Mobius Forensic Toolkit

0.5.13

Tutorial

May 21, 2012
by Eduardo Aguiar
`aguiar.eao@gmail.com`

Contents

1 Introduction

Nowadays, open source forensic tools are domain specific tools. Each tool tries to cover a little part of the investigation scope, and some of them do it very well. Unfortunately, they lack integration, and their development is made harder because of the absence of common code, and therefore of code reuse. Their outputs are not standardized, and most of them uses command line interface.

Mobius Forensic Toolkit is a framework to develop forensic tools. It is written in Python, using PYGTK and PyCairo. It is very extensible through extensions, whose are programs that share services, program environment and have access to a case model.

This tutorial is not intend to be a complete guide to the tools developed so far, but it is simply a hands-on guide and may grow further with the releases to come.

2

Setting up your case

2.1 Creating case

The first step to use Mobius is to create a case. A case is an abstraction and might be anything you might call a case, such as an investigation case, a part of an investigation case. It is basically a container of evidences.

To create a case, hit new case button at toolbar, or hit `File`→`New` menu option (see figure 2.1).

In this example, a new case named `Untitled Case 01` has been created (see figure 2.2).

To set up this case, hit the `File`→`Properties` menu option. It will open the case properties dialog (figure 2.3), where you can edit your case properties. `base folder` is where Mobius and its extensions save information about your case, so choose a suitable folder.

You can save your case by pressing the `save` button. It will open a file chooser dialog where you can enter your case filename. Mobius case files have an extension `.case`, which is added by default.

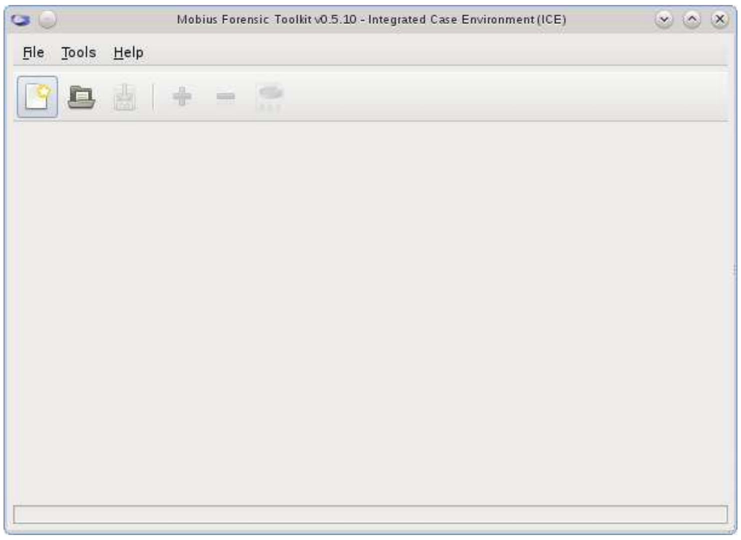

Mobius Forensic Toolkit v0.5.10 - Integrated Case Environment (ICE)
File
Tools
Help

Figure 2.1: Mobius main window

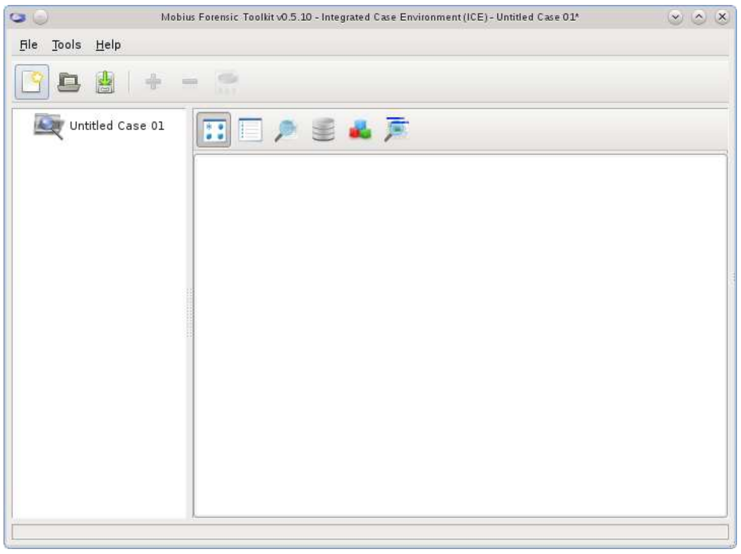
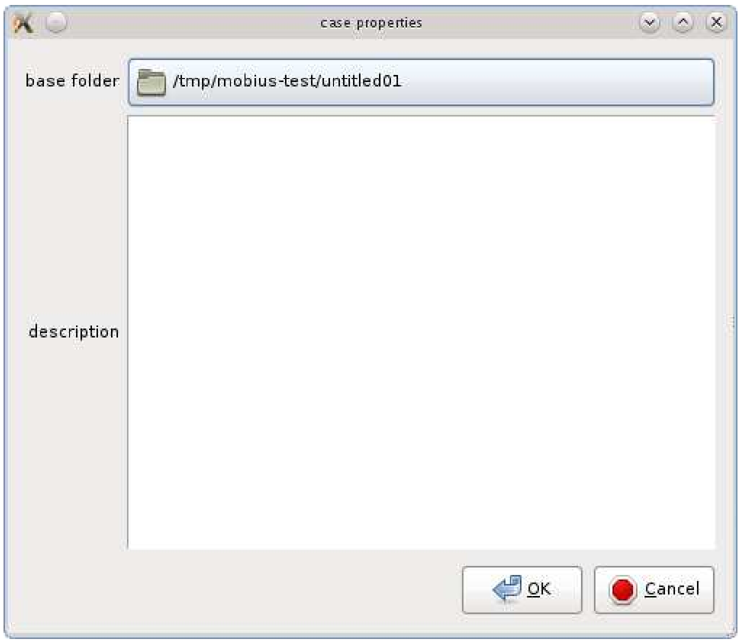

Mobius Forensic Toolkit v0.5.10 - Integrated Case Environment (ICE) - Untitled Case 01*
File
Tools
Help
Untitled Case 01

Figure 2.2: new case window

2.2 Adding items to case

Once your case has been created successfully, you can start to add evidences to it. Evidences are divided in categories, such as `harddisk`, `floppy` and so on. In section 4.1 we will see how to create new categories on the fly.

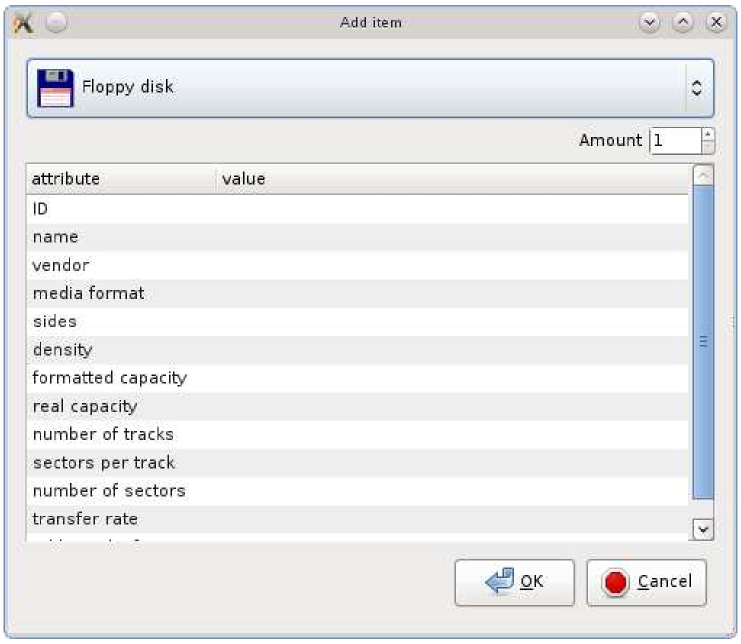
To add an item, you must select its container. Click on a case item and then



base folder /tmp/mobius-test/untitled01

description

Figure 2.3: new case properties dialog



attribute	value
ID	
name	
vendor	
media format	
sides	
density	
formatted capacity	
real capacity	
number of tracks	
sectors per track	
number of sectors	
transfer rate	

