



Tutorial

1.14

Mobius Forensic Toolkit

(c) 2008-2019 Eduardo Aguiar

Contents

1	Introduction	1
2	Setting up your case	3
2.1	Creating a case	3
2.2	Adding items to case	4
3	Managing datasources	7
3.1	Data view mode	7
4	Managing categories	9
4.1	Creating categories	10
5	Managing parts	11
5.1	Automatic startup	11
6	Browsing registry files	13
7	Browsing Skype log files	17
8	Generating reports	19
8.1	Creating a report template	19
8.2	Running a report template	19
9	Imaging floppy disks	25
9.1	Running Floppy Imager	25

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 C++ and Python, using PYGTK and PyCairo. It is very extensible through extensions, 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 a 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).



Figure 2.1: Mobius main window

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

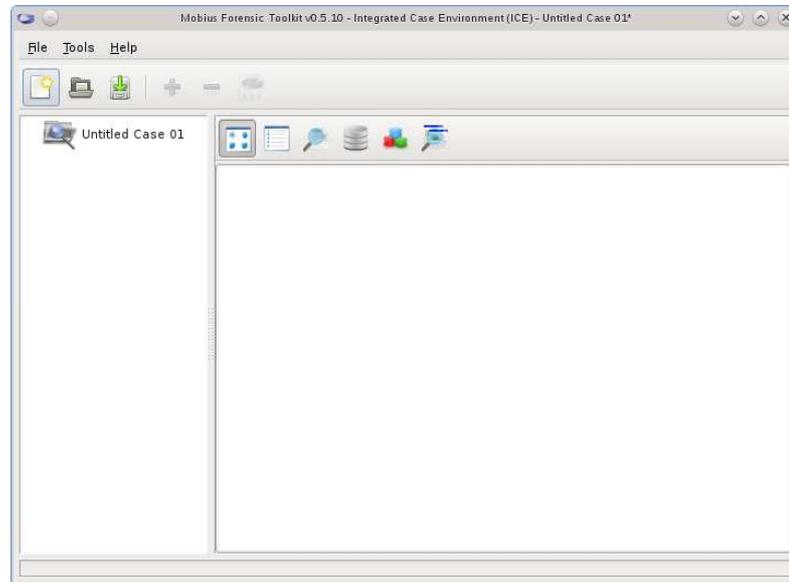


Figure 2.2: new case window

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. The **base folder** attribute is the folder 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.

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 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. Therefore, 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™ 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**.

