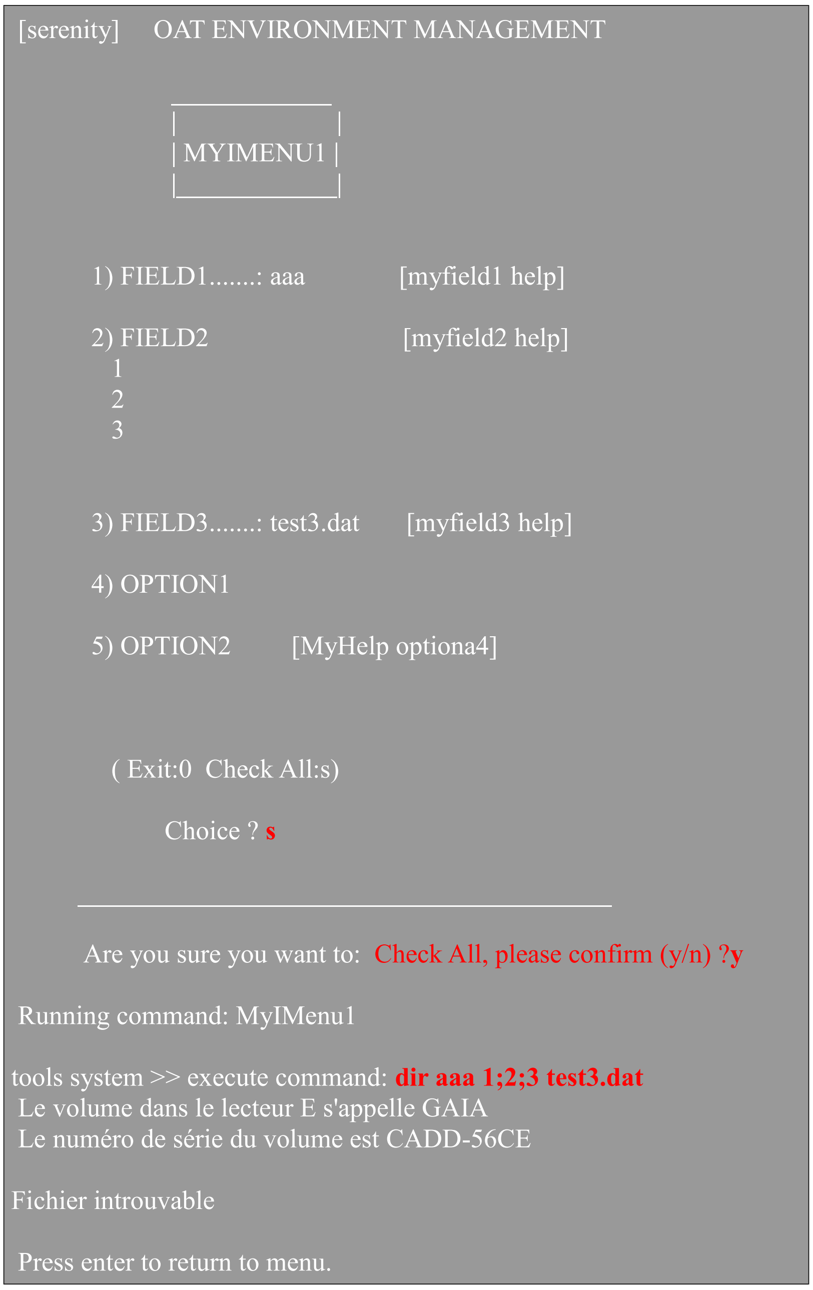
Figure 5.

Interactive menu Validation

What we call the Validation step is when the user type "s" to submit the IMenu command.