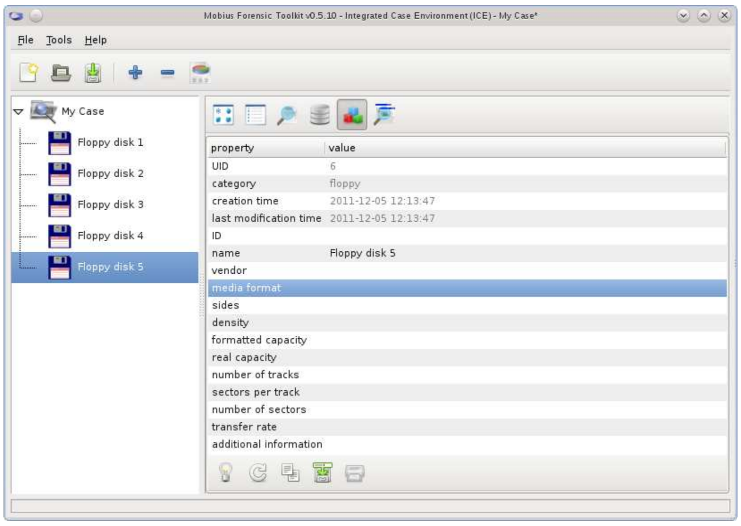
Figure 2.4: add item dialog

click on `add` icon. It will open then Add Item dialog.

Choose a category and optionally the amount of items to be created at once. You can also set some attributes that are common to the items being inserted. The `Generic item` can be used to represent anything without having to create a new category. Usually it is used as a container, and may represent, for example, a place (John Doe's) or a document (Investigation Request 055). To group items you can also use the `set` item, which is a generic set. That way, to group 154 floppies, you can create a set and 154 floppies under it.

In the example shown in figure 2.5, we have created 5 floppy disks.

Alternatively, since release 0.5 you can drag and drop a file directly into case, on any item, to create a file object (figure 2.6). Some files have special meaning when dragged. For example, the `.log` file generated by Logicube Talon^{TM} is parsed and instead of a file item, a `harddisk` item is created, with some attributes filled, and associated to a datasource of type `talon`.



property	value
UID	6
category	floppy
creation time	2011-12-05 12:13:47
last modification time	2011-12-05 12:13:47
ID	
name	Floppy disk 5
vendor	
media format	
sides	
density	
formatted capacity	
real capacity	
number of tracks	
sectors per track	
number of sectors	
transfer rate	
additional information	

Figure 2.5: case with 5 floppy disks

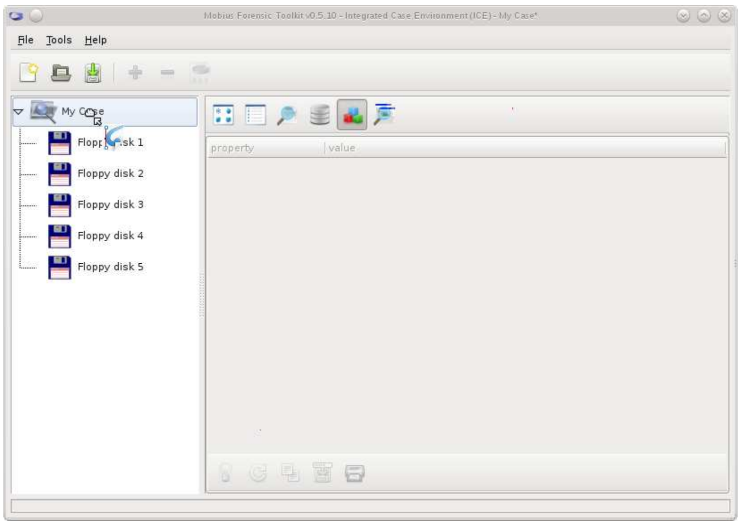


Figure 2.6: add item through drag and drop

3
Managing datasources

Each case item can have an associated datasource. A datasource is an object that represents a stream of bytes.

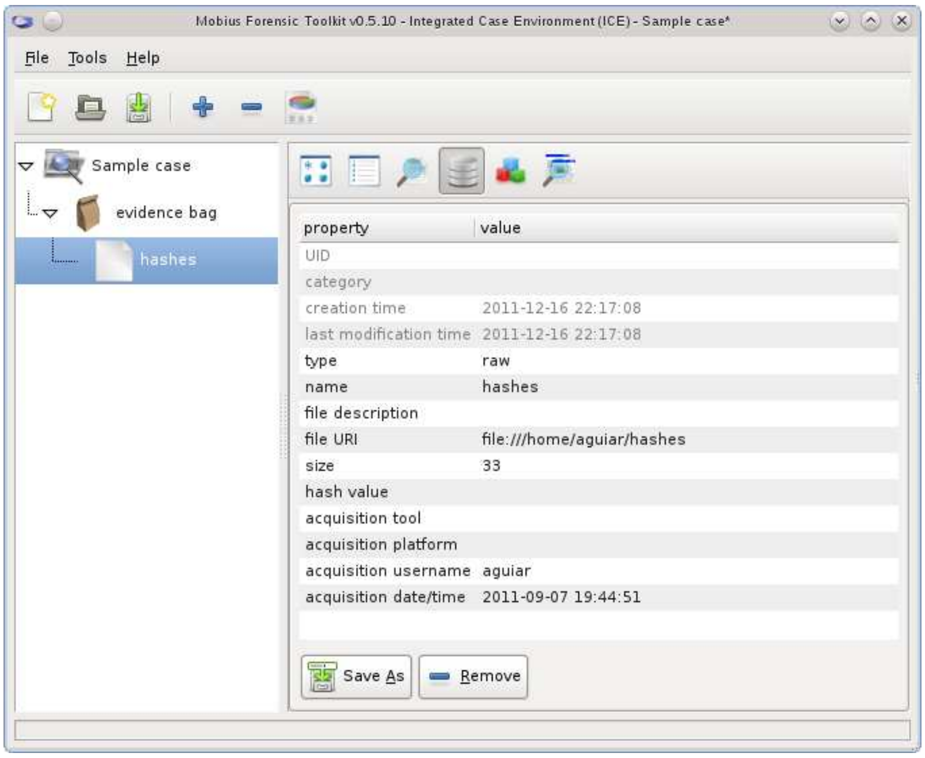
In section 2.2 we saw how to drag and drop a file to create an item. In fact, two objects were created: an item and an associated `raw` datasource, pointing to the dragged file path.

To manage datasources, click on `datasource view mode` icon. This view can be used to associate a datasource to and disassociate from an item, change datasource attributes, and export datasources (see figure 3.1).

Click on an item to see the datasource and its attributes. You can change the datasource associated to an item by dragging and dropping a new file directly into Data Sourcerer window.