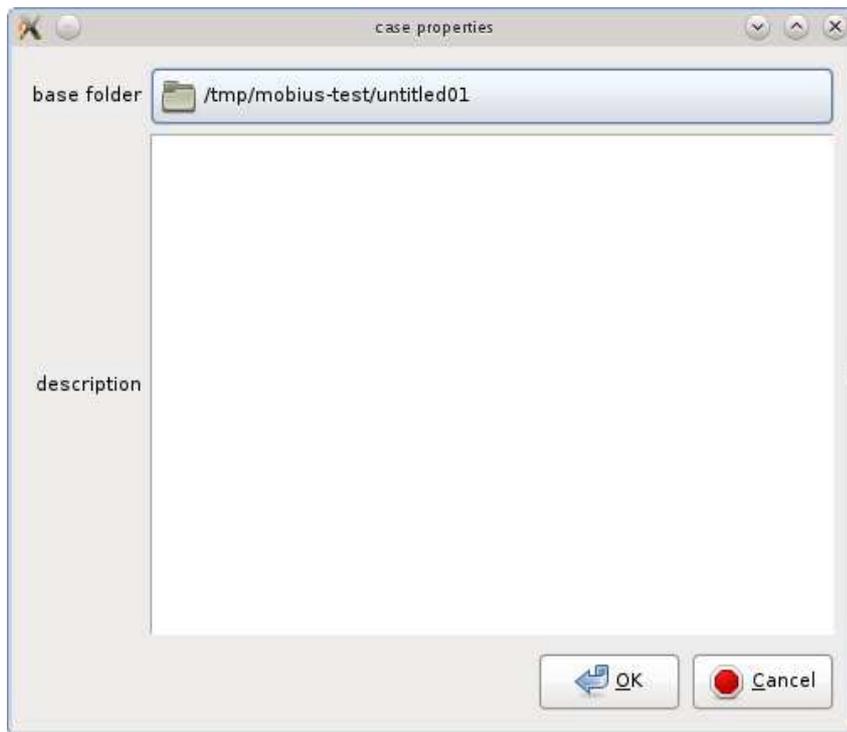


Figure 2.3: new case properties dialog

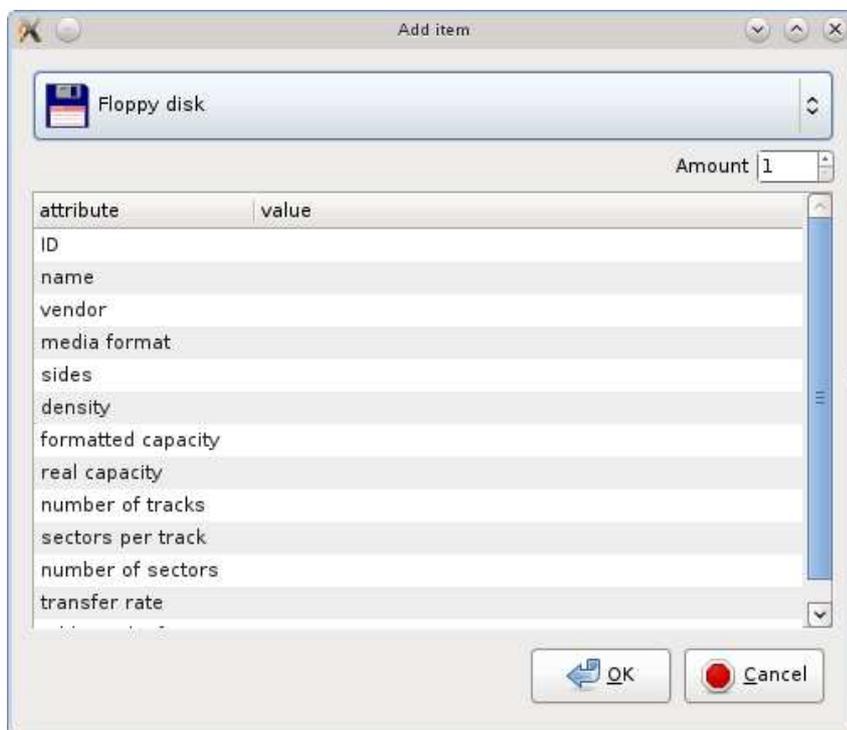


Figure 2.4: add item dialog

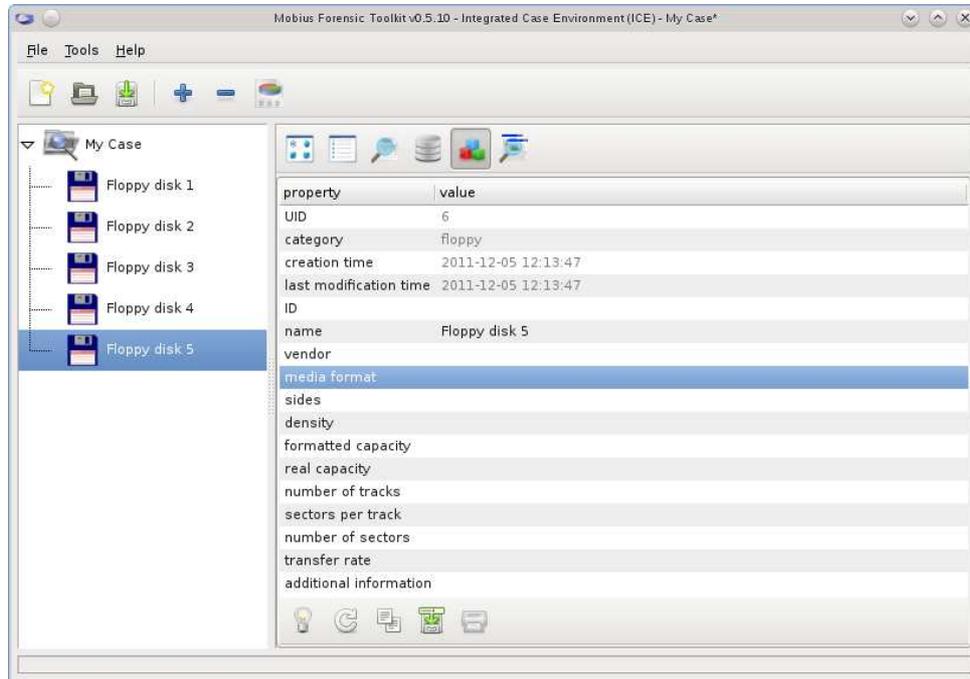


Figure 2.5: case with 5 floppy disks

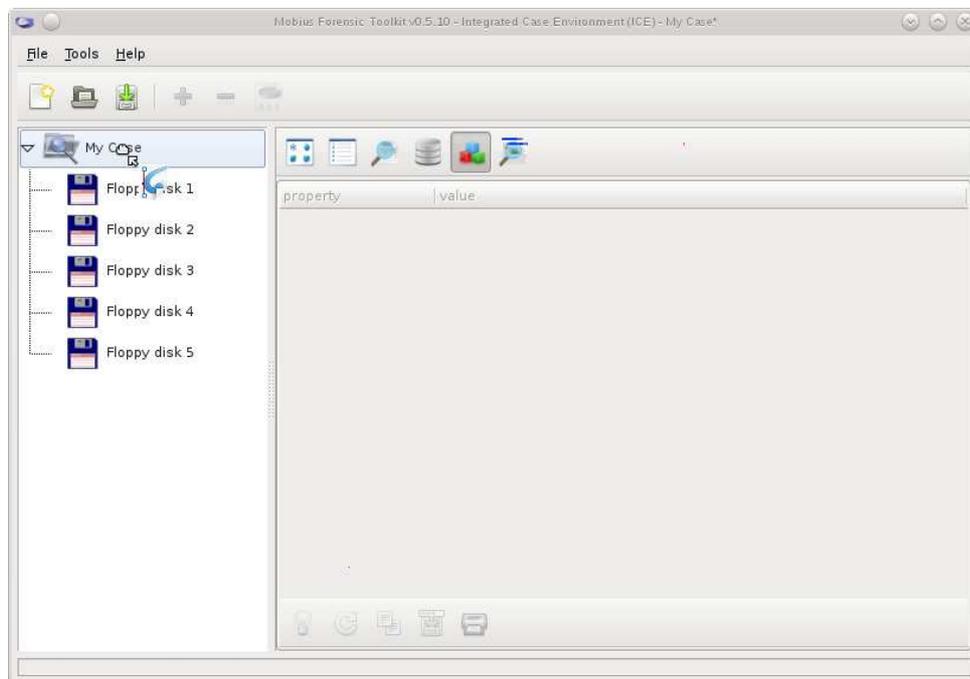


Figure 2.6: add item through drag and drop

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 `imagefile` datasource, pointing to the dragged file URL.