All the fields are checked back again, pattern replacement is processed inside the command string, then the command is run.

e.g.: If the User was typing "s" on figure 5 :



```
[serenity]    OAT ENVIRONMENT MANAGEMENT

               _____________
              |             |
              | MYIMENU1 |
              |_____________|


        1) FIELD1.......: aaa              [myfield1 help]

        2) FIELD2                          [myfield2 help]
          1
          2
          3


        3) FIELD3.......: test3.dat        [myfield3 help]

        4) OPTION1

        5) OPTION2          [MyHelp optiona4]



          ( Exit:0  Check All:s)

               Choice ? s

       __________________________________________

       Are you sure you want to:  Check All, please confirm (y/n) ?y

 Running command: MyIMenu1

tools system >> execute command: dir aaa 1;2;3 test3.dat
 Le volume dans le lecteur E s'appelle GAIA
 Le numéro de série du volume est CADD-56CE

Fichier introuvable

 Press enter to return to menu.
```

6.3 MIXTE MENUS

Please check the third sample: menu2.xml into the directory :
<KIKONF_INSTALL_DIR>/samples/apimenu to see a more complex imbrication of menus and imenus.

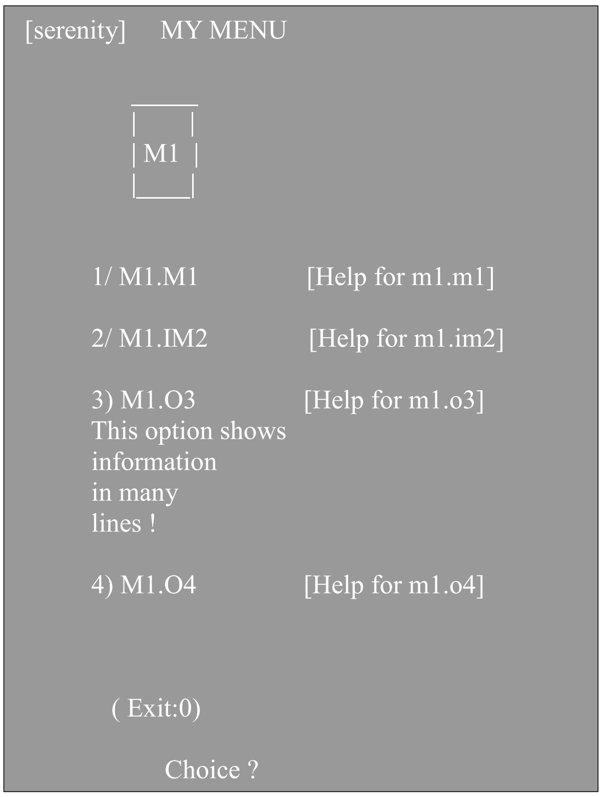
7 USING THE PROGRAMATIC INTERFACE

A sample documented and using the programatic is at <APIMENU_INSTALL_DIR>/bin/apitest.

You can also check the Apimenu Pydoc described at *"ANNEXE : The ApiMenu Pydoc"*.

7.1 RUNNING THE SUPPLIED SAMPLE

To run the progam sample type :
$ apitest



```
[serenity]    MY MENU

           _____
          |     |
          | M1  |
          |_____|


      1/ M1.M1             [Help for m1.m1]

      2/ M1.IM2            [Help for m1.im2]

      3) M1.O3             [Help for m1.o3]
      This option shows
      information
      in many
      lines !

      4) M1.O4             [Help for m1.o4]



        ( Exit:0)

             Choice ?
```

8 ANNEXE : THE APIMENU PYDOC

```
Help on module lib.apimenup in lib:

NAME
    lib.apimenup

FILE
    apmenu\core\lib\apimenup.py

DESCRIPTION
    ## Copyright (c) 2007-2008, Patrick Germain Placidoux
    ## All rights reserved.
    ##
    ## This file is part of kikonf.
    ##
    ## kikonf is free software: you can redistribute it and/or modify
    ## it under the terms of the GNU General Public License as published by
    ## the Free Software Foundation, either version 3 of the License, or
    ## (at your option) any later version.
    ##
    ## kikonf is distributed in the hope that it will be useful,
    ## but WITHOUT ANY WARRANTY; without even the implied warranty of
    ## MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    ## GNU General Public License for more details.
    ##
    ## You should have received a copy of the GNU General Public License
    ## along with kikonf.  If not, see <http://www.gnu.org/licenses/>.
    ##
    ## Home: http://www.kikonf.com
    ## Contact: kikonf@gmx.com

CLASSES
    BaseMenu
        IMenu
        Menu
    Config
    IOption
    Option
```