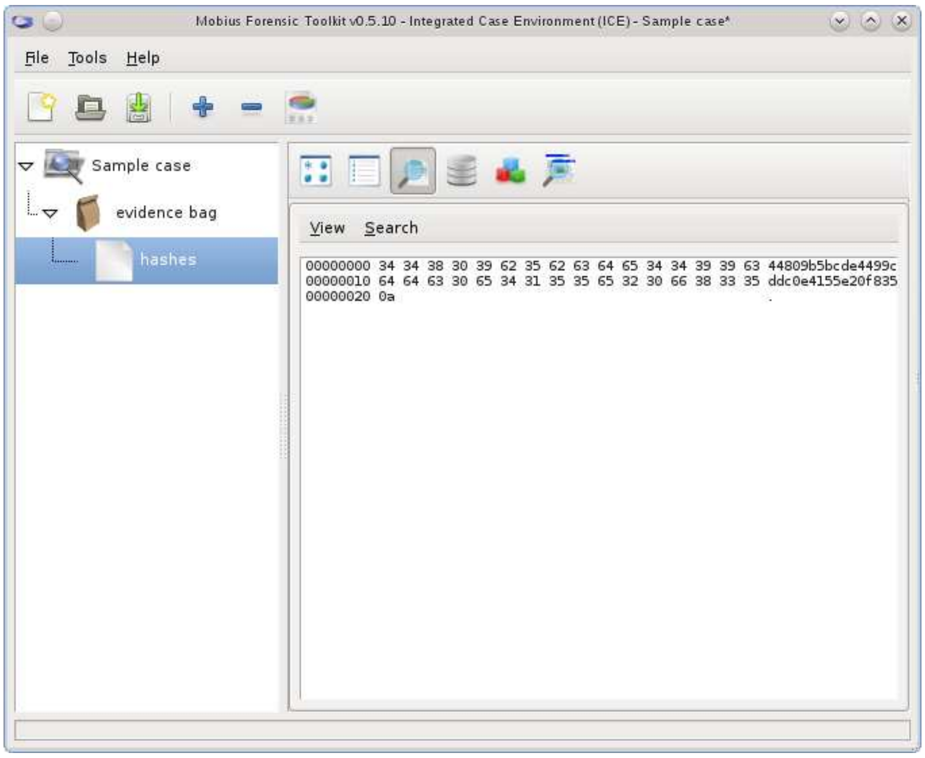
3.1 Data view mode

Once a case item has an associated datasource you can see its data using the data view mode (figure 3.2). To jump to a certain address, hit `CTRL+J`, and enter an address, or any valid python expression, like: `0x2000 * 35 + 43`, and hit `OK`.


Mobius Forensic Toolkit v0.5.10 - Integrated Case Environment (ICE) - Sample case*
File
Tools
Help
Sample case
evidence bag
hashes
Save As
Remove

property	value
UID	
category	
creation time	2011-12-16 22:17:08
last modification time	2011-12-16 22:17:08
type	raw
name	hashes
file description	
file URI	file:///home/aguiar/hashes
size	33
hash value	
acquisition tool	
acquisition platform	
acquisition username	aguiar
acquisition date/time	2011-09-07 19:44:51

Figure 3.1: datasource view mode


Mobius Forensic Toolkit v0.5.10 - Integrated Case Environment (ICE) - Sample case*
File
Tools
Help
Sample case
evidence bag
hashes
View
Search

```
00000000 34 34 38 30 39 62 35 62 63 64 65 34 34 39 39 63 44809b5bcde4499c
00000010 64 64 63 30 65 34 31 35 35 65 32 30 66 38 33 35 ddc0e4155e20f835
00000020 0a                                              .
```

Figure 3.2: data view mode

4 Managing categories

The Category Manager extension is used to create, modificate and delete both categories and their attributes on the fly (figure 4.1).

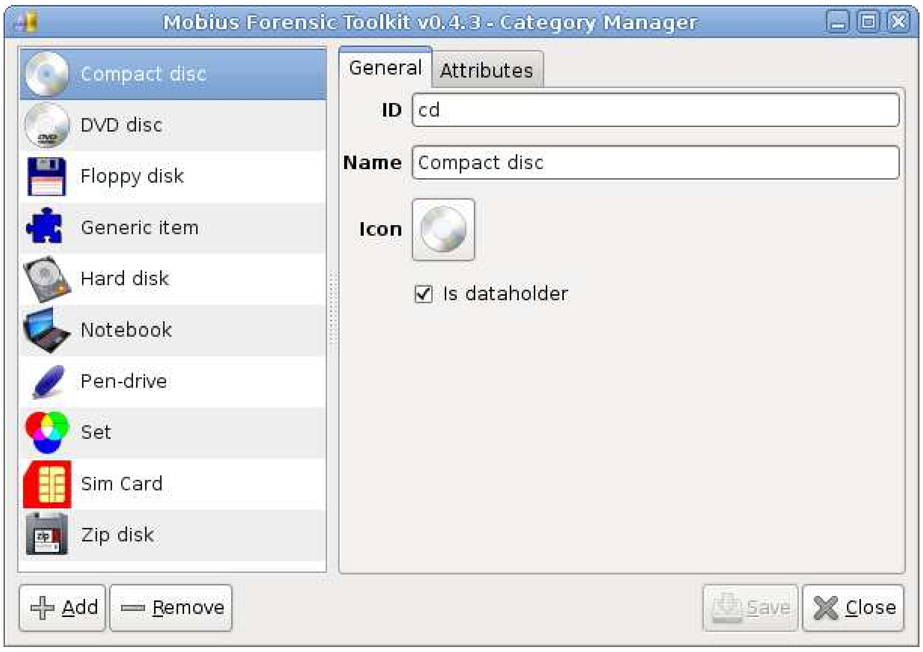
4.1 Creating categories

To create a category, hit `add` button below category listview (leftmost button). A new category named `<NEW CATEGORY>` is created. Now click on it to edit its icon, ID and name.

To edit its attributes, click on `Attributes` tab folder.

After modifications, click `save` button. Now this category will be available for all cases, and items of that type can be added to the current case.

You can also use Category Manager to modify existing categories and its attributes, and even to translate attributes descriptions to your language, as long as you keep their IDs from changing. You can add attributes to an existing category or even remove some attributes.



ID: cd
Name: Compact disc
☑ Is dataholder

Figure 4.1: Category Manager extension

5 Managing parts

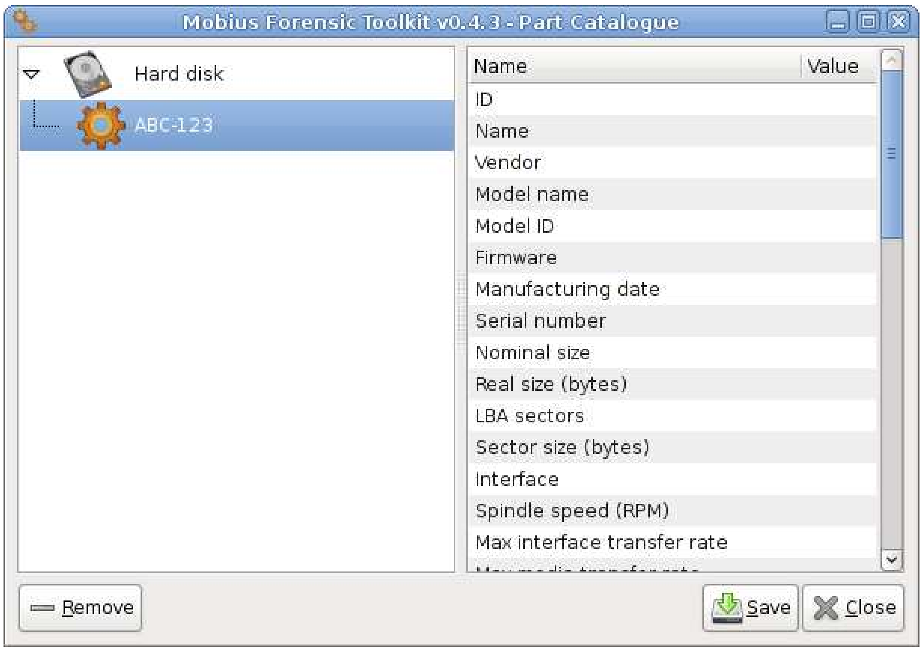
The Part Catalogue extension was created to fulfill attributes of common parts. If you have harddisks with part-number `ABC-123`, you can fill the attributes which are common to this kind of harddisk, lefting the ones that are device dependent blank.

5.1 Automatic startup

Part Catalogue is started everytime you fill an attribute whose ID is `part_id`. If this part-id is already recorded in Part Catalogue database, it will fulfill item attributes with those attributes you have set to this part. If not, it will open a window to register this new part and its attributes.

To test this, add a harddisk to current case, jump to the attributes view mode and change Part ID to **ABC-123**. The Part Catalogue will open a window like the one shown in figure 5.1.

Enter attributes which are common to this part number and hit `save` button. The next time you enter a harddisk with part ID **ABC-123**, the Part Catalogue extension will automatically fill its attributes.