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 use the same dialog to change the datasource.

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**.

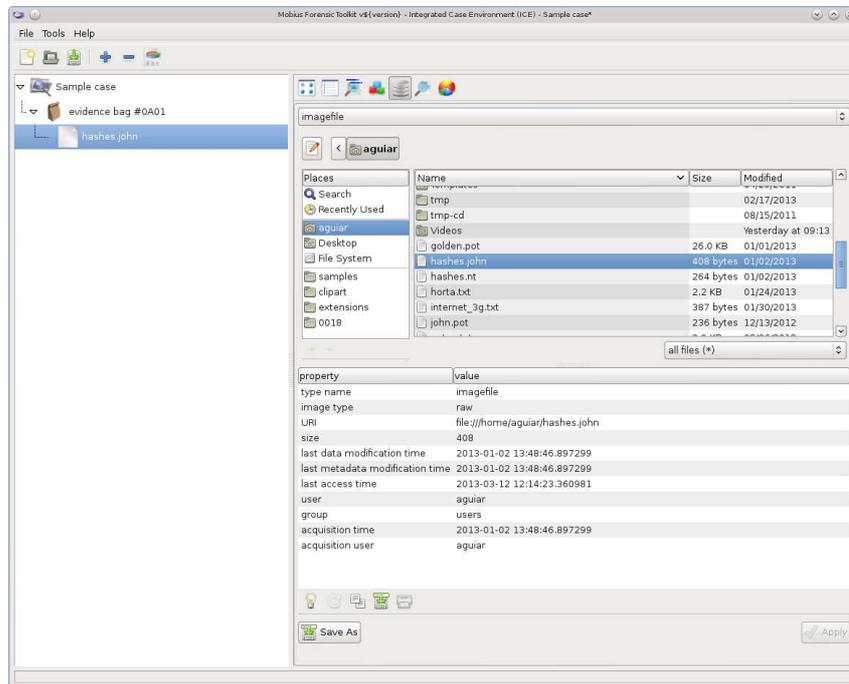


Figure 3.1: datasource view mode

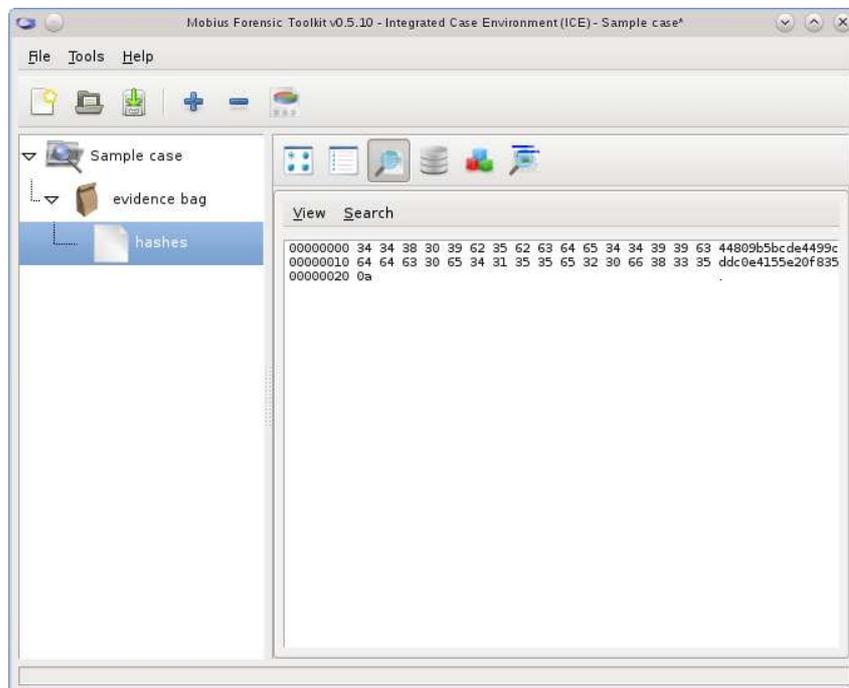


Figure 3.2: data view mode

4

Managing categories

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



Figure 4.1: Category Manager extension

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.

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, leaving 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.



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 (figure 6.1).

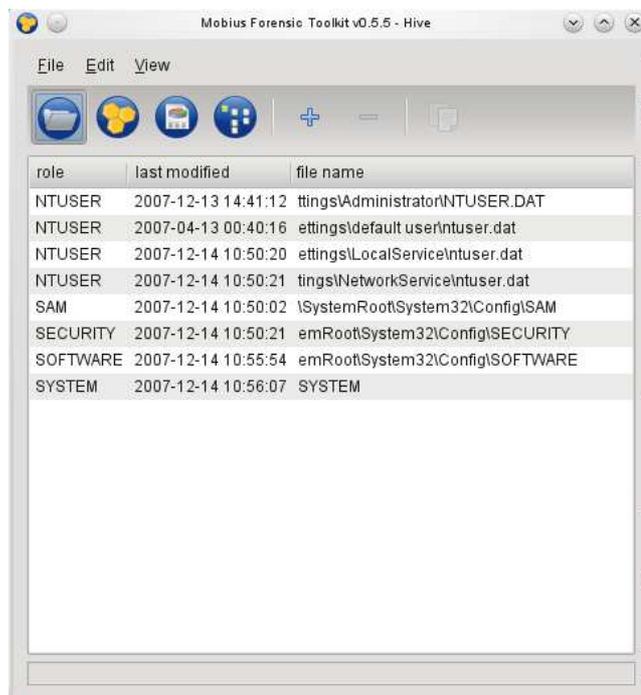


Figure 6.1: The Hive extension — showing registry files

After adding files to the registry, you can view both the logical registry structure (figure 6.2) and the physical file structure (figure 6.3).