8.1 CLASS CONFIG

```
| Methods defined here:
|
| __init__(...)
|    fct_menu : A function to build Menus
|       Menu are build on the flow.
|       Each time ApiMenu nedds to build a new Menu this function (or method) is called with an
|       unique parameter the parent Menu instance.
|
|    title : The main menu title.
|
|    show_host : True/False : If True shows the localhost name on the rigth.
|
|    When ApiMenu is called with an xml file, the values of the coming list of parameter
|    are retreived from the xml file.
|    Most of these parameters are defined within the xml file througth the menu lang dictionary
|    into <APIMENU_INSTALL_DIR>/langs.
|
|    screen_max_lines : If not guiven takes it value from the global variable:
|       SCREEN_MAX_LINES
|       Called with an xml file, Apimenu passes the value from the Tag/Attribute:
|                config@screen_max_lines.
|       Number of lines. This the maximum size allowed for a Menu screen.
|
|    up_car : If not guiven takes it value from the global variable: UP_CAR.
|       Called with an xml file, Apimenu passes the value from the Tag/Attribute: config@up_car.
|       Page Up charater.When a Menu screen exceed the screen_max_lines this character symbol
|       is shown.
|       Typing this character on the Menu will PageUp.
|
|    up_message : If not guiven takes it value from the global variable: UP_MESSAGE
|       Called with an xml file, Apimenu passes the value from the Tag/Attribute:
|       config@up_message.
|       Page Up message. When a Menu screen exceed the screen_max_lines this message is
|       shown.
|
|    down_car : If not guiven takes it value from the global variable: DOWN_CAR
|       Called with an xml file, Apimenu passes the value from the Tag/Attribute:
|       config@down_car.
|       Page Down charater. When a Menu screen has been paged Up this character Appear.
|       Typing this character on the Menu will PageDown.
|
|    down_message : If not guiven takes it value from the global variable: DOWN_MESSAGE
|       Called with an xml file, Apimenu passes the value from the Tag/Attribute:
|       config@down_message.
```

Page Down message. When a Menu screen has been paged Up this message shown.

exit_car : If not guiven takes it value from the global variable: EXIT_CAR
Called with an xml file, Apimenu passes the value from the Tag/Attribute: config@exit_car.
Exit message. Some menu may need an action for the user to exit. In this case character Appear.
Typing this character on the Menu will Exit.

exit_message : If not guiven takes it value from the global variable: EXIT_MESSAGE
Called with an xml file, Apimenu passes the value from the Tag/Attribute: config@exit_message.
Exit message. Some menu may need an action for the user to exit. In this case this message is shown.

check_all_car : If not guiven takes it value from the global variable: CHECK_ALL_CAR
Called with an xml file, Apimenu passes the value from the Tag/Attribute: config@check_all_car.
For Input Menu only.
An Input Menu screen lists a set of Input fields.
These Input fields are proposed for validation (validation action may be to run a command) througth the check_all_car character.
Typing this character will validate the IMenu.

check_all_message : If not guiven takes it value from the global variable: CHECK_ALL_MESSAGE
Called with an xml file, Apimenu passes the value from the Tag/Attribute: config@check_all_message.
An Input Menu screen lists a set of Input fields.
These Input fields are proposed for validation (validation action may be to run a command) througth the check_all_car character.
This message is shown to invite the user to run Validation.

choice_message : If not guiven takes it value from the global variable: CHOICE_MESSAGE
Called with an xml file, Apimenu passes the value from the Tag/Attribute: config@choice_message.

confirm_message : If not guiven takes it value from the global variable: CONFIRM_MESSAGE
Called with an xml file, Apimenu passes the value from the Tag/Attribute: config@confirm_message.
Confirmation message. Before to run a command some menu may require acceptance.
This message is the acceptance invitation message.

confirm_exit_message : If not guiven takes it value from the global variable: CONFIRM_EXIT_MESSAGE
Called with an xml file, Apimenu passes the value from the Tag/Attribute: config@confirm_exit_message.
Exit message. Before to exit, some menu may require acceptance.