Name	Value
ID	
Name	
Vendor	
Model name	
Model ID	
Firmware	
Manufacturing date	
Serial number	
Nominal size	
Real size (bytes)	
LBA sectors	
Sector size (bytes)	
Interface	
Spindle speed (RPM)	
Max interface transfer rate	

Figure 5.1: Part Catalogue extension

6 Browsing registry files

The Hive extension is used to browse registry files. To start Hive, click on `tools→Hive` menu option. Once the Hive extension is opened you can add registry files to it either by dragging and dropping them into file listview area or by selecting the `add files` option (figures 6.1 and 6.2).

After adding files to the registry, you can view both the logical registry structure (figure 6.3) and the physical file structure (figure 6.4).

The "report view" shows the reports that are available (figure 6.5). In any report you can drag the `information` icon (the leftmost icon at the bottom toolbar) into the case tree view, creating a report-data object (figure 6.6). The report-data objects can be visualized using the Report Viewer extension and the data can be copied to the clipboard or exported to an .xml file using the other options available at the bottom toolbar.

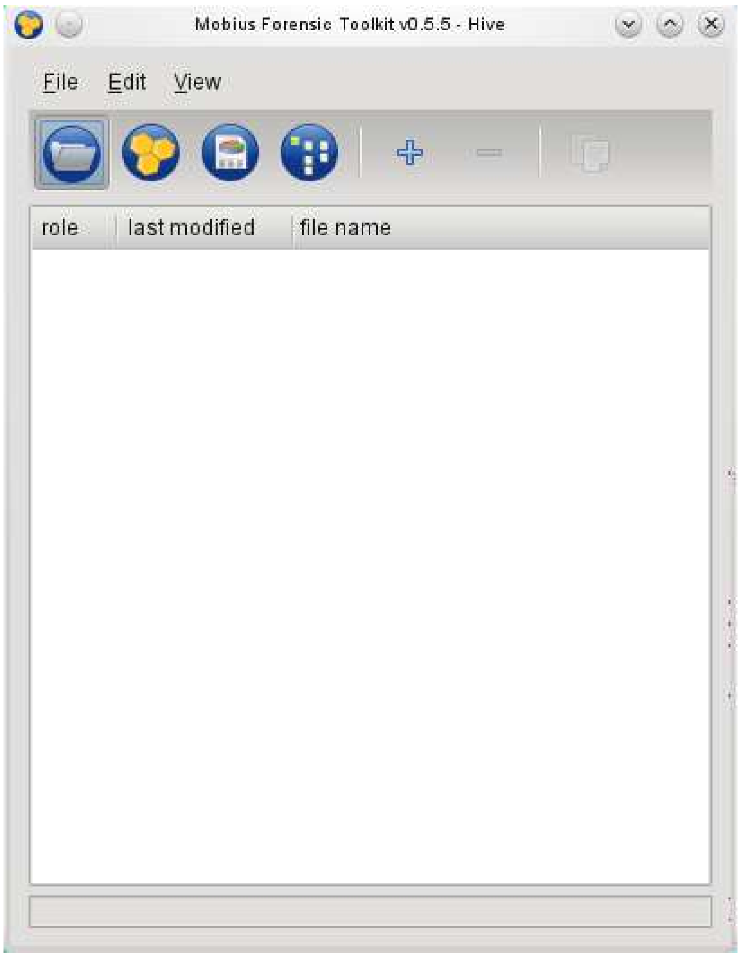
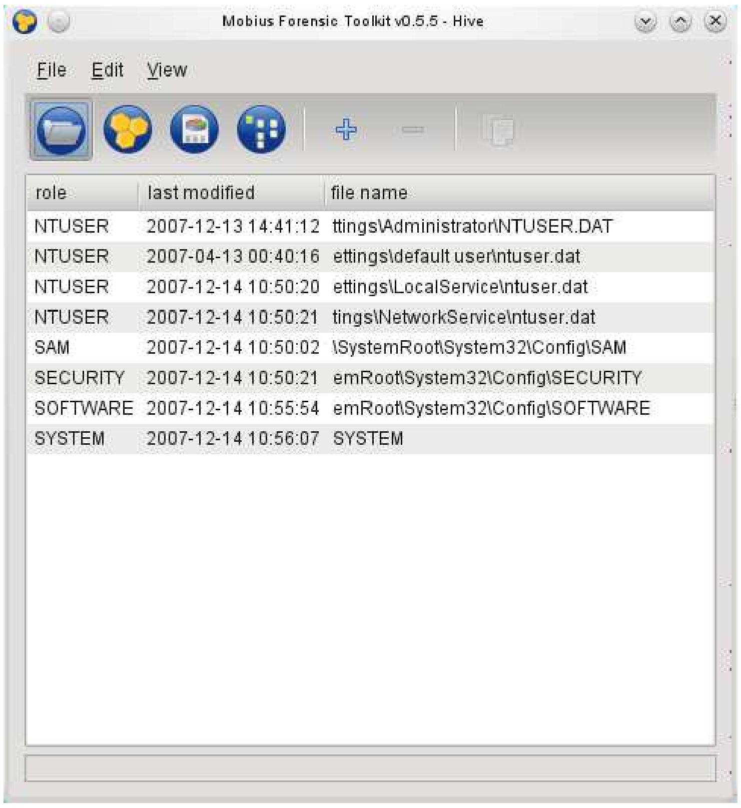
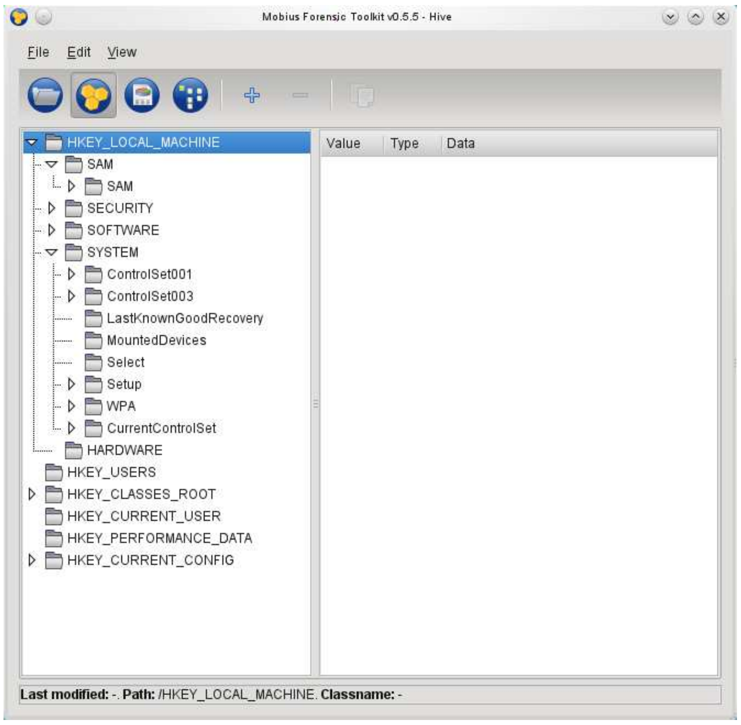