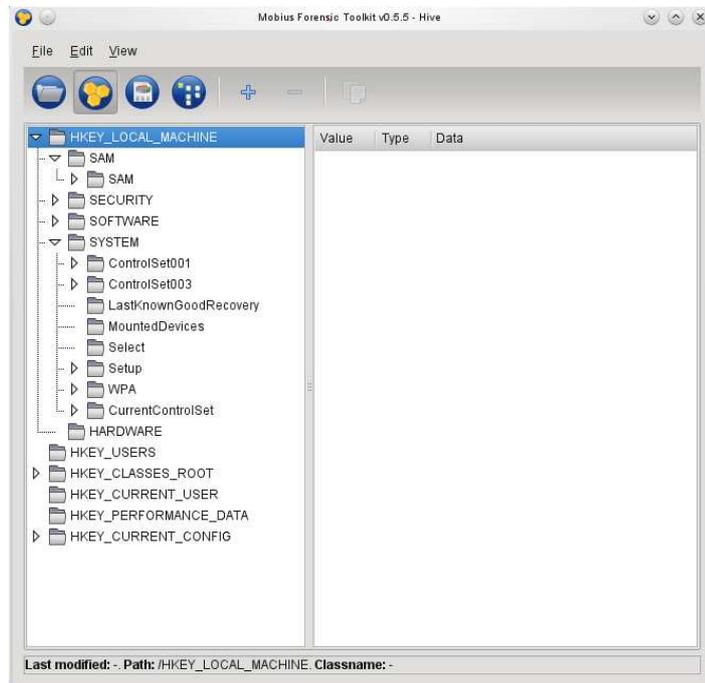


Figure 6.2: The Hive extension — logical registry view

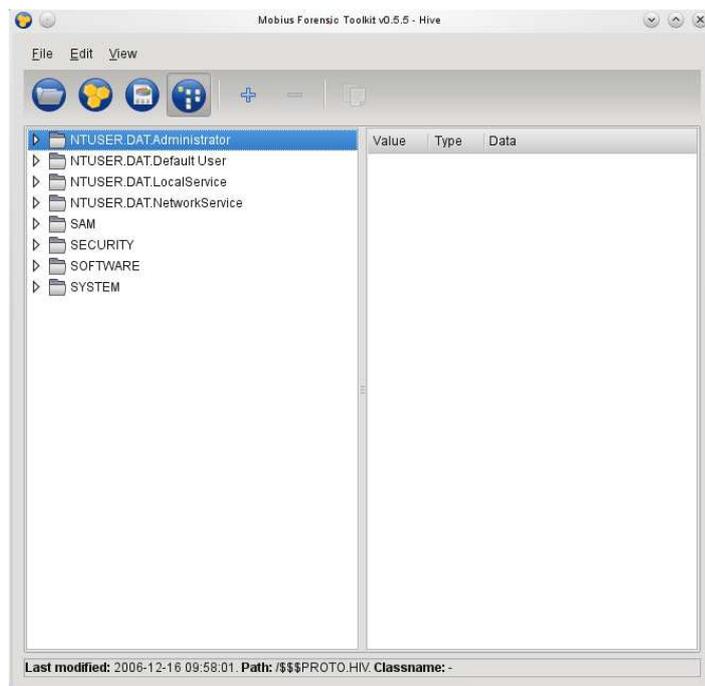


Figure 6.3: The Hive extension — physical registry view

The “report view” shows the reports that are available (figure 6.4). 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 6.5). 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.