This message is the exit invitation message.

wait_message : If not guiven takes it value from the global variable: WAIT_MESSAGE
Called with an xml file, Apimenu passes the value from the Tag/Attribute: config@wait_message.
Wait message. Running a command, some menu may need a wait message.
This message is the way message.

option_upper : If not guiven takes it value from the global variable: OPTION_UPPER
Called with an xml file, Apimenu passes the value from the Tag/Attribute: config@up_car.
If True, Menu title are shown in Upper Cases.

option_check_message1 : If not guiven takes it value from the global variable: OPTION_CHECK_MESSAGE1
Called with an xml file, Apimenu passes the value from the Tag/Attribute: config@option_check_message1.
This message may be shown into the validation sequence of the value typed by the user.

option_check_message2 : If not guiven takes it value from the global variable: OPTION_CHECK_MESSAGE2
Called with an xml file, Apimenu passes the value from the Tag/Attribute: config@option_check_message2.
This message may be shown into the validation sequence of the value typed by the user.

input_field_message1 : If not guiven takes it value from the global variable: INPUT_FIELD_MESSAGE1
Called with an xml file, Apimenu passes the value from the Tag/Attribute: config@input_field_message1.
This message may be shown into the input sequence for an Input field (from an IMenu screen).

input_field_message2 : If not guiven takes it value from the global variable: INPUT_FIELD_MESSAGE2
Called with an xml file, Apimenu passes the value from the Tag/Attribute: config@input_field_message2.
This message may be shown into the input sequence for an Input field (from an IMenu screen).

input_field_default_message : If not guiven takes it value from the global variable: INPUT_FIELD_DEFAULT_MESSAGE
Called with an xml file, Apimenu passes the value from the Tag/Attribute: config@input_field_default_message.
This message is displayed showing the default value for an Input field (from an IMenu screen).

input_field_checkin_message : If not guiven takes it value from the global variable: INPUT_FIELD_CHECKIN_MESSAGE
Called with an xml file, Apimenu passes the value from the Tag/Attribute:

```
|       config@input_field_checkin_message.
|       The message is shown when an entry for an Input field (from an IMenu screen) must be
       chossen beteewn
|       an arbitrary list of predefined values.
|
|     command_label : If not guiven takes it value from the global variable:
|       COMMAND_LABEL
|       Called with an xml file, Apimenu passes the value from the Tag/Attribute:
|       config@command_label.
|       This label is displayed at the begining of the output of the Running Command.
|
|    option_help_indent : If not guiven takes it value from the global variable:
|       OPTION_HELP_INDENT
|       Called with an xml file, Apimenu passes the value from the Tag/Attribute:
|       config@option_help_indent.
|       Integer, how much to indent the help information on the Menu screen.
|
|    option_value_indent : If not guiven takes it value from the global variable:
|       OPTION_VALUE_INDENT
|       Called with an xml file, Apimenu passes the value from the Tag/Attribute:
|       config@option_value_indent.
|       This label is displayed at the begining of the output of the Running Command.
|       Integer, how much to indent the value information on the Menu screen.
|
|    skip_line : If not guiven takes it value from the global variable: SKIP_LINE
|       Called with an xml file, Apimenu passes the value from the Tag/Attribute:
|       config@skip_line.
|       If True, will skip a line between each child composing the Menu screen.
|
|    lang_dir : If not guiven takes it value from the global variable: LANG_DIR
|       Called with an xml file, Apimenu passes the value from the Tag/Attribute: config@lang_dir.
|       Where is the lang dictionary.
|       Generally into : <APIMENU_INSTALL_DIR>/lang
|
|    verbose : Default 0.
|          Integer, the verbose level.
|
| drawLine(self)
|
| printIMenu(self, menu)
|
| printMenu(self, menu)
|
| run(self, menu)
```