Figure 6.1: The Hive extension — file list view



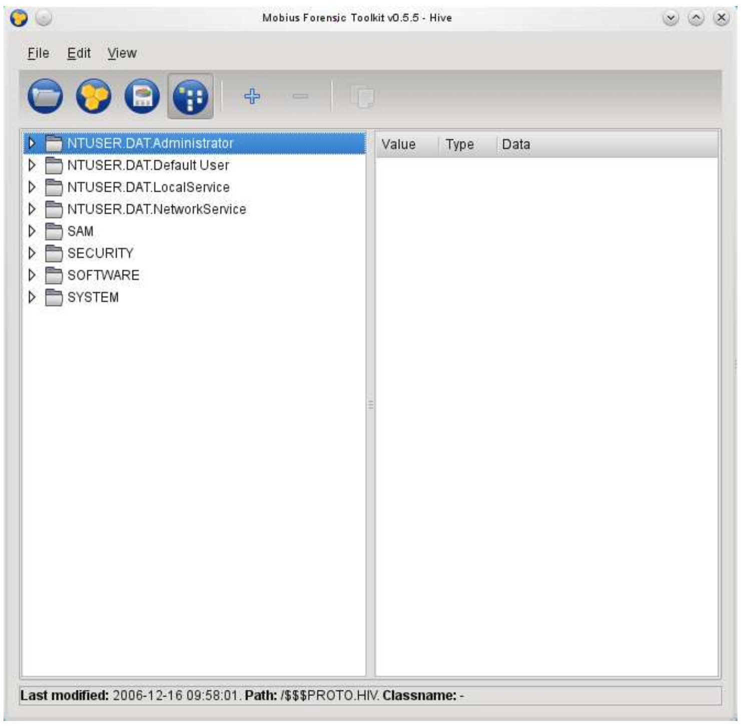
role	last modified	file name
NTUSER	2007-12-13 14:41:12	ttings\Administrator\NTUSER.DAT
NTUSER	2007-04-13 00:40:16	ettings\default user\ntuser.dat
NTUSER	2007-12-14 10:50:20	ettings\LocalService\ntuser.dat
NTUSER	2007-12-14 10:50:21	tings\NetworkService\ntuser.dat
SAM	2007-12-14 10:50:02	\SystemRoot\System32\Config\SAM
SECURITY	2007-12-14 10:50:21	emRoot\System32\Config\SECURITY
SOFTWARE	2007-12-14 10:55:54	emRoot\System32\Config\SOFTWARE
SYSTEM	2007-12-14 10:56:07	SYSTEM

Figure 6.2: The Hive extension — showing registry files



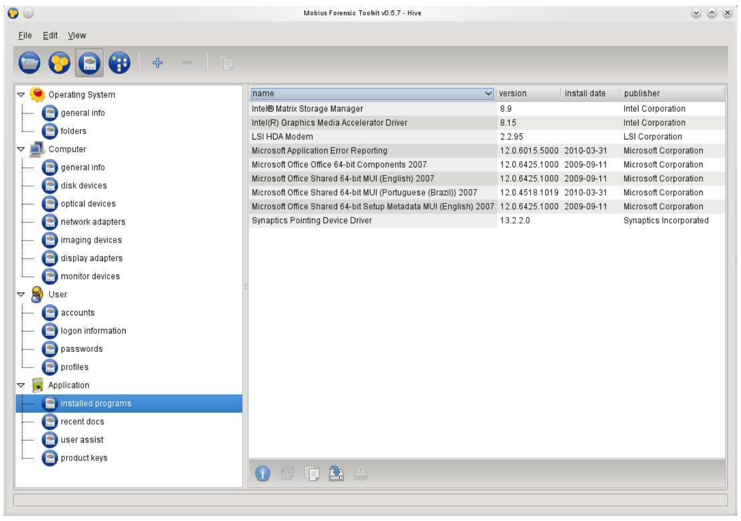
Last modified: -. **Path:** /HKEY_LOCAL_MACHINE. **Classname:** -

Figure 6.3: The Hive extension — logical registry view



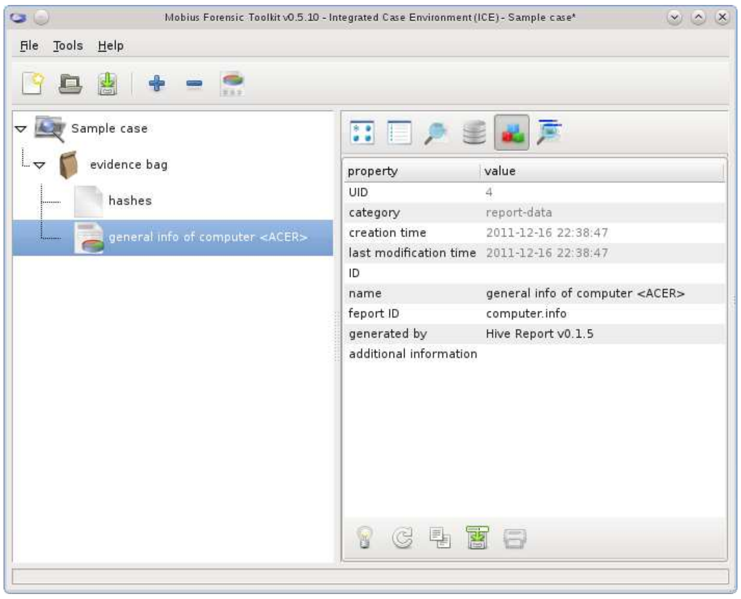
Last modified: 2006-12-16 09:58:01. **Path:** /$$$PROTO.HIV. **Classname:** -

Figure 6.4: The Hive extension — physical registry view



name	version	install date	publisher
Intel® Matrix Storage Manager	8.9		Intel Corporation
Intel(R) Graphics Media Accelerator Driver	8.15		Intel Corporation
LSI HDA Modem	2.2.95		LSI Corporation
Microsoft Application Error Reporting	12.0.6015.5000	2010-03-31	Microsoft Corporation
Microsoft Office Office 64-bit Components 2007	12.0.6425.1000	2009-09-11	Microsoft Corporation
Microsoft Office Shared 64-bit MUI (English) 2007	12.0.6425.1000	2009-09-11	Microsoft Corporation
Microsoft Office Shared 64-bit MUI (Portuguese (Brazil)) 2007	12.0.4518.1019	2010-03-31	Microsoft Corporation
Microsoft Office Shared 64-bit Setup Metadata MUI (English) 2007	12.0.6425.1000	2009-09-11	Microsoft Corporation
Synaptics Pointing Device Driver	13.2.2.0		Synaptics Incorporated

Figure 6.5: The Hive extension — report view


Mobius Forensic Toolkit v0.5.10 - Integrated Case Environment (ICE) - Sample case*
File
Tools
Help
Sample case
evidence bag
hashes
general info of computer <ACER>

property	value
UID	4
category	report-data
creation time	2011-12-16 22:38:47
last modification time	2011-12-16 22:38:47
ID	
name	general info of computer <ACER>
feport ID	computer.info
generated by	Hive Report v0.1.5
additional information	

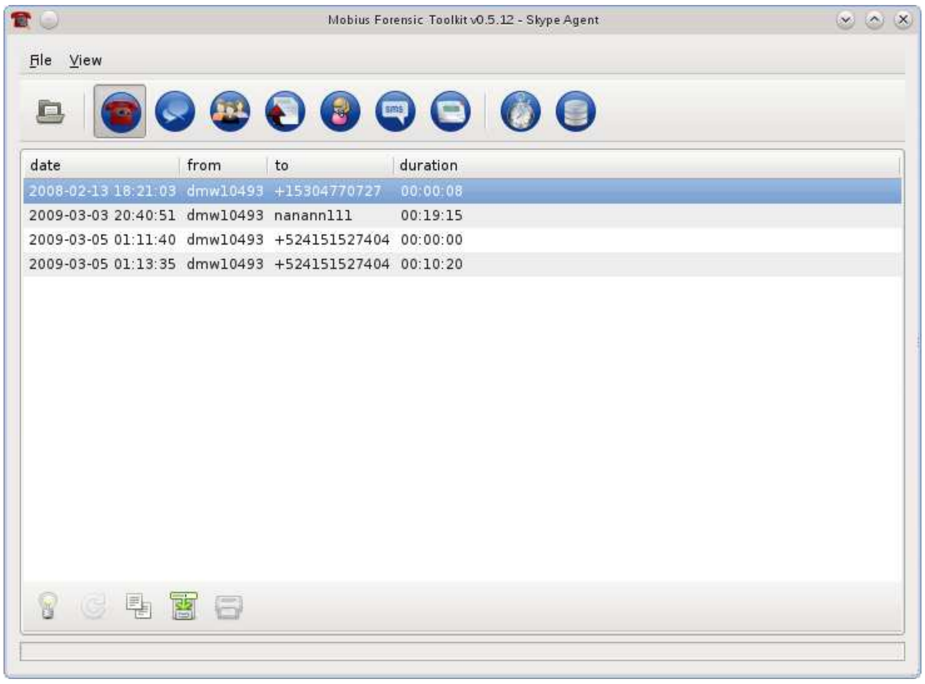
Figure 6.6: The Hive extension — Report data dragged into case treeview

7 Browsing Skype log files

The Skype Agent extension is used to browse Skype log files. To start Skype Agent, click on `tools`→`Skype Agent` menu option. Once the Skype Agent extension is opened you can open Skype log files by selecting the `open` option (see figure 7.1).