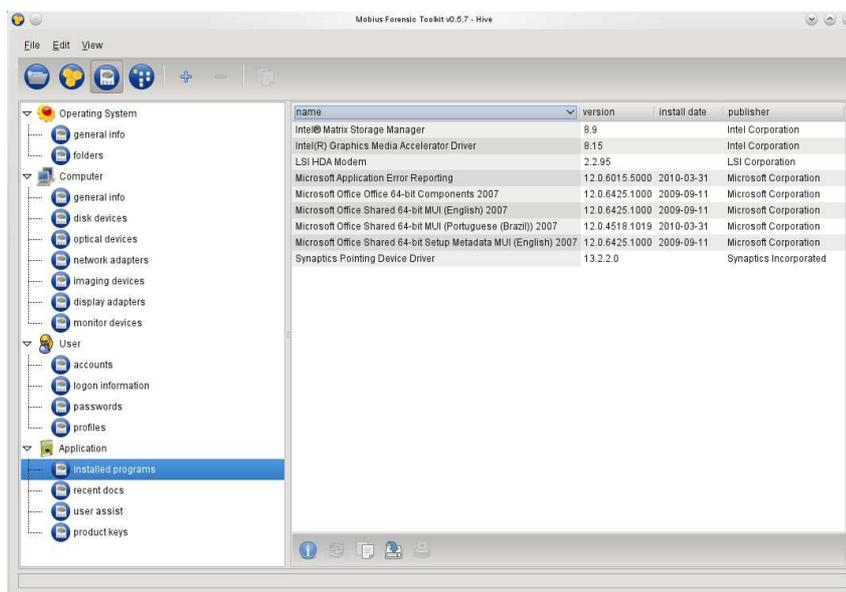


Figure 6.4: The Hive extension — report view

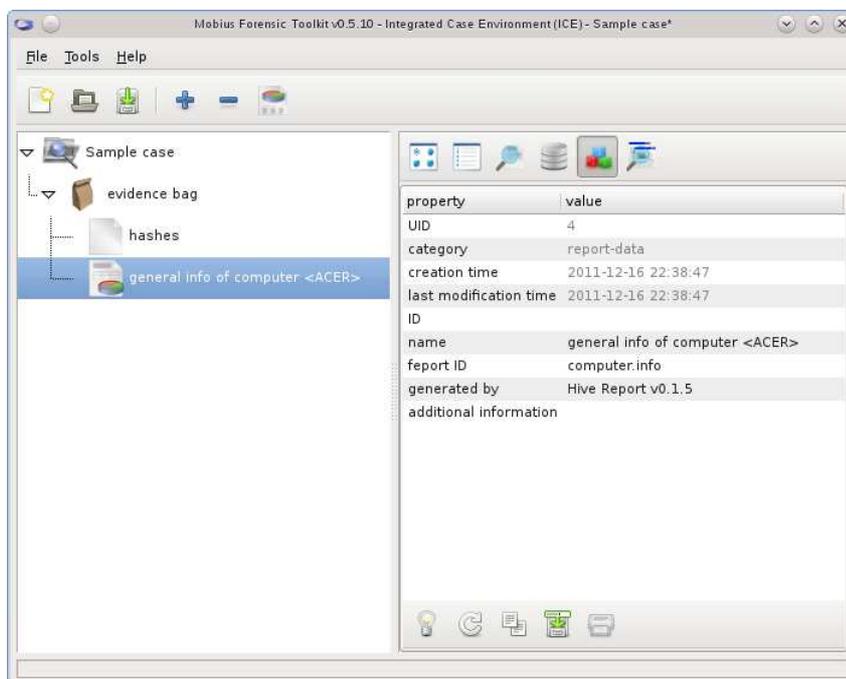


Figure 6.5: The Hive extension — Report data dragged into case treeview

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.

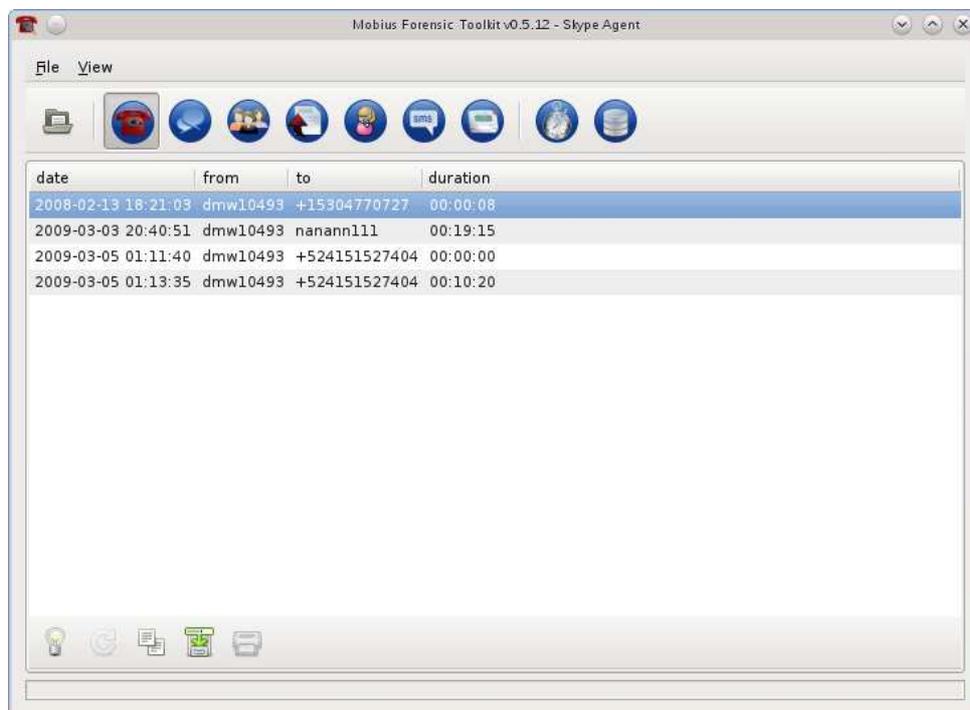


Figure 7.1: The Skype Agent extension

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

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

Hit 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 palette 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.