8.2 CLASS BASEMENU : NOT USED DIRECTLY

```
| Methods defined here:
|
| __init__(...)
|    title : This is the menu title
|
|    sub_title : This is the menu title
|
|    fct_exit_command : Default None.
|       When called programatically. This function interceptor is called before exiting a Menu.
|
|    do_check_all : Default True.
|       For IMenu (Input Menu) only.
|       Will invite the user to check all the Input field and run the validation sequence.
|
|    confirm_exit : Default False.
|       Does this Menu require an acceptance to Exit.
|
|    fct_enter : Default None.
|       When called programatically. This function interceptor is called before entering a Menu.
|
|    fct_leave_forward : Default None.
|       When called programatically. This function interceptor is called before leaving forward
|       a Menu.
|
|    fct_leave_backward : Default None.
|       When called programatically. This function interceptor is called before leaving backward
|       a Menu.
|
| add(self, child)
|
| calcPages(self, screen_max_lines, do_skip_line)
|
| clear(self)
|
| delChild(self, index)
|
| delete(self)
|
| doConfirmExit(self)
|
| doDelete(self)
|
| enter(self, config)
|
| getChild(self, index)
|
| getChildNumber(self)
|
| getChilds(self)
|
| getContents(self)
|
| getCurrentPage(self)
|
| getCurrentPageIndex(self)
|
| getFct_exit_command(self)
|
| getHelp(self)
|
| getIndex(self)
|
| getLinesCount(self)
|
| getName(self)
|
| getPageNumber(self)
|
| getPages(self)
|
| getPipeIndex(self)
|
| getSubTitle(self)
|
| getTitle(self)
|
| isDeletable(self)
|
| isDeleted(self)
|
| isFed(self)
|
| leave_backward(self, config)
|
| leave_forward(self, config)
|
| pageDown(self)
|
| pageUp(self)
|
| pageZero(self)
|
|  setNotFed(self)
```

8.3 CLASS MENU(BASEMENU)

```
| Methods defined here:
|
| __init__(...)
|    title : This is the menu title
|
|    sub_title : This is the menu title
|
|    fct_exit_command : Default None.
|       When called programatically. This function interceptor is called before exiting a Menu.
|
|    confirm_exit : Default False.
|       Does this Menu require an acceptance to Exit.
|
|    fct_enter : Default None.
|       When called programatically. This function interceptor is called before entering a Menu.
|
|    fct_leave_forward : Default None.
|       When called programatically. This function interceptor is called before leaving forward a
|       Menu.
|
|    fct_leave_backward : Default None.
|       When called programatically. This function interceptor is called before leaving backward a
|        Menu.
|
| ----------------------------------------------------------------------
| Methods inherited from BaseMenu:
|
| add(self, child)
|
| calcPages(self, screen_max_lines, do_skip_line)
|
| clear(self)
|
| delChild(self, index)
|
| delete(self)
|
| doConfirmExit(self)
|
| doDelete(self)
|
| enter(self, config)
|
| getChild(self, index)
```

```
|
| getChildNumber(self)
|
| getChilds(self)
|
| getContents(self)
|
| getCurrentPage(self)
|
| getCurrentPageIndex(self)
|
| getFct_exit_command(self)
|
| getHelp(self)
|
| getIndex(self)
|
| getLinesCount(self)
|
| getName(self)
|
| getPageNumber(self)
|
| getPages(self)
|
| getPipeIndex(self)
|
| getSubTitle(self)
|
| getTitle(self)
|
| isDeletable(self)
|
| isDeleted(self)
|
| isFed(self)
|
| leave_backward(self, config)
|
| leave_forward(self, config)
|
| pageDown(self)
|
| pageUp(self)
|
| pageZero(self)
|
| setNotFed(self)
```