Skype log files are usually named `main.db` and are located under the folder `ApplicationData/Skype`, inside user's profile folder. The best way to use Skype Agent is to open all the log files related to a Skype account at once.

The Skype Agent extension has views for: calls, chats, contacts, file transfers, profile data, SMS and voicemails. It also has a timeline view that reports all those entries ordered by date and also a record raw mode view, that shows records and tags.



Mobius Forensic Toolkit v0.5.12 - Skype Agent

date	from	to	duration
2008-02-13 18:21:03	dmw10493	+15304770727	00:00:08
2009-03-03 20:40:51	dmw10493	nanann111	00:19:15
2009-03-05 01:11:40	dmw10493	+524151527404	00:00:00
2009-03-05 01:13:35	dmw10493	+524151527404	00:10:20

Figure 7.1: The Skype Agent extension

8 Generating reports

The Report Wizard extension is used to create report templates. Report templates are indeed programs that are able to generate a report, based upon a data model.

Once a report template is created, it can be executed using the current case as data model.

8.1 Creating a report template

Click on `tools`→`Report Wizard` menu option. A window like the one shown in figure 8.1 will be opened.

Click either on `new report` icon or `file->new` menu option to create a new report template. A pop-up window will appear. Enter an unique ID for this template. In this example, use `myreport` as ID.

By dragging items from the component pallette at the bottom part of window, you can compose a rather complex program, with `if`, `else`, `for`, among other components (figure 8.2).

In this example, compose a report template akin to that shown in figure 8.3. Do not forget to click on `save` option to save the report template.

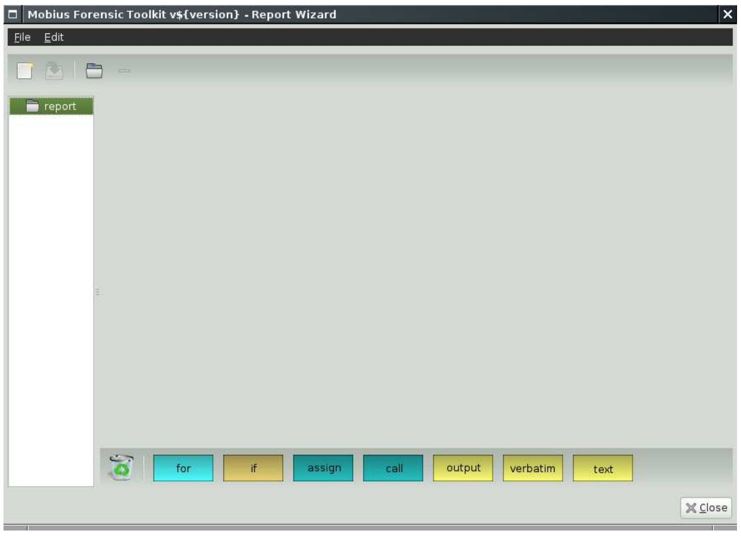
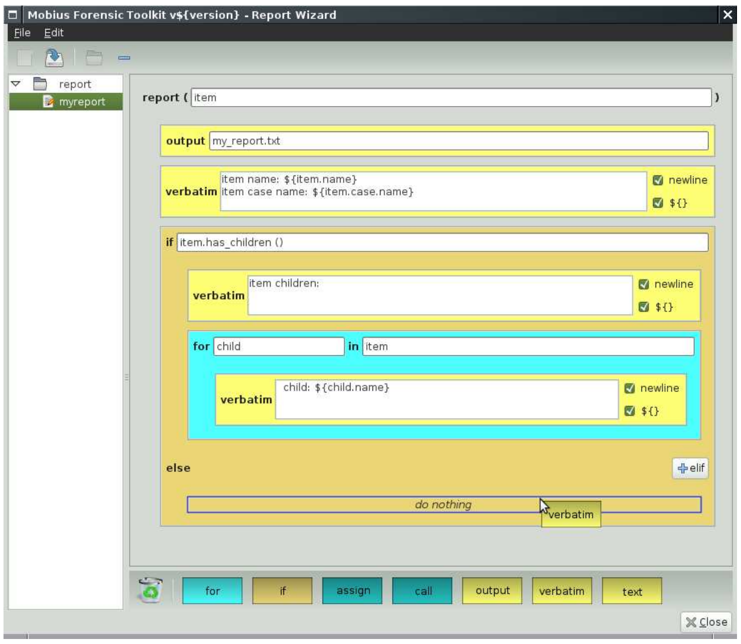


Figure 8.1: Report Wizard running

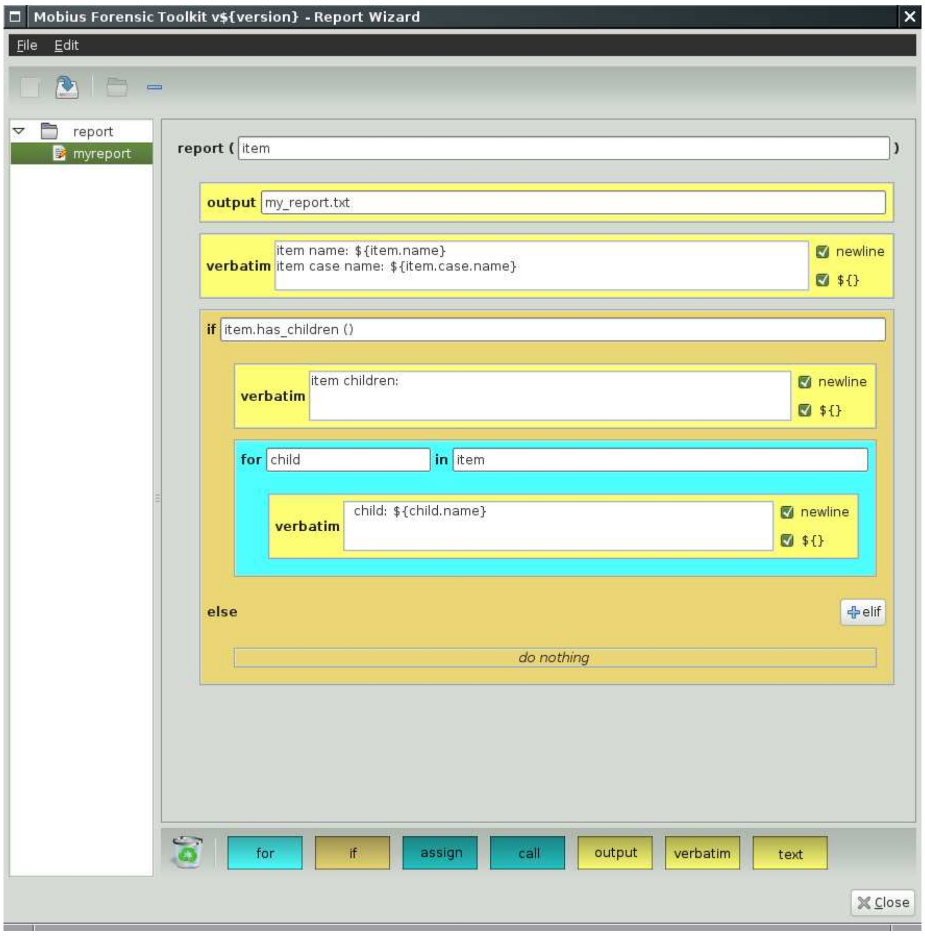


```
report ( item )
output my_report.txt
verbatim  item name: ${item.name}
          item case name: ${item.case.name}      newline  ${}
if item.has_children ()
    verbatim  item children:                     newline  ${}
    for child in item
        verbatim  child: ${child.name}           newline  ${}
else                                             elif
    do nothing
```