8.2 Running a report template

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

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:

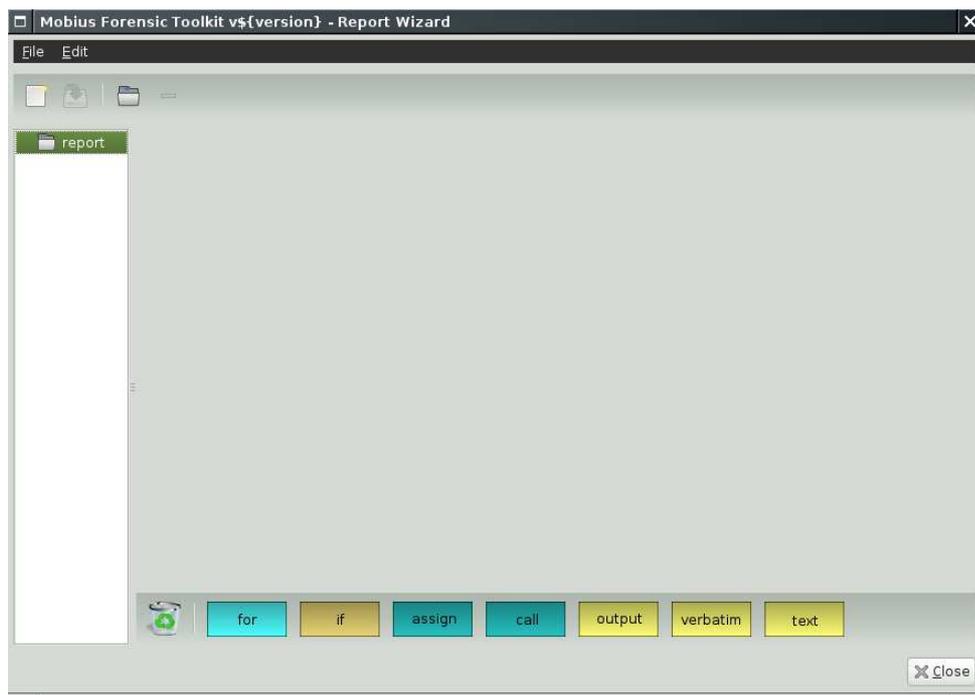


Figure 8.1: Report Wizard running

```
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
```