8.4 CLASS IMENU(BASEMENU)

| Methods defined here:
|
| __init__(...)
| **title :** This is the menu title
|
| **sub_title :** This is the menu title
|
| **fct_exit_command :** Default None.
| When called programatically. This function interceptor is called before exiting a Menu.
|
| **do_check_all :** Default True.
| For IMenu (Input Menu) only.
| Will invite the user to check all the Input field and run the validation sequence.
|
| **confirm :** Default False.
| Does the command associated with this IMenu needs Acceptance before to run the
| Validation sequence.
|
| **command :** Default None
| This command is called running the Validation sequence.
| All variables within the command string are replaced with the corresponding Input field
| value.
| e.g.: if command is : ls $field1 $field2
| And the values for these 2 IOptions are field1 = 'myfile1' and field = 'myfile2'.
| The command turns to : ls myfile1 myfile2.
|
| **fct_command :** Default None.
| When called programatically. This function interceptor is called at the Validation sequence.
|
| >> Rules for command and fct_command :
| ....
| if fct_command only is given: this fonction is called at the Validation sequence.
| if command only is given: This system command is called on check_all (default:s) key.
| if fct_command and command are given: The fonction fct_command is called, with the
| command parameter at the Validation sequence.
|
| **confirm_exit :** Default False.
| Does this Menu require an acceptance to Exit.
|
| **fct_enter :** Default None.
| When called programatically. This function interceptor is called before entering a Menu.
|
| **fct_leave_forward :** Default None.
| When called programatically. This function interceptor is called before leaving forward a

```
|          Menu.
|
|      fct_leave_backward : Default None.
|          When called programatically. This function interceptor is called before leaving backward a
|           Menu.
|
|  checkCommand(self)
|
|  doCheckAll(self)
|
|  doConfirm(self)
|
|  getCommand(self)
|
|  getFct_command(self)
|
|  getVerbose_exec_command(self)
|
|  ----------------------------------------------------------------------
|  Methods inherited from BaseMenu:
|
|  add(self, child)
|
|  calcPages(self, screen_max_lines, do_skip_line)
|
|  clear(self)
|
|  delChild(self, index)
|
|  delete(self)
|
|  doConfirmExit(self)
|
|  doDelete(self)
|
|  enter(self, config)
|
|  getChild(self, index)
|
|  getChildNumber(self)
|
|  getChilds(self)
|
|  getContents(self)
|
|  getCurrentPage(self)
|
|  getCurrentPageIndex(self)
```

```
|
| getFct_exit_command(self)
|
| getHelp(self)
|
| getIndex(self)
|
| getLinesCount(self)
|
| getName(self)
|
| getPageNumber(self)
|
| getPages(self)
|
| getPipeIndex(self)
|
| getSubTitle(self)
|
| getTitle(self)
|
| isDeletable(self)
|
| isDeleted(self)
|
| isFed(self)
|
| leave_backward(self, config)
|
| leave_forward(self, config)
|
| pageDown(self)
|
| pageUp(self)
|
| pageZero(self)
|
| setNotFed(self)
```

8.5 CLASS OPTION

```
| Methods defined here:
|
| __init__(...)
|    title : This is the Option title
|
|    help :  Default None
|       Short Help string for this Menu Option.
|
|    contents : Long text information for this Menu Option.
|
|    command : Default None
|       This command is called when this Option is chosen.
|
|    fct_command : Default None.
|       When called programatically. When this Option is chosen, this function interceptor is called.
|
|    confirm : Default False
|       Does this Option requires Acceptance before to execute.
|
|    verbose_exec_command : Default False.
|       If True the called command is explicitly shown on the command output display.
|
| doConfirm(self)
|
| getCommand(self)
|
| getContents(self)
|
| getFct_command(self)
|
| getHelp(self)
|
| getIndex(self)
|
| getLinesCount(self)
|
| getName(self)
|
| getVerbose_exec_command(self)

FUNCTIONS
   default_fct_menu(config, menu)

   digest(menu_file, verbose=0)
```

feedIMenuChild(menu, menu_node)

feedMenuChild(menu, menu_node)