Figure 8.2: Dragging components to the report template

8.2 Running a report template

Once you have created a report template, you can run it using the selected case item as input.



report (item)

output my_report.txt

verbatim
item name: ${item.name}
item case name: ${item.case.name}
✓ newline
✓ ${}

if item.has_children ()

verbatim
item children:
✓ newline
✓ ${}

for child in item

verbatim
child: ${child.name}
✓ newline
✓ ${}

do nothing

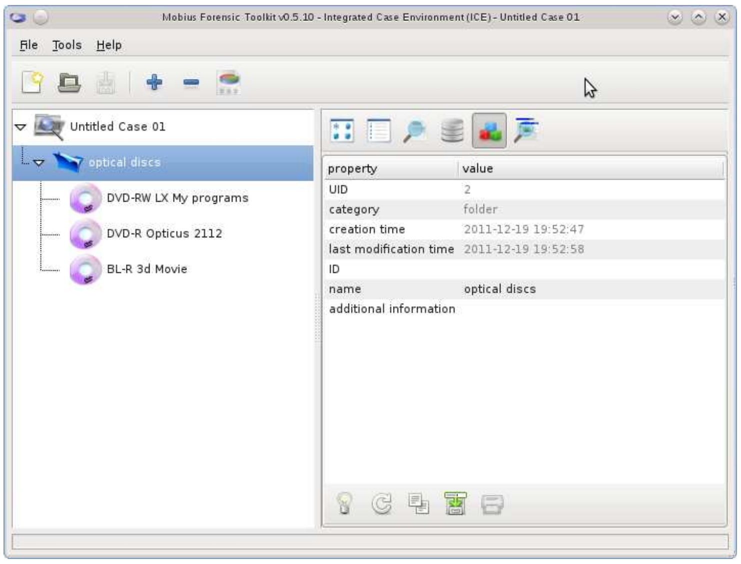
Figure 8.3: `myreport` template

Open your case, or create a new one, like the one shown in figure 8.4.

Select an item and click on `run report` icon. A dialog like the one shown in 8.5 will be shown. Select the output directory, and enter the report template ID.

The report template will be run using the selected item as input. Running the `myreport` template created before in section 8.1, it will output a file named `my_report.txt` at output directory, with the following content:

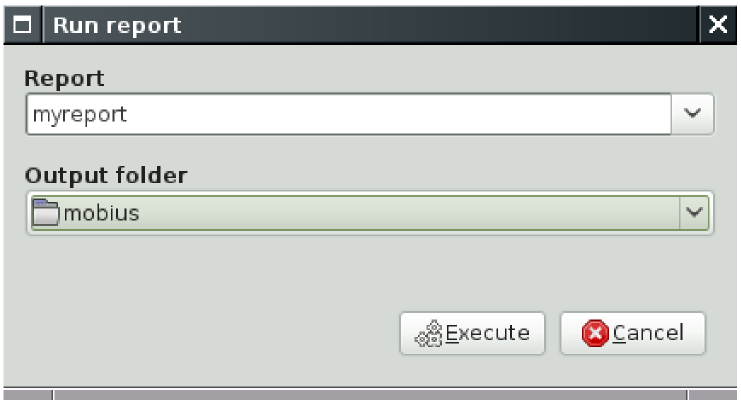
```
item name: Optical discs
item case name: Untitled Case #01
item children:
  child: DVD-RW LX My programs
  child: DVD-R Opticus 2112
  child: BL-R 3d Movie
```


Mobius Forensic Toolkit v0.5.10 - Integrated Case Environment (ICE) - Untitled Case 01

- Untitled Case 01
  - optical discs
    - DVD-RW LX My programs
    - DVD-R Opticus 2112
    - BL-R 3d Movie

property	value
UID	2
category	folder
creation time	2011-12-19 19:52:47
last modification time	2011-12-19 19:52:58
ID	
name	optical discs
additional information	

Figure 8.4: Sample case


Run report

Report: myreport

Output folder: mobius

Figure 8.5: Report dialog

9
Imaging floppy disks

The Floppy Imager extension was designed to create logical image files from floppy disks and also to collect their metadata as well. It runs only in Linux systems. To run, `/dev/fd0` must have permission **0666**.

9.1 Running Floppy Imager

To start Floppy Imager, click on `tools`→`Floppy Imager` menu option. A window like the one shown in figure 9.1 will be opened. The Floppy Imager is only active when you select a floppy disk inside case treeview. Any other item will handle Floppy Imager inactive.

Click on any floppy item, insert a floppy into drive `/dev/fd0` and hit the `retrieve` button. The Floppy Imager will collect floppy metadata, filling item's attributes according. Each block on sector map represents a sector. Gray blocks are undefined, blue ones are sectors successfully read and red are bad sectors (figure 9.2).

A floppy disk can be imaged more than once. If you select an already imaged floppy disk and hit the `retrieve`, Floppy Imager will try to read only bad sectors. Usually, if you eject and re-insert floppy disk, the Floppy Imager will recover some bad sectors.

Floppy imagefiles are saved at folder `casedir/image`, where `casedir` is the

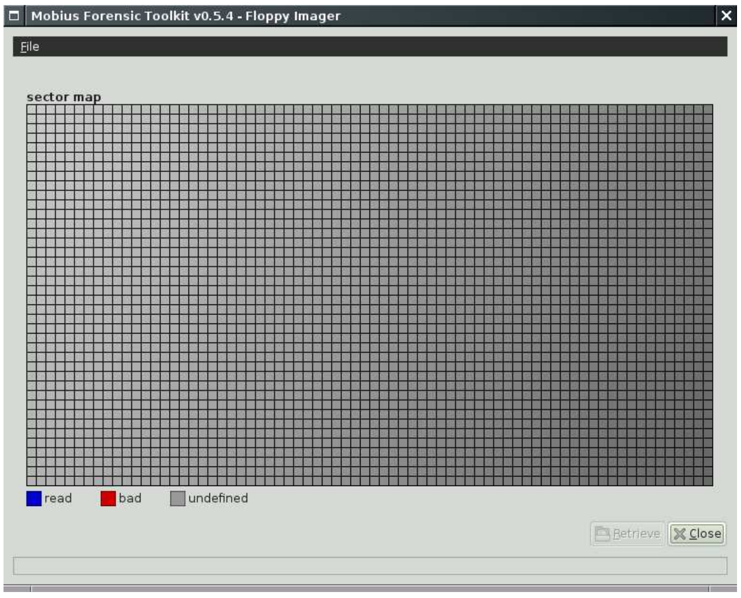
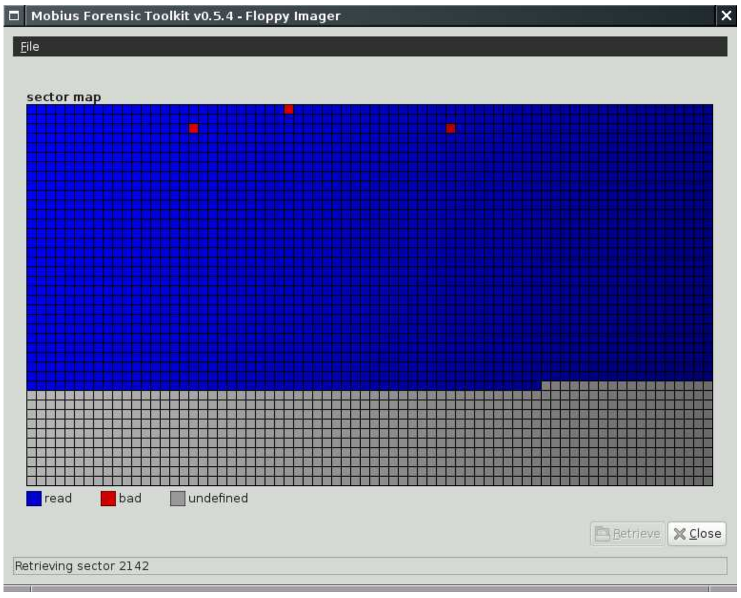


Figure 9.1: Floppy Imager extension



Retrieving sector 2142

Figure 9.2: The Floppy Imager running

case base folder.