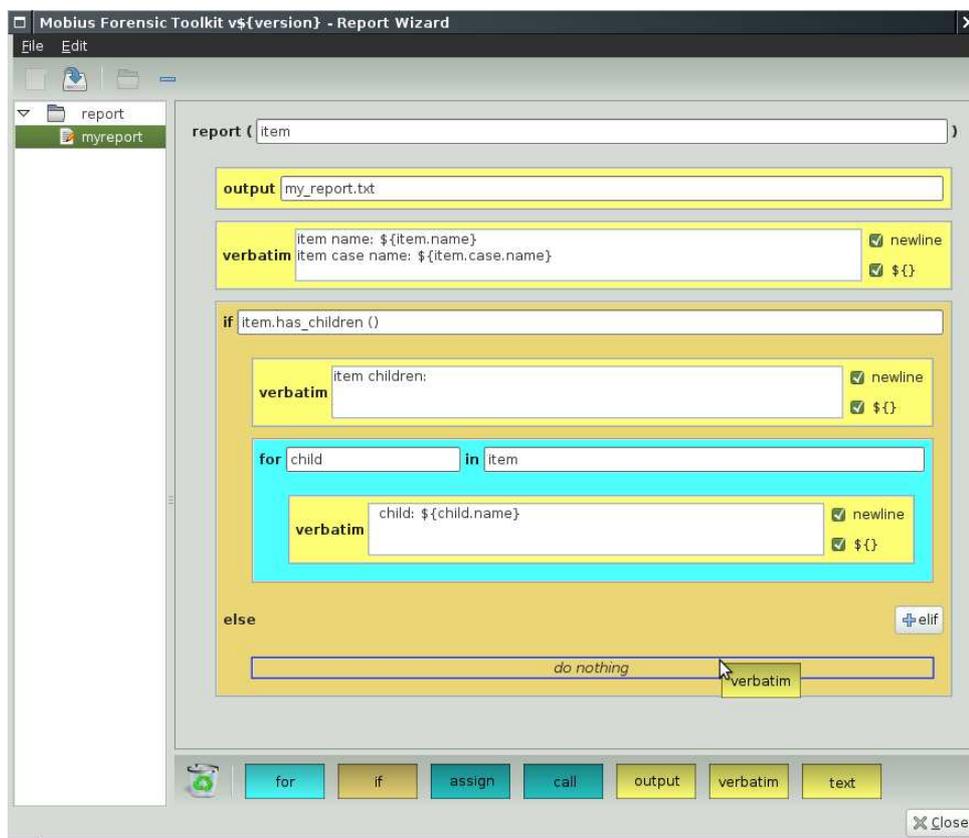


Figure 8.2: Dragging components to the report template

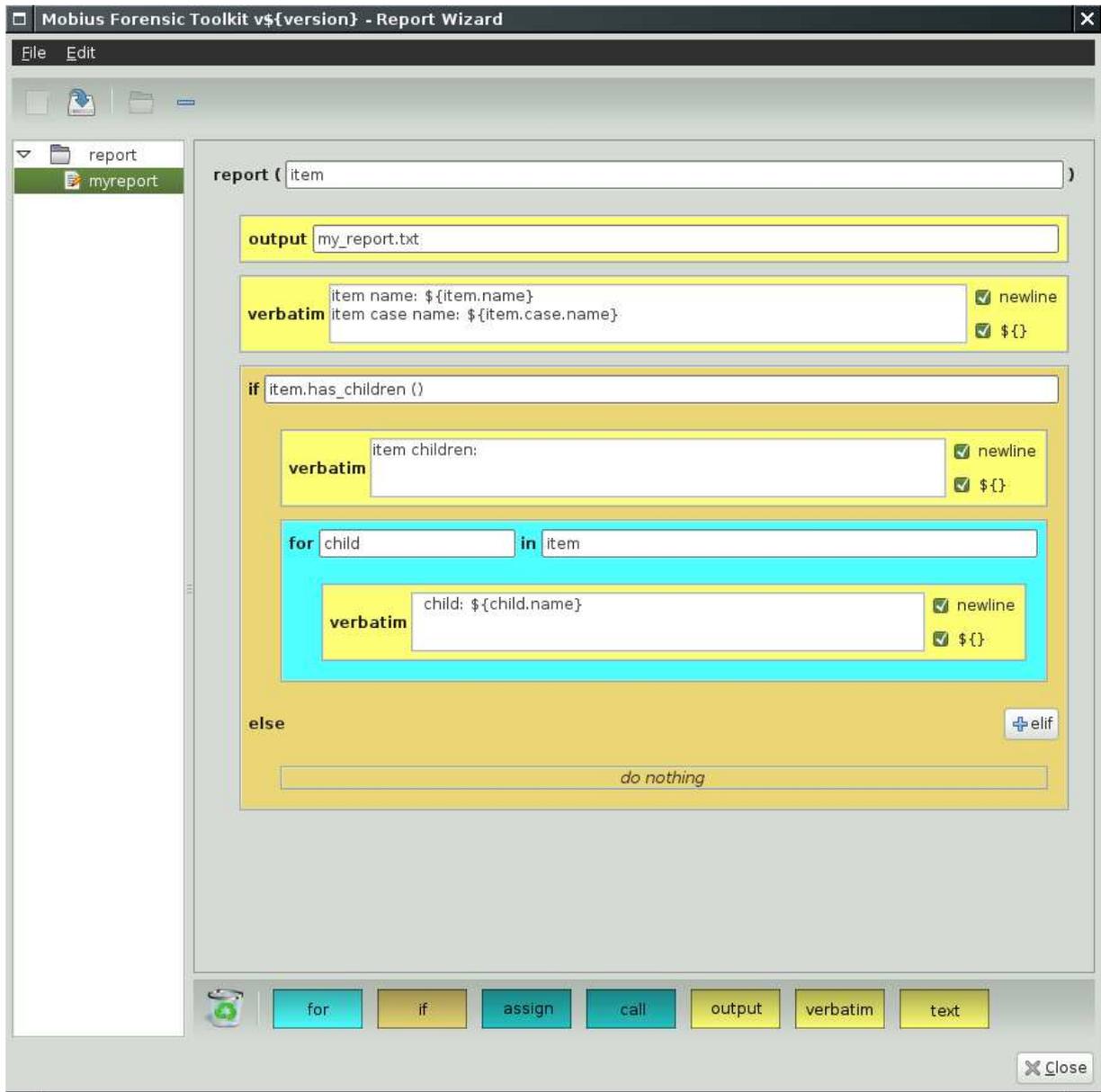


Figure 8.3: myreport template

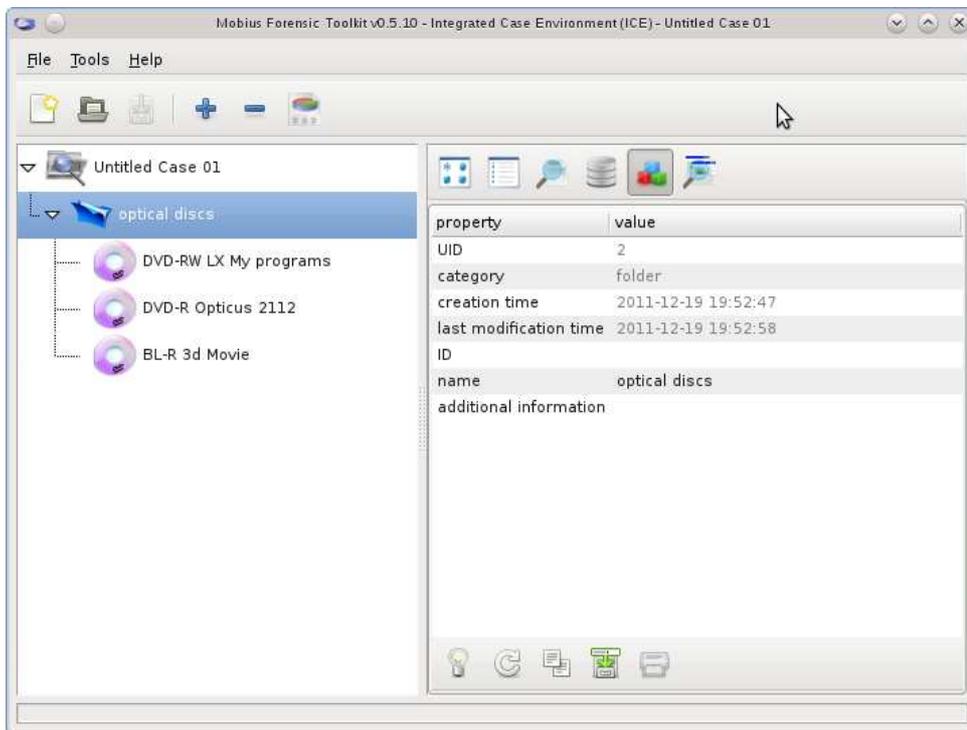


Figure 8.4: Sample case

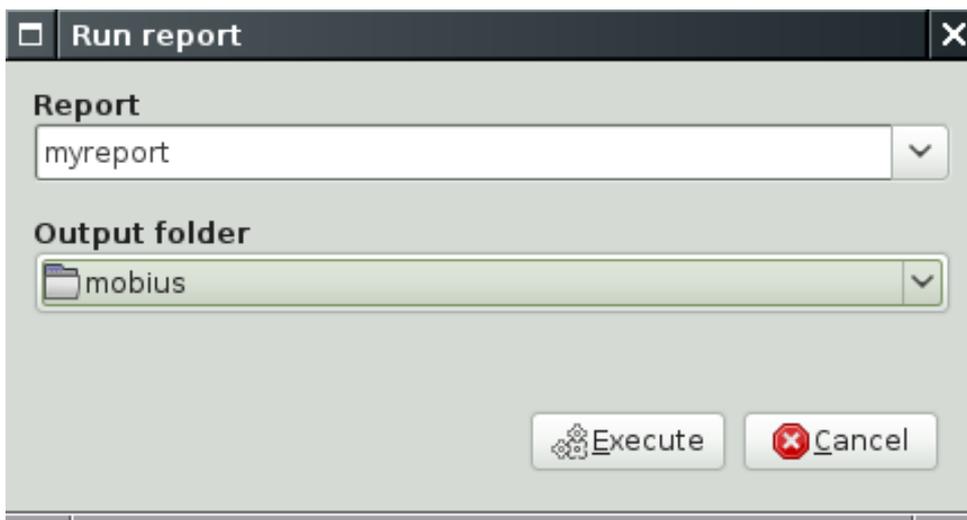


Figure 8.5: Report dialog

9

Imaging floppy disks