8.6 CLASS IOPTION

```
| Methods defined here:
|
| __init__(...)
|     name : This is the Field Name
|
|     help : A short Help for this IMenu IOption.
|
|     lhelp : A short Help for this IMenu IOption.
|
|     value : Default None
|         A default value for this IMenu IOption.
|
|     wkvalue : A default value for this IMenu IOption.
|
|     contents : Long text information for this IMenu IOption.
|
|     wkcontents : a wk expression to check the user Input value.
|         This wk expression will be run on the Validation sequence of the parent IMenu.
|         For more information about wk expression see the wk project at sourceforge.net
|         or the wk book into the <APIMENU_INSTALL_DIR>/doc diretory.
|
|     lock : Default False
|         If True this IOption wont be allowed for Input.
|
|     command : Default None
|         This command is called when this Option is chosen.
|
|     fct_command : Default None.
|         When called programatically. When this Option is chosen, this function interceptor is called.
|
|     list_separator_car : When the Input value can be a list, this is the line break separator
|         character.
|
| checkIOValue(self, value, error_prefix=None)
|
| getCommand(self)
|
| getContents(self)
|
| getFct_command(self)
|
| getHelp(self)
|
| getIODescs(self)
```

```
|
| getIODftValue(self)
|
| getIOValue(self)
|
| getIndex(self)
|
| getLHelp(self)
|
| getLabel(self)
|
| getLinesCount(self)
|
| getList_separator_car(self)
|
| getName(self)
|
| getValue(self)
|
| isLocked(self)
|
| isValue(self)
|
| setIOValue(self, value, error_prefix=None)
```

8.7 DATA

```
    APIMENU_HOME = r'E:\Projets\DEPLOYMENT\kikonf-5.2\core\kikonf'
    CHECK_ALL_CAR = 'c'
    CHECK_ALL_MESSAGE = '%lang/menu.en/check_all_message'
    CHOICE_INDENT = 12
    CHOICE_MESSAGE = '%lang/menu.en/choice_message'
    COMMAND_LABEL = '%lang/menu.en/command_label'
    CONFIRM_EXIT_MESSAGE = '%lang/menu.en/confirm_exit_message'
    CONFIRM_MESSAGE = '%lang/menu.en/confirm_message'
    DOWN_CAR = '-'
    DOWN_MESSAGE = '%lang/menu.en/down_message'
    EXIT_CAR = '0'
    EXIT_MESSAGE = '%lang/menu.en/exit_message'
    INDENT = 10
    INPUT_FIELD_CHECKIN_MESSAGE = '%lang/menu.en/input_field_checkin_messa...
    INPUT_FIELD_DEFAULT_MESSAGE = '%lang/menu.en/input_field_default_messa...
    INPUT_FIELD_MESSAGE1 = '%lang/menu.en/input_field_message1'
    INPUT_FIELD_MESSAGE2 = '%lang/menu.en/input_field_message2'
    LANG_DIR = '$install_dir/conf'
    LINE_LENGTH = 40
    MULTILANG = <lib.multilangue.MultiLang instance at 0x00C36A80>
    OPTION_CHECK_MESSAGE1 = '%lang/menu.en/option_check_message1'
    OPTION_CHECK_MESSAGE2 = '%lang/menu.en/option_check_message2'
    OPTION_HELP_INDENT = 20
    OPTION_INDENT = 2
    OPTION_UPPER = True
    OPTION_VALUE_INDENT = 20
    SCREEN_MAX_LINES = 25
    SELF_MODULE = 'apimenup'
    SHOW_HOST = True
    SKIP_LINE = False
    UP_CAR = '+'
    UP_MESSAGE = '%lang/menu.en/up_message'
    WAIT_MESSAGE = '%lang/menu.en/wait_message'


>>>
```

9 ANNEXE : THE APIMENU LANG DICTIONARY

This is an exemple excerpt from the file :<APIMENU_INSTALL_PATH>/lang/menu.en.
Please checks this file for more acurate information.

up_message = Up
down_message = Down
exit_message = Exit
check_all_message = Check All
choice_message = Choice
confirm_message = Are you sure you want to: $1, please confirm
confirm_exit_message = Are you sure you want to exit, please confirm
wait_message = Press enter to return to menu.
option_check_message1 = Value must be a digit or one of: $1.
option_check_message2 = Choice must be a valid option menu !
input_field_message1 =Enter the value for field
input_field_message2 = Many entries are allowed using the separator
input_field_default_message = default
input_field_checkin_message = Choose one of
command_label = Running command: $1