10 Developing extensions

The Mobius Forensic Toolkit is implemented through extensions. Each extension is a separated program that runs in its own independent namespace. The Extension Builder is an extension that was specifically made to edit extensions. It is a complete IDE that handles the underlying extensions and services structure, with code editing capabilities.

To start Extension Builder, click on `tools`→`Extension Builder` menu option. A window like the one shown in figure 10.1 will be opened.

10.1 Opening an extension

After you have started Extension Builder, click on `Open` menu option or on the corresponding icon in the toolbar, to open an extension.

Mobius Forensic Toolkit distribution files (`.tar.gz`, `.tar.bz2`, or `.zip`) have a directory named `extensions` where you can find all extensions that are distributed inside those packages. Feel free to open those extensions, and even to create new ones based upon their source codes. In this example, we have selected all extensions from `extensions` directory (figure 10.2).

To use an extension you have modified, you must install it using Mobius main window `tools` option.

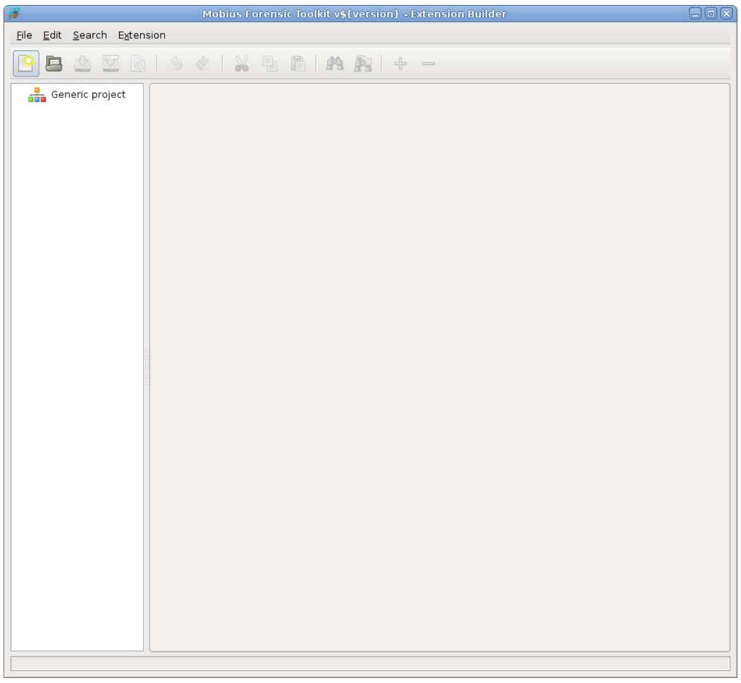
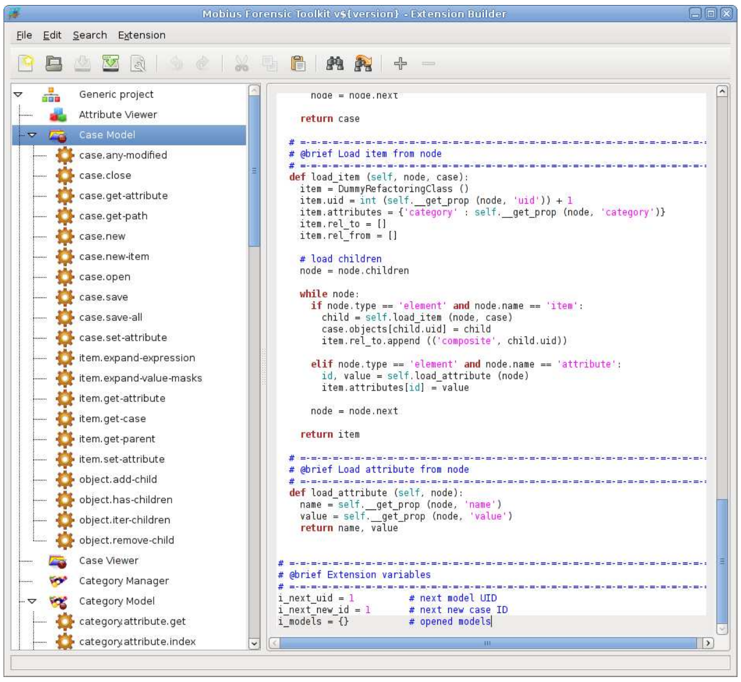

Mobius Forensic Toolkit v${version} - Extension Builder
File Edit Search Extension
Generic project

Figure 10.1: Extension Builder running

10.2 Creating a new extension

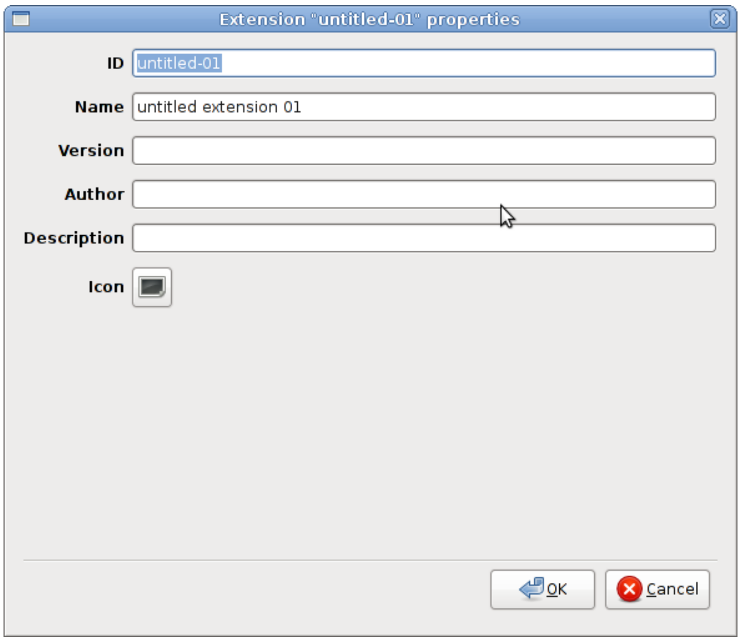
As told before, you can open an existing extension, modify its source codes and save it as a new extension. But you can also start with a fresh new one. Click on `New` menu option or on the corresponding icon at toolbar, to create an extension.

Change your extension properties using `properties` option, and it will open up a dialog (figure 10.3).



```
      node = node.next

    return case

  # =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
  # @brief Load item from node
  # =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
  def load_item (self, node, case):
    item = DummyRefactoringClass ()
    item.uid = int (self.__get_prop (node, 'uid')) + 1
    item.attributes = {'category' : self.__get_prop (node, 'category')}
    item.rel_to = []
    item.rel_from = []

    # load children
    node = node.children

    while node:
      if node.type == 'element' and node.name == 'item':
        child = self.load_item (node, case)
        case.objects[child.uid] = child
        item.rel_to.append (('composite', child.uid))

      elif node.type == 'element' and node.name == 'attribute':
        id, value = self.load_attribute (node)
        item.attributes[id] = value

      node = node.next

    return item

  # =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
  # @brief Load attribute from node
  # =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
  def load_attribute (self, node):
    name = self.__get_prop (node, 'name')
    value = self.__get_prop (node, 'value')
    return name, value


# =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# @brief Extension variables
# =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
i_next_uid = 1          # next model UID
i_next_new_id = 1       # next new case ID
i_models = {}           # opened models
```

Figure 10.2: Extension Builder showing extensions



ID	untitled-01
Name	untitled extension 01
Version	
Author	
Description	
Icon	

Figure 10.3: Extension Builder properties dialog