The Floppy Imager extension was designed to create logical image files from floppy disks and 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, the blue ones are sectors successfully read and the red ones 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 case base folder.

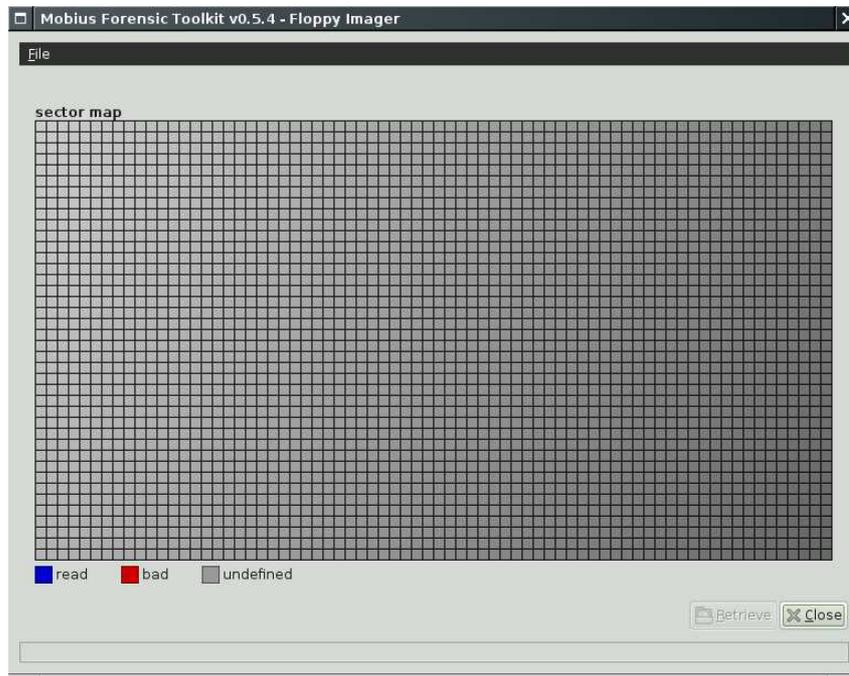


Figure 9.1: Floppy Imager extension

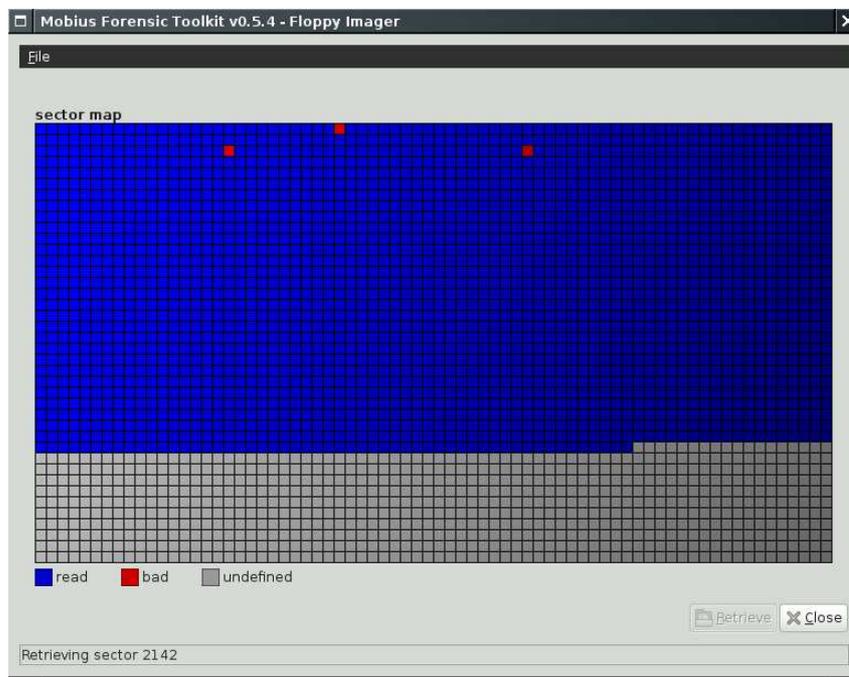


Figure 9.2: The Floppy Imager running